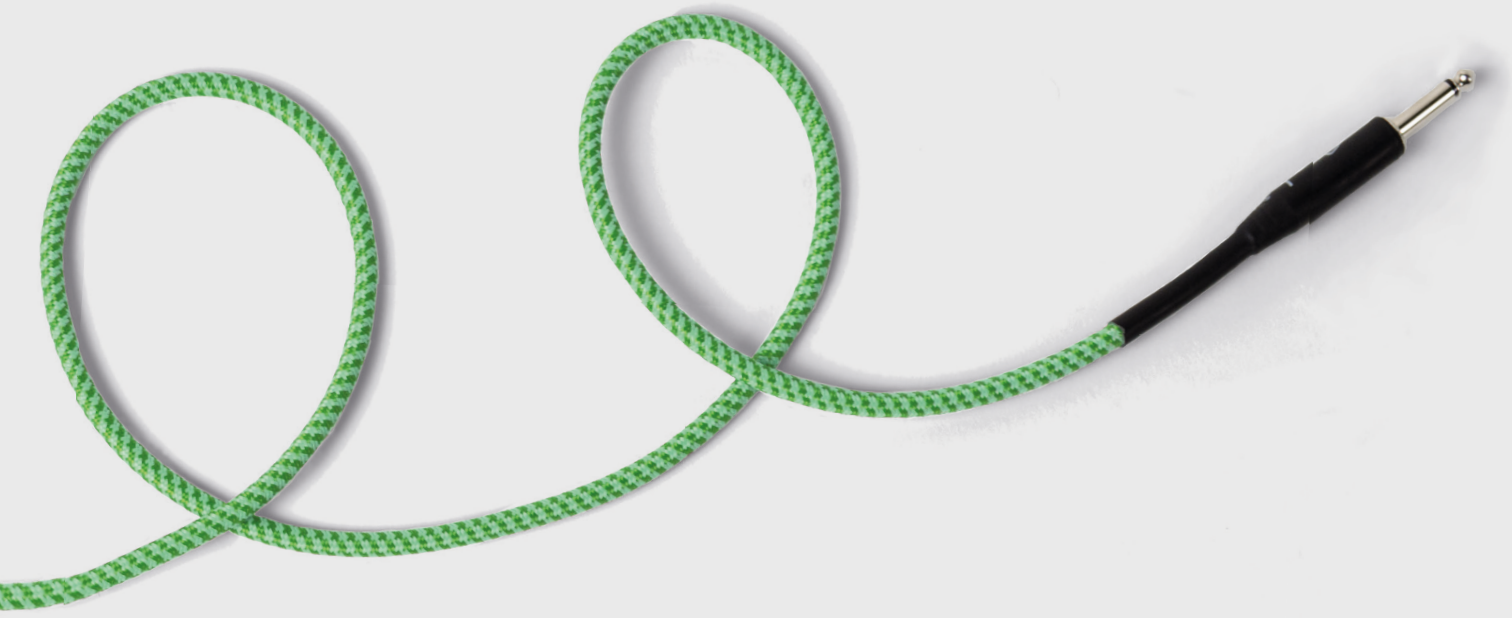
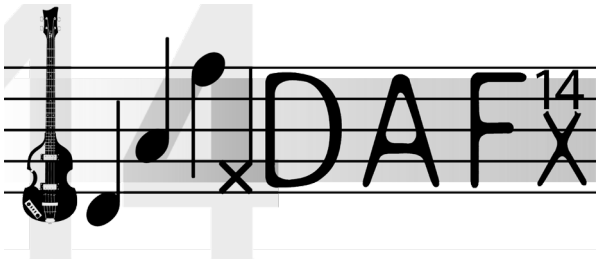
Sascha Disch
Bernd Edler
Jürgen Herre
Meinard Müller
Rudolf Rabenstein
Stefan Turowski

17th International Conference on Digital Audio Effects

Erlangen, Germany, September 1-5, 2014



Fraunhofer IIS
AUDIO LABS
LMS
FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG



17th International Conference on Digital Audio Effects

Erlangen, Germany, September 1-5, 2014

http://www.dafx14.fau.de/

Fraunhofer IIS

AUDIO LABS

LMS

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Credits:
Proceedings produced and cover designed by
Alexander Adami, Fabian-Robert Stöter and Nils Werner
LaTeXstyle by Paolo Annibale
DAFx-14 Logo by Sascha Disch
Cover photo by Valentin Schilling

ISBN 978-3-00-046825-4

Copies may be ordered from:
info@audiolabs-erlangen.de
International Audio Laboratories Erlangen
Am Wolfsmantel 33
91058 Erlangen (Germany)

Foreword

Welcome to DAFx-14

It is our great pleasure to welcome you to the 17th International Conference on Digital Audio Effects, held 01.09.-05.09.2014 in Erlangen, Germany. Playing a leading role in the developement and market introduction of today's audio codecs, Erlangen has a long history in perceptual audio coding and general audio engineering. Since the late 1970s, the Fraunhofer Institute for Integrated Circuits (IIS) and the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) both have continuously contributed to audio engineering with cutting-edge research and are still shaping the future of audio and multimedia. The fruitful collaboration between IIS and FAU is also reflected by the joint organization of this year's edition of the DAFx conference. We are deeply honoured of having the opportunity to host this event at our institutions.

DAFx-14 has a five day programme full of interesting scientific events. The first part of the conference (01.09.-02.09.2014), which takes place at the IIS (Tennenlohe), mainly consists of tutorials, demos, and poster presentations. In particular, we have three tutorials on "Multipitch Analysis of Music Signals" by Anssi Klapuri, on "Audio Structure Analysis of Music" by Meinard Müller, and "Perceptual Audio Coding" by Jürgen Herre, Sascha Disch, and Bernd Edler. The second part of the conference (03.09.-05.09.2014) is held at the FAU campus (Südgelände Technische Fakultät H11) and comprises the oral presentations and further poster presentations of the peer-reviewed scientific contributions.

The DAFx program also features three distinguished keynote speakers, who give their talks on Tuesday, Thursday, and Friday. The first keynote with the title "Improving Time-Frequency Upmix through Time-Domain Processing" is given by Christof Faller, who is managing director at Illusonic and teaching at the Swiss Federal Institute of Technology (EPFL) in Lausanne. The second keynote by Geoffroy Peeters from the Institut de Recherche et Coordination Acoustique/Musique (IRCAM) in Paris addresses the topic of audio indexing for music analysis and music creativity. The third keynote by Christian Hoyer, who is the head of the Bubenreutheum museum association, will explain the historic roots of Frankonian world-class musical instrument building in the Erlangen region and, alongside, point out a quite surprising relation between the Beatles and Erlangen.

The present volume of the proceedings of DAFx-14 contains the complete manuscripts of all peer-reviewed papers presented at the conference. A total of 63 papers entered the review process, out of which 44 contributions were selected for the scientific programme. The mode of presentation was determined after the accept/reject decision and has no relation to the quality of the papers. From the 44 papers, 23 papers were chosen for oral presentation (organized in six sessions), whereas 21 papers were chosen for poster presentation (organized in two sessions). Oral presentations have a 20-minute slot (including setup and questions/answers of the audience), whereas the posters are presented in a 90-minute poster session as well as during coffee breaks. Furthermore, poster presenters have the opportunity (in two minutes and two slides) to announce orally their poster during two "Fast Forward" sessions.

Besides the scientific programme, we also prepared several social events in the evenings, including a welcome reception on Monday, a concert & reception on Tuesday, a conference banquet with music on Wednesday, as well as a visit to Nuremberg on Thursday.

We are very proud to present to you the proceedings of DAFx-14. The conference program was made possible thanks to the hard work of many people including the members of the local organization team, the reviewers from the DAFx-14 programme committee and the DAFx board members. Special

thanks go to this year's sponsors - Dolby Germany GmbH, iZotope Inc., Native Instruments GmbH and Soundtoys Inc. - and also to the supporting institutions - the Fraunhofer Institut Integrated Circuits (IIS), the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the International Audio Laboratories Erlangen (AudioLabs). Last, but not least, the DAFx-14 programme is possible only thanks to the excellent contributions of our community in response to our call for participation. The biggest acknowledgment therefore goes to you, the authors, researchers and participants of this conference.

The DAFx-14 Conference Committee

Sascha Disch (Fraunhofer IIS, General Chair)
Jürgen Herre (FAU, AudioLabs, General Chair)
Rudolf Rabenstein (FAU, LMS, General Chair)
Bernd Edler (FAU, AudioLabs, Scientific Coordinator)
Meinard Müller (FAU, AudioLabs, Scientific Coordinator)
Stefan Turowski (FAU, AudioLabs, Technical Coordinator)

Conference Committees

DAFx Board

Organizing Committee

Programme Committee, Reviewers

Sascha Disch
(Fraunhofer Institut für Integrierte Schaltungen (IIS), Erlangen)

Bernd Edler
(Friedrich-Alexander University Erlangen-Nürnberg (FAU), International Audio Laboratories Erlangen)

Gianpaolo Evangelista
(University of Music and Performing Arts Vienna)

Sebastian Ewert
(Queen Mary University of London)

John ffitch
(University of Bath)

Federico Fontana
(University of Udine)

Volker Gnann
(RWTH Aachen, IENT)

Pierre Hanna
(University of Bordeaux 1)

Jürgen Herre
(Friedrich-Alexander University Erlangen-Nürnberg (FAU), International Audio Laboratories Erlangen)

Robert Hoeldrich
(University of Music and Performing Arts Graz)

Andrew Horner
(HKUST)

Jari Kleimola
(Aalto University, Espoo)

Matthieu Kowalski
(Univ Paris-Sud)

John Lato
(NUI Maynooth)

Victor Lazzarini
(NUI Maynooth)

Jonathan Le Roux
(NTT Corporation)

Heidi-Maria Lehtonen
(Aalto University)

Tapio Lokki
(Aalto University)

Simon Lui
(MIT)

Tom Lysaght
(NUI Maynooth)

Piotr Majdak
(Acoustics Research Institute, Austrian Academy of Sciences)

Sylvain Marchand
(University of Bordeaux 1)

Roland Maas
(Friedrich-Alexander University Erlangen-Nürnberg (FAU))

Rémi Mignot
(Aalto University)

Meinard Mueller
(Friedrich-Alexander University Erlangen-Nürnberg (FAU), International Audio Laboratories Erlangen)

Damian Murphy
(University of York)

Gautham Mysore
(Advanced Technology Labs, Adobe Systems Inc.)

Thibaud Necciari
(Acoustics Research Institute)

Søren Nielsen
(SoundFocus)

Markus Noisternig
(IRCAM)

Jouni Paulus
(Fraunhofer Institut für Integrierte Schaltungen (IIS), Erlangen)

Geoffroy Peeters
(Ircam - CNRS SMTS)

Laurent Pottier
(UJM-CIEREC)

Rudolf Rabenstein
(Friedrich-Alexander University Erlangen-Nürnberg (FAU), Chair of Multimedia Communications)

Lise Regnier
(IRCAM)

Marc Rébillat
(Equipe Audition, Ecole Normale Supérieure)

Sigurd Saue
(Norwegian University of Science and Technology (NTNU), Trodheim)

Lauri Savioja
(Aalto University School of Science and Technology)

Alex Southern
(Aalto University)

Fabian-Robert Stöter
(Friedrich-Alexander University Erlangen-Nürnberg (FAU), International Audio Laboratories Erlangen)

Bob Sturm
(Aalborg University Copenhagen)

Nicolas Sturmel
(GIPSA-Lab)

Clara Suied
(IRBA)

Sakari Tervo
(Aalto University)

Tony Tew
(University of York)

Joe Timoney
(NUI Maynooth)

Stefan Turowski
(Friedrich-Alexander University Erlangen-Nürnberg (FAU), International Audio Laboratories Erlangen)

Jan Tro
(Norwegian University of Science and Technology)

Vesa Valimaki
(Department of Signal Processing and Acoustics, Aalto University School of Electrical Engineering)

Maarten Van Walstijn
(Queens University Belfast)

Tuomas Virtanen
(Tampere University of Technology)

Rory Walsh
(DkIT Ireland)

Jez Wells
(University of York)

Udo Zoelzer
(Helmut Schmidt University Hamburg)

Additional Reviewers

Dominique Fourer
(University of Bordeaux 1)

Jonathan Botts
(Aalto University, Espoo)

Jukka Pätynen
(Aalto University, Espoo)

Contents

Keynotes

Christof Faller: *Improving Time-Frequency Upmix through Time-Domain Processing*

Abstract Upmix has been broadly used in the professional (broadcast) and consumer (home cinema) domains, to convert stereo signals to 5.1 surround. Our motivation to add time-domain methods (such as reverberators, early reflections, equalisers, exciters, and compressors) came originally from the desire of scalability. At the advent of 3D multi-channel surround, we wanted an upmix that would be scalable to almost any number of output channels. It became quickly clear, for instance, that ambience signals to be reproduced with many loudspeakers need to be generated very differently than in a 5.1 upmix. The initial efforts in adding reverberators were frustrating: while one could hear the potential (amazing envelopment), difficult items sounded too often too bad. Ultimately, time-domain processing improved the quality of our upmix beyond scalability. Specifically, I'll describe: early reflections for depth in three dimensions, reverberators for generation of multi-channel ambience signals, equalisation of the center channel, and the use of exciters to enhance room signals.

Geoffroy Peeters: *Audio Indexing for Music Analysis and Music Creativity*

Abstract Since the end of the 90's, audio signal analysis has been used more and more in connection with machine learning for the development of audio indexing. One of the specific types of audio content targeted by this indexing technologies is music and the corresponding research field named Music Information Retrieval (MIR). MIR attempt to develop tools for the automatic analysis of music (score, tempo, chord, key, instrumentation, genre, mood, tag classification).

In this talk, I will review the development of this research field, its connection with other research fields and the motivation for its development: from the initial search and navigation over large music collections paradigm (music search engine) to the more recent computational musicology, ethnomusicology and the use of MIR for music creativity.

Christian Hoyer: *The Beatles and Erlangen? Bubenreuth near Erlangen - the Place where the World-Famous Instruments are Made*

Abstract Legendary bands, orchestras, stars and virtuosos like Yehudi Menuhin, the Bavarian Radio Orchestra, Peter Kraus, Elvis, the Beatles and the Rolling Stones - they all were playing Bubenreuth instruments. In the post-war years, displaced persons from the Sudeten region (Czechoslovakia) brought musical instrument manufacturing and related industries to the region around Erlangen. The region is home to one of every tenth German musical instrument manufacturers. Bubenreuth in particular was transformed from a farming village to a metropolis of German string instrument making. The community council of Bubenreuth - then a small village of less than 500 inhabitants - decided by 1949 that more than 2.000 people would be resettled there in the following years to come.

Whether it is those learning to play an instrument, musicians in philharmonic orchestras or rock stars - they all appreciate Franconian violins and guitars. Both small artisan workshops and semi-industrial manufacturers produce quality products for the home market, but mainly for export. The viola da gamba-shaped electric bass designed by Walter Höfner in 1956 and played by Sir Paul McCartney exemplifies the story of Bubenreuth's roots in the instrument making tradition of the 17th century and how it extends to the manufacturing of electric guitars today. A museum was formed in 2009 in order to maintain the cultural heritage of Bubenreuth.

Tutorials

Anssi Klapuri: *Multipitch Analysis of Music Signals*

Abstract Pitch analysis is an essential part of making sense of music signals. Whereas skilled human musicians perform the task seemingly easily, computational extraction of the note pitches and expressive nuances from polyphonic music signals has turned out to be hard. This tutorial starts from the fundamentals of pitch estimation, explaining the basic challenges of the task (robustness to different sound sources, robustness to polyphony and additive noise, octave ambiguity, inharmonicity, missing data, time-frequency resolution) and the processing principles and sources of information that can be used to tackle those challenges. Among the processing principles, we will discuss why autocorrelation-type estimators (as used in speech processing) do not work for polyphonic data and how they can be amended; how phase information can be utilized; how timbral information must be either explicitly modeled or normalized away; etc. Examples pictures and sounds will be presented in order to illustrate what kind of data we are dealing with and to develop intuition. Towards the end of the talk, I will describe some state-of-the-art systems by different researchers, and from my own experience, mention some of the practical challenges that I have encountered when developing real-time multipitch estimation on mobile devices in last few years.

Meinard Müller: *Audio Structure Analysis of Music*

Abstract One of the attributes distinguishing music from other sound sources is the hierarchical structure in which music is organized. Individual sound events corresponding to individual notes form larger structures such as motives, phrases, and chords, and these elements again form larger constructs that determine the overall layout of the composition. One important goal of audio structure analysis is to divide up a given music recording into temporal segments that correspond to musical parts and to group these segments into musically meaningful categories. One challenge is that there are many different criteria for segmenting and structuring music. This results in conceptually different approaches, which may be loosely categorized in repetition-based, novelty-based, and homogeneity-based approaches. Furthermore, one has to account for different musical dimensions such as melody, harmony, rhythm, and timbre. In this tutorial, I will give an overview of current approaches for the computational analysis of the structure of music recordings, which has been a very active research problem within the area of music information retrieval. As one example, I present a novel audio thumbnailing procedure to determine the audio segment that best represents a given music recording. Furthermore, I show how path and block structures of self-similarity matrices, the most important tool used in automated structure analysis, can be enhanced and transformed. Finally, I report on a recent novelty-based segmentation approach that combines homogeneity and repetition principles in a single representation referred to as structure feature.

Jürgen Herre, Bernd Edler, Sascha Disch: *Perceptual Audio Coding*

Abstract Perceptual audio has been a key ingredient in the multimedia revolution, enabling the availability of high-quality audio over channels with limited channel capacity, such as the Internet, broadcasting or wireless services. Today, mp3 and other perceptual audio coding technologies are ubiquitous in devices, such as CD/DVD players, computers, portable music players and cellular phones. This tutorial covers the basics of perceptual audio coding, starting with what it means to operate according to psychoacoustic principles rather than Mean Square Error (MSE). The most relevant psychoacoustic effects will be briefly reviewed. From the modules of a perceptual audio coder, the filterbank and strategies for quantization and coding are examined in some detail. Furthermore, we discuss tools for joint stereo coding of two channels. Alongside, the most common coding artefacts that originate from violating perceptual transparency criteria will be demonstrated and explained.

Beyond these concepts, modern perceptual audio coders feature tools that can significantly boost their performance further at low bitrates, for example, audio bandwidth extension, parametric stereo or unified speech and audio coding. Some sound examples will be given to illustrate these new advanced tools. Finally, an overview of today's state of the art in compression efficiency is given as well as an outlook of some currently ongoing coding developments.

Sascha Spors, Matthias Geier, Max Schäfer: ***Sound Field Synthesis with the SoundScape Renderer***

Abstract Sound field synthesis with massive-multichannel loudspeaker arrays has been an active research field for the last few decades. Several rendering methods for multiple loudspeakers have been developed including Wave Field Synthesis, Ambisonics, and Vector Base Amplitude Panning. Different loudspeaker installations exist at many institutions throughout Europe. While the their operating software is often home-made and specific to the particular loudspeaker set-up, there exists also a versatile open-source software tool for real-time spatial audio reproduction, the SoundScape Renderer (SSR). It can be adapted to various loudspeaker configurations and provides modules for the most common rendering methods. For headphone use also spatial sound by binaural synthesis is supported. The tutorial gives an introduction to the most common sound field rendering methods, presents the SoundScape Renderer and some of its rendering methods, and allows hands-on experience for a limited number of participants using the 128 loudspeaker array at the Chair of Multimedia Communications and Signal Processing (LMS).

FINITE DIFFERENCE SCHEMES ON HEXAGONAL GRIDS FOR THIN LINEAR PLATES WITH FINITE VOLUME BOUNDARIES

Brian Hamilton, *

Acoustics and Audio Group,
University of Edinburgh
brian.hamilton@ed.ac.uk

Alberto Torin,

Acoustics and Audio Group,
University of Edinburgh
a.torin@sms.ed.ac.uk

ABSTRACT

The thin plate is a key structure in various musical instruments, including many percussion instruments and the soundboard of the piano, and also is the mechanism underlying electromechanical plate reverberation. As such, it is a suitable candidate for physical modelling approaches to audio effects and sound synthesis, such as finite difference methods—though great attention must be paid to the problem of numerical dispersion, in the interest of reducing perceptual artefacts. In this paper, we present two finite difference schemes on hexagonal grids for such a thin plate system. Numerical dispersion and computational costs are analysed and compared to the standard 13-point Cartesian scheme. An equivalent finite volume scheme can be related to the 13-point Cartesian scheme and a 19-point hexagonal scheme, allowing for fitted boundary conditions of the clamped type. Theoretical modes for a clamped circular plate are compared to simulations. It is shown that better agreement is obtained for the hexagonal scheme than the Cartesian scheme.

1. INTRODUCTION

The vibration of thin linear plates is a starting point for the modelling and sound synthesis of many musical systems, such as cymbals, gongs, stiff membranes, soundboards, and instrument bodies. Plate vibration is also important for plate reverberation as a digital audio effect. Among the various approaches adopted for the simulation of linear plates, modal techniques are an attractive option, and can be extended to non-linear equations as well [1, 2]. Finite difference and finite element methods have also been extensively adopted [3].

In the modelling of plates using finite difference methods, minimising numerical dispersion is critical, as it can introduce artefacts, such as a mistuning of modes and incorrect modal densities [4]. The latter effect is due to a loss of bandwidth in the simulations, giving rise to sparsity in frequencies leading to the Nyquist frequency. Numerical dispersion has been, and continues to be, extensively studied for the second-order wave equation [5, 6], but aside from [7], this topic has been neglected for the case of linear plates. Research has instead focused on simulating the non-linear aspects of plate vibration, which are arguably more interesting pursuits [8].

The regular hexagonal grid is an alternative to the regular Cartesian (square) grid in 2-D, and it has been shown to provide computationally efficient finite difference schemes for the second-order wave equation [6], mainly due to the isotropy of discrete Laplacians on the hexagonal grid [9]. It is thus of interest to study discrete biharmonic operators (bilaplacians) on the hexagonal grid, which, to our knowledge, have not been used for time-domain plate simulations. Aside from some sparse references found throughout the numerical methods and scientific computing literature [9, 10, 11, 12], relatively little research has featured the hexagonal discrete biharmonics that will be employed in this study.

This paper is organised as follows. In Section 2, the model equation for the plate is introduced and in Section 3, the hexagonal finite difference schemes are presented along with von Neumann stability conditions. In Section 4, numerical dispersion and computational efficiency are analysed. In Section 5, finite volume formulations are presented to implement boundary conditions, and stability conditions for the boundary value problem are given in terms of matrix eigenvalues. Section 6 features circular plate simulations in order to validate the numerical schemes. Conclusions and future directions of study are given in Section 7.

2. THIN PLATE VIBRATION

Linear lossless vibrations of plates are governed by the following equation [13]

$$\partial_t^2 w + \kappa^2 \Delta^2 w = 0\,, \tag{1}$$

where $w(t, \mathbf{x})$ represents the transverse displacement of the plate, t is time and $t \in \mathbb{R}^+$, $\mathbf{x} := (x, y) \in \mathbb{R}^2$ and Δ is the 2-D Laplacian operator, $\Delta := \partial_x^2 + \partial_y^2$, and thus Δ^2 is the biharmonic operator, or bilaplacian. The notation ∂_t denotes partial differentiation with respect to t, and similarly for ∂_x and ∂_y. κ is a constant defined by

$$\kappa = \sqrt{\frac{Ea^2}{12d(1-\nu^2)}}\,, \tag{2}$$

where d is the plate density in kg/m^3, a is the thickness in m, E is Young's modulus in Pa, and ν is the dimensionless Poisson's ratio. All of these parameters are positive.

Eq. (1) holds as long as the transverse displacement w is small in comparison with the thickness a (small deflections regime) [14], and is the 2-D analogue of the Euler-Bernouilli equation for a beam [13]. For deflections of the same order of magnitude as a, this linear equation no longer holds; some simplifying assumptions on the system must be dropped and a more complicated, non-linear equation must be taken into account (von Kármán-Föppl equations [15]).

* This work was supported by the European Research Council, under grant StG-2011-279068-NESS, and by the Natural Sciences and Engineering Research Council of Canada.

3. NUMERICAL SCHEME

3.1. Temporal and spatial grids

We discretise time with the temporal grid $\mathbb{T} := \{nk, n \in \mathbb{Z}^+\}$, where k is the time-step. Space will be discretised with a spatial grid $\mathbb{G}$ which is either a square (Cartesian) grid: $\mathbb{G}_C := h\mathbb{Z}^2$ or a hexagonal grid $\mathbb{G}_H$ defined by

$$\mathbb{G}_H := \{h\mathbf{V}\mathbf{z} \in \mathbb{R}^2, \mathbf{z} \in \mathbb{Z}^2\}, \quad \mathbf{V} = \begin{bmatrix} 1 & -1/2 \\ 0 & \sqrt{3}/2 \end{bmatrix}. \quad (3)$$

3.2. Difference operators

Let $\hat{w}(t, \mathbf{x})$ represent an approximation to the solution of interest $w(t, \mathbf{x})$. A temporal shift operator may be defined as

$$s_{t\pm}\hat{w} := \hat{w}(t \pm k, \mathbf{x}), \quad (4)$$

and a centered time-difference operator can then be written as

$$\delta_{tt} := \frac{1}{k^2}(s_{t+} - 2 + s_{t-}) = \partial_t^2 + O(k^2). \quad (5)$$

Let us define the spatial shift operator

$$s_{\mathbf{r},h}\hat{w} := \hat{w}(t, \mathbf{x} + \mathbf{r}h) \quad (6)$$

where $\mathbf{r} \in \mathbb{R}^2$. The simplest discrete Laplacian on the regular Cartesian grid is then

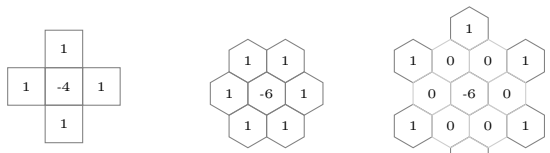
$$\delta_{C,\Delta} := \frac{1}{h^2}\sum_{\mathbf{r}\in\Omega_C}(s_{\mathbf{r},h} - 1) = \Delta + O(h^2), \quad (7)$$

where Ω_C is the set of four unit vectors in $\mathbb{Z}^2$. On the hexagonal grid we consider the following two discrete Laplacians

$$\delta_{H,\Delta} := \frac{2}{3h^2}\sum_{\mathbf{r}\in\Omega_H}(s_{\mathbf{r},h} - 1) = \Delta + \frac{1}{16}h^2\Delta^2 + O(h^4), \quad (8)$$

$$\delta^*_{H,\Delta} := \frac{2}{9h^2}\sum_{\mathbf{r}\in\Omega^*_H}(s_{\mathbf{r},h} - 1) = \Delta + \frac{3}{16}h^2\Delta^2 + O(h^4), \quad (9)$$

where Ω_H and Ω^*_H are the sets of six vectors with norms h and $\sqrt{3}h$ in $\mathbb{G}_H$ respectively. These discrete Laplacians on their respective grids are illustrated in Fig. 1.



(a) $\delta_{C,\Delta}$ (b) $\frac{3}{2}\delta_{H,\Delta}$ (c) $\frac{9}{2}\delta^*_{H,\Delta}$

Figure 1: Stencil weights for discrete Laplacians, scaled by h^2

Now we can construct discrete biharmonics by the composition of discrete Laplacians in the following manner

$$\delta_{C,\Delta^2} := (\delta_{C,\Delta})^2 = \Delta^2 + O(h^2), \quad (10)$$

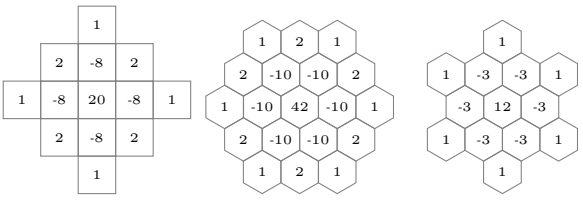
$$\delta_{H,\Delta^2} := (\delta_{H,\Delta})^2 = \Delta^2 + \frac{1}{8}h^2\Delta^3 + O(h^4). \quad (11)$$

The Cartesian biharmonic δ_{C,Δ^2} is a stencil that employs 13 points. The second-order error in (10) is anisotropic so it is not displayed. The hexagonal biharmonic δ_{H,Δ^2} is a 19-point stencil, and has an isotropic second-order error term (the triharmonic operator), which is due to the isotropic second-order error term in $\delta_{H,\Delta}$.

Another biharmonic on the hexagonal grid, using only 13 points [can be written as a linear combination of $\delta_{H,\Delta}$ and $\delta^*_{H,\Delta}$:

$$\delta^\star_{H,\Delta^2} := \frac{8}{h^2}(\delta^*_{H,\Delta} - \delta_{H,\Delta}) = \Delta^2 + O(h^2). \quad (12)$$

This discrete biharmonic is different from δ_{C,Δ^2} and δ_{H,Δ^2} in that it cannot be decomposed into the composition of two discrete Laplacians. The three discrete biharmonics are shown on their respective grids in Fig. 2.



(a) δ_{C,Δ^2} (b) $\frac{9}{4}\delta_{H,\Delta^2}$ (c) $\frac{9}{16}\delta^\star_{H,\Delta^2}$

Figure 2: Stencil weights for discrete biharmonics, scaled by h^4

.

3.3. Finite difference schemes

Combining these operators gives three finite difference schemes for (1)

$$\delta_{tt}\hat{w} + \kappa^2\delta_{\Delta^2}\hat{w} = 0, \quad (t, \mathbf{x}) \in \mathbb{T} \times \mathbb{G}, \quad (13)$$

with possible choices of $\delta_{\Delta^2} \in \{\delta_{C,\Delta^2}, \delta_{H,\Delta^2}, \delta^\star_{H,\Delta^2}\}$ and its appropriate spatial grid $\mathbb{G} \in \{\mathbb{G}_C, \mathbb{G}_H\}$. Each scheme has the time recursion

$$\hat{w}^+ = (2 - \mu^2\delta^h_{\Delta^2})\hat{w} - \hat{w}^-, \quad (14)$$

where $\hat{w}^\pm := s_{t\pm}\hat{w}$, $\mu := \kappa k/h^2$ is a free parameter to be set, analogous to the Courant number in wave equation schemes, and $\delta^h_{\Delta^2} := h^4\delta_{\Delta^2}$. The recursion begins from the two known (or approximated) values $\hat{w}(0, \mathbf{x})$ and $\hat{w}(k, \mathbf{x})$ determined from the initial conditions. Note that this explicit update is parallelisable, and thus, well-suited to GPU implementations [16].

3.4. Stability analysis

To determine stability conditions, we can take the Z-transform of (13) to get the following quadratic equation in $z \in \mathbb{C}$

$$z + \mu^2\Lambda - 2 + z^{-1} = 0, \quad (15)$$

where $\Lambda = \Lambda(\boldsymbol{\xi})$ is the Fourier symbol of the operator $\delta^h_{\Delta^2}$ and $\boldsymbol{\xi} \in \mathbb{R}^2$ are the spatial frequencies. For now, we assume that $\Lambda(\boldsymbol{\xi})$ has the property $\Lambda \geq 0$. A stability condition (disallowing exponential growth) is found from the condition $|z| \leq 1$, which leads to

$$\mu \leq \mu_{\max} := \sqrt{4/\Lambda_{\max}}, \quad (16)$$

where $\Lambda_{\max} := \max_{\boldsymbol{\xi}}\Lambda$ for the spatial frequencies $\boldsymbol{\xi} \in \mathbb{R}^2$. For the three biharmonics $\delta_{C,\Delta^2}, \delta_{H,\Delta^2}, \delta^\star_{H,\Delta^2}$ (scaled by h^4) we have respectively

$$\Lambda_{C,\max} = 64, \quad \Lambda_{H,\max} = 36, \quad \Lambda^\star_{H,\max} = 48. \quad (17)$$

The first two values are given by previous studies [11], and the latter can be found by examining $\Lambda^\star_{H,\max}$. Stability limits for the schemes in (13) are respectively

$$\mu_{C,\max} = 1/4\,, \quad \mu_{H,\max} = 1/3\,, \quad \mu^\star_{H,\max} = \sqrt{1/12}\,. \quad (18)$$

Note that both of the hexagonal schemes give higher $\mu_{\max}$ than the Cartesian scheme, which allows for a larger time-step when h is fixed. On the other hand, if k is fixed to $k = 1/F_s$, as is common in sound synthesis applications, this implies a smaller minimum grid spacing (spatial step). Setting h as small as possible is generally a good choice for numerical dispersion and maximising the temporal bandwidth in the approximation [4]. However, this also increases the density of the spatial grid, and the hexagonal grid is already $2/\sqrt{3} \approx 1.15$ times more dense than the square grid for the same h. More will be said about this in Section 4.1.

4. NUMERICAL DISPERSION

The dispersion relation for our plate equation is

$$\omega = \pm\kappa|\boldsymbol{\xi}|^2\,, \quad (19)$$

where $\omega \in \mathbb{R}$ represents the temporal frequency in rad/s and $|\boldsymbol{\xi}|$ is the *wavenumber* in rad/m. The plate system is dispersive, as seen by its phase velocity:

$$v_\phi = \kappa|\boldsymbol{\xi}|\,. \quad (20)$$

In other words, plane-waves with small wavenumbers travel slower than plane-waves with large wavenumbers.

In order to analyse numerical dispersion of the finite difference scheme it helps to define a normalised spatial frequency $\boldsymbol{\xi}_h := \boldsymbol{\xi}h$ and a normalised frequency $\omega_k := \omega k$. We can then write the Fourier symbol for each discrete Laplacian $\delta_{C,\Delta}, \delta_{H,\Delta}, \delta^*_{H,\Delta}$, scaled by h^2, as

$$\Gamma_C(\boldsymbol{\xi}_h) := -2\sum_{\mathbf{r}\in\Omega_C} \sin^2(\boldsymbol{\xi}_h\cdot\mathbf{r}/2)\,, \quad (21a)$$

$$\Gamma_H(\boldsymbol{\xi}_h) := -\frac{4}{3}\sum_{\mathbf{r}\in\Omega_H} \sin^2(\boldsymbol{\xi}_h\cdot\mathbf{r}/2)\,, \quad (21b)$$

$$\Gamma^*_H(\boldsymbol{\xi}_h) := -\frac{4}{9}\sum_{\mathbf{r}\in\Omega^*_H} \sin^2(\boldsymbol{\xi}_h\cdot\mathbf{r}/2)\,. \quad (21c)$$

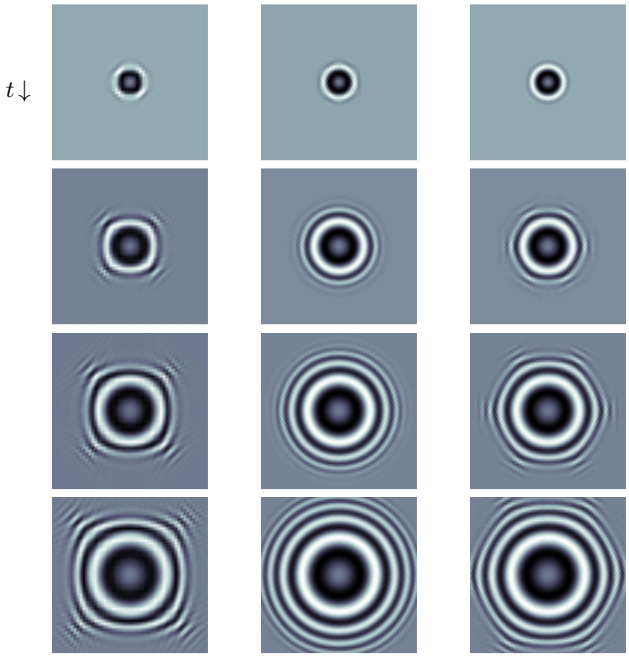
This allows us to build Λ_C, Λ_H, $\Lambda^\star_H$ as follows

$$\Lambda_C = (\Gamma_C)^2\,, \quad \Lambda_H = (\Gamma_H)^2\,, \quad \Lambda^\star_H = 8(\Gamma^*_H - \Gamma_H)\,. \quad (22)$$

Clearly, Λ_C and Λ_H are non-negative. Examining $\Lambda^\star_H$ gives the same result, but we leave this out for brevity. We can then write the *relative phase velocity* as

$$v_{\text{rel}}(\boldsymbol{\xi}_h) := \frac{\omega_k(\boldsymbol{\xi}_h)}{\mu|\boldsymbol{\xi}_h|^2}\,, \quad \omega_k(\boldsymbol{\xi}_h) := 2\arcsin\left(\frac{\mu}{2}\sqrt{\Lambda}\right)\,, \quad (23)$$

for $\omega_k \in (0,\pi]$ and $\boldsymbol{\xi}_h \in \mathbb{B}$, where $\mathbb{B}$ is the wavenumber cell of the grid. For the square grid, $\mathbb{B}$ is a square centered at zero with sides of length 2π, whereas for the hexagonal grid, $\mathbb{B}$ is the Voronoi cell (a hexagon) of the lattice spanned by the vectors:$(2\pi, 2\pi/\sqrt{3})^T$ and $(0, 4\pi/\sqrt{3})^T$ [6]. The relative phase velocity should ideally be unity everywhere. Figs. 3(a)-(c) display the relative phase velocities of the finite difference schemes with $\mu = \mu_{\max}$. Note that the hexagonal wavenumber cell is slightly bigger than the Cartesian wavenumber cell, this is ultimately a result of the denser grid for the same h. Also, the isotropic characteristic to δ_{H,Δ^2} can be seen in Fig. 3b. Some simulations, without taking boundaries into account, are presented in Fig. 4 to demonstrate how the directional dependence of the schemes are reflected in the numerical approximation. The initial conditions $\hat{w}(0,\mathbf{x})$ and $\hat{w}(k,\mathbf{x})$ are set to a spatial Gaussian for each case, and the simulations are stopped at the same time instant. It is clear that the approximation in Fig. 4b has less directional dependence than the other two.



(a) 13-pt Cartesian (b) 19-pt hexagonal (c) 13-pt hexagonal

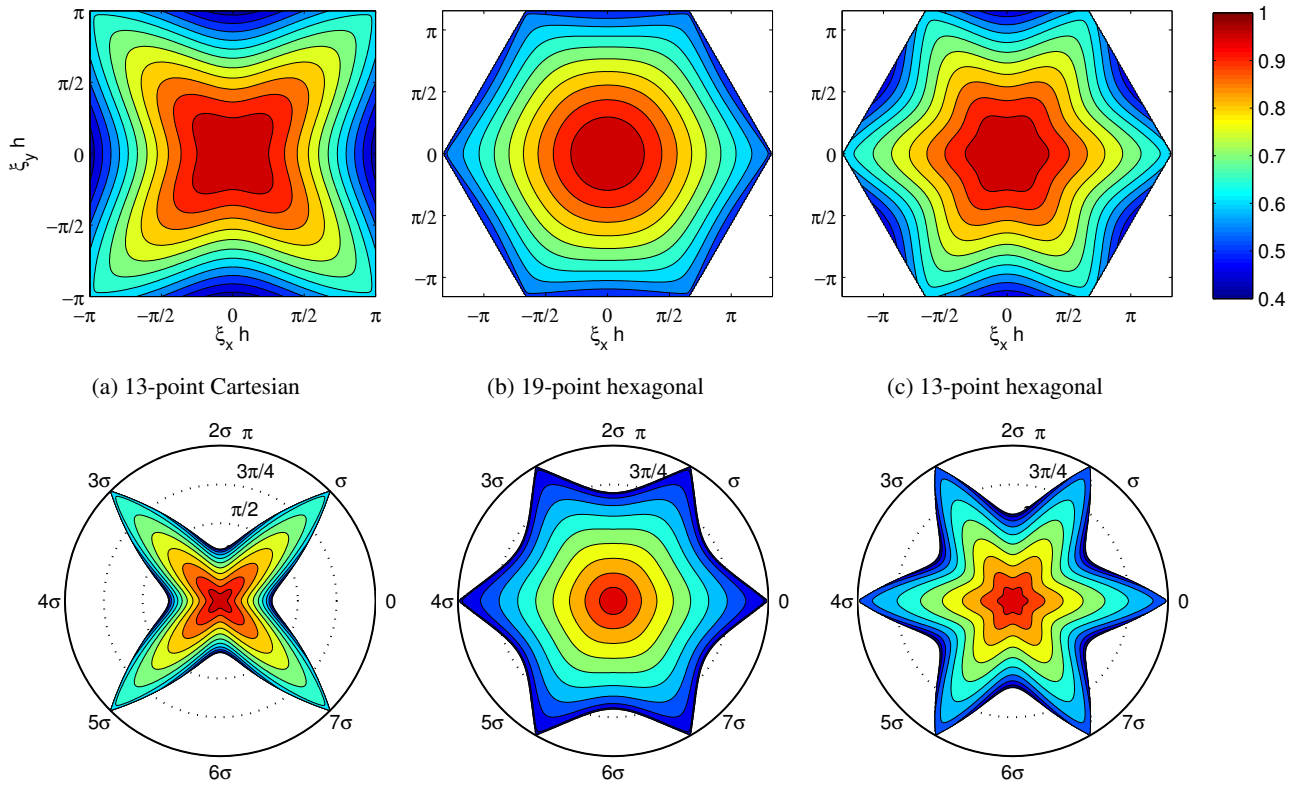
Figure 4: Spatial response to same initial conditions (Gaussian), demonstrating (an)isotropy. Time-step fixed across schemes. Snapshots after 9, 18, 27, and 36 time-steps (top to bottom).

We would like to compare the dispersion for the Cartesian scheme to the hexagonal schemes, but this can be difficult since they are defined in different wavenumber cells. To make for a better comparison we can use the dispersion relation to reassign the relative phase velocities to ω_k and an angle of propagation $\theta \in [0, 2\pi]$, giving a function $v_{\text{rel}}(\omega_k(\boldsymbol{\xi}_h), \theta(\boldsymbol{\xi}_h))$ where

$$\theta(\boldsymbol{\xi}_h) = \arctan\left((\boldsymbol{\xi}_h\cdot\hat{\mathbf{y}})/(\boldsymbol{\xi}_h\cdot\hat{\mathbf{x}})\right)\,. \quad (24)$$

where $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ are the standard unit vectors in $\mathbb{R}^2$. Now we have a single domain on which to compare relative phase velocities for the Cartesian and hexagonal schemes. These relative phase velocities, are displayed in Figs. 3(d-e).

It is from this point of view (temporal frequencies) that we see large variations between the schemes. There are two effects of numerical dispersion that are prevalent here. The first is that the high spatial frequencies are compressed into a small band of temporal frequencies along the worst-case directions (multiples of $\pi/2$ for the Cartesian scheme, odd multiples of $\pi/6$ for hexagonal schemes). This will cause an unnatural modal density within those bands of frequencies. The second effect is that the spatial Nyquist



(a) 13-point Cartesian (b) 19-point hexagonal (c) 13-point hexagonal

(d) 13-point Cartesian (e) 19-point hexagonal (f) 13-point hexagonal

Figure 3: Contour plots of relative phase velocity as a function of $\boldsymbol{\xi}_h \in \mathbb{B}$, ($\boldsymbol{\xi}_h = (\xi_x h, \xi_y h)$) (top row), and $\omega_k \in [0, \pi]$ (radial) and $\theta \in [0, 2\pi]$ (bottom row), where $\sigma = \pi/4$. Contours mark 5% deviations in relative phase velocity.

does not remap to the temporal Nyquist in every direction, creating directional *cutoff frequencies*. Thus, above the smallest directional cutoff frequencies the modal density will be incorrect. These effects are worst in the Cartesian scheme, while the 19-point hexagonal scheme experiences the least of these effects.

4.1. Normalising for computational cost

It can be argued that this is still not a fair comparison between Cartesian and hexagonal schemes, since δ_{H,Δ^2} uses 19 spatial points instead of 13 for δ_{C,Δ^2}. Furthermore, for a fixed time-step (fixed sample rate) and $\mu = \mu_{\max}$ the hexagonal grid will be more dense than the Cartesian one. This ultimately leads to more computation per unit time and space. In principle, it is always possible to oversample the grid in order to achieve the same levels of accuracy or simulated bandwidth with the 13-point Cartesian scheme, so we have to somehow normalise for computational costs.

Three different methods can be adopted to evaluate the finite difference schemes. First, we will consider the same time-step for each scheme (no normalisation of computational cost), then we will normalise for spatiotemporal grid ($\mathbb{T} \times \mathbb{G}$) densities, and finally we will consider normalised spatiotemporal densities of addition operations.

Let the time-step for each scheme be set to $k = \chi k'$, where k' is a constant and χ will represent computational cost normalisation factors with respect to the Cartesian scheme. As such, χ is always set as $\chi = 1$ for the Cartesian scheme. When $\chi = 1$ for all schemes, normalisation for computational cost is ignored. On the other hand, when χ is chosen as $\chi = \sqrt{4\eta\mu}$ with $\eta = 1$ for the Cartesian scheme and $\eta = 2/\sqrt{3}$ for the hexagonal schemes, then we have normalised for density of points in space and time, with respect to the Cartesian scheme. When $\chi = \sqrt{(4/13)\gamma\eta\mu}$, where γ is the number of points in the stencil, then the schemes will be normalised for the density of additions per space and time, with respect to the Cartesian scheme. We neglect multiplications for brevity.

The relative phase velocities with these normalisations along the respective worst-case directions are shown in Fig. 5. It can be seen that, even after normalising for the extra computational costs, the hexagonal schemes are more efficient at reducing numerical dispersion than the Cartesian scheme. In parallel implementations of finite difference schemes for plates, such as [17], the normalisation for additions, which are easily parallelised, may not be important so we ignore this normalisation for the following discussion.

For the plate problem, reducing the time-step results in a squared increase in the total number of operations (2x increase in F_s equals 4x computational cost). With this in mind, we can compare schemes in terms of a *relative computational efficiency* to attain a certain accuracy in the relative phase velocity up to a given frequency, as in [5] for wave equation schemes. For example, with data taken from Fig. 5, we can calculate that the 19-point and 13-point hexagonal schemes are respectively 2.1 and 1.8 times more efficient than the 13-point Cartesian scheme for a one-percent rela-

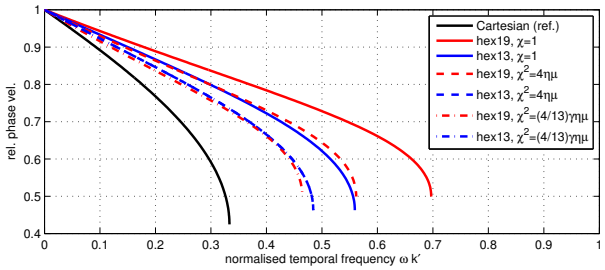


Figure 5: Relative phase velocity along worst-case direction for three schemes, with various normalisations for computational cost. The time-step in each case is set as $k = \chi k'$ for k' fixed, so $\chi = 1$ implies no normalisation (same time-step), $\chi^2 = 4\eta\mu$ normalises for spatiotemporal grid densities, and $\chi^2 = (4/13)\gamma\eta\mu$ normalises for spatiotemporal density of additions. Note, $\gamma = 13$ for the 13-point hexagonal scheme, so the $\chi^2 = 4\eta\mu$ and $\chi^2 = (4/13)\gamma\eta\mu$ curves overlap.

tive phase velocity error tolerance. Such relative efficiency numbers could be given for the entire range of phase velocity errors, but it is unknown how much numerical dispersion is tolerated for audio applications of this plate model, and whether numerical dispersion is perceptually distinguishable from the system's underlying dispersion.

Perhaps a more useful comparison is in terms of the global cutoff frequencies after normalisation, as this gives an idea of the modal density across the temporal range of frequencies, and thus a measure of how 'rich' the output sound will be. In terms of global cutoff frequencies, we can calculate that the 19-point and 13-point hexagonal schemes are respectively 4.3 and 2.8 times more efficient than the 13-point Cartesian scheme.

5. FINITE VOLUME BOUNDARIES

In this section, we present a finite volume formulation of the 13-point Cartesian scheme and the 19-point hexagonal scheme, in order to simplify the implementation of certain boundary conditions. The 13-point discrete biharmonic on the hexagonal grid does not decompose into the composition two discrete Laplacians, so it cannot be easily be interpreted within the following finite volume framework.

Let $\mathcal{V}$ denote a closed 2-D volume and $\partial\mathcal{V}$ its boundary. The finite grid under consideration can then be written as $\overline{\mathbb{G}} := \mathbb{G} \cap \mathcal{V}$. We start by rewriting (1) as the system of two equations:

$$\partial_t v = \kappa \Delta m \,, \tag{25a}$$

$$\partial_t m = -\kappa \Delta v \,, \tag{25b}$$

where $m = m(t, \mathbf{x})$ is the initial moment and $v = v(t, \mathbf{x})$ is the initial velocity, which is related to w by:

$$v = \partial_t w \tag{26}$$

In this system, the two initial conditions to specify are $v(0, \mathbf{x})$ and $m(0, \mathbf{x})$. Boundary conditions for the plate can be of the clamped type:

$$v = \mathbf{n} \cdot \nabla v = 0 \,, \quad \mathbf{x} \in \partial\mathcal{V} \tag{27}$$

where $v = 0$ denotes a homogeneous Dirichlet condition and $\mathbf{n} \cdot \nabla v = 0$ denotes a homogeneous Neumann boundary condition. Another set of Dirichlet boundary conditions is the following:

$$v = m = 0 \,, \quad \mathbf{x} \in \partial\mathcal{V} \,. \tag{28}$$

This set of conditions may be a simplified form of the "simply supported" conditions for certain geometries, such as rectangular plates with Cartesian grids.

Consider a tiling of closed cells $\mathcal{C}_i$ whose interiors are pairwise disjoint, and the tiling fills up the volume $\mathcal{V}$, i.e. $\bigcup_i \mathcal{C}_i = \mathcal{V}$. Now consider one cell surrounding some point $\mathbf{x}_i \in \overline{\mathbb{G}}$. For now we will focus on one of the two equations, as they are similar. Integrating both sides of (25a) over the volume of the cell and applying the divergence theorem we have:

$$\int_{\mathcal{C}_i} \partial_t v \, dS = \kappa \int_{\partial\mathcal{C}_i} \mathbf{n} \cdot \nabla m \, dr \,, \tag{29}$$

where $\partial\mathcal{C}_i$ denotes the *boundary of* $\mathcal{C}_i$ and where $\mathbf{n}$ is the normal vector pointing out of the cell at $r \in \mathcal{C}_i$. Now, consider that this cell has neighbouring cells $\mathcal{C}_j$ with indices j in the set of neighbour indices N_i. The interiors of cells are pairwise disjoint but their closures can intersect. Let these intersections be denoted by $\mathcal{S}_{ij} := \mathcal{C}_i \cap \mathcal{C}_j$; these are the sides of the cell. Furthermore, let $\mathcal{S}_{i(b)} := \mathcal{C}_i \cap \partial\mathcal{V}$ denote the boundary side of the cell. Since $\partial\mathcal{C}_i = (\bigcup_j \mathcal{S}_{ij}) \cup \mathcal{S}_{i(b)}$ we can write (29) as

$$\int_{\mathcal{C}_i} \partial_t v \, dS = \kappa \sum_{j \in N_i} \int_{\mathcal{S}_{ij}} \mathbf{n} \cdot \nabla m \, dr + \kappa \int_{\mathcal{S}_{i(b)}} \mathbf{n} \cdot \nabla m \, dr \,, \tag{30}$$

The last term describes one half of the system at the boundary and this can be set to zero for Neumann conditions. Let the 2-D volume (area) of the cell be V_i and the length of each side be S_{ij} and similarly for the boundary side $S_{i(b)}$. We define the first-order spatial and time differences:

$$\delta_{t\pm} := \pm \frac{1}{k}(s_{t\pm} - 1) \,, \tag{31a}$$

$$\delta_{ij} \hat{w}_j := \frac{1}{h_{ij}}(\hat{w}_j - \hat{w}_i) \,, \tag{31b}$$

where $h_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|$. Consider the variables $\hat{v}_i := \hat{v}(t + k/2, \mathbf{x}_i)$ and $\hat{m}_i := \hat{m}(t, \mathbf{x}_i)$, representing approximations to $v(t + k/2, \mathbf{x}_i)$ and $m(t, \mathbf{x}_i)$ respectively. Neglecting for now the boundary term, we can approximate (30) with the following, and (25b) by the same procedure:

$$V_i \delta_{t-} \hat{v}_i = \quad \kappa \sum_{j \in N_i} S_{ij} \delta_{ij} \hat{m}_i \,, \tag{32a}$$

$$V_i \delta_{t+} \hat{m}_i = -\kappa \sum_{j \in N_i} S_{ij} \delta_{ij} \hat{v}_i \,. \tag{32b}$$

Note that the time differences are centered, since $\hat{v}_i$ is staggered in time. The spatial difference will also be centered about the sides of the cells for the grids (square and hexagonal) considered here. Now rearranging for the update equations we have:

$$\hat{v}_i = \hat{v}_i^- + \frac{\kappa k}{V_i} \sum_{j \in N_i} S_{ij} \delta_{ij} \hat{m}_i \,, \tag{33a}$$

$$\hat{m}_i^+ = \hat{m}_i - \frac{\kappa k}{V_i} \sum_{j \in N_i} S_{ij} \delta_{ij} \hat{v}_i \,, \tag{33b}$$

where $\hat{v}_i^- := s_{t-} \hat{v}_i$ and $\hat{m}_i^+ := s_{t+} \hat{m}_i$. The update does not change when Neumann conditions are applied because the neglected boundary term would be set to zero. For the clamped conditions, it then suffices to fix $\hat{v}_i = 0$ when $S_{i(b)} > 0$. For

the conditions (28), we also fix $\hat{m}_i^+ = 0$ when $S_{i(b)} > 0$. If, on the other hand, we update both values at the boundaries, then this implies the non-physical, yet well-posed, boundary conditions:

$$\mathbf{n} \cdot \nabla m = \mathbf{n} \cdot \nabla v = 0\,, \quad \mathbf{x} \in \partial\mathcal{V}\,. \tag{34}$$

We include this boundary condition because it arises naturally from the finite volume framework and it may provide interesting artificial reverberation.

To establish the link with the finite difference schemes, we will now consider square and regular hexagonal tilings of $\mathcal{V}$. These tilings may be *locally irregular* [18], which means that cells on the interior are regular polygons from the Voronoi tessellations of $\mathbb{G}_C$ or $\mathbb{G}_H$, but cells that intersect with the boundary of $\mathcal{V}$ may be irregular. Now consider a cell $\mathcal{C}_i$ with $S_{i(b)} = 0$ and $S_{j(b)} = 0$ for $j \in N_i$. It is straightforward to show [19] that we can recover the following discrete Laplacians from the finite volume formulations:

$$\frac{1}{V_i}\sum_{j\in N_i} S_{ij}\delta_{ij}\hat{v}_i = \delta_{C,\Delta}\hat{v}_i\,, \quad \mathbf{x}_i \in \mathbb{G}_C\,, \tag{35a}$$

$$\frac{1}{V_i}\sum_{j\in N_i} S_{ij}\delta_{ij}\hat{v}_i = \delta_{H,\Delta}\hat{v}_i\,, \quad \mathbf{x}_i \in \mathbb{G}_H\,. \tag{35b}$$

Then, using the identity $\delta_{t+}\delta_{t-}\hat{v}_i = \delta_{tt}\hat{v}_i$, it follows that Eqs. (32a) and (32b) simplify to the second-order 13-point Cartesian and 19-point hexagonal schemes respectively in $\hat{v}_i$. The variable $\hat{w}_i^+$ can be recovered from $\hat{v}_i = \delta_{t+}\hat{w}_i$.

5.1. Matrix formulation and stability

The approximations $\hat{v}$ and $\hat{m}$ can be written as the $N \times 1$ vectors $\mathbf{v}$ and $\mathbf{m}$ with the values of $\hat{v}_i$ and $\hat{m}_i$ for $\mathbf{x}_i \in \overline{\mathbb{G}}$ ($N = |\overline{\mathbb{G}}|$) at a particular time t. The system (32) can be rewritten in the matrix-vector form:

$$\delta_{t-}\mathbf{v} = \quad \kappa\mathbf{L}_1\mathbf{m}\,, \tag{36a}$$

$$\delta_{t+}\mathbf{m} = -\kappa\mathbf{L}_2\mathbf{v}\,, \tag{36b}$$

where $\mathbf{L}_1$ and $\mathbf{L}_2$ are $N \times N$ matrices corresponding to δ_Δ with Dirichlet conditions possibly imposed. These matrices can be defined as follows. Consider $\mathbf{L}$ to be either $\mathbf{L}_1$ or $\mathbf{L}_2$. For each row i of the matrix $\mathbf{L}$, the entries l_{ij} can be written as:

$$l_{ij} = \frac{S_{ij}}{V_i h_{ij}}\,, \quad i \neq j\,, \tag{37a}$$

$$l_{ii} = -\sum_{i\neq j} l_{ij}\,. \tag{37b}$$

In order to impose Dirichlet conditions, $\mathbf{L}$ must be modified on rows pertaining to boundary nodes. To impose the condition $v = 0$, we set $l_{ij} = 0$ in $\mathbf{L}_1$ when $S_{i(b)} > 0$. Similarly, to impose the condition $m = 0$, we set $l_{ij} = 0$ in $\mathbf{L}_2$ when $S_{i(b)} > 0$. If the boundary condition is (34), then $\mathbf{L}_1 = \mathbf{L}_2$.

Stability of the system (36) can be checked as follows. Recombining the system into one variable, we have

$$\mathbf{v}^+ = (2\mathbf{I} - \mu^2\mathbf{B})\mathbf{v} - \mathbf{v}^-\,, \tag{38}$$

where $\mathbf{v}^\pm := s_{t\pm}\mathbf{v}$, $\mathbf{B} = h^4\mathbf{L}_1\mathbf{L}_2$, and where $\mathbf{I}$ is the $N \times N$ identity matrix. Here, h represents the minimum h_{ij} with $j \in N_i$ ($i \neq j$) and $\mathbf{x}_i \in \overline{\mathbb{G}}$. Similarly to the stability analysis presented for the initial value problem, we have the following "matrix method" [20] type stability condition

$$\mu \leq \sqrt{4/\rho(\mathbf{B})}\,, \tag{39}$$

provided that $\mathbf{B}$ is positive semi-definite (PSD), and where $\rho(\mathbf{B})$ denotes the spectral radius of $\mathbf{B}$. That $\mathbf{B}$ is PSD follows from (37) and Gerschgorin's theorem [21]. It is assumed that the tiling is constructed such that $\rho(\mathbf{B}) \leq \Lambda_{\max}$, and thus $\mu = \mu_{\max}$ (as given previously) will be sufficient for stability. Energy methods [19] should be employed to get a more instructive stability condition for the finite-volume meshing pre-processing step, but these will be left for a future study.

6. SIMULATIONS

6.1. Modes of clamped circular plate

In order to validate these schemes, we simulate a clamped circular plate with tabulated values for the modal frequencies from [22]. The circular plate of interest has a radius of one metre and $\kappa = 20$. The time-step is set to $k = 1/F_s$ where $F_s = 8000$ Hz for the Cartesian scheme, and $F_s = 6300$ Hz for the hexagonal scheme in order to (approximately) normalise for the spatiotemporal density of points. For both schemes we employ a "staircase" approximation and a "fitted" approximation to the circular domain. These tilings are shown in Fig. 6.

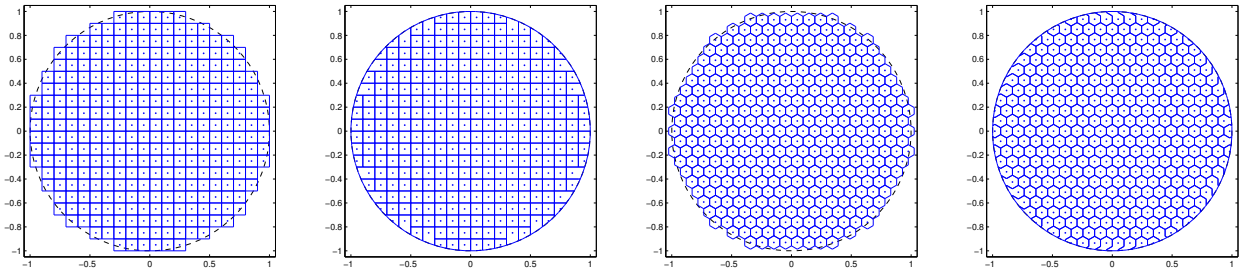
A normalised Kronecker delta (in space and time) is used as an excitation for the plate. The spectra of the resulting impulse responses, for low frequencies, are shown in Fig. 7. It can be observed that the fitted approximations are better than their staircase counterparts in both cases (Cartesian and hexagonal). However, as numerical dispersion is significant in both cases the modes are misrepresented above 250 Hz. We can also observe that in the Cartesian case, certain modal frequencies are in numerically split degenerate mode pairs (e.g. at 220 Hz and 270 Hz). This is a consequence of anisotropy, and it is clear that the hexagonal scheme offers an improvement in this respect.

6.2. Modal density and cutoff frequencies

Next we demonstrate the effects of the minimum directional cutoff frequencies, as discussed in Section 4. In Fig. 8, the same impulse responses are plotted, but now over the entire range of simulated temporal frequencies. The minimum cutoff frequencies are denoted with vertical dashed lines. These refer to $0.33F_s$ for the Cartesian scheme, and $0.70F_s$ for the hexagonal scheme.

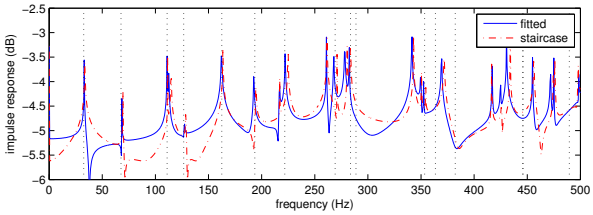
For the Cartesian case in Fig. 8a, we can see the sparsity of modes increases above the (minimum) cutoff frequency, leading up to the maximum cutoff frequency (the Nyquist). As seen in Fig. 8b, the hexagonal scheme (normalised for computational cost) has a higher cutoff frequency and one can notice that the density of modes near 2000 Hz is greater than in the Cartesian case. With a 8000 Hz sample rate, in Fig. 8c, the hexagonal scheme provides a richer spectrum, albeit at a higher computational cost. For these simulations μ was set to $\mu_{\max}$ and (39) was satisfied.

Finally, we show the hexagonal circular plate with double Dirichlet conditions (28) and the double Neumann conditions (34). The spectra that were obtained are shown in Fig. 9. It can be seen that these conditions provide spectra qualitatively similar to the clamped conditions.

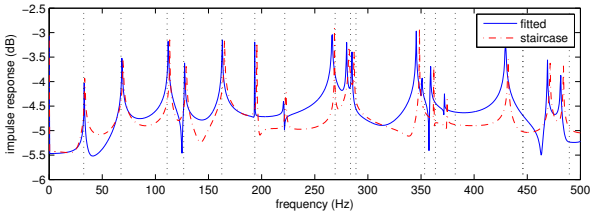


(a) Square, staircase (b) Square, fitted (c) Hexagonal, staircase (d) Hexagonal, fitted

Figure 6: Finite volume tilings representing circular plate of radius one. Square and hexagonal staircase and fitted tilings.

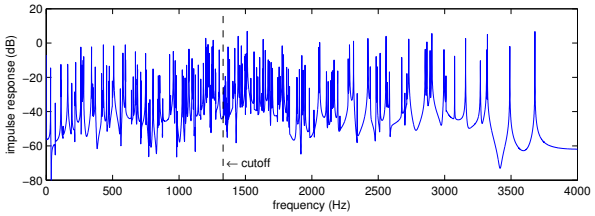


(a) Cartesian grid, $F_s = 8000$ Hz

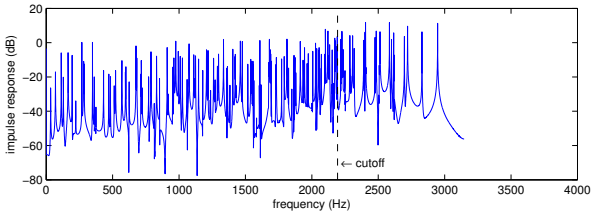


(b) Hexagonal grid, $F_s = 6300$ Hz

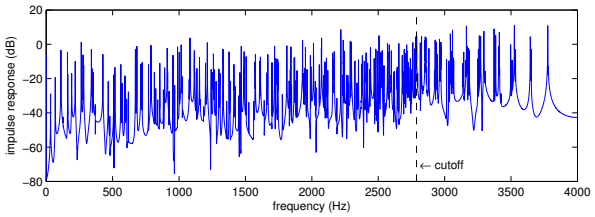
Figure 7: Impulse responses and analytical modes (dotted lines) of clamped circular plate



(a) Cartesian grid, $F_s = 8000$ Hz



(b) Hexagonal grid, $F_s = 6300$ Hz



(c) Hexagonal grid, $F_s = 8000$ Hz

Figure 8: Comparison of spectra, clamped circular plate

7. CONCLUSIONS

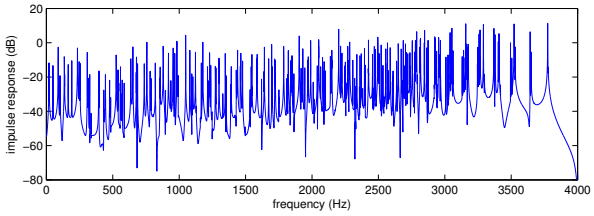
In this paper, we have presented two finite difference schemes for thin plate vibration using hexagonal grids. Stability conditions were presented and numerical dispersion was analysed. It was shown that better computational efficiency in terms of minimising numerical dispersion can be achieved using hexagonal grids rather than Cartesian (square) grids. Equivalent finite volume schemes were presented for the 13-point Cartesian and 19-point hexagonal finite difference schemes in order to implement clamped boundary conditions over irregular geometries. Simulations of clamped circular plates were presented and it was seen that finite volume grids that conformed to the domain were more accurate than "staircase" approximations. Furthermore, modal accuracy was generally better with the hexagonal scheme for a comparable computational cost with the Cartesian scheme.

One issue that will be addressed in future work is a more thorough analysis of boundary conditions. The system (25) is but a simplified version of a more complex system involving bending and twisting moments [13], which naturally leads to the correct boundary conditions in the simply supported and free case. This complete system is arguably more difficult to simulate with unstructured grids within a finite volume framework, and this constitutes a major challenge at the moment.

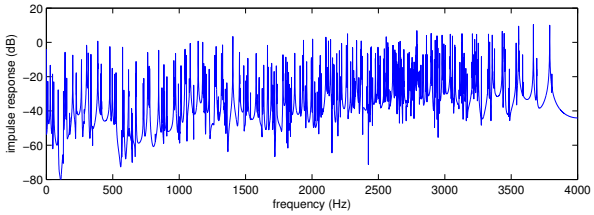
Another interesting direction for future study is the simulation of non-linear phenomena. Finite difference simulations of von Kármán equations have been performed in the past over Cartesian grids [23], but to our knowledge no similar study has been performed over different grids. Such simulations rely on a discrete version of the "triple self-adjointness" property of the non-linear operator [24], which will present new challenges over non-Cartesian grids.

Sound examples and animations from these schemes will be available at:

`http://www2.ph.ed.ac.uk/~s1164563/dafx14.`



(a) Hexagonal grid, double Dirichlet conditions, $F_s = 8000$ Hz



(b) Hexagonal grid, double Neumann conditions, $F_s = 8000$ Hz

Figure 9: Impulse response spectra from circular plate

8. REFERENCES

[1] M. Ducceschi, C. Touzé, and S. Bilbao, "Sound synthesis of gongs obtained from nonlinear thin plates vibrations: Comparison between a modal approach and a finite difference scheme." in *Proc. Stockholm Musical Acoustics Conf. (SMAC)*, Stockholm, Sweden, 2013.

[2] M. Ducceschi, C. Touzé, S. Bilbao, and C. J. Webb, "Nonlinear dynamics of rectangular plates: investigation of modal interaction in free and forced vibrations," *Acta Mechanica*, vol. 225, no. 1, pp. 213–232, 2014.

[3] C. Lambourg, A. Chaigne, and D. Matignon, "Time-domain simulation of damped impacted plates. II. numerical model and results," *J. Acoustical Society of America*, vol. 109, no. 4, pp. 1433–1447, 2001.

[4] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. Chichester, UK: Wiley, 2009.

[5] M. van Walstijn and K. Kowalczyk, "On the numerical solution of the 2D wave equation with compact FDTD schemes," in *Proc. Digital Audio Effects (DAFx)*, Espoo, Finland, 2008, pp. 205–212.

[6] B. Hamilton and S. Bilbao, "Hexagonal vs. rectilinear grids for explicit finite difference schemes for the two-dimensional wave equation," in *Proc. Int. Cong. Acoustics (ICA)*, Montréal, Canada, 2013.

[7] S. Bilbao, L. Savioja, and J. O. Smith III, "Parameterized finite difference schemes for plates: Stability, the reduction of directional dispersion and frequency warping," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1488–1495, 2007.

[8] M. Ducceschi, O. Cadot, C. Touzé, and S. Bilbao, "Dynamics of the wave turbulence spectrum in vibrating plates: A numerical investigation using a conservative finite difference scheme," *Physica D: Nonlinear Phenomena*, vol. 280, pp. 73–85, 2014.

[9] L. Collatz, *The numerical treatment of differential equations*, 3rd ed. Berlin: Springer, 1966.

[10] P. Voroshko, "Construction of finite difference diagrams of the engineering theory of elasticity, on the basis of integral representations of the resolvent functions," *Strength of Materials*, vol. 4, no. 8, pp. 943–947, 1972.

[11] J. Tuomela, "Fourth-order schemes for the wave equation, Maxwell equations, and linearized elastodynamic equations," *Numerical Methods for Partial Differential Equations*, vol. 10, no. 1, pp. 33–63, 1994.

[12] D. P. Playne and K. A. Hawick, "Stencil methods and graphical processing units for simulating field," in *Proc. 9th Int. Conf. on Foundations of Computer Science*, 2013.

[13] K. F. Graff, *Wave motion in elastic solids*. Courier Dover Publications, 1975.

[14] S. Timoshenko and S. Woinowsky-Krieger, *Theory of plates and shells*. New York, USA: McGraw-Hill Inc., 1959, vol. 2.

[15] A. H. Nayfeh and D. T. Mook, *Nonlinear oscillations*. New York: John Wiley and Sons, 1979.

[16] S. Bilbao and C. J. Webb, "Physical modeling of timpani drums in 3D on GPGPUs," *J. Audio Engineering Society*, vol. 61, no. 10, pp. 737–748, 2013.

[17] A. Torin and S. Bilbao, "A 3D multi-plate environment for sound synthesis," in *Proc. Digital Audio Effects (DAFx)*, Maynooth, Ireland, 2013.

[18] B. Heinrich, "Boundary value problems and irregular networks," in *Finite Difference Methods on Irregular Networks*. Springer, 1987, pp. 17–39.

[19] S. Bilbao, "Modeling of complex geometries and boundary conditions in finite difference/finite volume time domain room acoustics simulation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1524–1533, Jul. 2013.

[20] J. Strikwerda, *Finite difference schemes and partial differential equations*. Philadelphia, PA: SIAM, 2004.

[21] C. Meyer, *Matrix analysis and applied linear algebra*. Philadelphia, PA: SIAM, 2000, vol. 2.

[22] A. W. Leissa, *Vibration of Plates*. Washington, D.C.: NASA, 1969.

[23] S. Bilbao, "A family of conservative finite difference schemes for the dynamical von Karman plate equations," *Numerical Methods for Partial Differential Equations*, vol. 24, no. 1, pp. 193–216, 2008.

[24] O. Thomas and S. Bilbao, "Geometrically nonlinear flexural vibrations of plates: In-plane boundary conditions and some symmetry properties," *J. Sound and Vibration*, vol. 315, no. 3, pp. 569–590, 2008.

PRIORITIZED COMPUTATION FOR NUMERICAL SOUND PROPAGATION

John Drake
Center for Human Modeling and Simulation,
University of Pennsylvania
Philadelphia, PA, USA
drake@seas.upenn.edu

Maxim Likhachev
The Robotics Institute,
Carnegie Mellon University
Pittsburgh, PA, USA
maxim@cs.cmu.edu

Alla Safonova
Center for Human Modeling and Simulation,
University of Pennsylvania
Philadelphia, PA, USA
alla@seas.upenn.edu

ABSTRACT

The finite difference time domain (FDTD) method is commonly used as a numerically accurate way of propagating sound. However, it requires extensive computation. We present a simple method for accelerating FDTD. Specifically, we modify the FDTD update loop to prioritize computation where it is needed most in order to faithfully propagate waves through the simulated space. We estimate for each potential cell update its importance to the simulation output and only update the N most important cells, where N is dependent on the time available for computation. In this paper, we explain the algorithm and discuss how it can bring enhanced accuracy and dynamism to real-time audio propagation.

1. INTRODUCTION

Faithful propagation of sound through arbitrary environments is a computationally complex problem. Audio propagation solutions must be re-evaluated as the source position(s), listener position(s), and environment geometry change over time. If the recomputation can be done very quickly, the method might be useful in real-time applications like virtual environment audio simulations.

In this paper we present a method to accelerate the finite difference time domain numerical sound propagation method so that it might be used in real-time applications under broader configurations. We prioritize computation where it is needed most to most accurately propagate a wave, eliminating computation where it would have little effect on the output.

2. PREVIOUS WORK

Sound propagation can broadly be split into two groups: geometric methods and numerical methods. Geometric methods often take advantage of analytic solutions to wave equation problems directly in terms of the geometry of the environment and assume that sound waves travel in straight lines. Numerical methods discretize and solve wave equation problems with numerical analysis.

Geometric methods include such techniques as image methods [1], ray tracing [2], beam tracing [3], and acoustic energy transfer methods [4]. In the early image method presented in [1], virtual image sources are created from the true sound source to represent the acoustical contribution of sounds reflected from geometry in the environment. Similar to graphics research on geometric tracing techniques, ray [2] and beam [3] tracing methods have been developed for audio propagation. Ray tracing samples an environment with a multitude of rays reflecting from surfaces. Errors are introduced in ray tracing methods when samples miss important features in the environment [5]. Beam tracing methods improve on this by sampling continuous areas of the environment with each cast beam and splitting each cast beam where the environment is discontinuous, such as at the edge of a beam-intersecting wall. However, it is hard to accurately handle effects like wave diffraction in arbitrary environments using these methods.

Numerical methods include finite element [6] and finite difference [7] methods. The finite difference time domain method (FDTD) is an especially popular numerical method in acoustics, though originally developed for electricity and magnetism [7]. In the FDTD method, a finite simulation lattice is overlaid on the environment to discretize it in space, and the output is computed at discrete time steps over many iterations. The method naturally allows the modeling of arbitrary environment configurations and captures wave phenomena like diffraction. However, it suffers from high memory and computation time demands, especially as a simulation's time or space discretization is refined. We focus on FDTD in this paper and look at it in greater detail in Section 3.

Several hybrid methods have been developed, merging geometric and numerical methods [8], [9]. Hybrid methods are good for real-time applications because different wave properties can be efficiently represented by different methods. A recent example is the "Wave-Ray Coupling" presented by Yeh et al. 2013 [10], where a geometric technique handles long-distance wave propagation in a large environment and a numerical method captures important wave diffraction effects (which the geometric technique cannot handle) close to the listener position. Acceleration of computationally expensive numerical methods like FDTD is necessary for the numerical components of hybrid methods to function well in real-time applications. For example, the adaptive rectangular decomposition method [11] is used instead of plain FDTD in [10]. Any additional acceleration to the numerical component of a

This work was supported by NSF Grant IIS-1018486.

hybrid system can critically provide better real-time performance over a broader range of configurations, e.g. a larger numerical simulation zone capturing a more complete simulation of diffracted waves in the environment. This paper offers one such acceleration technique.

3. BACKGROUND

Sound is a wave phenomena dependent on the wave equation. Analytic solutions to the equation exist for simple configurations, but no complete analytic solutions exist for complex environments.

The FDTD method provides a way to solve the equation in complex environments in a discrete way. The environment is discretized in space into a regular grid. The sizes of the grid cells in each dimension do not have to match but for simplicity they will be equal here, represented by h. $p^n(i,j,k)$ represents the pressure at location i,j,k at time n. Throughout, we use the speed of sound $c \approx 340$m/s.

Following is a brief derivation of an FDTD update equation, based on the description in [11].

$$\frac{\partial^2 p}{\partial t^2} - c^2 \nabla^2 p = F(x,t) \tag{1}$$

$F(x,t)$ is a forcing term representing sound inputs. It is zero without inputs.

∇^2 represents the 3D Laplacian operator, $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$, the divergence of the gradient at a point in the pressure field.

The Laplacian operator can be discretized for a grid representation in many ways[12]. We use the L_2 discretization for speed and simplicity:

$$\nabla^2 p^n(i,j,k) \approx$$

$$lap(i,j,k) = \frac{1}{h^2}\begin{pmatrix} -6p^n(i,j,k) + \\ p^n(i-1,j,k) + p^n(i+1,j,k) + \\ p^n(i,j-1,k) + p^n(i,j+1,k) + \\ p^n(i,j,k-1) + p^n(i,j,k+1) \end{pmatrix} \tag{2}$$

Or more succinctly with K as a Discrete Laplacian Matrix and P a long vector of all pressure values in the grid:

$$\nabla^2 p \approx \frac{1}{h^2} KP \tag{3}$$

yielding

$$\frac{\partial^2 P}{\partial t^2} - \frac{c^2}{h^2} KP = F(t) \tag{4}$$

Discretizing in time with time step $\triangle t$ and using the leapfrog integrator yields the following update equation.

$$P^{n+1} = 2P^n - P^{n-1} + \left(\frac{c\triangle t}{h}\right)^2 KP^n + \triangle t^2 F^n(t) \tag{5}$$

Time step size $\triangle t$ depends, for the sake of numerical stability in the simulation, on the grid resolution according to Courant-Friedrichs-Lewy condition $\triangle t < \frac{h}{c\sqrt{3}}$

To model the interface of air and environment surfaces, any of many absorbing boundary conditions (ABCs) can be used. Please see [13], based on the original Perfectly Matched Layer (PML) work [14] for a very helpful derivation of the PML ABC for the single field parameter "scalar" FDTD context presented above. The works [15],[16] present the formulation of a simple surface ABC which also may be used in this context.

4. PRIORITIZED FDTD

Full FDTD simulation demands the evaluation of a large number of computations. There are many time steps needed, and in each one, potentially every grid cell representing the simulation space needs to be updated. Each FDTD time step depends on the previous time step, but within each time step, every cell update computation is independent. Also, if one cell is updated, the effect of that update is only relevant to its neighboring cells whose discretized Laplacian estimations in the subsequent time step include a term reading the value of that previously-updated cell. Areas of uniform pressure remain static until disturbed by impinging waves and also become static again after those waves pass by. These properties together allow us to accelerate FDTD by prioritizing computation where it is needed most and omitting it elsewhere.

We incorporate these properties into one tunable system by the introduction of a cell update importance function and prioritized selection of which updates to execute. We concentrate on accelerating FDTD impulse response simulation of low frequency diffracting waves. Higher frequency reflecting components can be simulated in real time with other methods, together forming a hybrid system as in [10]. Our method of acceleration is orthogonal to other approaches like parallelization, so we believe it can be applied on top of other methods for further improved results.

4.1. FDTD Setup

We initialize our FDTD simulation grid with a grid cell size, h appropriate for low frequency waves (e.g. simulation frequency= 1KHz, $h \leq \frac{c}{2KHz}$ guided by the Nyquist-Shannon sampling theorem). These low frequency waves have a greater tendency to diffract in a significant way in human-scale environments than do higher frequency waves.

We focus on the task of recording an impulse response, rather than continuously propagating an arbitrary source wave. Simulated impulse responses can be efficiently convolved with arbitrary source waves after simulation to auralize output. The impulse response context allows us accelerate computation more than we could if the source emitted an arbitrary wave. This is true because with an impulse, more of the pressure field is likely to be static and unimportant (in front of or behind the impulse wave) at any arbitrary time step than if the field were inundated with a continuous series of waves.

To record an impulse response, we use an input pulse defined by the following Gaussian function. Its parameters were chosen to make the pulse peak near the origin and to be short but not so short that it causes major ringing oscillations when played into the 1KHz grid simulation.

$$\begin{aligned} pulseCenterTime &= 0.0025 \\ spread &= 0.001 \\ f(t) &= e^{-\frac{(t-pulseCenterTime)^2}{2*spread^2}} \end{aligned} \tag{6}$$

Note that any impulse response recorded from $t = 0$ using this impulse must be shifted in time by $-pulseCenterTime$.

PML ABC zones are placed around the boundaries of the simulation space. It is suggested in [17] to make the PML thickness at least 65-70% of the longest wavelength of interest. Because we primarily focused on the recording of impulse responses and to aid computation times we use $n_a = 10$, which corresponds to a "longest wavelength of interest" of $0.65c/h/10 = 130$Hz.

The profile of our input pulse approximately corresponds to a sinusoidal wave section with a frequency around 200Hz, so the PML thickness is reasonable.

4.2. FDTD Computation

Described below is our method to accelerate FDTD computation in the context described in Section 4.1.

4.2.1. Importance Function

An importance function, notated as $importance(i, j, k)$, estimates the importance of a cell update during simulation. Before the first FDTD update pass, the $importance$ function is initialized for every cell to zero. At the moment of the start of the input pulse, the sound source location and its immediate neighbors are given artificially elevated $importance$ values to seed their evaluation when they are first important.

In the last moments of a simulation, many regions of the environment are so far from the listener position(s) that no wave leaving those regions and traveling at the speed of sound could reach the listener(s) before the end of the simulation. We include this in all importance functions by forcing importance to zero when a cell update can be omitted. Let L represent the listener position in grid units. Let dur be the duration of the simulation. $Omit$ determines when a cell update would be made to a cell too far from the receiver position to possibly have any effect on the output of the simulation.

$$Omit(i, j, k, t) = \frac{h \cdot |[i, j, k] - L|}{c} > (dur - t) \qquad (7)$$

Idealy, an importance function would look into the future and determine how much of an effect an update will have on the eventual output of the simulation. Since this is not possible to do in any less time than it would take to run the simulation to that future time or even in less time than it takes to do a single update, we approximate the importance function by estimating the effect an update could have on the immediate region surrounding it in the following time step. In FDTD simulation, the discretized Laplacian approximation (lap) is the only term in the update equation (Equation 5, inside K) which interacts with neighboring cells, so our importance functions incorporate the same values which lap uses. Importance functions we used take these forms:

$importance_1$ (m_1) is something like the Laplacian approximation, but the absolute value of each term is used and all coefficients are one. It is always non-negative.

$$m_1(i, j, k) = \begin{pmatrix} |p^n(i, j, k)| + \\ |p^n(i-1, j, k)| + |p^n(i+1, j, k)| + \\ |p^n(i, j-1, k)| + |p^n(i, j+1, k)| + \\ |p^n(i, j, k-1)| + |p^n(i, j, k+1)| \end{pmatrix}$$

$$importance_1(i, j, k) = \begin{Bmatrix} 0 & \text{if } Omit(i, j, k, t) \\ m_1(i, j, k) & \text{otherwise} \end{Bmatrix} \qquad (8)$$

$importance_2$ (m_2) is something like a gradient magnitude. m_2 yields the largest magnitude of a difference between any two nearby pressure values ($Nearby(i, j, k)$ denotes the set of nearby pressure values and includes the value of the cell itself). It is always non-negative.

$$Nearby(i, j, k) = \begin{Bmatrix} p^n(i, j, k)), \\ p^n(i-1, j, k)), p^n(i+1, j, k)), \\ p^n(i, j-1, k)), p^n(i, j+1, k)), \\ p^n(i, j, k-1)), p^n(i, j, k+1)) \end{Bmatrix}$$

$$m_2(i, j, k) = \max_{\substack{p_1 \in Nearby(i,j,k) \\ p_2 \in Nearby(i,j,k)}} (p_1 - p_2)$$

$$importance_2(i, j, k) = \begin{Bmatrix} 0 & \text{if } Omit(i, j, k, t) \\ m_2(i, j, k) & \text{otherwise} \end{Bmatrix} \qquad (9)$$

4.2.2. Most Important Cell Retrieval

In every FDTD update pass, a limited number of cells with the highest $importance$ values are recomputed. To efficently ascertain which cells have the highest $importance$, we keep a list of update candidate cells which we call $candidates$. An average case $O(n)$ time partial sorting algorithm is used to partially sort $candidates$ once per time step according to the importance function. It ensures that the first N cells in the list have greater importance than all others. Provided their importances are non-zero, these cells are updated in the usual FDTD manner to finish evaluation of a time step.

The number of cells to recompute, N, can be estimated according to the approximate volume of the pulse wavefront, defined here by the inner and outer radii, r_i and r_o, of an impulse wave in an open environment. t is the simulation time.

$$\begin{aligned} r_o &= c \cdot t \\ r_i &= c \cdot (t - 2 \cdot pulseCenterTime) \\ ApproxWaveVol(t) &\approx \frac{4}{3}\pi {r_o}^3 - \frac{4}{3}\pi {r_i}^3 \end{aligned} \qquad (10)$$

$$N(t) = ApproxWaveVol(t) \div h^3 \qquad (11)$$

4.2.3. Refreshing the Importance Function Efficiently

To avoid unnecessarily evaluating the $importance$ function for every cell at every time step, we keep a set of cell references which we call $refreshSet$. Once per time step, the $importance$ for every cell stored in $refreshSet$ is evaluated and all other cells are ignored. After this, $refreshSet$ is cleared. In a single time step of FDTD, the $importance$ values of *only* updated cells and their immediate neighbors can change. So, whenever a cell is updated in one time step, the cell and its neighbors are added to $refreshSet$ for $importance$ re-evaluation before updates at the next time step.

4.2.4. Pseudocode

Some specific implementation details have been simplified, such as code to avoid duplicate additions to $refreshSet$ and the special case for PML zones. See Algorithm 1 and UpdateSimState.

5. DISCUSSION

5.1. Analysis

Figure 1 shows how N changes during a simulation. The unshaded portion shows the amount of computation avoided by our method.

begin
  initialize;
  for $t = 0$...*dur at increments of* $\triangle t$ **do**
    UpdateSimState;
    Force pressure at source cells;
    Record IR output at listener cells;
    Prepare for next time step;

Algorithm 1: FDTD Outer Loop

begin
  initialize;
  $candidates$ =List of all sim cells;
  for $element \in refreshSet$ **do**
    Refresh $importance$ of $element$.
  $refreshSet = \emptyset$;
  N = compute as in Eq. 11;
  Partially sort $candidates$ as in 4.2.2;
  for $i = 1...N$ **do**
    updateCell($candidates[i]$);
    $refreshSet+ = candidates[i]$;
    $refreshSet+ =$ All neighbors of $candidates[i]$;

Procedure UpdateSimState

The dashed line at the top marks the total number of cells in the FDTD grid. At the beginning of a simulation, computation is limited by N, and at the end of a simulation, it is limited by the $Omit$ function. The superimposed curves show the numbers of updates done on actual runs of our algorithm in our test environments. The configuration of the environment affects how many updates are done by affecting the number of zero-importance cells at different times. Zero-importance cells are not updated even if N is larger than the number of nonzero-importance cells). Figure 2 is similar to Figure 1, but the simulation duration is five times longer.

The first profile, Figure 1, demonstrates the context where our method is most useful: impulse response simulations of limited duration, such as the numerical simulation component in a hybrid system like Yeh et al.'s Wave-ray Coupling [10]. However, even in less ideal situations like Figure 2, our method always saves some computation at the beginning and end of the simulation and provides a pricipled approach to restricting computation in the middle of the simulation too, by limiting N.

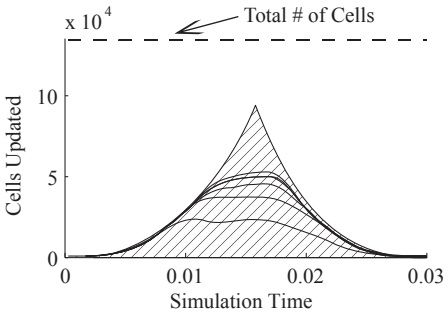


Figure 1: *With simulation parameters from our trials in Section 6, the shaded region is computed, the rest omitted.*

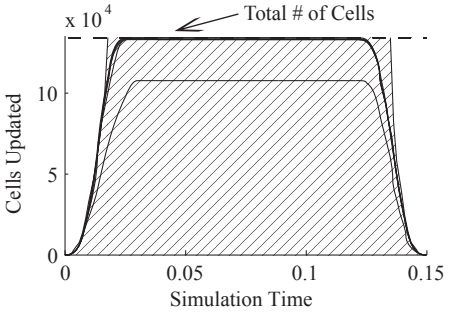


Figure 2: *Longer duration simulation worst-case behavior.*

If the simulation computes output for a continuous input instead of an impulse response or if multiple source positions are present, our method can still be used. $ApproxWaveVol$ would have to be changed, which would change the shape of the early portions of Figures 1 and 2 (for lesser performance), but the end portions of the simulation would remain the same, since the same distance-based cell omission can be done. Conversely, adding additional listener positions affects the later portions of simulation while not affecting the beginning.

5.2. N Approximation

In an environment with many absorbing surfaces (or areas open to the simulation boundary PML regions), a tighter bound on the wavefront volume can be made by observing that if there were no obstacles in the environment, an expanding spherical wave would eventually begin to leave the simulation grid. The portions of the wave which have left the space can be subtracted from the volume as calculated in 4.2.2.

Our presented approximation for N, estimating the wave volume in an empty environment, is not always quite enough to capture important updates at the front of a wave because the importance function is only an estimation of true update importance. In the "worst case" of a completely open environment while the propagated wave forms a spherical shell, we find it helps to inflate the very tight wavefront volume figure by up to 20% to ensure that important cells on the leading edge of the wavefront are not missed in simulation. In our experiment trials, we did not have to inflate the N value, because absorbers in our environments and the edges of the simulation space reduced the actual simulated wave volume below the estimated wave volume before significant deterioration took place. If, as discussed in the previous paragraph, a tighter volume bound were used, some inflation might indeed be needed in all cases.

6. RESULTS

We computed average performance statistics on an Intel i7-860 machine. The code was compiled and linked from C++ source with the MSVC 2008 compiler. We ran the simulation thirty six times for our method and thirty six times for plain FDTD. The trials were divided evenly between six environments which together stress our algorithm and represent plausible hybrid simulation scenarios: four open artificial environments with various wall configurations between source and listener, one completely open

	Time Prioritized	Time FDTD	Speedup in Time
Open	1.26s	4.29s	3.41
Env. A	1.24s	4.28s	3.44
Env. B	1.04s	4.31s	4.13
Env. C	1.17s	4.27s	3.66
Env. D	1.27s	4.29s	3.38
Building	0.71s	3.59s	5.07
Column Averages	1.11s	4.17s	3.85

Table 1: *Performance results in terms of time.*

	Updates Prioritized	Updates FDTD	Speedup in Cell Updates
Open	15.0%	100%	6.66
Env. A	14.5%	99.1%	6.82
Env. B	12.3%	99.1%	8.05
Env. C	13.7%	99.4%	7.23
Env. D	15.5%	99.5%	6.86
Building	8.2%	80.8%	9.86
Column Averages	13.1%	96.3%	7.58

Table 2: *Performance results in terms of cell updates. "Updates FDTD" is not always 100% because updates are not made within environment obstacles.*

environment with no walls, and one generated from interior geometry of a real building. The environment dimensions were all fixed at 7m $\times$ 7m $\times$ 2.5m and the simulated duration (0.03s, long enough to receive all impulse wave diffractions in our trials) was also equal across all trials. These results are shown in Tables 1, 2, and 3. "Time" columns show average computation times, "Updates" columns show average percentages of updates performed out of the maximum possible, and "Speedup" columns show the relative performance of our method over plain FDTD. In all trials, FDTD updates were not made for cells within solid objects. The real building environment had the largest number of occluded cells (around 20%).

As seen in Table 1, our approach improves average simulation speeds by a factor of 3.85. When the real building environment trial is considered alone, the improvement was over a factor of 5. Memory usage with our method was around 70% greater than plain FDTD, to store the $candidates$ list, $refreshSet$, and other data to do things like avoid duplicate neighbor additions to $refreshSet$ efficiently.

Figure 3 shows response waveform comparisons of our method to full FDTD simulation for three environments, at three different inflation factors for N and one deflation factor for N. Each plot has a response from our method overlaid with the full FDTD response. The first two environments (Env. C and Env. D respectively in Table 1) are artificial and mostly open, so they exhibit a mild case of the problem explained in Section 5.2, where N is close to the actual wave volume and the importance function does not perfectly indicate which cells must be updated. Mild N inflation helps those results converge. The third environment is the real building environment and has many reflecting surfaces which the other tested environments do not have. Relative computation times as N inflation factors change are given in the caption of Figure 3.

	0.8 N Factor	1.0 N Factor	1.2 N Factor	2.0 N Factor
Open	42967.8	3014.0	271.3	69.7
Env. A	6148.4	517.6	138.7	49.2
Env. B	21.9	3.7	2.9	2.8
Env. C	76.5	5.9	1.9	1.3
Env. D	4343.0	241.8	25.3	12.7
Building	91.4	39.4	46.0	36.6

Table 3: *Sum of squared error between waveform outputs of the full FDTD and the prioritized method. Compare with Figure 3*

7. CONCLUSIONS & FUTURE WORK

Our prioritized computation method accelerates FDTD wave propagation. It is especially helpful in the context of a hybrid simulation where a method like FDTD captures the effect of an impulse wave diffracting in an environment. Our acceleration allows such an impulse response simulation to be repeated more rapidly, with better discretization, or larger environment size, to improve real time results.

Our implementation was not parallelized, but it also does not prevent the use of parallelization. In fact, preliminary results show that running our method on a single CPU thread is faster than a four-way parallel FDTD implementation we tested on four CPU threads, at least under the trial configurations tested in Section 6. It is future work for us to parallelize prioritized FDTD computation.

We kept the complete set of simulated cells in the $candidates$ list, but this list could instead be grown, starting with just the sound source cell, by appending the contents of $refreshSet$ at each time step. Old irrelevant cells could likewise be removed over time. If $candidates$ were made shorter like this or if special consideration were given to the high temporal coherence of $candidates$, the running time of the partial sorting algorithm could be reduced.

The partial sorting of $candidates$ creates an overhead over plain FDTD in the middle of long simulations (like ones with reflections which must be simulated), when N approaches the full volume of the space. In these situations, because we do not change the pressure field representation, computation can easily be switched between our method and plain FDTD to avoid the overhead. Computation savings would still be found at the beginning and end of the simulation time. Alternatively, an upper bound can be placed on the size of N to guarantee that performance is always better than ordinary FDTD while still maintaining simulation quality in a pricipled way. While potentially deleterious in the "worst case," an upper bound like this could ordinarily be used because the wave volume in a simulation typically does not approach the total environment volume.

Finally, future work also includes testing the effectiveness of prioritized computation in numerical techniques other than FDTD.

8. REFERENCES

[1] B. M. Gibbs and D. K. Jones, "A simple image method for calculating the distribution of sound pressure levels within an enclosure," *Acustica*, vol. 26, pp. 24–32, 1972.

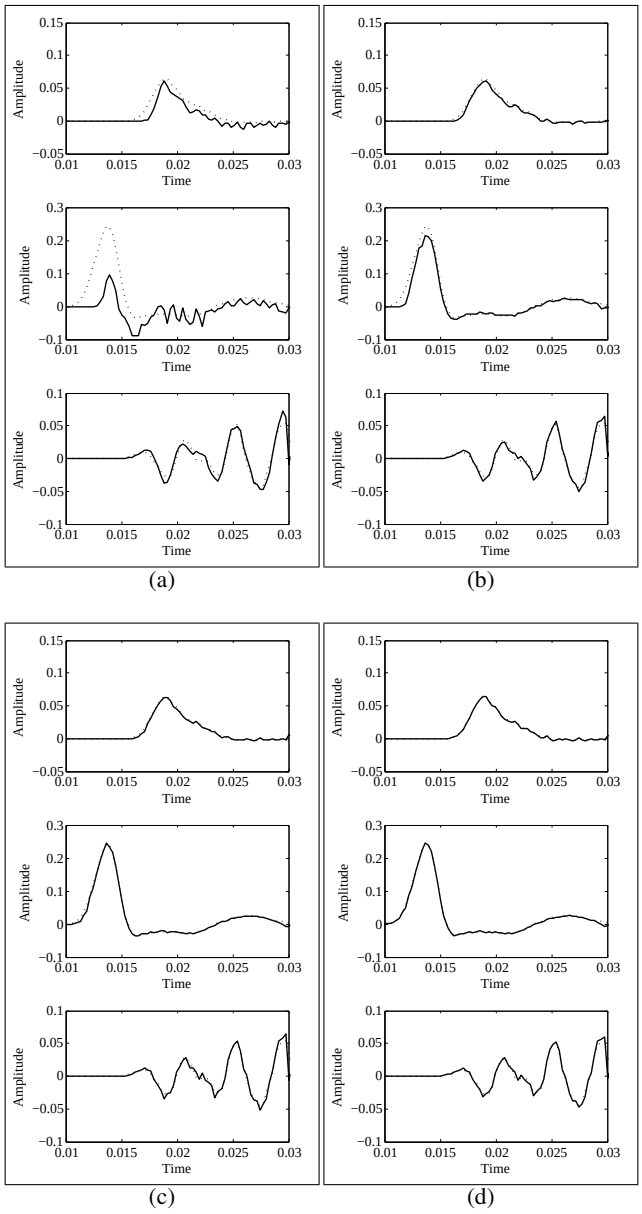


Figure 3: *The broken lines are the full FDTD ground truth output and the solid lines are the prioritized computation output. The three waveforms correspond to Env. C, Env. D, and Building, respectively, in Tables 1, 2 and 3.*
(a): 20% N deflation (10% shorter computation time than (b), but quality suffers in some environments)
(b): No N inflation (quality suffers a little in worst-case open environments – see Section5.2)
(c): 20% N inflation (5% longer computation time than (b))
(d): 100% N inflation (12% longer computation time than (b))

[2] A. Krokstad, S. Strøm, and S. Sørsdal, "Calculating the acoustical room response by the use of a ray tracing technique," *J. Sound Vib.*, vol. 8, no. 1, pp. 118–125, 1968.

[3] Norm Dadoun, David G. Kirkpatrick, and John P. Walsh, "The geometry of beam tracing," *Proc. of the first annual symposium on Computational geometry*, pp. 55–61, 1985.

[4] Nicolas Tsingos, *Simulating High Quality Virtual Sound Fields for Interactive Graphics Applications*, Ph.D. thesis, Universite J. Fourier, Grenoble I, December 1998.

[5] Hilmar Lehnert, "Systematic errors of the ray-tracing algorithm," *Applied Acoustics*, vol. 38, pp. 207–221, 1993.

[6] Emmanuel Granier, Mendel Kleiner, Bengt-Inge Dalenbäck, and Peter Svensson, "Experimental auralization of car audio installations," *J. Audio Eng. Soc*, vol. 44, no. 10, pp. 835–849, 1996.

[7] Kane S. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. AP-14, no. 3, pp. 301–307, May 1966.

[8] Ying Wang, Safieddin Safavi-Naeini, and Sujeet K. Chaudhuri, "A hybrid technique based on combining ray tracing and fdtd methods for site-specific modeling of indoor radio wave propagation," *IEEE Transactions on Antennas and Propagation*, vol. 48, pp. 743–754, 2000.

[9] S. Hampel, S. Langer, and A. P. Cisilino, "Coupling boundary elements to a raytracing procedure," *International Journal for Numerical Methods in Engineering*, vol. 73, pp. 427–445, 2008.

[10] Hengchin Yeh, Ravish Mehra, Zhimin Ren, Lakulish Antani, Ming C. Lin, and Dinesh Manocha, "Wave-ray coupling for interactive sound propagation in large complex scenes," *Proc. of ACM SIGGRAPH Asia (TOG)*, 2013.

[11] Nikunj Raghuvanshi, Rahul Narain, and Ming C. Lin, "Efficient and accurate sound propagation using adaptive rectangular decomposition," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 5, 2009.

[12] Steve Schaffer, "Higher order multi-grid methods," *Mathematics of Computation*, vol. 43, no. 167, 1984.

[13] D. Zhou, W. P. Huang, C. L. Xu, D. G. Fang, and B. Chen, "The perfectly matched layer boundary condition for scalar finite-difference time-domain method," *IEEE Photonics Technology Letters*, vol. 13, no. 5, pp. 454–456, May 2001.

[14] Jean-Pierre Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of Computational Physics*, vol. 114, pp. 185–200, 1994.

[15] Tapio Lokki, Alex Southern, and Lauri Savioja, "Studies on seat dip effect with 3d fdtd modeling," *Proc. of Forum Acusticum*, 2011.

[16] Konrad Kowalczyk and Maarten van Walstijn, "Room acoustics simulation using 3-d compact explicit fdtd schemes," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, January 2011.

[17] Yotka S. Rickard, Natalia K. Geogieva, and Wei-Ping Huang, "Application and optimization of pml abc for the 3-d wave equation in the time domain," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 2, pp. 286–295, February 2003.

SINUSOIDAL SYNTHESIS USING A FORCE-BASED ALGORITHM

Ryoho Kobayashi
Faculty of Environment and Information Studies,
Keio University SFC
Kanagawa, Japan
ryoho@sfc.keio.ac.jp

ABSTRACT

In this paper we propose a synthesis method using a force-based algorithm to control frequencies of multiple sine waves. In order to implement this synthesis method, we analyze an existing sound source using a fast Fourier transform (FFT). Spectral peaks which have large magnitudes are regarded as heavy partials and assigned large attractive forces. A few hundred sine waves with stationary amplitudes are placed in a frequency space on which forces generated in the analysis phase are applied. The frequencies of the partials gravitate to the nearest peak of the reference spectrum from the source sound. As more sine waves are combined at the large peaks, the sound synthesized by the partials gradually transforms into the reference spectrum. In order to prevent the frequencies of the partials from gravitating onto localized peaks, each partial is assigned a repulsive force against all others. Through successful control of these attractive and repulsive forces, roughness and speed variation of the synthesis can be achieved. Moreover, by increasing or decreasing the number of partials according to the total amplitude of the source sound, amplitude envelope following is achieved.

1. INTRODUCTION

A force-based (or force-directed) algorithm is commonly known as a graph drawing algorithm [1]. A graph is a common data structure which is constructed from a set of vertices and edges, where the edges connect pairs of vertices [2]. The synthesis method proposed in this paper is inspired by this algorithm, and utilizes the algorithm in order to generate sound.

The motivation of this research is to accomplish a new synthesis method as an application of the sinusoidal partial editing technique [3, 4]. The basic premise of the synthesis method is placing multiple sinusoidal waves which have separate frequencies, and applying a one dimensional force-based algorithm in a frequency domain to control the frequencies of the waves. This method is different from general spectral editors such as spectral SPEAR [5] in that it is not developed for flexible sound editing, but rather for generating characteristic time-varying sounds between noises and recorded sound materials.

The goal of this research is to develop a synthesis method which can generate various sounds from musical tones and noises with a small number of intuitive parameters. In order to achieve this goal, we prepare an existing sound to generate attractive forces and apply them to the force-based algorithm. Strong attractive forces are assigned to large peaks in the spectrum by analyzing the reference sound source using the Fourier transform. A user can vary the similarity of the sound to the reference sound by controlling the forces applied.

All programs presented in this paper are written in Objective-C and C++ and are executed in Mac OSX.

2. STRUCTURES

The structures used to achieve the synthesis method are described in this section.

2.1. Analysis of a reference sound source

The first step is the analysis of a reference sound selected by a user. The reference sound is usually provided as a sound file except for real-time processing, which is presented in section 3.3. Any valid sound source is allowed.

The Short-Time Fourier Transform (STFT) analysis [6] is used for this step, where amplitudes A for each frequency f are detected by

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-2\pi i k n / N} \tag{1}$$

$$A(k) = |X(k)| \tag{2}$$

$$F(k) = \frac{kR}{N} \tag{3}$$

where $x(n)$ consists of N samples of a windowed waveform and R represents the sampling rate.

2.2. Distribution of partials

The synthesis phase for the proposed method begins by generating partials in a specific range of frequencies. The amplitudes of the partials are calculated from the number of partials and are unchangeable. There are two options for determining the number of partials.

1. The user specifies the number of partials and it does not change. The synthesized result does not follow the amplitude of the input reference source.
2. The user specifies the maximum number of partials and the number of active partials is determined in proportion to the amplitude of the reference sound as (4).

$$\nu = \nu_{max} \sum_{k=0}^{N-1} \alpha A(k) \tag{4}$$

where α represents a constant number for scaling the amplitude. The active or inactive partials are randomly chosen. In this step, since the frequencies of the partials are random, an unpitched sound is typically created.

2.3. Attractive force

Attractive forces, which are applied to the partials, are generated from the spectrum detected from the reference sound. A partial is attracted to neighboring frequency components, where the user can specify the number of effective frequency components. The force is inversely proportional to the square of the distance between the target frequency component and the frequency of the partial

$$f_a(P_f(i)) = \sum_{0<|F(k)-P_f(i)|<\tau} \frac{\mathrm{sgn}(F(k)-P_f(i))g_a A(k)}{|F(k)-P_f(i)|^2} \quad (5)$$

where $f_a(P_f(i))$ represents an attractive force for partial $P(i)$ of which the frequency is $P_f(i)$, g_a is a constant value to adjust the strength of the force, and τ corresponds to the range of the effective frequency components.

Figure 1 depicts an example of attractive forces which are applied to a partial. Three peaks of the spectrum are used for calculation in this figure. A large and close peak such as A has a profound effect while a distant peak such as C has little effect. As a result, the effect of peak A is significant, thus this partial shifts to the left side (lower in the frequency domain).

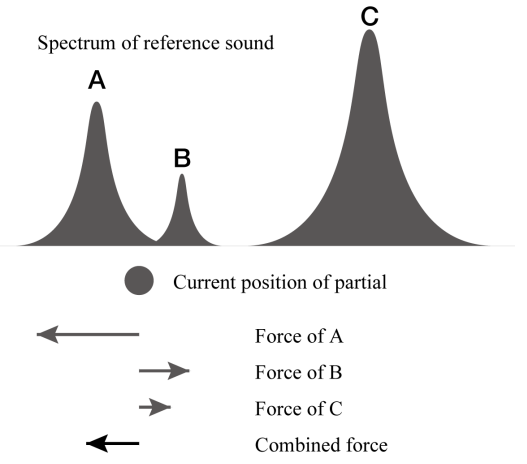


Figure 1: *Attractive forces applied to a partial*

2.4. Repulsive force

To avoid congestion of partials at a small peak in the spectrum, repulsive forces are generated between every pair of partials. The force is inversely proportional to the square of the distance between the partials

$$f_r(P(i)) = \sum_{P_f(j)\neq P_f(i)} \frac{\mathrm{sgn}(P_f(i)-P_f(j))g_r}{|P_f(i)-P_f(j)|^2} \quad (6)$$

where $f_r(P(i))$ represents a repulsive force for partial $P(i)$. By using all pairs of partials for the calculation, partials depart from condensations.

Figure 2 represents repulsive forces between each pair of three partials. Since the partials repel close partials more strongly, partial B is moved up.

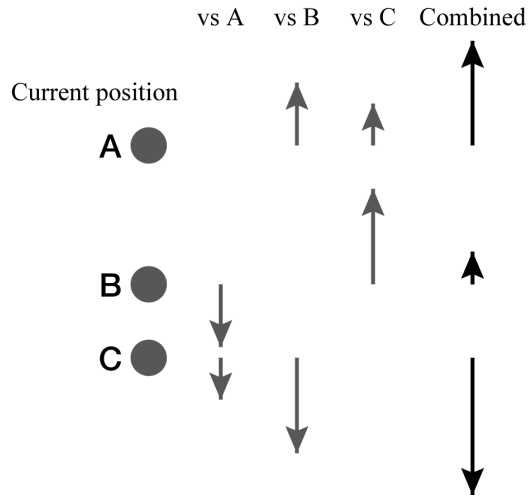


Figure 2: *Repulsive forces applied to three partials*

When some partials have the same frequencies, the repulsive forces between these partials are unable to activate and the attractive forces are always congruent. To separate these partials, two options which user can select are prepared.

1. Random repulsive forces are applied to each partial which is placed at the same frequency with other partials.
2. New frequencies, which is unrelated to current status, are redistributed to the partials.

When some partials have the same frequencies, the repulsive forces between these partials are not activated and the attractive forces are always congruent. In order to separate these partials, there are two options for the user:

1. Random repulsive forces are applied to each partial which is placed at the same frequency with other partials.
2. New frequencies, which are unrelated to the current status, are redistributed to the partials.

Since it is infrequent that multiple partials have exactly the same frequency, the difference in the final synthesized sound qualities is minimal between the aforementioned options.

2.5. Resistance

When the reference sound has a static frequency component, the partials have the risk of periodic vibration around a spectral peak. This is because the attractive forces convert back and forth between potential and kinetic energy. Therefore, the oscillations are inhibited by implementing resistance.

$$f(P(i)) = r(f_a(P(i) + f_r(P(i)) \quad (7)$$

produces total force $f(P(i))$ for partial $P(i)$. r is a resistance value between 0 and 1.

2.6. Synthesis

The forces, which are derived in section 2.5, are applied to partials at every frame by addition of the forces

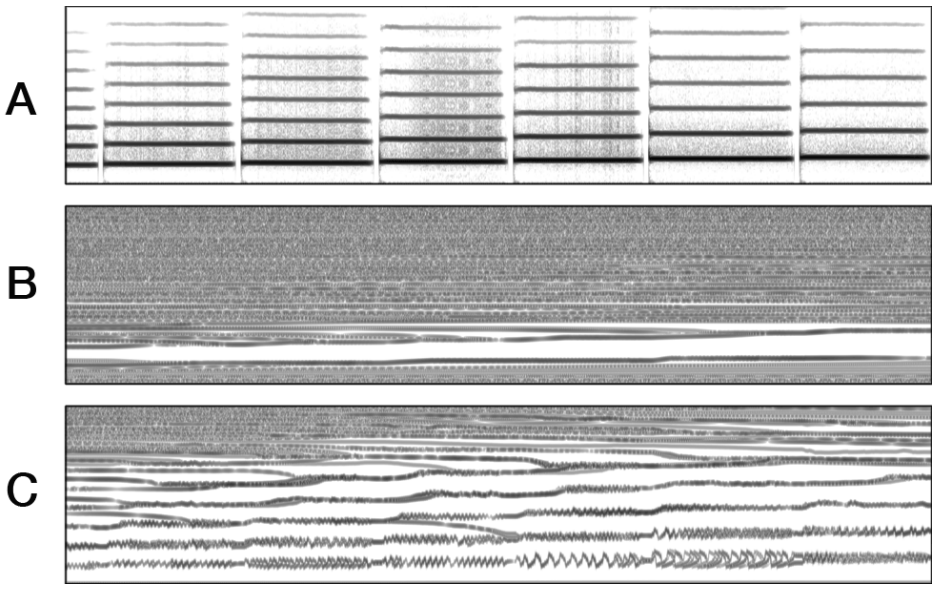


Figure 3: *An example result of the synthesis (the horizontal axis is time and the vertical axis is frequency). "A" is the reference sound (Horn; from 0Hz to 2,000Hz), while "B" and "C" are the synthesized sounds which consist of 1000 partials. The attractive force (ga) for "B" is 0.3, and gr for "C" is 5.*

$$P_f(n,i) = P_f(n-1,i) + f(P(n-1,i)) \tag{8}$$

where n represents the current time frame.

The sound synthesis is accomplished using a common oscillator bank synthesis technique [7, 8] which is realized by

$$y(n) = \sum_{\forall i} A\cos[2\pi P_f(n,i)n + \phi_i] \tag{9}$$

where A represents a constant amplitude for each partial, and an initial phase ϕ_i is randomly distributed.

An example of the synthesis result is depicted in Figure 3. A is generated from a horn sound; B and C are the synthesis results which consist of 1000 partials. The beginning of the synthesized sound starts with an unpitched noise at the left in the figure and the pitched tone is constructed gradually.

It is also possible to use the inverse Fourier transform for both simplified and/or real-time calculation which is proposed in the section 3.3.

3. APPLICATIONS

In this section, we present some applications for this synthesis method.

3.1. Dynamic control of parameters

This synthesis method can generate various sounds by adjusting the parameters. In particular, the coefficients for attractive force g_a, repulsive force g_r, and resistance r are important for controlling the similarity to the reference sound and the quickness of transitions.

It is possible for users to control these parameters by preparing time-varying functions. These functions are written in files which the synthesis program reads in order to synthesize the sound.

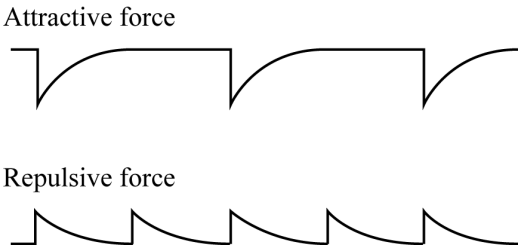


Figure 4: *An example of the force functions used to generate rhythmic sound.*

For instance, periodic changes in timbre are generated by using the functions depicted in Figure 4, where a rhythm structure is created. Control of the resistance value is effective to generate and remove vibrations. Moreover, various effects are realized by applying these forces to the amplitudes of the reference sound without preparing function files.

3.2. Timbre morphing

Timbre morphing is a technique used to create a new sound by combining multiple sounds and transforming the timbres gradually [9]. Sinusoidal modeling, which is used for the synthesis method in this paper, is utilized in some cases of timbre morphing [10, 11]. This synthesis method does not analyze the multidirectional features of the reference sound and it is therefore difficult to generate a high-quality and well-defined result. However, uncomplicated timbre morphing is accomplished by using this synthesis method.

A sound file which consists of several segments from separate sound sources is required to accomplish timbre morphing, using

the file as a reference. Each particle for this synthesis method varies continuously; therefore, the generated timbre transforms gradually at the transition of the reference sounds.

By decreasing the attractive force and increasing the repulsive force at the transition, the synthesized sound converges to the desired sound, and a smooth morphing result is achieved.

3.3. Real-time processing

The synthesis method proposed in this paper generates a wide variety of sounds and controls them through a small number of parameters; therefore, this method has the potential for realizing a novel and intuitive user interface for generating sounds. At this time, it is difficult for off-the-shelf personal computers to generate high-quality results in real time due to the large calculations required.

By implementing the ideas below, real-time processing is accomplished in computationally limited environments.

1. Decreasing the frame rate(e.g. 30fps)
2. Using a Graphics Processing Unit (GPU)
3. Decreasing the number of partials (e.g. 200)
4. Using the IFFT for synthesis

GPUs are able to realize fast parallel processing; hence it is effective to calculate the frequencies of many partials using them. In this research, the GLSL (OpenGL Shading Language) is used [12, 13].

The IFFT (inverse fast Fourier transform) is a common synthesis method which is used for decreasing the calculation time. In addition to this, the calculation volume is reduced by decreasing the FFT size and frame rate. However, deteriorations in resolution in both the time and frequency domains are observed, meaning the generated sound also has a loss in quality.

4. CONCLUSION

In this paper, structures and applications of a new synthesis method, which is constructed using sinusoidal editing and a force-based algorithm, were proposed. Although this method is a complicated tool at this time, examples of productive features are indicated.

The following two points are considered important for the future prospects of this research:

1. Development of real-time processing and an intuitive and interactive user interface.
2. Adoption of advanced physical laws.

As mentioned in section 3, this method has the potential to realize various sound synthesis. However, it is difficult to utilize this method for musical performances and real-time and/or interactive compositions due to the large volume of calculations required.

This method also has the possibility to involve various physical laws, for instance, recent research results of fluid mechanics indicate large potentials for controlling a vast amount of particles and streams. The synthesis method presented in this paper has already accomplished the creation of a wide variety of sounds using a simple force-based model. It is proposed that more powerful, flexible, and intuitive synthesis methods can be realized by utilizing advanced models.

5. REFERENCES

[1] Thomas M. J. Fruchterman and Edward M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[2] John Adrian Bondy and Uppaluri Siva Ramachanda Murty, *Graph Theory with Applications*, The Macmillan Press Ltd., London, 1976.

[3] Robert McAulay and Thomas F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[4] Kelly Fitz and Lippold Haken, "Sinusoidal modeling and manipulation using lemur," *Computer Music Journal*, vol. 20, no. 4, pp. 44–59, 1996.

[5] Michael Klingbeil, "Software for spectral analysis, editing, and synthesis," in *Proceedings of the International Computer Music Conference*, 2005, pp. 107–110.

[6] Jont B. Allen, "Short term spectral analysis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics, Speech, and Processing*, vol. 25, no. 3, pp. 235–238, 1977.

[7] G. DiGiugno, "A 256 digital oscillator bank," in *Proceedings of the International Computer Music Conference*, MIT, Cambridge, 1976, pp. 188–91.

[8] F. Richard Moore, *Elements of Computer Music*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1990.

[9] John M. Grey, *An Exploration of Musical Timbre*, Ph.D. thesis, Stanford University, 1975.

[10] Lippold Haken Edwin Tellman and Bryan Holloway, "Timbre morphing of sounds with unequal numbers of features," *Journal of the Audio Engineering Society*, vol. 43, no. 9, pp. 678–689, 1995.

[11] Kelly Fitz Lippold Haken and Paul Christensen, *Analysis, Synthesis, and Perception of Musical Sounds*, chapter Beyond traditional sampling synthesis: Real-time timbre morphing using additive synthesis, pp. 122–14, James W. Beauchamp Eds. Springer, New York, 2007.

[12] John D. Owens et al., "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[13] Ricardo Marroquim and André Maximo, "Introduction to gpu programming with glsl," in *Tutorials of the XXII Brazilian Symposium on Computer Graphics and Image Processing*, Rio de Janeiro, Brazil, 2009, pp. 3–16.

6. APPENDIX: SOUND EXAMPLES

Sound examples are available online at the following address.

http://www.ryoho.com/software/sinfba/

A METHOD OF MORPHING SPECTRAL ENVELOPES OF THE SINGING VOICE FOR USE WITH BACKING VOCALS

Matthew Roddy
Dept. of Computer Science and Information Systems, University of Limerick, Ireland

Jacqueline Walker
Dept. of Electronic and Computer Engineering, University of Limerick, Ireland

ABSTRACT

The voice morphing process presented in this paper is based on the observation that, in many styles of music, it is often desirable for a backing vocalist to blend his or her timbre with that of the lead vocalist when the two voices are singing the same phonetic material concurrently. This paper proposes a novel application of recent morphing research for use with a source backing vocal and a target lead vocal. The function of the process is to alter the timbre of the backing vocal using spectral envelope information extracted from both vocal signals to achieve varying degrees of blending. Several original features are proposed for the unique usage context, including the use of LSFs as voice morphing parameters, and an original control algorithm that performs crossfades between synthesized and unsynthesized audio on the basis of voiced/unvoiced decisions.

1. INTRODUCTION

Sound morphing is a term that has been used to describe a wide range of processes and, as of yet, there is no consensus on a standard definition for the term due to variations in usage contexts, goals and methods. Despite the disparities in definitions, Caetano [1] remarks that, in most applications, the aim of morphing can be defined as "obtaining a sound that is perceptually intermediate between two (or more), such that our goal becomes to hybridize perceptually salient features of sounds related to timbre dimensions." The goal of achieving perceptually intermediate timbres is complicated by the multidimensional nature of timbre perception [2]. Classifications of the dimensions associated with timbre [3, 4] usually distinguish between features derived from the temporal envelope of the sound (e.g temporal centroid, log-attack time), and features derived from the spectral envelope of sounds (e.g spectral centroid, spectral tilt).

When attempting to achieve perceptually intermediate spectral features between sounds, many morphing systems adopt sinusoidal models in which the partials of a sound are represented as a sum of sinusoids that, in the case of musical sounds, are often quasi-harmonically related. A common strategy in morphing systems is to establish correspondences between the partials of two sounds and to interpolate the frequency and amplitude values [5, 6]. Methods based on this approach do not account for resonance peaks or formants that are delineated by the contour of the sound's spectral envelope. Consequently, the resulting intermediate spectral envelopes often display undesirable timbral behavior in which formant peaks are smoothed rather than shifted in frequency. Therefore, when hybridizing the non-temporal dimensions of timbre the challenge is finding parameterizations of the spectral envelope that can be interpolated to create perceptually linear shifts in timbre. Some spectral envelope parameterizations that have been proposed are: linear prediction coefficients (LPC) [7], cepstral coefficients (CC) [8], reflection coefficients (RC) [7], and line spectral frequencies (LSF) [9].

Different parameterizations of the spectral envelopes of musical instrument sounds were recently compared at IRCAM [10] using spectral shape features as timbral measures to determine which representations provided the most linear shift in peaks and spectral shape. They found that, of the parameterizations surveyed, LSFs provided the most perceptually linear morphs. This supports previous proposals [9, 11] for the use of LSFs as good parameters for formant modification. In the morphing process introduced below, this research is used in conjunction with research into the formant behavior of singers that has indicated that individual singers will sometimes alter the formant structures of vowels to blend in or stand out in an ensemble situation. Goodwin [12] found that singers in choirs lowered the intensity of their second and third formants, and sometimes shifted the formants down in frequency to blend better. Ternström [13] concluded that singers in barbershop quartets spread out the spacings of their formants to stand out for intonation purposes.

This paper presents a novel voice morphing process that is intended to be used as a studio tool to blend a backing vocal with a lead vocal. The process uses the spectral envelope of a lead vocalist to alter the spectral envelope of the backing vocalist on a frame by frame basis while preserving pitch information. The morphing process is built upon the observation that it is common in many music styles for a backing vocalist to sing the same phonetic material concurrently with the lead vocalist. Given this specific context, the formants of the two signals will be similar, and differences in the spectral envelopes can be attributed to differences in either the singer's pronunciation or the timbral characteristics of the individual's voice. It can be aesthetically desirable in this situation for vocalists to blend their timbre with other vocalists [12, 13]. In this context, if the spectral envelope of the backing vocalist is morphed with that of the lead vocalist, and the morphing method creates a perceptually linear morph, the formants that define phonetic information will remain intelligible and only the envelope information that affects the singer's individual timbre will be altered. Furthermore, since perceptually intermediary timbres between the two can be achieved using LSFs, the process can be used as a subtle effect.

This proposed morphing process could be useful in studio situations where the lead vocalist and a backing vocalist have contrasting timbres. In this scenario, the current common practice to achieve a blended timbre is to multitrack the lead vocalist perform-

ing both the lead and backing parts. In this situation, the timbral results are limited to either being perceptually blended (when the lead vocalist records both parts) or perceptually distinct (when the backing vocalist records their part). The proposed morphing process allows for a larger variety of combined vocal textures by creating gradations in the amount of blending between the two voices. The combined texture created by the two voices can be perceptually blended, perceptually distinct or any gradation in between the two depending on the LSF settings that are used.

The objectives of this voice morphing process differ from those of most morphing processes since the objective is not to achieve the target vocal sound, but rather to use its spectral envelope to modify the timbre of the source vocal, preserving its original harmonic structure and hence its fundamental frequency. The objectives of this morphing process share some similarities with those discussed in [14], in which features from two voices are combined to create a hybrid voice that retains one voice's pitch information.

The proposed morphing process falls within the bounds of some definitions of cross-synthesis in which an "effect takes two sound inputs and generates a third one which is a combination of the two input sounds. The idea is to combine two sounds by spectrally shaping the first sound by the second one and preserving the pitch of the first sound." [15] If this definition is adopted then the proposed process would be defined as cross-synthesis with a preliminary morphing stage in which the spectral envelope of the second sound is altered using envelope features extracted from the first sound.

In the next section the signal model used to morph the envelopes is described and an overview of the structure of an analysis/synthesis system that implements the process is presented. In section 3 the calculation of the LSF spectral envelope parameterization is discussed. In section 4 an original control algorithm that performs crossfades between the synthesized audio and the unsynthesized backing vocal audio is discussed. In section 5 a subjective discussion of the sonic results and the limitations of the process are presented as well as our conclusions.

2. SIGNAL MODEL AND THE STRUCTURE OF THE PROCESS

2.1. Source-filter signal model

This morphing process uses spectral modeling synthesis (SMS), as described by Xavier Serra [16], to synthesize a morphed version of a backing vocal signal. SMS models a sound $x(t)$, by splitting it into two components, a sinusoidal component $x_h(t)$, and a stochastic residual component $x_r(t)$. The sinusoidal component models the quasi-harmonic element of sounds by first detecting spectral peaks according to a quadratic peak-picking algorithm [17], followed by a refinement of these peaks on the basis of harmonic content. This harmonic component of the sound is modeled as a sum of sinusoids using:

$$x_h(t) = \sum_{k=0}^{K(t)} a_k(t) \exp[j\phi_k(t)] \tag{1}$$

where $a_k(t)$ and $\phi_k(t)$ are the amplitude and phase of the k$^{\text{th}}$ harmonic. The residual component is modeled by subtracting the harmonic component from the original signal. The residual is then synthesized using noise passed through a time-varying filter. When using SMS to synthesize the human voice, the residual generally models unvoiced sounds such as consonants and aspiration noise.

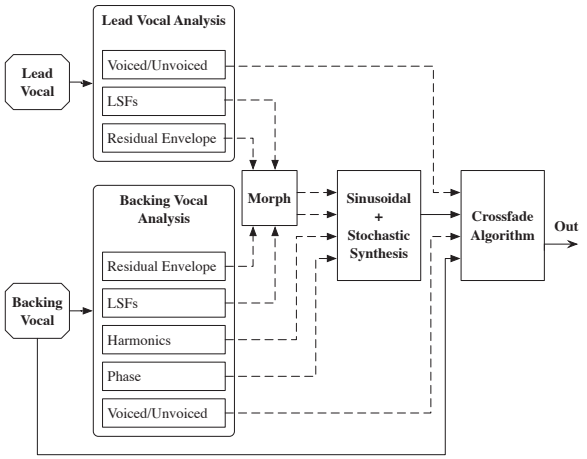


Figure 1: *Flow chart diagram of the morphing process. Dashed lines represent the flow of extracted data. Solid lines represent the flow of audio.*

The synthesis strategy adopted in this morphing process differs from traditional SMS in its use of a source-filter model which considers the amplitudes of the harmonics separately from the harmonics themselves. This model, proposed in [18], divides the harmonic component of a sound into an excitation source, in which the amplitudes of the harmonics are set to unity ($a_k = 1$), and a time-varying filter given by:

$$H(f,t) = |H(f,t)| \exp[j\psi(f,t)] \tag{2}$$

where $|H(f,t)|$ is the amplitude, and $\psi(f,t)$ is the phase of the system. The time-varying filter is derived using spectral envelope estimation methods described in section 3. The model for the representation of the harmonic element is then given by:

$$y_h(t) = \sum_{k=0}^{K(t)} |H[t, f_k(t)]| \exp[j(\phi_k(t) + \psi(f_k(t)))] \tag{3}$$

where $f_k(t) \approx k f_0(t)$, $\phi_k(t)$ is the excitation phase, and $\psi[f_k(t)]$ is the instantaneous phase of the k$^{\text{th}}$ harmonic. As such, the time-varying filter models the curve of the spectral envelope according to the formant structure and individual timbral characteristics of the singer. This approach, which was originally proposed for musical instruments, is adopted for the singing voice instead of traditional source-filter models, such as linear predictive coding, since it offers greater flexibility for timbral manipulation.

2.2. Process Structure

This morphing process belongs to the class of audio effects discussed by Verfaille *et al.* [19] known as external-adaptive audio effects. External-adaptive effects use features extracted from an external secondary input signal as control information to modify a primary input signal. In the case of this morphing process, features used to control the source-filter model described above are

extracted from the lead vocalist's signal (x_{Lv}) to alter the backing vocalist's signal (x_{Bv}) on a frame-by-frame basis. The structure of the process (shown in Fig. 1) can be divided into four stages: an analysis stage, a morphing stage, a synthesis stage, and a control stage.

During the analysis stage the spectral envelopes of the harmonic components of both the lead and backing vocal frames are estimated and parameterized as LSFs using a process described in section 3. The residual envelopes are extracted by subtracting their harmonic components from their respective magnitude spectra. Decimation is then used to create line-segment representations of the residual envelopes. Voiced/unvoiced information is also extracted from the two vocals using a two way mismatch (TWM) algorithm [20]. In addition to the three features listed above that are extracted from both voices, two additional features, the frequencies of harmonics and phase information, are extracted from the backing vocal. These two features are used, unaltered, during the synthesis process. By using the original phase and harmonic structures, the pitch information of the backing vocalist's audio is preserved and only its timbral qualities are altered.

During the morphing stage of the process, the parametric representations of both the harmonic and residual envelopes (LSFs and line segments, respectively) are morphed using:

$$M(\alpha) = \alpha S_{Lv} + [1-\alpha]S_{Bv} \qquad 0 \leq \alpha \leq 1 \qquad (4)$$

where S_{Lv} and S_{Bv} are arrays containing the spectral envelope parameters of the lead and backing vocals respectively. The variable α is the morph factor that controls the amount of timbral blending. The morphed parameters are input into the SMS system during the synthesis stage of the process along with the original harmonic frequencies and phase information of the backing vocalist. The final control stage of the process (described in section 4) uses the voiced/unvoiced information extracted during the analysis stage to perform crossfades between audio produced by the SMS system and the original unvoiced backing vocal audio.

The overall structure of the effect, and the unique control algorithm (discussed in section 4) were designed with the intention of laying the ground-work for a real-time SMS implementation. A possible real-time effect could be implemented using a side-chain to input the lead vocal signal. A similar real-time SMS application has been discussed in [21].

3. MORPHING USING LINE SPECTRAL FREQUENCIES

The chosen method of calculating LSFs begins with the magnitudes of the harmonic component of x_h, which are derived using the peak-picking algorithm. The harmonic component is first squared and then interpolated to create the power spectrum $|X(\omega)|^2$. An inverse-Fourier transform is performed on the power spectrum to calculate autocorrelation coefficients ($r_{xx}(\tau)$) according to the Wiener-Khinchin theorem:

$$r_{xx}(\tau) = \mathcal{F}^{-1}\{|X(\omega)|^2\} \qquad (5)$$

The first p autocorrelation coefficients are used to calculate p linear prediction coefficients using Levinson-Durbin recursion to solve the normal equations:

$$\sum_{k=1}^{p} a_k r_{xx}(i-k) = r_{xx}(i) \quad , i = 1, \ldots, p. \qquad (6)$$

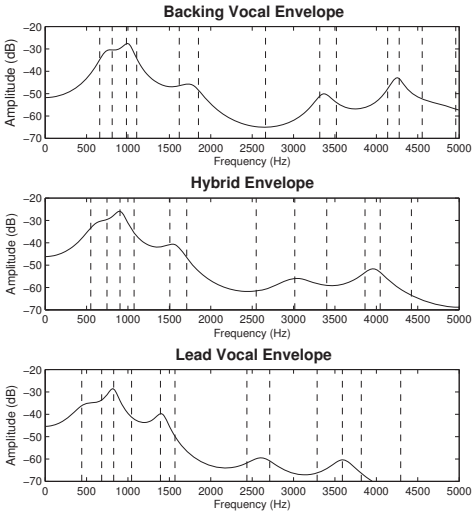


Figure 2: *Spectral envelopes demonstrating the effect of morphing sung [a] vowels using LSFs (overlaid in dashed lines). The hybrid envelope shows the resulting formant shift behavior when a morphing factor (α) of 0.5 is used.*

LSFs are then derived from the linear prediction coefficients (a_k) by considering the coefficients as a filter representing the resonances of the vocal tract. Based on the interconnected tube model of the vocal tract, two polynomials are created that correspond to a complete closure and a complete opening at the source end of the interconnected tubes [22]. The polynomials are generated from the linear prediction coefficients by adding an extra feedback term that is either positive or negative, modeling energy reflection at a completely closed glottis or a completely open glottis respectively. The roots of these polynomials are the LSFs. A thorough explanation of the process of calculating LSFs from linear prediction coefficients, as well as the reverse process, is given in [22].

In the line spectral domain, the LSFs from the backing vocal are morphed with the LSFs from the lead vocals using equation (4). An example of morphed LSFs and the hybrid spectrum created using this process are shown in Fig. 2. The figure shows a clear shift in the amplitudes and central frequencies of the of the third and fourth formants, demonstrating the good interpolation characteristics discussed in [9, 11, 10]. These morphed LSFs are then converted into the linear prediction coefficients that constitute the all-pole filter $H[f_k(t)]$ discussed in section 2.1. Using

$$H[\omega_k] = \frac{1}{1 + \sum_{n=1}^{p} a(n) \exp[-j\omega_k n T_s]} \qquad (7)$$

where $\omega_k = 2\pi f_k$ and T_s is the sampling interval, the linear prediction filter is evaluated at the individual harmonic frequencies.

4. CROSSFADE ALGORITHM

An important feature of this morphing process is a control algorithm that performs crossfades (shown in Fig. 3) between the original unvoiced consonants of the backing vocal and morphed

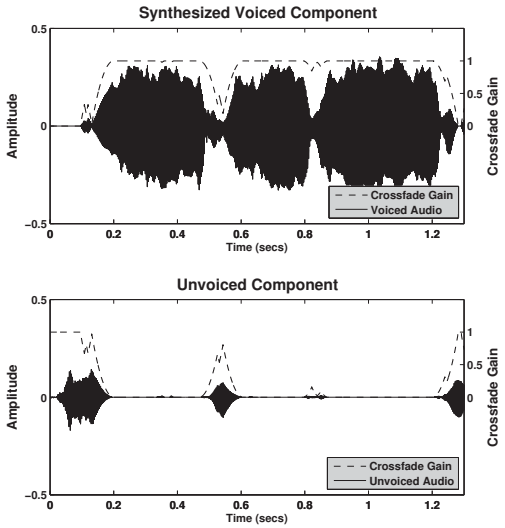


Figure 3: *The synthesized harmonic plus stochastic audio (top figure), the unsynthesized original audio (bottom figure), with their respective crossfade gain values. Crossfades with an exponential value of 2 and a fade length of 2 windows (2048 samples) were used.*

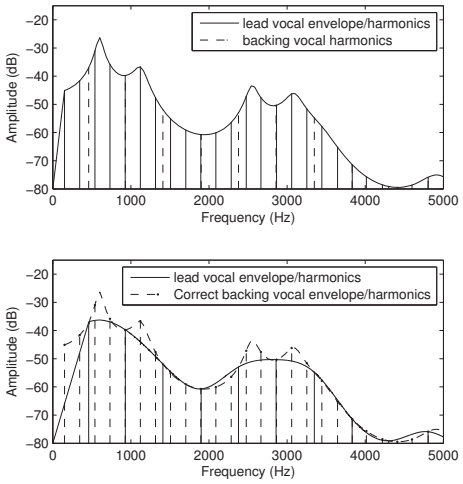


Figure 4: *Demonstration of the vowel spectra of a phoneme ([a]) created when the target lead vocal has either a lower (a) or higher (b) fundamental frequency relative to the backing vocalist. In (a) the lead vocalist has a lower fundamental ($f_0 = 147$ Hz) and the backing vocalist has a higher fundamental (f_0 = 497 Hz). In (b) the fundamental frequencies are swapped.*

voiced sounds. This reconstruction algorithm for the morphing process uses the voiced/unvoiced classifications for the frame plus a fade position inherited from the previous frame. The crossfades are performed by indexing tables created with user-defined exponential curves. The fades are designed to be at unity gain and the number of samples needed to complete a fade is specified by the user in window lengths. In the experiments discussed below in section 5, the hop size of 256 samples is taken into account when performing the crossfades by applying the indexed gain amount to only 256 samples at a time. The length of the fade was set to 3072 samples with an analysis window-length of 1024 samples and a sampling frequency of 44100 Hz.

The crossfades address a number of issues that are unique to the application context. Firstly, although the morphing process is designed to operate under the condition that both voices are singing the same phonetic material concurrently, there will almost always be discrepancies in the timing of the two voices. To avoid the spectral envelope of a consonant being imposed on the harmonic structure of a vowel, or vice versa, the algorithm checks whether either of the two voices contain unvoiced sounds in their corresponding frames. If so, the algorithm either fades towards the original unsynthesized audio or it remains with the unsynthesized audio at full gain, depending on the initial position of the fade. An equally important reason for using a crossfading system is that the transients of consonants synthesized using the filtered noise of SMS are considered to lack realism due to a loss of sharpness in their attack [17, 23]. A reason for performing a gradual crossfade is to make up for inaccuracies in voiced/unvoiced decisions made by the TWM algorithm during the analysis stage. These inaccuracies can be observed in Fig. 3 by the presence of jagged lines during either steady state voiced sections or during transitions. They represent decisions that change quickly over the course of a small number of frames. They are usually a single voiced frame surrounded by unvoiced frames, or vice versa. The use of gradual transitions masks the overall impact that these isolated voicing classifications have.

5. DISCUSSION

5.1. Informal Testing

The effectiveness of the two principal features of this morphing process (the use of LSFs and the reinsertion of unvoiced consonants using crossfades) were informally tested by comparing the morphing process with a second SMS-based morphing process [24] that uses synthesized unvoiced segments and morphs voiced segments using simple interpolation of the spectral envelopes created by the harmonic components. From a five second recording of a backing vocal, two sets of processed backing vocals were created: one using the morphing process presented here, and another using the second envelope interpolation process used for comparison. In each of the sets, the backing vocal was synthesized using the morphing factors: $\alpha = 0$, 0.5, 1.0. To assess the realism of the resulting audio, the two sets were first played independent of their corresponding lead vocal. Subsequently, the same processed backing vocals were played in conjunction with their corresponding lead vocal to informally assess the level of perceptual blending.

An initial observation was that the realism contributed by the reintroduction of the original unvoiced consonants using the crossfade algorithm was significant when compared with the envelope interpolation process without the reinsertion of consonants. Similar to what was found by [17, 23], the use of SMS to model unvoiced segments was considered to result in consonants that lacked

definition due to being modeled by the noise residual. A drawback of the use of the crossfades was that, as α increased, there were noticeable artifacts that appeared during the transitions between synthesized and unsynthesized audio. These artifacts are due to the differences between the two spectral envelopes that are perceptually highlighted by rapid changes. The effect of these artifacts can be reduced by increasing the length of the crossfade. When considering the realism contributed by the LSFs, as the α value was increased, the resulting voiced sounds of the LSF-based morphing process remained defined and realistic, due to the linear shift in timbral features. In contrast, the voiced segments synthesized using the second SMS morphing process lacked definition at $\alpha = 0.5$, due to the peak smoothing behavior that occurs during the interpolation of envelopes. When the two sets of processed backing vocals were played in conjunction with the lead vocal it was considered that the formant shift behavior due to the use of LSFs increased the level of perceptual blend between the two voices as the α value was increased. With the second SMS morphing process, this was not always the case due to the peak smoothing behavior.

5.2. Limitation

One of the limitations of the morphing process presented here is that it cannot be used to effectively blend backing vocals that have a lower fundamental than their corresponding lead vocals. This is due to the envelope-sampling behavior of harmonics. As shown in Fig. 4, the harmonics sample the vowel envelope at frequencies that are approximately integer multiples of the fundamental. Given the case of a backing vocal with a lower fundamental than the lead vocal, the lead vocal vowel envelope will not be sampled at a high enough rate for the backing vocalist to accurately recreate the formants of the vowel. In addition, the harmonics of the backing vocal that are at lower frequencies than the fundamental of the lead vocal cannot be designated appropriate amplitude values since there is no vowel envelope information at frequencies below the fundamental.

5.3. Conclusion

The voice morphing process presented in this paper uses LSFs to modify the timbral characteristics of a backing vocal, including the frequencies and strengths of formants, to achieve different levels of blending with a target lead vocal. In choral situations, formant modification by singers has been observed in which formant strengths have been lowered and centre frequencies slightly shifted for the purpose of blending [12]. Although the actions of a choral singer and the timbral modifications produced by this process create different results, both are motivated by the objective of producing a homogeneity of timbre through modification of the spectral envelope. For this reason, this process is proposed as a potentially valuable artistic tool for blending two voices.

6. REFERENCES

[1] M. F. Caetano and X. Rodet, "Automatic timbral morphing of musical instrument sounds by high-level descriptors," in *Proceedings of the International Computer Music Conference*, 2010, pp. 254–261.

[2] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *The Journal of the Acoustical Society of America*, vol. 61, p. 1270, 1977.

[3] K. Jensen, "The timbre model," *Journal of the Acoustical Society of America*, vol. 112, no. 5, p. 2238–2251, 2002.

[4] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," IRCAM, Tech. Rep., 2004.

[5] E. Tellman, L. Haken, and B. Holloway, "Morphing between timbres with different numbers of features," *Journal of the Audio Engineering Society*, vol. 62, no. 2, pp. 678–689, 1995.

[6] K. Fitz and L. Haken, "Sinusoidal modeling and manipulation using lemur," *Computer Music Journal*, vol. 20, no. 4, pp. 44–59, 1996.

[7] J. A. Moorer, "The use of linear prediction of speech in computer music applications," *Journal of the Audio Engineering Society*, vol. 27, no. 3, pp. 134–140, 1979.

[8] M. Slaney, M. Covell, and B. Lassiter, "Automatic audio morphing," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 2, 1996, pp. 1001–1004.

[9] K. K. Paliwal, "Interpolation properties of linear prediction parametric representations," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[10] M. Caetano and X. Rodet, "Musical instrument sound morphing guided by perceptually motivated features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 8, pp. 1666–1675, Aug. 2013.

[11] R. W. Morris and M. A. Clements, "Modification of formants in the line spectrum domain," *Signal Processing Letters, IEEE*, vol. 9, no. 1, pp. 19–21, 2002.

[12] A. W. Goodwin, "An acoustical study of individual voices in choral blend," *Journal of Research in Music Education*, vol. 28, no. 2, 1980.

[13] S. Ternstrom and G. Kalin, "Formant frequency adjustment in barbershop quartet singing," in *Intenational Congress on Acoustics*, 2007.

[14] P. Depalle, G. Garcia, and X. Rodet, "The recreation of a castrato voice, Farinelli's voice," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1995, p. 242–245.

[15] U. Zölzer, Ed., *DAFX: Digital Audio Effects*. John Wiley & Sons, 2011, ch. Glossary, pp. 589–594.

[16] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[17] X. Serra, "A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition," Ph.D. dissertation, Stanford University, 1989.

[18] M. Caetano and X. Rodet, "A source-filter model for musical instrument sound transformation," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012, pp. 137–140.

[19] V. Verfaille, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (a-DAFx): a new class of sound transformations," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 5, pp. 1817–1831, Sep. 2006.

[20] R. C. Maher and J. W. Beauchamp, "Fundamental frequency estimation of musical signals using a two-way mismatch procedure," *The Journal of the Acoustical Society of America*, vol. 95, pp. 2254–2263, 1994.

[21] P. Cano, A. Loscos, J. Bonada, M. De Boer, and X. Serra, "Voice morphing system for impersonating in karaoke applications," in *Proceedings of the International Computer Music Conference*, 2000, pp. 109–112.

[22] I. V. McLoughlin, "Line spectral pairs," *Signal Processing*, vol. 88, no. 3, pp. 448–467, Mar. 2008.

[23] T. S. Verma and T. H. Meng, "Extending spectral modeling synthesis with transient modeling synthesis," *Computer Music Journal*, vol. 24, no. 2, pp. 47–59, 2000.

[24] J. Bonada, X. Serra, X. Amatriain, and A. Loscos, *DAFX: Digital Audio Effects*. John Wiley & Sons, 2011, ch. Spectral Processing, pp. 393–445.

SHORT-TIME TIME-REVERSAL ON AUDIO SIGNALS

Hyung-Suk Kim
CCRMA,
Stanford University
Stanford, CA, USA
hskim08@ccrma.stanford.edu

Julius O. Smith
CCRMA,
Stanford University
Stanford, CA, USA
jos@ccrma.stanford.edu

ABSTRACT

We present an analysis of short-time time-reversal on audio signals. Based on our analysis, we define parameters that can be used to control the digital effect and explain the effect each parameter has on the output. We further study the case of 50% overlap-add, then use this for a real-time implementation. Depending on the window length, the effect can modify the output sound variously, from adding overtones to adding reverse echoes. We suggest example use cases and digital effects setups for usage in sound design and recording.

1. INTRODUCTION

Overlap-add (OLA) methods are widely used in digital audio effects. Examples include time stretching, pitch shifting, phase vocoder, and more complex effects based on the short-time Fourier transform (STFT). [1, 2, 3, 4, 5, 6, 7, 8]. In this paper, we explore a special case of OLA effects termed short-time time-reversal (STTR) — reversal of overlapping short time intervals.

Time reversal is widely used in many fields including acoustics, ultrasound, underwater communications, and biomedical engineering as a method for focusing propagated signals ([9, 10]). Contrarily, it does not seem to be a noticeable topic in the digital audio effects literature. The application of time reversal in audio effects is generally not covered because the system becomes non-causal. For short time intervals, however, it is possible to add a short delay to the output, a buffering period similar to that of delay line effects, to alleviate non-causality. It is worth noting that though STTR is linear, it is not time invariant.

Time reversal audio effects are available on the market. Grain Reverser, a Max for Live plugin, and Reverse Grain from Native Instruments are examples. These audio effects are designed to be temporal not spectral. As we will examine in later sections, time reversal of shorter time intervals, 30 ms or less, with overlap-add creates complex spectral and temporal effects and opens new possibilities for sound design. However, due to the nature of the effect it may be hard to control and may create unexpected and unpleasant results. We shed light on this through Fourier analysis.

The remainder of this paper is structured as follows. We mathematically define STTR and look at the Fourier analysis of STTR (§2), cover the parameters of STTR and examine the effects of each parameter (§3), explore a special case with 50% OLA (§4), look at a real-time implementation of the 50% OLA case (§5), and discuss observations using the implementation (§6).

2. FOURIER ANALYSIS

In this section, we define STTR and perform a Fourier transform to study its effects in the frequency domain.

2.1. Short-Time Time-Reversal

Let $x(t)$ be the input signal and $w(t)$ be the window function of length L with constant overlap-add for step size R: (Equation 2.1)

$$\sum_{m=-\infty}^{\infty} w(t - mR) = 1 \tag{2.1}$$

The STTR signal $y(t)$ is formed by the following steps.

Step 1. Window the input signal $x(t)$ with $w(t - mR)$.

Step 2. Reverse the signal under the window:

- **(a)** Move the windowed signal to the origin.
- **(b)** Reverse the windowed signal.
- **(c)** Move it back to the original position.

Step 3. Sum the reversed signals.

Following the steps we get

$$y(t) = \sum_{m=-\infty}^{\infty} x(-t + 2mR)w(t - mR). \tag{2.2}$$

Note that without the time reversal, the time shifts for $x(t)$ from Step 2 would cancel out.

2.2. General Derivation

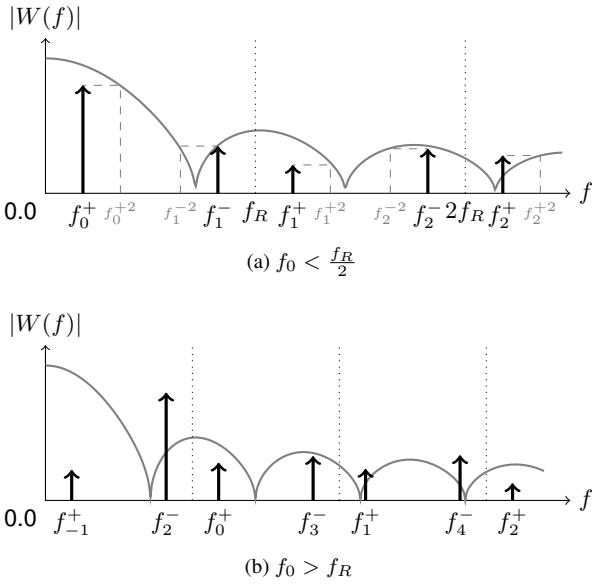
The Fourier transform of $y(t)$ becomes

$$Y(f) = \sum_{m=-\infty}^{\infty} e^{-2\pi i f m 2R} X(-f) * e^{-2\pi i f m R} W(f). \tag{2.3}$$

We can expand the convolution in equation (2.3) and use the Fourier transform of an impulse train to simplify $Y(f)$ to

$$Y(f) = \frac{1}{R} \sum_{k=-\infty}^{\infty} X\left(f - \frac{k}{R}\right) W\left(2f - \frac{k}{R}\right). \tag{2.4}$$

For a detailed derivation of equation (2.4), see the Appendix.



(a) $f_0 < \frac{f_R}{2}$

(b) $f_0 > f_R$

Figure 1: A visualization of equation (2.5), the STTR output for a single sinusoid, for $f_0 < \frac{f_R}{2}$ (1a) and $f_0 > f_R$ (1b). The dotted vertical lines mark multiples of f_R. In 1a, the dashed lines show the magnitude $|W(f_k^{\pm 2})|$ for frequency $f_k^{\pm}$. Compared to the window function spectrum $|W(f)|$, the impulse functions need to be scaled by $\frac{1}{2R}$.

2.3. Single Sinusoid Input

From a quick glance equation (2.4) may not intuitively make sense. We can gain insight into the effect of STTR in the frequency domain by looking at the simple case of a single sinusoid.

For a single sinusoid $x(t) = \cos(2\pi f_0 t)$, equation (2.4) becomes,

$$Y(f) = \frac{1}{2R} \sum_{k=-\infty}^{\infty} \Big\{ W(f_k^{+2})\delta(f - f_k^{+}) + W(f_k^{-2})\delta(f - f_k^{-}) \Big\} \tag{2.5}$$

where $f_R = \frac{1}{R}$, $f_k^{\pm} = kf_R \pm f_0$ and $f_k^{\pm 2} = kf_R \pm 2f_0$.

At each integer multiple of frequency f_R, we get two peaks at offsets $\pm f_0$, a weighted copy or "reflection" of the original frequency spectrum ($f_k^{\pm}$). The weights are given by not the corresponding sample of the window spectrum but that at offsets $\pm 2f_0$, twice the frequency offsets ($f_k^{\pm 2}$). Figure 1 visualizes equation (2.5) for two cases, $f_0 < f_R/2$ and $f_0 > f_R$. When $f_0 < f_R/2$ finding the correct weights for each peak at $f_k^{\pm}$ is trivial. It quickly gets complicated when $f_0 > f_R$ (Figure 1b). Adjacent peaks are not from the same reflection, i.e. f_1^- is not the closest peak to f_1^+. The weights for each peaks are from even further away points. Furthermore, the original frequency f_0 is not necessarily the peak with the greatest amplitude.

Figure 2b shows the spectrogram of the STTR output for a linear sine sweep for a short window length. The pattern on this figure can be explained by equation (2.5). We cover the parameters of STTR and the observed effects of each parameter in §3.

2.4. Gaussian White Noise

We look at the discrete STTR to analyze the output for Gaussian white noise. Let $x[n]$ be an uncorrelated Gaussian white noise process and $y[n]$ the output after STTR. Since all samples of $y[n]$ are linear combinations of $x[n]$, we know that they are also Gaussian random variables.

We now look at the covariance matrix to verify if all samples in $y[n]$ are uncorrelated. We first look at the case of 50% OLA ($R = L/2$), then extend this to the generalized case. For a given section along the alignment of half the window length, like one slot in Figure 3, the output will be the weighted linear sum of the surrounding time slots. Let X be a $3R \times 1$ random vector with the values of $x[n]$ for $n = [mR, (m+3)R)$, the span of 2 overlapping windows, and Y be a $R \times 1$ random vector with the values of $y[n]$ for $n = [(m+1)R, (m+2)R)$, where the two windows overlap. We can formulate Y as follows,

$$Y = \begin{pmatrix} A & 0 & B \end{pmatrix} X$$

where

$$A_{ij} = \begin{cases} w[j] & i = R - 1 - j; \\ 0 & \text{otherwise,} \end{cases}$$

and

$$B_{ij} = \begin{cases} w[j+R] & i = R - 1 - j; \\ 0 & \text{otherwise.} \end{cases}$$

That is, A and B are cross diagonal matrices with the split window components of $w[n]$ for each overlapping component from the corresponding parts of $x[n]$. Since the covariance matrix of X, V_X is the identity matrix I, the covariance matrix of Y is

$$V_Y = \begin{pmatrix} A & 0 & B \end{pmatrix} I \begin{pmatrix} A \\ 0 \\ B \end{pmatrix} = AA^T + BB^T \tag{2.6}$$

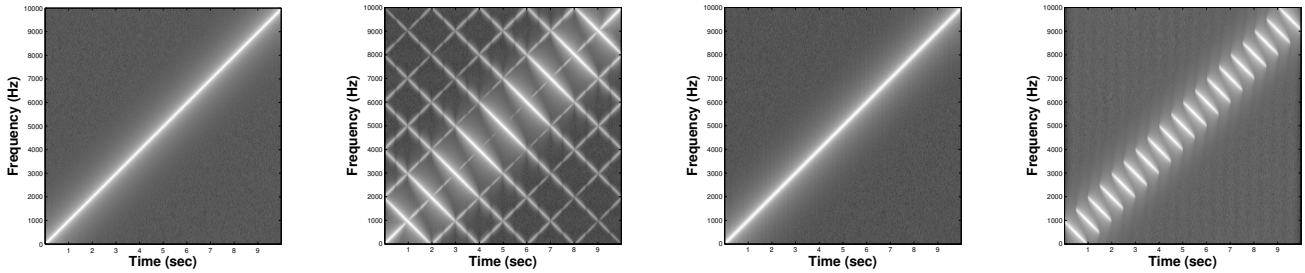
Since A and B are cross diagonal matrices, V_Y is a diagonal matrix and thus all elements of Y are uncorrelated. We can generalize equation (2.6) to any overlap ratio by splitting $w[n]$ onto more cross diagonal matrices. This holds true regardless of the window type. The values of the main diagonal, however, will not be constant ($V_Y \neq I$) but will be dependent on $w[n]$ and the overlap ratio. This means the window type and overlap ratio will be imprinted on the variance for each sample within a given slot. See [11] for an analysis of the effects of OLA on noise.

3. PARAMETERS

Equation (2.5) gives us insight into the parameters that can be used to change the audible effects of STTR. First we can change the window type as well as the length of the window L. Also, we can change R, the step size.

3.1. Window Type

The window type defines the shape of the function $W(f)$. This affects the weights of the overtones. Choosing a window with high sidelobe levels, e.g., a rectangular window, will in general increase the power of the overtones. By smoothly changing the window shape it is possible to shape the overtones. Compared to the sidelobe levels, the mainlobe width has a subtle effect of spreading the peak energy, i.e., the frequency with maximum power over a number of sinusoid peaks. It is worth noting that the peak frequency is not necessarily the original sinusoid frequency (Figure 1b).



(a) Input signal (b) Window length 1 ms (c) Window length 33 ms (d) Window length 1000 ms

Figure 2: Spectrogram plots showing the effect of STTR window length for 50% overlap-add. The sampling frequency for all signals in this figure is 20 kHz. Figure 2a shows the spectrogram of the input signal, a 10 second linear sine sweep from 0 Hz to 10 kHz. For short window lengths the "reflected" overtones of the signal are visible (2b). As the window length and hop size increase, the reflections are pulled in closer to the main diagonal, decreasing the visibility of STTR on the spectrogram (2c). Further increasing the window length, the time reversal structure becomes visible (2d).

3.2. Step Size

For short window lengths, the step size R changes the reflection frequencies kf_R. Decreasing R will increase spacing between overtones $f_R = 1/R$. The step size will also change the overtone weights as can be seen in Figure 1. We can regulate the overtone weights in regard to the window length by defining the overlap ratio $\alpha = R/L$ and using this as a parameter instead of R.

3.3. Window Length

The window length L determines the width of the window spectrum $W(f)$. As L increases the width of $W(f)$ decreases, eventually resembling an impulse function. At the same time the time reversal effect becomes more audible due to the longer durations that become reversed.

Figure 2 shows the spectrograms of STTR on a linear sine sweep from 0 Hz to 10 kHz with different window lengths. For shorter window lengths (2b), we see the overtones explained in §2.3. For window lengths around 30 ms (2c), the width of $W(f)$ decreases to the point that the reflections disappear on the spectrogram. However, STTR affects the timbre adding roughness or shimmer to the sine sweep. At longer lengths, window lengths beyond 100 ms (2d), we see the overlapping reverse sweeps.

3.4. Relation between Parameters

Though we cover the effects of each parameter separately, it must be noted that they are not independent. The spectrum $W(f)$ depends on both the window type and the window length. The weights of each overtone depend on both $W(f)$ and R.

Furthermore, the step size R must be chosen so that equation (2.1) holds. R cannot be an arbitrary value and is dependent on the type of window as well as its length. When the window sidelobe level is negligible above some frequency f_c, all step-sizes $R < f_s/f_c$ will yield substantially constant overlap-add, where f_s denotes the sampling rate [1, 7].

We can reduce the complexity by fixing the overlap ratio, α. For a fixed α, the window length becomes the parameter that changes the effect of STTR most, since $R = \alpha L$. In the following sections we cover the case where $\alpha = 0.5$ (50% OLA).

4. SPECIAL CASE STUDY: 50% OLA

Here we examine a case for a fixed 50% overlap ratio ($\alpha = 0.5$). The price of fixing α is to lose the freedom of changing the weights of the overtones. However, it simplifies the process of designing a window function.

For a window function to work for 50% OLA, it must satisfy the following constraints.

1. Non-negative $w(t)$ is assumed to be non-negative:

$$w(t) > 0$$

2. Symmetry As with most window functions, we expect $w(t)$ to be even:

$$w(t) = w(-t)$$

3. Constant OLA From equation (2.1) with $C = 1$ and $R = \frac{L}{2}$,

$$w(t) + w\left(\frac{L}{2} + t\right) = 1.$$

From the constraints above, we find that we can choose any shape for the interval $t = [-\frac{L}{2}, -\frac{L}{4})$, a quarter of the window, with the only constraint being $w(\pm\frac{L}{4}) = \frac{1}{2}$. This opens possibilities for designing various windows to create different overtone weights, including linear mixtures of known constant OLA windows.

5. IMPLEMENTATION

In this section we cover an audio plug-in implementation of 50% OLA STTR. Figure 3 shows the timing relations between the input buffer and the output buffer. This can be implemented efficiently using a single delay line and two output taps. In general, for an arbitrary overlap ratio α, we need $\lceil \frac{1}{\alpha} \rceil$ taps. The length of the delay line is $2L_{max}$, where L_{max} is the longest allowed window length. This is constant regardless of the step size. We can also add another output tap on the delay line to delay the input signal to match that of the STTR signal.

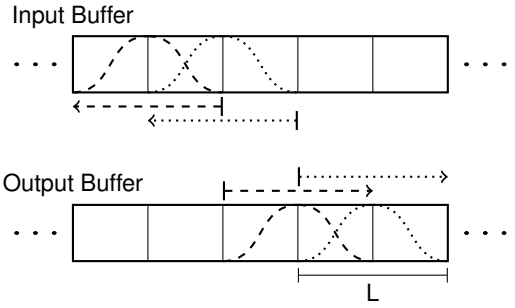


Figure 3: Relation between input signal and output signal for 50% OLA STTR. It shows two overlapping windows and the corresponding read and write directions. We can see that the first sample of a windowed input signal will be the last to be rendered to the output, two window lengths later.

We implemented 50% OLA STTR as an audio plugin with JUCE[1]. For practical purposes, we exposed the parameters, window length, window shape and wet/dry mix. The window length parameter is on a log scale ranging from 0.1 ms to 0.5 s. The window shape parameter, ranging from 0 to 1, mixes a rectangular window with a Hann window with 0 being the rectangular window and 1 being the Hann window. The wet/dry mix weighs the output of STTR with the original signal. This is particularly useful for taming the harshness of STTR caused by the overtones. We will look at some example uses in §6.2.

6. OBSERVATIONS

Implementation of an audio plug-in allows the real-time exploration of the digital audio effect. In this section, we test various input signals and present the findings. We cover the perceptual qualities of STTR and suggest example use cases.

6.1. Perceptual Effects of STTR Depending on Window Length

In §2, we covered the effects of window length on a single sinusoid and Gaussian white noise. Here we will make a qualitative assessment on the effects of STTR on more complex audio signals.

For window lengths of less than 1 ms, STTR creates many regularly spaced overtones. This causes the output to sound harsh, metallic and aliased, but with no stretching of the original frequencies. The effects are mostly spectral with almost no effect in the time domain.

For window lengths between 1 ms and 30 ms, we start hearing deflections in the transitions, that is, the pitch, like that of a singing voice, starts moving in a different direction than the original signal. Tonal sounds start sounding detuned.

From 30 ms to 100 ms, the sounds start to flutter. STTR starts having a temporal effect. For sounds like guitar, it adds a shimmering effect, similar to a mixture of chorus and reverb.

Beyond 100 ms, we hear the time reversal. Mixing some of the input signal makes it a reverse echo effect. Due to the delay in the implementation, when mixed with the input signal the delay becomes noticeable at larger window sizes, which also contributes to the timbre.

6.2. Example Usage

Based on the observations in the previous section we have found example use cases for our implementation of 50% OLA STTR.[2]

One obvious use is to set a long window length, mix the output with the dry signal and use it to create a reverse echo effect. This works particularly well with arpeggiated instruments such as guitars or pianos.

STTR can be used to change the direction of pitch by setting the window length around 1 ms and 30 ms. Since this extends the spectrum, it is recommended to add a low pass filter to reduce the extreme overtones. This can be used on pitched sounds such as a speech or a car accelerating to make versions with different pitch trajectories.

STTR can also be used to extend the spectrum and add sparkle when set to very short window lengths. For this use, it is recommended to use a low pass filter or band pass filter as an input stage to control the aliasing effects and also a low pass filter on the output stage to reduce extreme overtones.

7. CONCLUSION

We have presented STTR, a novel digital audio effect for manipulating an input signal both spectrally and temporally. Despite its simple implementation, one delay line and a few output taps, it is possible to achieve a variety of effects by changing the window length. STTR opens up new methods for designing and manipulating sounds. We conclude this paper by examining possible extensions of STTR.

We examined the case of 50% OLA STTR and found the degrees of freedom for designing window functions to shape the overtone. It is worth looking further into the effects of the shape of the window function on the timbre and find window design principals for 50% OLA STTR.

Another aspect to further investigate is the effect of time varying window lengths. We hypothesize that for short window lengths, the effect will be similar to a chorus effect (time varying comb filters), yet the spectral peaks will move in alternating directions which may cause a different perceptual effect. We have yet to see what the effect will be at longer window lengths.

Pitch synchronous STTR is also a promising direction to explore. At short window lengths, STTR expands the spectrum of an input signal. Together with a pitch tracker, it may be possible to harmonize a musical signal tonally or atonally. This can also be used to bend the direction of pitch by taking advantage of the fact that we have overtones moving in both directions.

8. REFERENCES

[1] Jont B. Allen and Larry R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, Nov. 1977.

[2] Mark Dolson, "The phase vocoder: A tutorial," vol. 10, no. 4, pp. 14–27, 1986.

[1] `http://www.juce.com`

[2] Sound examples can be found at `http://ccrma.stanford.edu/~hskim08/sttr/`.

[3] James A. Moorer, "The use of the phase vocoder in computer music applications," vol. 26, no. 1/2, pp. 42–45, Jan./Feb. 1978.

[4] Jean Laroche and Mark Dolson, "New phase-vocoder techniques for real-time pitch shifting, chorusing, harmonizing, and other exotic audio modifications," *Journal of the Audio Engineering Society*, vol. 47, no. 11, pp. 928–936, Nov. 1999.

[5] Axel Röbel, "A new approach to transient processing in the phase vocoder," 2003.

[6] S. Roucos and A. M. Wilgus, "High quality time scale modification for speech," Mar. 1985, pp. 493–496, Introduces the SOLA (Synchronous OverLap-Add) technique for time scale modification.

[7] Julius O. Smith, *Spectral Audio Signal Processing*, `http://ccrma.stanford.edu/~jos/sasp/`, Dec. 2011, online book.

[8] Udo Zölzer, *DAFX*, Digital Audio Effects. John Wiley & Sons, Mar. 2011.

[9] Sylvain Yon, Mickael Tanter, and Mathias Fink, "Sound focusing in rooms: The time-reversal approach," *The Journal of the Acoustical Society of America*, vol. 113, no. 3, pp. 1533–1543, 2003.

[10] Mathias Fink, Gabriel Montaldo, and Mickael Tanter, "Time-reversal acoustics in biomedical engineering," *Annual review of biomedical engineering*, vol. 5, no. 1, pp. 465–497, 2003.

[11] Pierre Hanna and Myriam Desainte-Catherine, "Adapting the overlap-add method to the synthesis of noise," pp. 101–104, 2002.

9. APPENDIX: DERIVATION OF EQUATION (2.4)

$$
\begin{aligned}
Y(f) &= \sum_{m=-\infty}^{\infty} \Big(e^{-2\pi i f m 2R} X(-f) * e^{-2\pi i f m R} W(f) \Big) \\
&= \sum_{m=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi i (f-\tau) m 2R} X(\tau - f) e^{-2\pi i \tau m R} W(\tau) d\tau \\
&= \int_{-\infty}^{\infty} X(\tau - f) W(\tau) \Big(\sum_{m=-\infty}^{\infty} e^{-2\pi i (2f-\tau) m R} \Big) d\tau \\
&= \int_{-\infty}^{\infty} X(\tau - f) W(\tau) \frac{1}{R} \sum_{k=-\infty}^{\infty} \Big(\delta\big(2f - \tau - \frac{k}{R}\big) \Big) d\tau \\
&= \frac{1}{R} \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} X(\tau - f) W(\tau) \delta\big(\tau - 2f + \frac{k}{R}\big) d\tau \\
&= \frac{1}{R} \sum_{k=-\infty}^{\infty} X\big(f - \frac{k}{R}\big) W\big(2f - \frac{k}{R}\big)
\end{aligned}
$$

On the third line, we use the Fourier transform of an impulse train, the Dirac comb function $\text{Ш}_T(t)$.

$$
\text{Ш}_T(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) = \frac{1}{T} \sum_{k=-\infty}^{\infty} e^{2\pi i k t / T}
$$

A STATISTICAL APPROACH TO AUTOMATED OFFLINE DYNAMIC PROCESSING IN THE AUDIO MASTERING PROCESS

Marcel Hilsamer,

Department of Digital Signal Processing
University of Kaiserslautern
Kaiserslautern, Germany
hilsamer@eit.uni-kl.de

Stephan Herzog,

Department of Digital Signal Processing
University of Kaiserslautern
Kaiserslautern, Germany
herzog@eit.uni-kl.de

ABSTRACT

Mastering audio is a complicated yet important step in music production. It is used for many purposes, an important one is to ensure a typical loudness for a piece of music within its genre. In order to automate this step we use a statistical model of the dynamic section. To allow a statistical approach we need to introduce some modifications to the compressor's side-chain or more precisely to its ballistics. We then develop an offline framework to determine compressor parameters for the music at hand such that the signal's statistic properties meet certain target properties, namely statistical central moments, which for example can be chosen genre specific. Finally the overall system is tested with songs which are available to us as unmastered, professionally mastered, and only compressed versions.

1. INTRODUCTION

Mastering audio is a complex task which requires an experienced sound designer. There is a huge amount of literature giving experience-based tips to sound designers on how to master mastering music. One of the best-known and most cited examples would be Bob Katz' "Mastering audio" [1].

An alternative approach to set the compressor parameters is the use of presets coming with today's software mastering tools like compressors and limiters. The drawback of these presets is, that they do not take into account the properties of the piece of music at hand, therefore a satisfying result can not be guaranteed.

Recently Giannoulis et. al. [2] proposed an automation of the compressors parameters based on the input signal's temporal behavior with only a single user controlled parameter, namely the compressor's threshold. Vickers [3] presented a method to automate a compressor based on the input signal statistics. His approach still has the need of the user controlled parameter threshold, as well as two target parameters, one to define the input-output relation and one to set a make-up gain, and of course the ballistics of the compressor.

We will present a new way to determine the control settings of the compressor depending on the material at hand and statistical determined target parameters without the need of any adjustments by the user. This will be done by means of the input signal statistics and an arbitrary reference, which could be the typical statistics of its genre. The statistical properties of music have been investigated in various ways. See for example [4], [5] and [6] which can be used as a starting point for more information on genre specific statistical properties, [7] gives an overview of statistical features which can be used to describe a compressor's behaviour.

In Section 2 we describe a classical compressor and its parameters. Section 3 introduces the statistical model of the dynamic section and defines some modification concerning the ballistics and signal feature detector of the compressor. This is used in Section 4 to present an offline procedure to estimate the compressor parameters needed to match certain target central moments based on the properties of the music at hand. This procedure will be evaluated in Section 5. Finally, Section 6 will summarise the proposed framework and its result and suggest fields of further research.

2. COMPRESSOR CONTROLS

There are countless compressor topologies, but all of them are using a signal path and a control path, the so called side-chain. This side-chain is typically realised in a feedforward structure. The principle block diagram is shown in Fig. 1, with the side-chain usually working on levels and consisting of four basic blocks

- a signal feature detector, like the Root Mean Square (RMS) or Peak detector to control the compressor based on the signal feature to be altered,
- the gain calculation based on an arbitrary input-output relationship, given as a characteristic curve,
- the ballistics, a gain smoothing stage with different time constants for rising and falling edges to reduce non-linear distortion on transient signals
- and finally the so called make-up gain, an increase of the calculated gain by a constant.

We will first describe these blocks in detail before we will discuss their contribution to the statistical model of the compressor. In the following we assume normalised pieces of music as input signals.

2.1. Signal feature detector

Classic analog compressors (or digital realisations of it) use either Peak or short time RMS detectors to control the compressor's characteristic [8]. This is done because of the simple realisation, not the musical or psychoacoustical meaning of these measures. With digital signal processing we can use arbitrary measures, like for example a complex loudness model with signal level dependent spectral weighting or a simplified model like in [3]. In order to keep the presented method a general framework we will use the term C_x for the input feature used in the side-chain. Analysing the compressors output y using the same detector yields the corresponding output feature C_y.

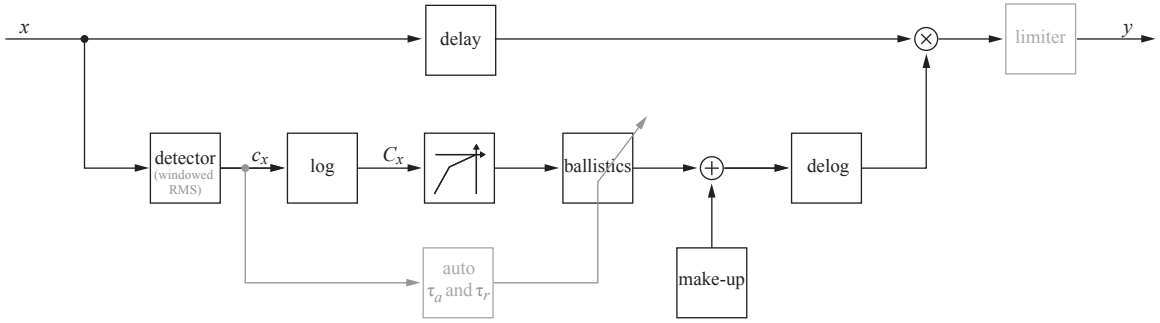


Figure 1: *principle block diagram of a compressor with suggested extensions (grey).*

2.2. Characteristic curve

The classical compressor characteristic is given by two parameters, namely the threshold (T) and the ratio (R). For levels below T the gain is 1, above T the slope of the characteristic is determined by the reciprocal of R. Fig. 2 illustrates the classical characteristic with $T = -30$ dB and $R = 3:1$ (solid) and the same characteristic with a make-up gain of $M = 20$ dB (dashed).

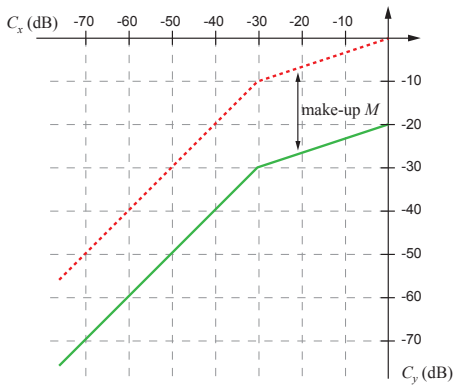


Figure 2: *classical characteristic curve of a compressor with (red,dashed) and without (green, solid) make-up gain.*

2.3. Attack and release times, the ballistics

The gain calculated based on the signal feature and the characteristic curve is usually smoothed to reduce non-linear distortion on transient signals and to suppress gain ripples produced by low frequencies. This is realised with different time constants for rising and falling gains, the attack time τ_a and the release time τ_r respectively. The user needs to select the attack and release times carefully which complicates the use of a compressor.

Katz [1] gives the following typical ranges: attack times ranging between 50 ms and 300 ms, common value 100 ms, release times lying between 50 ms and 500 ms, common value range between 150 ms and 250 ms. These values should be set suitable for the music's temporal behaviour, or in Katz' words: *"it's counterproductive to go against the natural dynamics of music"*.

Due to the ballistics, the gain reduction or the actual compression depends not only on the input signal's characteristic level C_x but also on its temporal characteristics.

2.4. Make-up gain

The make-up gain is an added constant (sometimes signal dependent and time-varying as e.g. in [9]) to raise the system's output gain. We will use this block to normalise the chosen signal characteristic. This means we do not want to reduce the maximum of C_x (say the RMS value), instead we want to amplify the lower (RMS) parts of the signal. In other words the make-up gain should bring the maximum output value to the maximum input value, for example 0 dBFS. This leads to

$$M = -T\left(1 - \frac{1}{R}\right) . \tag{1}$$

In Fig. 2 the dashed line shows the characteristic curve with a make-up gain according to equation (1) applied.

3. STATISTICAL MODEL OF A COMPRESSOR

The description of a compressor by means of signal statistics has recently been proposed by Shuttleworth [7]. He uses different signal features, i.e. inflection points as a peak measure, short and long term RMS as well as their partition into several frequency bands, to investigate the effects of a compressor by its Probability Density Function (PDF).

The statistic properties of any signal s is completely characterised by its PDF f_s. It is obvious that the compressor generates an output y with a PDF depending on the input's PDF and the compressor parameters. This holds of course for the PDF of any signal feature C_x (e.g. the RMS levels) used in the side-chain, namely f_{C_y} and f_{C_x} respectively, with

$$f_{C_y} = g\left[f_{C_x}, T, R, M\right] . \tag{2}$$

The influence of each block of the side-chain on f_{C_y} is now discussed to find an analytic expression for the function g.

The detector determines the signal feature to be altered. The static characteristic curve of the compressor is mainly responsible for the transformation of the PDF. This usually leads to a non-linear relation between input and output PDF that can be expressed analytically. The make-up gain just shifts the PDF by M and is

therefore a simple linear transformation. The ballistics however lead to a transformation which depends not only on the level of the input's signal feature C_x, but also on its temporal characteristics and thus can not be modelled appropriately in a statistical manner directly. Fig. 3 illustrates this effect in the time domain.

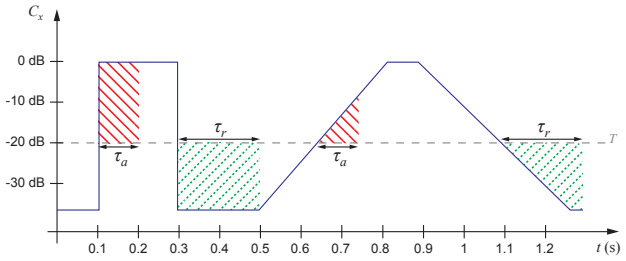


Figure 3: *Illustration of the ballistics in the time-domain, red (solid lines) areas are above T but not fully compressed due to $\tau_a = 100$ms, green (dashed lines) areas are below T but partly compressed due to $\tau_r = 200$ms.*

The red/green areas show the regions above/below threshold in which the gain does not follow the static characteristic due to the attack/release time. Their sizes are a direct measure for the variation from the static characteristic. By comparing the first red and green areas with the second ones it can be stated, that the faster the signal's feature cross the threshold the larger these areas are.

The ballistics behave like different low pass filters for descending and ascending gains with corner frequencies inverse proportional to the attack and release time and thus can not follow fast changes in C_x. The effect therefore can be lowered by simply adapting the attack and release time on the signal variation in time. We propose the use of automated ballistics based on recent research by Giannoulis et. al. [2]. In the following we will give a brief summary of their automated ballistics, extend their approach and show a computational efficient way to realise the proposed automation.

3.1. Auto ballistics

Giannoulis et. al. proposed an algorithm to ease the adjustment of the compressor parameters for the user. They automated the attack and release time based on the so called Spectral Flux (SF) of the input signal, which is a common tool for note onset detection. It sums the bins of the input short time spectrum in which the energy raises (positive half-wave rectifier) from one block to another, normalised by the total signal energy. This measure will be low for steady state signals and will show peaks for abrupt changes in pitch or transient level increase.

The basic concept of this approach is to use short response times for material consisting of strong transients and longer time constants for signals with constant levels over time. Giannoulis et. al. set the attack and release time inversely proportional to the SF. Listening tests with both, professionals and amateurs showed good results for this signal dependent automation of the ballistics [2].

This method seems a promising starting point to automate the ballistics in a musical sense as well as to lower the time dependency of the PDF transformation as described before. To do so, we will need to extend this approach to satisfy the following two conditions: First falling transients to automate the release time separately need to be included, whereas changes in pitch without significant changes in signal level should to be excluded in the measure. Second the measure should correlate strongly with the slope of level increase respectively decrease. Finally we will discuss the use of this measure with an arbitrary signal feature detector.

In [2] the SF alters the attack time as well as the release time which leads to good musical performance. Following this approach and taking into account that falling transients, i.e. an abrupt end of a sound, should alter the release time of a compressor, we propose to denote SF, as it is a measure for increasing energy or onset, by SF^+ and extend it by SF^- which sums falling energy bins (negative half-wave rectifier) or in other words is a measure for note offsets or falling transients.

By adding SF^+ and SF^- to SF^Σ the change of a note without a significant change in loudness will no longer be detected. A positive value will show an abrupt raise in signal level which can be used to scale the attack time of the compressor. A negative value indicates an abrupt fall in signal level and can therefore be used to scale the release time.

Following [2], for our simulations[1] we used a $N = 1024$ sample Fast Fourier Transformation (FFT) with a hann window and an overlap of $N/2$ to produce a value every 512th input sample (hop size $h = 512$) or every block k. Windowing the data prior to the FFT is important in order to reduce so called end-effects and therefore smooth the spectrum and thus the SF.

Fig. 4 (b)[2] illustrates SF^+, SF^- and SF^Σ for a sine-wave with abrupt changing level and frequency. It is clearly visible, that SF^Σ is a good measure to detect rising and falling transients in signal level.

In order to reduce the computational effort we will realise a similar measure in the time domain. As the introduced SF^Σ basically indicates the differences in signal energy between two adjacent blocks we can also use the differentiation of the length N RMS value of the windowed[3] input signal $\hat{x}$ calculated with a moving averager, namely

$$\Delta_{RMS}[k] = \sqrt{\sum_{i=0}^{N-1} \hat{x}^2[kh+i]} - \sqrt{\sum_{i=0}^{N-1} \hat{x}^2[(k-1)h+i]}\ . \quad (3)$$

Fig. 4 (c) illustrates this measure. It can be seen that the proposed measure is similar to SF^Σ. It even correlates more with the amount of signal energy increase respectively decrease. This is not surprising since SF was introduced to detect note onsets and not their levels, hence due to the normalisation by the block energy every change in level will be detected almost equally.

Δ_{RMS} satisfies the required modifications, namely positive values are proportional to the slope of rising transients, negative values are proportional to the slope of falling transients and changes in frequency are ignored. Using this measure to scale the ballistics of the compressor guarantees a signal dependent attack and release time which will smooth the gain accordingly to the temporal behaviour of the input signal.

The use of any signal feature detector other than RMS leads to the need of a different measure as the characteristic and of course

[1] With a sampling rate of $f_s = 48$kHz.

[2] The chosen test signal is equivalent to that in [2] for a better comparability of the results.

[3] Windowing in time domain calculations seems unusual, but the reduction of end-effects will help to suppress the RMS error due to the difference between integration time and the unknown signal's period. This leads to smoother, less oscillating results. A correction factor must be applied. For an overview of different windows and their correction factors see [10].

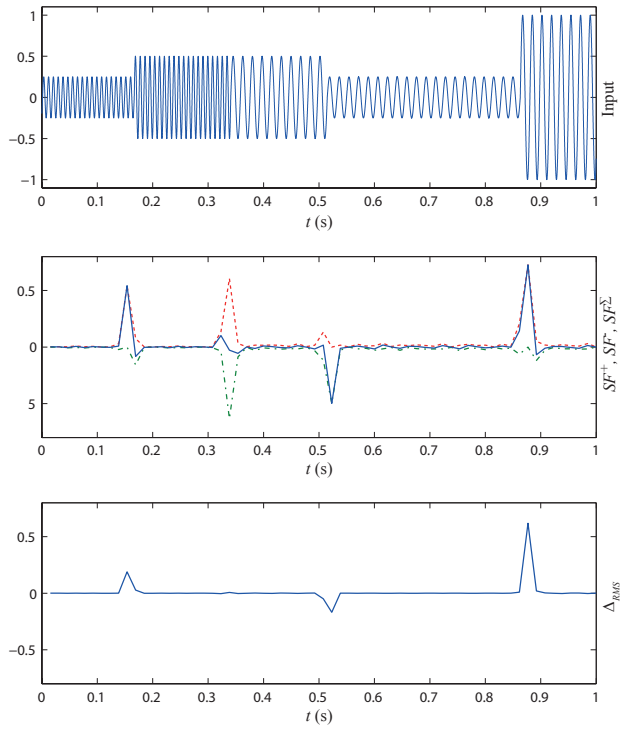


Figure 4: *(a) input signal, sine-wave with abrupt changing amplitude and frequency. (b) SF$^+$ (red, dashed), SF$^-$ (green, dash-dotted) and their sum (blue, solid) with an FFT length of $N = 1024$ using a hann window and 50% overlap. (c) Δ_{RMS} with block length $N = 1024$ using a hann window and 50% overlap.*

the ballistics will not work on the RMS values anymore. Therefore we will introduce the general measure

$$\Delta_{C_x}[k] = \hat{C}_x[kh] - \hat{C}_x[(k-1)h] \tag{4}$$

with again windowed input data $\hat{x}$.

In order to achieve approximately equal areas shown in Fig. 3, the actual attack and release times should be calculated block-wise by

$$\begin{aligned}\tau_a[k] &= \tau_{a_{max}}\left(1 - 2\max\left\{0, \Delta_{C_x}[k]\right\}\right)\\ \tau_r[k] &= \tau_{r_{max}}\left(1 + 2\min\left\{0, \Delta_{C_x}[k]\right\}\right).\end{aligned}$$

The maximum attack and release times should be set accordingly to typical values as given in Section 2.3, for example $\tau_{a_{max}} = 100$ms and $\tau_{r_{max}} = 200$ms.

Using this automation, we can simplify the temporal characteristic-dependent, non-linear characteristic of the compressor to a temporal characteristic-independent system with kindly smoothed non-linearity and therefore a statistical analysis becomes acceptable. The smoothing is because of the now signal dependent ballistics, or in other words signal independent error areas in Fig. 3 and is ignored in the following parameter estimation process as it would not modify the needed parameters significantly.

4. AUTOMATED COMPRESSOR USING THE STATISTIC MODEL

We assume similar statistic properties for pieces of mastered music within a specific genre. This assumption is based on genre-specific instrumentation and arrangements as well as a similar overall sound and loudness as sound designers usually let themselves be inspired by currently popular productions within the same genre. This leads to genre-specific PDFs for the RMS levels [5].

As a compressor can alter the PDF of an input signal, if the transformation of the input PDF f_{C_x} to the output f_{C_y} is known analytically, one could invert this transformation and thus the proper parameters needed to meet a genre specific target PDF could be determined explicitly.

The input signal feature distribution is given by f_{C_x} and can be approximated by the histogram of the levels of the detector's output. This distribution will be altered by the compressor, more precisely its characteristic. We will now formulate analytic expressions for this alteration with the use of a PDF transformation.

4.1. Transformation of the PDF due to a compressor and its inversion

A compressor with a characteristic curve based on threshold and ratio alters the detected signal characteristic (without taking the ballistic into account) to the output distribution

$$f_{C_y} = \begin{cases} R f_{C_x}\left[C_x = RC_y\right], & C_x \geq T,\ C_y \geq \frac{T}{R} \\ f_{C_x}\left[C_x = C_y - M\right], & C_x < T,\ C_y < \frac{T}{R}. \end{cases} \tag{5}$$

We now know how the compressor modifies the input PDF. With a genre typical target PDF, which could be the mean of a statistical relevant number of analysed mastered songs within a genre we can try to minimise the difference between this target PDF and f_{C_y} by adjusting R and T. An exact match in general is not possible as equation (5) does not arbitrarily modifies the input PDF. However it is possible to match certain (central or standardised) moments of the target PDF for example its mean and variance. Hence we will now analyse the transformation of these moments due to the compressor.

The i-th central moment $\mu_{i,y}$ is transformed to

$$\begin{aligned}\mu_{i,y} = &\int_{-\infty}^{T} \left(M + C_x - \mu_{1,y}\right)^i f_{C_x} dC_x \\ &+ \int_{T}^{\infty} \left(\frac{C_x}{R} - \mu_{1,y}\right)^i f_{C_x} dC_x .\end{aligned} \tag{6}$$

With the help of equation (6) it is possible to determine the central moments of the output of the compressor based on its input and parameters R and T. As the parameter T is part of the integral limits it is not possible to directly invert equation (6) to determine R and T to match certain target moments $\mu_{i,t}$. In addition the input signal PDF f_{C_x} is not known as an analytic expression and needs to be approximated by a histogram.

We define the cumulative sums over the product of C_x^i and the histogram $\hat{f}_{C_x}$ for all possible thresholds T as

$$m_i[T] = \sum_{l=C_{x_{min}}}^{T} l^i \hat{f}_{C_x}[l] . \tag{7}$$

This vector is an approximation for the i-th non central moment of the compressor input $\int_{-\infty}^{T} C_x^i f_{C_x} dC_x$ and can be used to evaluate the output central moments using simple vector additions. We will need two target moments to determine the two parameters R and T, in the following we will use the first two, namely the mean and variance.

The mean value is then approximated by

$$\mu_{1,y} = Mm_0[T] + m_1[T] + \frac{1}{R}\left(m_1[0] - m_1[T]\right) \tag{8}$$

and the variance to

$$\begin{aligned}\mu_{2,y} =& M^2 m_0[T] + 2Mm_1[T] + m_2[T] \\ &+ \frac{m_2[0] - m_2[T]}{R^2} - \mu_{1,y}^2 \ ,\end{aligned} \tag{9}$$

with $m_i[0]$ representing the i-th moment with $T = 0$dB, or in other words of the whole histogram of C_x.

By rearranging equation (8) to solve for R we obtain the vector

$$R_{\mu_1}[T] = \frac{Tm_0[T] - m_1[T] + \mu_{1,x}}{Tm_0[T] - m_1[T] + \mu_{1,t}} \tag{10}$$

containing the values for R which are needed to meet the target mean value.

By evaluating equation (9) at the points determined in (10) we obtain a vector containing all output variances $\mu_{2,y}\left[R_{\mu_1}[T]\right]$ for each parameter combination. Then the minimum of the squared difference of the variance $\mu_{2,y}$ of the output and its target $\mu_{2,t}$ determines the desired parameters T_{est} and R_{est}. In cases were an exact match in both moments is not possible with the use of just a compressor, this approach will lead to an exact match in mean $\mu_{1,y}$ and a minimum squared error in variance $\mu_{2,y}$. To ensure an exact match in variance and a minimum error in mean for all cases, equation (9) should be rearranged to solve for $R_{\mu_2}[T]$, then the minimum squared difference between $\mu_{1,y}$ and the target mean $\mu_{1,t}$ determines the desired parameters. The described procedure of course can easily be expanded to higher central moments or even standardised moments like the skewness or the kurtosis.

R_{μ_1} can contain negative values, as for higher thresholds, depending on the input signal and the target mean value, it will become impossible to match it with the make-up gain not letting C_y exceed 0 dBFS. Then a negative ratio R and therefore a negative slope of the characteristic above T will cause C_y to exceed 0 dBFS around $C_x = T$ and still meet 0 dBFS for $C_x = 0$ dBFs. This of course is not what we desire, so we will only use the range with positive values for R_{μ_1} to evaluate $\mu_{2,y}$.

4.2. Limiter to suppress overshoots due to attack time

Even though the make-up gain M is defined to meet 0dB at the output for a 0dB input, the non-zero attack time produces peaks of very short duration exceeding 0dB. In order to prevent the output signal from clipping these peaks have to be eliminated. This can be done with what is typically called a brick-wall limiter.

The use of such a limiter in the mastering process is a typical procedure as it can be used to make the piece of music at hand louder without changing its sound significantly. As distortion of short duration are nearly inaudible even gain reductions of several dB are possible for short peaks as long as the limiter's attack and release time are short enough [1]. We therefore propose the use of a limiter with very short attack and release time and a so called look ahead, an infinite ratio and a threshold set near to 0 dB. Our simulations show good results using $\tau_{l,a} = 0.5$ ms, $\tau_{l,r} = 50$ ms and $T_l = -0.1$ dB in terms of no clipping and no significant detoriation in the matching of the target mean and variance[4]. The limiter is shown as the last block of the framework in Fig. 1.

5. SIMULATION RESULTS

To evaluate the proposed automated compressor we first tested the method to determine the parameters R and T described in Section 4.1. Then we used the proposed automatic compressor with songs which are available to us in unmastered and professional mastered versions[5] in two scenarios: first to match the mean and variance of unmastered songs to those of the same songs processed with a classic compressor and second to match the mean and variance of these songs to those of the corresponding mastered version. Finally the results were validated in an informal listening test. For all simulations we solved the mean value for R to choose R and T to achieve the minimum squared error in variance.

By using a compressor with the proposed auto ballistics it should be possible to exactly reconstruct the chosen parameters R and T from the input and output PDF with the described parameter estimation method from section 4.1. In order to test this method we processed songs with a compressor employing the described windowed RMS detector and automatic ballistics using randomly chosen parameters R_{ref} and T_{ref}. The resulting versions were used to calculate the target moments for the parameter estimation process. The estimated parameters R_{est} and T_{est} were identical (with small variations due to the chosen histogram density) to the ones employed for generating the reference signal, as expected.

The complete automatic compressor was tested in the first scenario by using unmastered songs processed with a classic compressor, more precisely the one coming with Apple's Logic Pro, using typical settings according to [1] and with the build-in limiter activated, as the reference signal. Then the raw unmastered versions were processed with the framework and the moments of its output were compared to those of the reference signal. The results are shown in Table 1.

	song 1	song 2	song 3	song 4	song 5
$\mu_{1,t}$	-10.57	-14.47	-9.25	-18.47	-21.89
$\mu_{1,x}$	-12.55	-16.95	-12.36	-19.55	-26.76
$\mu_{1,y}$	-10.60	-14.50	-9.29	-18.47	-21.90
μ_{1,y_l}	-10.87	-15.01	-9.80	-19.15	-23.13
$\mu_{2,t}$	14.60	35.22	20.28	48.46	90.71
$\mu_{2,x}$	21.20	47.05	31.27	51.78	91.29
$\mu_{2,y}$	17.18	37.93	25.24	49.01	91.12
μ_{2,y_l}	17.12	37.90	24.96	49.01	91.10

Table 1: *Mean and variance values of input signals, their targets calculated from the compressed signals, the output moments without ($\mu_{1,y}$, $\mu_{2,y}$) and with limiter (μ_{1,y_l}, μ_{2,y_l}).*

It can clearly be seen, that the overall system shows a good matching of the mean values $\mu_{1,y}$ for each song. The variance of the output $\mu_{2,y}$ is kindly higher than its target, which is due to

[4] A small decrease in both, mean and variance, will occur (see Table 1).

[5] Song 1 - 3 from [11] and song 4 and 5 from the album 'Mind Meets Matter' by Claude Pauly.

the ballistics, leading to higher levels during attack and lower levels during release than modelled. This effect is of short duration and was therefore not audible in our tests. Furthermore this effect could be reduced by introducing an additive correction term to the target's variance, depending on $\tau_{a_{max}}$ and $\tau_{r_{max}}$. The discussed kind reduction of the moments due to the usage of the limiter is also clearly visible. This good overall matching is achieved although the parameters R and T of the automated compressor differ from those used to create the reference, which is not surprising as Logic's compressor and the automatic compressor employ different signal feature detectors.

In the second scenario we matched the unmastered versions to the mastered versions of the same songs. As mentioned before an exact match of the PDFs is not possible. This is mainly due to the possibly excessive use of a limiter (see Fig. 5 around -4dB), and in this case, in addition, the possible use of an equalizer and/or a multi-band compressor during the professional mastering process, as these effects can not be modelled by just a compressor. However the statistical moments of the output meet the targets in all our simulations very well (comparable to the results in Table 1). As an example, Fig. 5 shows the PDF of one of the songs for its unmastered and mastered version as well as the output of the automatic compressor. The target moments were $\mu_{1,t} = -9.89$ dB, $\mu_{2,t} = 16.80$, the resulting compressor parameters $T = -16.5$ dB and $R = 1.281$ and the resulting moments at the output were $\mu_{1,y} = -9.92$ dB and $\mu_{2,y} = 19.20$ before and $\mu_{1,y_l} = -10.28$ dB and $\mu_{2,y_l} = 18.96$ after limiting.

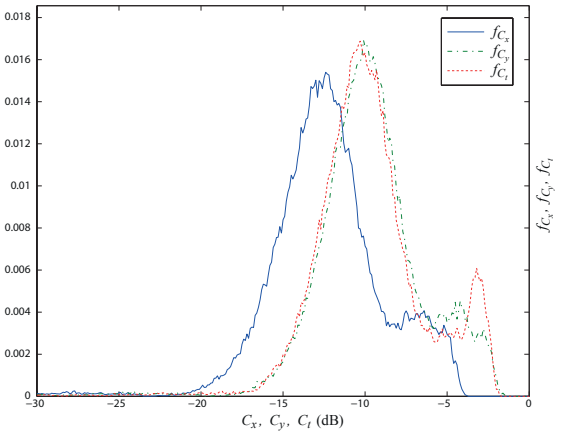


Figure 5: *PDF of the RMS values of the input (blue, solid), the output (green, dash-dotted) and the target (red, dashed).*

Finally, in order to evaluate the overall performance of the system we validated the results in an informal listening test, comparing the target wave file with the automatic compressor's output. In all examples there was no significant change in the overall loudness between both versions, which is not surprisingly due to the use of an RMS detector in combination with the relative exact match in its mean. In addition, no pumping or any other audible distortions were present, which indicates a good working automation of the ballistics. Finally the overall sound of the songs processed with the automatic compressor, besides some spectral differences compared to the mastered versions due to the use of an equalizer or multi-band compressor, was similar to that of their targets.

6. CONCLUSION

We presented a method to determine the parameters R and T for a compressor to ensure its output, based on the piece of music at hand, to meet certain target moments. The only modifications to a classic compressor needed for this statistical approach in mastering audio was automating the ballistics in the compressor and the use of a simple brick-wall limiter to eliminate overshoots due to the ballistics. In total this leads to a framework to automate the audio mastering process. With the use of target moments which differ significantly between different genres the proposed framework is able to match these moments and therefore a genre specific loudness and sound automatically.

Further research should focus on the identification of the most significant statistical moments to use as targets. In addition the framework can easily be extended to an automated multiband compressor by using several paths with different bandpass filters and an instance of the proposed compressor in parallel, which will help to take a genre typical spectral distribution during automated mastering into account. A useful starting point for getting proper multiband target moments or even PDFs can be [4] or [7].

7. REFERENCES

[1] Bob Katz, *Mastering audio: the art and the science*, Taylor & Francis, 2007.

[2] Dimitrios Giannoulis, Michael Massberg, and Joshua D. Reiss, "Parameter automation in a dynamic range compressor," *J. Audio Eng. Soc*, vol. 61, no. 10, pp. 716–726, 2013.

[3] Earl Vickers, "Automatic long-term loudness and dynamics matching," in *Audio Engineering Society Convention 111*, Nov 2001.

[4] Goran Ferenc, "Dynamic properties of musical signals in genre ethnic electronica," in *INFOTEH JAHORINA Vol. 11*, 2012.

[5] Miomir Mijic, Drasko Masovic, Dragana Sumarac Pavlovic, and Milan Petrovic, "Statistical properties of music signals," in *Audio Engineering Society Convention 126*, May 2009.

[6] Patrick Agostini, "Investigation of statistical properties of music signals," Bachelor's thesis, Dept. of Digital Signal Processing, University of Kaiserslautern, Germany, 2014.

[7] Tim Shuttleworth, "Evaluation of dynamics processors effects using signal statistics," in *Audio Engineering Society Convention 135*, Oct 2013.

[8] Guy W. McNally, "Dynamic range control of digital audio signals," *J. Audio Eng. Soc*, vol. 32, no. 5, pp. 316–327, 1984.

[9] Jacob A. Maddams, Saoirse Finn, and Joshua D. Reiss, "An autonomous method for multi-track dynamic range compression," in *Proceedings of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, 2012, vol. 15.

[10] F.J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, Jan 1978.

[11] Analog dimension - mastering studio, "Audio samples," http://www.analogdimension.com/audiosamples.html.

REVISITING IMPLICIT FINITE DIFFERENCE SCHEMES FOR 3-D ROOM ACOUSTICS SIMULATIONS ON GPU

Brian Hamilton, Stefan Bilbao,*
Acoustics and Audio Group,
University of Edinburgh
first.lastname@ed.ac.uk

Craig J. Webb,
Oxford e-Research Centre
University of Oxford
craig.webb@oerc.ox.ac.uk

ABSTRACT

Implicit finite difference schemes for the 3-D wave equation using a 27-point stencil on the cubic grid are presented, for use in room acoustics modelling and artificial reverberation. The system of equations that arises from the implicit formulation is solved using the Jacobi iterative method. Numerical dispersion is analysed and computational efficiency is compared to second-order accurate 27-point explicit schemes. Timing results from GPU implementations demonstrate that the proposed algorithms scale over their explicit counterparts as expected: by a factor of $M + 2$, where M is a fixed number of Jacobi iterations (eight can be sufficient in single precision). Thus, the accuracy of the approximation can be improved over explicit counterparts with only a linear increase in computational costs, rather than the quartic (in operations) and cubic (in memory) increases incurred when oversampling the grid. These implicit schemes are advantageous in situations where less than 1% dispersion error is desired.

1. INTRODUCTION

Room acoustics simulations are important for the purposes of auralization and artificial reverberation. There are many models and techniques used in room acoustics simulations; see [1, 2] for a review. One popular starting point for room acoustics modelling is the second-order scalar wave equation with impedance boundary conditions [3]. This model problem can be discretised with finite difference (FD) operators on regular spatial grids, and solutions can be approximated through explicit (leapfrog) time integration [4] at a sample rate of choice (e.g. 44.1 kHz). Explicit time-stepping FD methods have been used extensively in the literature for simulating room acoustics [5, 6, 7] in various equivalent formulations [8, 9, 10, 11].

FD methods can be computationally expensive for large 3-D spaces due to the fact that the solution is approximated for the entire domain at each time-step. Furthermore, numerical dispersion affects the approximation, to a large degree, in high frequencies. This may require that the spatial grid be oversampled, which incurs cubic increases in memory usage and quartic increases in the operation count. Explicit schemes are well-suited to implementation on graphics processing units (GPU), allowing for real-time low-frequency [12] and offline full-bandwidth applications [13].

Numerical dispersion can be improved by employing implicit generalisations of explicit schemes [14, 15], however, implicit schemes require a linear system to be solved at each time-step. This extra burden at each time-step can be alleviated somewhat when the implicit scheme allows for an alternating direction implicit (ADI) decomposition [16, 15], since the overall system in 3-D ADI schemes can be decomposed into three tridiagonal systems that can be solved efficiently with the Thomas algorithm [16]. However, the Thomas algorithm is serial in nature, so it is not easily parallelised. Furthermore, the formulation of impedance boundary conditions that are compatible with the ADI decomposition and the Thomas algorithm seems to be an open problem [7]. On the other hand, simple iterative methods [17] can be employed to tackle the implicit system, free from ADI constraints. The Jacobi method is a simple iterative method whose iterations reduce to sparse matrix-vector multiplications (SpMV) that are easily parallelised on a GPU. The purpose of this paper is then to revisit implicit schemes in the context of the Jacobi method and identify schemes that are suitable for room acoustics applications and GPU implementations.

This paper is laid out as follows. The model problem is introduced in Section 2, followed by the implicit finite difference schemes in Section 3 and conditions for stability in Section 4. The Jacobi method is described in Section 5, and optimal parameters for the implicit schemes are chosen in Section 6. Numerical dispersion and computational efficiency are analysed in Section 7. In Section 8, numerical experiments are conducted in order to validate the implicit schemes, check convergence rates for the Jacobi solve, and test the stability of the proposed schemes in finite precision arithmetic. Section 9 presents timing results from CUDA implementations of the implicit schemes on an Nvidia Tesla K20 GPU card, followed by conclusions and future work in Section 10.

2. MODEL PROBLEM

2.1. Initial and boundary value problem

The 3-D wave equation can be written as

$$\Box\Psi := \partial_t^2\Psi - c^2\Delta\Psi = 0\,. \tag{1}$$

Here, t is time and $t \in \mathbb{R}^+$, $\boldsymbol{x} := (x, y, z) \in \mathbb{R}^3$, c is the wave speed, assumed to be a constant, and Δ is the 3-D Laplacian operator, $\Delta := \partial_x^2 + \partial_y^2 + \partial_z^2$. The box symbol ($\Box$) represents the d'Alembert operator and the scalar field $\Psi = \Psi(t, \boldsymbol{x})$ represents the acoustic velocity potential [3]. Two initial conditions, $\Psi(0, \boldsymbol{x})$ and $\partial_t\Psi(0, \boldsymbol{x})$, are required to complete the initial value problem (IVP).

For the boundary value problem (BVP), let $\mathcal{V}$ denote a closed 3-D volume and $\partial\mathcal{V}$ its boundary. Frequency-independent lossy boundaries can be written as

$$\boldsymbol{n}\cdot\nabla\Psi = (\gamma/c)\partial_t\Psi\,, \quad \boldsymbol{x} \in \partial\mathcal{V}\,, \tag{2}$$

where γ represents the *specific acoustic admittance* with $\gamma \in \mathbb{R}$, $\gamma \geq 0$ and where $\boldsymbol{n}$ is the outward normal vector at $\boldsymbol{x} \in \partial\mathcal{V}$.

* This work was supported by the European Research Council, under grant StG-2011-279068-NESS, and by the Natural Sciences and Engineering Research Council of Canada.

These become first-order absorbing boundary conditions of the Engquist-Majda type for $\gamma = 1$. When $\gamma = 0$ the condition (2) is a homogeneous Neumann (lossless) boundary condition.

3. AN IMPLICIT FINITE DIFFERENCE SCHEME

3.1. Discretising time and space

Time can be discretised by restricting t to the grid of points $\mathbb{T} := \{nk, n \in \mathbb{Z}^+\}$, where k is the time-step, and the spatial domain can be discretised using a cubic grid: $\mathbb{G} := h\mathbb{Z}^3$. The finite spatial grid to consider can then be written as $\overline{\mathbb{G}} := \mathbb{G} \cap \mathcal{V}$. For the purposes of this paper, the closed volume of interest will be a box-shaped room. Furthermore, it will be assumed, for convenience and comparison with published literature [7], that the "boundary nodes" of the grid are precisely on the boundary $\partial\mathcal{V}$.

3.2. Difference operators

Let $u(t, \boldsymbol{x})$, which will be restricted to $\mathbb{T} \times \mathbb{G}$ or $\mathbb{T} \times \overline{\mathbb{G}}$, represent an approximation to the solution of interest $\Psi(t, \boldsymbol{x})$. A time-shift operator may be defined as

$$e_{t\pm}u := u(t \pm k, \boldsymbol{x}) \,, \tag{3}$$

and the following abbreviation will be employed throughout this paper: $u^{\pm} := u(t \pm k, \boldsymbol{x})$. Centered time-difference operators can be written as

$$\delta_{t\cdot} := (e_{t+} - e_{t-})/(2k) = \partial_t + O(k^2) \,, \tag{4a}$$

$$\delta_{tt} := (e_{t+} - 2 + e_{t-})/k^2 = \partial_t^2 + O(k^2) \,. \tag{4b}$$

A parameterised 27-point discrete Laplacian (stencil) can be defined on the cubic grid as

$$\delta_\Delta u := \sum_q \frac{6\alpha_q}{|\Omega_q| q h^2} \sum_{\boldsymbol{v} \in \Omega_q} (u(t, \boldsymbol{x}+\boldsymbol{v}h) - u(t, \boldsymbol{x})) = \Delta u + O(h^2) \,, \tag{5}$$

where $q \in \{1, 2, 3\}$, $\Omega_q := \{\boldsymbol{x} \in \mathbb{Z}^3 : \|\boldsymbol{x}\|^2 = q\}$, where $|\Omega_q|$ denotes the cardinality of the set Ω_q, and $\boldsymbol{\alpha} := (\alpha_1, \alpha_2, \alpha_3) \in \mathbb{R}^3$. $\sum_q \alpha_q = 1$ is required for consistency. The 27-point stencil vectors are displayed in Fig. 1.

3.3. Difference scheme for IVP

An implicit finite difference scheme for (1) can now be written as

$$\delta_\square u := \left(1 + \beta h^2 \delta_\Delta\right) \delta_{tt} u - c^2 \delta_\Delta u = 0 \,, \quad (t, \boldsymbol{x}) \in \mathbb{T} \times \mathbb{G} \,, \tag{6}$$

where $\beta \in \mathbb{R}$ is a free parameter. The scheme is consistent since $\delta_\square u \to \square u$ as $h \to 0$ for a fixed Courant number $\lambda := ck/h$. Starting from the two known (or approximated) values $u(0, \boldsymbol{x})$ and $u(k, \boldsymbol{x})$ determined by the initial conditions, the unknown variable u^+ is related to the two previous states by

$$(1 + \beta\delta_\Delta^h) u^+ = ((\lambda^2 + 2\beta)\delta_\Delta^h + 2)u - (1 + \beta\delta_\Delta^h) u^- \,, \tag{7}$$

where $\delta_\Delta^h := h^2 \delta_\Delta$. The unknown variable cannot be isolated algebraically unless $\beta = 0$, in which case the scheme is explicit. For $\beta \neq 0$ the scheme is implicit, and a linear system of equations must be solved at each time-step. This family of implicit schemes generalises the 27-point compact explicit schemes analysed in [7]. The operator δ_Δ^h expressed in a similar notation to that used in [7] can be found in the Appendix.

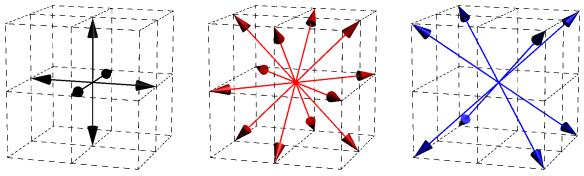


Figure 1: Stencil vectors for δ_Δ: Ω_1 (black), Ω_2 (red), Ω_3 (blue)

3.4. Matrix update for BVP

Imposing the boundary condition (2) reduces (7) to a finite system of equations which can be written as a matrix update. The approximation for the BVP at a particular time t can be written as the $N \times 1$ vector $\mathbf{u}$ with the values of u for $\boldsymbol{x} \in \overline{\mathbb{G}}$ ($N = |\overline{\mathbb{G}}|$). Similarly, $\mathbf{u}^\pm$ is a vector of $u^\pm$ values. The operator ∂_t in the lossy case (2) can be discretised with $\delta_{t\cdot}$ and the spatial derivatives are approximated with centered spatial differences, following [7]. The update equation in matrix form becomes

$$(\gamma\lambda\mathbf{Q} + \mathbf{I} + \beta\mathbf{L})\mathbf{u}^+ = ((\lambda^2 + 2\beta)\mathbf{L} + 2\mathbf{I})\mathbf{u} + (\gamma\lambda\mathbf{Q} - \mathbf{I} - \beta\mathbf{L})\mathbf{u}^- \,, \tag{8}$$

where $\mathbf{L}$ is the $N \times N$ Laplacian matrix corresponding to δ_Δ^h with discretised Neumann conditions, $\mathbf{I}$ is the $N \times N$ identity matrix, and $\mathbf{Q}$ is a non-negative diagonal matrix. Constructions for the matrices $\mathbf{L}$ and $\mathbf{Q}$ are given in the Appendix. This matrix update encapsulates the point-wise explicit updates presented in [7] for interior, wall, edge, and corner nodes, in the special case of frequency-independent boundaries.

4. NUMERICAL STABILITY

4.1. Stability for the IVP

First we consider stability conditions for the initial value problem. The recursion in (7) must be numerically stable for $\|u - \Psi\|_h \to 0$ as $h \to 0$ (for λ fixed) by the Lax-Richtmyer theorem, where $\|f\|_h$ denotes the spatial L^2-norm of $f(\boldsymbol{x})$ on $\mathbb{G}$ or $\overline{\mathbb{G}}$. Stability conditions for (7) can be found by taking the Z-transform in time and the Fourier transform in space [18]. After some cancellation we obtain the following quadratic in $z \in \mathbb{C}$:

$$(1 - 4\beta\Lambda)z + 4\Lambda(\lambda^2 + 2\beta) - 2 + (1 - 4\beta\Lambda)z^{-1} = 0 \,, \tag{9}$$

where Λ is the Fourier symbol of the operator $-\frac{1}{4}\delta_\Delta^h$. Solving for the roots of the quadratic (9) it can be shown [8] that $|z| \leq 1$ as long as

$$\left| \frac{4\Lambda(\lambda^2 + 2\beta) - 2}{1 - 4\beta\Lambda} \right| \leq 2 \,, \tag{10}$$

given that Λ is non-negative, which is satisfied when

$$-2\alpha_1 \leq \alpha_2 \leq 2\alpha_1 + 1 \,. \tag{11}$$

Condition (10) then simplifies to the following

$$\beta < \frac{1}{4\Lambda_{\max}} \,, \quad \lambda \leq \lambda_{\max} := \sqrt{\frac{1}{\Lambda_{\max}} - 4\beta} \,, \tag{12}$$

where $\Lambda_{\max} := \max_{\boldsymbol{\xi}} \Lambda$ for the spatial frequencies $\boldsymbol{\xi} \in \mathbb{R}^3$. We can extract $\Lambda_{\max}$ from previous studies [8] since this must reduce to the explicit case when $\beta = 0$. We have then

$$\Lambda_{\max} = \max(1, 2\alpha_1 + \alpha_2, 2\alpha_1 - \alpha_2 + 1) \,. \tag{13}$$

Note, the stability conditions allow linear growth in the solution, but this is valid since linear growth is permitted in the underlying system [18].

4.2. Stability for the BVP

Stability conditions for the lossless boundary value problem are straightforward to obtain using the matrix method [18]. Using the ansatz $\mathbf{u} = z\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is an eigenvector of $\mathbf{L}$, we get, analogous to (9), the following quadratic in z with matrix coefficients:

$$z(\mathbf{I}+\beta\mathbf{L})\boldsymbol{\phi} - ((\lambda^2+2\beta)\mathbf{L}+2\mathbf{I})\boldsymbol{\phi} + z^{-1}(\mathbf{I}+\beta\mathbf{L})\boldsymbol{\phi} = 0\,. \quad (14)$$

This can be reduced to a set of decoupled scalar equations, and thus, we can obtain a sufficient condition for stability in terms of the spectrum of $\mathbf{L}$. Comparing with (9) we can see that (12) is sufficient for stability as long as $\mathbf{L} \leq 0$ (negative semi-definite) and $\rho(\mathbf{L}) \leq 4\Lambda_{\max}$, where $\rho(\mathbf{L})$ denotes the spectral radius of $\mathbf{L}$. The first condition on $\mathbf{L}$ is easily verified using Gerschgorin's theorem [17]. That the condition $\rho(\mathbf{L}) \leq 4\Lambda_{\max}$ is satisfied, with $\mathbf{L}$ as defined in the Appendix, follows from stability in the explicit case [7].

A matrix-type stability analysis becomes more difficult after including the additional matrix $\mathbf{Q}$, with $\gamma > 0$ for lossy boundaries, because the matrix coefficients of the resulting quadratic equation no longer commute. It is possible to show, through the use of energy techniques [19, 10] or by investigating reflection coefficients at the boundaries [7], that the lossy case is stable as long as the lossless case is stable and the boundaries remain passive ($\gamma \geq 0$). A detailed proof is left out for brevity.

5. SOLVING THE LINEAR SYSTEM

5.1. Jacobi method

To solve the linear system of equations with the Jacobi method we first write (8) in the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, where

$$\mathbf{A} = (\gamma\lambda\mathbf{Q} + \mathbf{I} + \beta\mathbf{L})\,, \quad \mathbf{x} = \mathbf{u}^+\,, \quad (15a)$$

$$\mathbf{b} = ((\lambda^2+2\beta)\mathbf{L} + 2\mathbf{I})\mathbf{u} + (\gamma\lambda\mathbf{Q} - \mathbf{I} - \beta\mathbf{L})\mathbf{u}^-\,. \quad (15b)$$

Next, we use the matrix splitting $\mathbf{A} = \mathbf{D} - \mathbf{N}$ where $\mathbf{D}$ is a diagonal matrix with just the main diagonal of $\mathbf{A}$. Starting from any initial guess $\mathbf{x}^0$ (a good choice is $\mathbf{x}^0 = \mathbf{u}$), the Jacobi iterative solve proceeds with

$$\mathbf{x}^{n+1} = \mathbf{H}\mathbf{x}^n + \mathbf{b}'\,, \quad (16)$$

where $\mathbf{H} = \mathbf{D}^{-1}\mathbf{N}$ is the iteration matrix (sparse), $\mathbf{b}' = \mathbf{D}^{-1}\mathbf{b}$ and where the superscript n on $\mathbf{x}^n$ denotes the nth iteration ($n \geq 0$). Note that $\mathbf{b}'$ only needs to be computed once per time-step. The entire iterative solve can be accomplished with only four states stored in memory since the space in memory that is used to store $\mathbf{u}^-$ can be overwritten after $\mathbf{b}'$ has been calculated. This Jacobi solve requires two SpMVs to compute $\mathbf{b}'$ and M subsequent SpMVs, where M is the number of iterations. Thus, the increase in operations over the explicit case is a factor of $M + 2$. The memory increase over the explicit case is a factor of two.

The iterative solve can be halted when the following condition on the relative error is satisfied:

$$\frac{\|\mathbf{b} - \mathbf{A}\mathbf{x}^{n+1}\|_h}{\|\mathbf{b}\|_h} \leq E\,, \quad \|\mathbf{b}\|_h > 0\,, \quad (17)$$

where E is some threshold, such as IEEE 754 single precision machine epsilon, $\varepsilon_{\mathrm{s}} \approx 1.2 \times 10^{-7}$, or double precision machine epsilon, $\varepsilon_{\mathrm{d}} \approx 2.2 \times 10^{-16}$. Calculating the relative residual requires one additional SpMV per iteration, as well as the calculation of two discrete norms.

It is worth pointing out that while we use a matrix representation to illustrate the iterative method, a practical implementation does not require construction or storage of the matrices involved. For practical implementations, one can 'unroll' each SpMV into a (parallelisable) for-loop, as in the explicit case [13]. In fact, the explicit case is expressed by a single Jacobi iteration ($\beta = 0 \Rightarrow \mathbf{H} = 0$). The matrices involved are sparse and have entries that are mostly constant or zero along the diagonals, and the non-zero entries change only for boundary nodes. The storage of these constants is negligible. Point-wise updates can be extracted from the matrices in the Appendix, or derived from the explicit case in [7], so they are left out for brevity.

5.2. Convergence of the Jacobi method

The Jacobi iterations will converge from any initial guess $\mathbf{x}^0$ as long as the matrix $\mathbf{A}$ is *diagonally dominant* [17]. For a diagonally dominant $\mathbf{A}$, in the lossless case, we require that

$$\left|1 - 6\beta\sum_q \frac{\alpha_q}{q}\right| \geq 6\sum_q \frac{|\beta\alpha_q|}{q}\,. \quad (18)$$

If we assume that $\alpha_q \geq 0$ then this reduces to

$$|\beta| < \frac{1}{12}\,, \quad (19)$$

and in the general case $|\beta|$ has to be sufficiently small. By examining $\mathbf{L}$ it can be seen that the rows pertaining to boundary nodes will not change (19). This is also left out for brevity.

It can be shown that with each iteration the residual decreases by a factor of approximately $1/\rho(\mathbf{H})$ [17], and using Gerschgorin's theorem the following bound on $\rho(\mathbf{H})$ can be obtained:

$$\rho(\mathbf{H}) \leq \Upsilon\,, \quad \Upsilon := \frac{6\sum_q \frac{1}{q}|\beta\alpha_q|}{\left|1 - 6\beta\sum_q \frac{1}{q}\alpha_q\right|}\,. \quad (20)$$

Thus, we can neglect the residual calculation and fix the number of iterations to $M = \lceil -\log_{10}(E)/\eta \rceil$ or $M = \lfloor -\log_{10}(E)/\eta \rfloor$, where $\eta := -\log_{10}(\Upsilon)$. The parameter η represents, approximately, the number of additional digits of relative accuracy obtained with each iteration.

6. ISOTROPIC AND FOURTH-ORDER SCHEMES

To reduce the space of free parameters let us introduce some additional constraints. In the interest of isotropic error we can impose the following constraint

$$\alpha_2 = 4/3 - 2\alpha_1\,, \quad (21)$$

with which we get

$$\delta_\Delta u = \Delta u + \frac{h^2}{12}\Delta^2 u + O(h^4)\,. \quad (22)$$

The error will be isotropic (direction-independent) up to the $O(h^4)$ term since the (isotropic) biharmonic operator Δ^2 appears in the

$O(h^2)$ term. Through the use of modified equation methods [14], it is straightforward to arrive at the condition

$$\lambda^2 = 1 - 12\beta\,, \tag{23}$$

to have a fourth-order local truncation error for the IVP

$$\delta_\Box u = \Box u + O(h^4)\,. \tag{24}$$

Under the isotropy constraint (21) the stability condition (12) reduces to

$$\lambda_{\max,\beta,\alpha_1} = \begin{cases} \sqrt{3/4 - 4\beta} & 1/12 \le \alpha_1 \le 5/12 \\ \sqrt{3/(12\alpha_1 - 1) - 4\beta} & 5/12 < \alpha_1 \end{cases}, \tag{25}$$

and the constraints (11) and (19) reduce the parameter space of stable fourth-order schemes to the following

$$1/12 \le \alpha_1 \le 5/12\,, \tag{26}$$

with $\lambda = \sqrt{5/8} \approx 0.79$ and $\beta = 1/32$. Finally, we can optimise α_1 with respect to η. Using (20) it can be shown that

$$\eta \in [\log_{10}(19/3), \log_{10}(7)] \approx [0.802, 0.845]\,,$$

for the region defined in (26). The optimal value, $\eta = 0.845$, is given by $\alpha_1 = 1/3$, which corresponds to a scheme with a 19-point stencil ($\alpha_3 = 0$).

7. NUMERICAL DISPERSION

At this point, we can analyse the numerical dispersion of the schemes that are suitable candidates for the Jacobi iterative solve. To further reduce the space of free parameters, we will restrict our attention to two cases: $\alpha_1 = 1/3$ and $\alpha_1 = 5/12$. The former leads to an isotropic 19-point stencil, and the latter is an isotropic 27-point stencil. The resulting finite difference schemes are implicit generalisations of the "IISO1" and "IISO2" (interpolated isotropic) explicit schemes [15, 7].

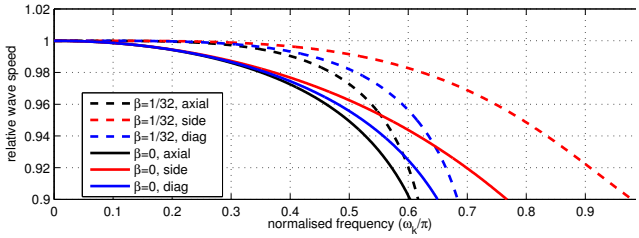
In order to analyse dispersion it helps to define a normalised spatial frequency $\boldsymbol{\xi}_h := \boldsymbol{\xi} h$ and a normalised temporal frequency $\omega_k := \omega k$. We can then write $\Lambda(\boldsymbol{\xi}_h)$ as

$$\Lambda(\boldsymbol{\xi}_h) = \sum_q \frac{3\alpha_q}{|\Omega_q| d} \sum_{\boldsymbol{v}\in\Omega_q} \sin^2(\boldsymbol{\xi}_h \cdot \boldsymbol{v}/2)\,, \tag{27}$$

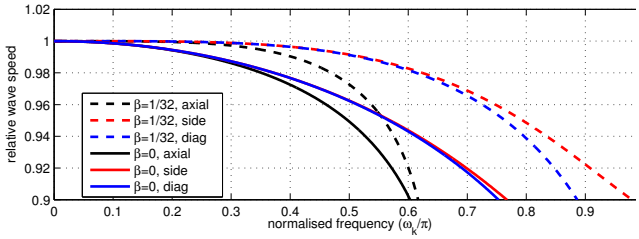
and the relative numerical wave speed (ideally unity), also known as the *dispersion coefficient*, is defined as

$$\nu(\boldsymbol{\xi}_h) := \frac{\omega_k(\boldsymbol{\xi}_h)}{\lambda|\boldsymbol{\xi}_h|}\,, \quad \omega_k(\boldsymbol{\xi}_h) := 2\arcsin\left(\lambda(\Lambda^{-1} - 4\beta)^{-\frac{1}{2}}\right)\,, \tag{28}$$

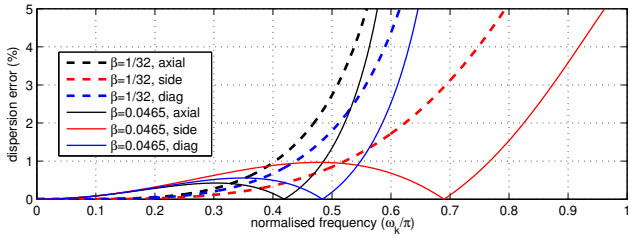
for $\omega_k \in (0, \pi]$ and $\boldsymbol{\xi}_h \in \mathbb{B}$, where $\mathbb{B}$ is the *wavenumber cell* of the grid, which is a cube centered at zero with sides of length 2π. Furthermore, by inverting the dispersion relation (in the region where it is surjective) we can plot the numerical wave speed as a function of spherical coordinates, where the radial coordinate represents the temporal frequency ω_k and where the two polar angles represent a plane-wave direction of propagation in $\mathbb{R}^3$ [15]. The wave speed errors can be seen in Fig. 2a for the schemes $\alpha_1 = 1/3$ and $\beta \in \{0, 1/32\}$ along the axial (center to face-center of $\mathbb{B}$), side diagonal (center to edge-center), and diagonal (center to vertex) directions; these are the directions in which the extreme cases



(a) Relative wave speed for $\alpha_1 = 1/3$ and $\beta \in \{0, 1/32\}$



(b) Relative wave speed for $\alpha_1 = 5/12$ and $\beta \in \{0, 1/32\}$



(c) Dispersion error for $\alpha_1 = 1/3$ and $\beta \in \{1/32, 0.0465\}$

Figure 2: Numerical dispersion for various schemes

are generally found [7]. Fig. 2b shows the dispersion coefficient along the same directions for the scheme with $\alpha_1 = 5/12$ and $\beta \in \{0, 1/32\}$. It can be seen from these figures that the fourth-order implicit schemes give improvements over their second-order explicit counterparts in each direction.

The fourth-order condition (23) can also be ignored in order to find a scheme optimised for some fixed amount of dispersion error that can be tolerated, where *dispersion error* is defined as $|1 - \nu| \times 100\%$. For example, the parameter $\beta = 0.0465$ is a good choice for a 1% dispersion error tolerance. The dispersion errors for $\alpha_1 = 1/3$ and $\beta \in \{1/32, 0.0465\}$ are shown in Fig. 2c. More optimised parameters will be given shortly. Note, the relative wave speeds are plotted only up to a 5% or 10% dispersion error for the purposes of showing detail. The minimum directional cutoff frequencies, above which the modal density will be incorrect, are not seen in the figures, but they are listed in Table 1 ($\omega_{k,\text{cutoff}}$). The cutoff frequencies for the implicit schemes are near to $\omega_{k,\text{cutoff}}$ for the IISO1 (or IISO2) explicit scheme, which is $(2/3)\pi$ rad/sample [7].

7.1. Relative computational efficiency

One can achieve any level of accuracy in the dispersion error up to any desired temporal frequency (in Hz) with any (convergent) scheme simply by reducing the spatial-step (for a fixed Courant number), due to consistency with the model equation. Of course, oversampling the grid incurs *cubic* increases in memory usage and *quartic* increases in the operation count, so this quickly becomes an impossible strategy for simulating large spaces. Nevertheless,

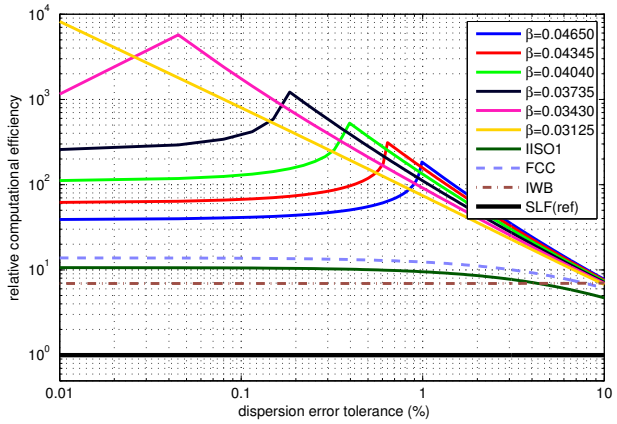


Figure 3: Relative computational efficiencies for various implicit schemes with $\alpha \in \{1/3, 5/12\}$ and explicit schemes, with the simplest scheme (SLF) as reference. Table 1 lists relative comparisons where IISO1 and FCC explicit schemes are reference schemes, also taking into account Jacobi iterations.

this strategy exists, so we must take into account some measures of computational costs in order to determine whether these implicit schemes are more or less effective than their simpler explicit counterparts with oversampled grids. First we will consider the spatiotemporal density of points required to achieve a certain dispersion error globally, and then we will include additional costs from the iterative solve.

As in [20, 7, 21], we start by investigating the relative computational efficiency (RCE), which is defined as the spatiotemporal density of points ($\mathbb{T} \times \mathbb{G}$) necessary to keep the dispersion error below some tolerance level, relative to that required by some reference scheme [20]. Thus, the RCE of a scheme, with some chosen reference scheme, is dimensionless and is a function of the dispersion error tolerance. As in [15, 7] we use the simplest explicit scheme [4], also known as the "standard leapfrog" (SLF), as a reference. The RCEs for the cases $\alpha_1 \in \{1/3, 5/12\}$ with various choices of β are shown in Fig. 3 for a 0.01-10% dispersion error tolerance. Along the axial directions, the schemes with $\alpha_1 \in \{1/3, 5/12\}$ have the same dispersion (worst-case), so the RCEs for the implicit schemes with $\alpha_1 = 5/12$ are the same as those with $\alpha_1 = 1/3$. The explicit IISO2 ($\alpha_1 = 5/12, \beta = 0$) scheme is also equivalent to IISO1 ($\alpha_1 = 1/3, \beta = 0$) in terms of its RCE.

Also included in Fig. 3 are the 13-point face-centered cubic (FCC) explicit scheme ($\alpha_1 = 0, \alpha_2 = 1$) (on its native grid [21]) and the 27-point "interpolated wideband" (IWB) explicit scheme ($\alpha_1 = 1/4, \alpha_2 = 1/2$), for comparison with existing literature [7, 21].[1] As can be seen in Fig. 3, the implicit schemes have higher RCEs than their explicit counterparts and, in particular, the fourth-order scheme ($\beta = 0.03125$) becomes exponentially (linear on a log scale) more efficient, relative to the second-order schemes, as the dispersion error tolerance approaches zero.

Now taking into account the additional iterations that are necessary for the Jacobi solve, the implicit schemes should be advantageous if the RCE for some desired dispersion error tolerance is

[1] It is worth pointing out that the implicit generalisations of the FCC and IWB explicit schemes were investigated, but they did not offer significant improvements over the explicit cases. This can be traced to the lack of an isotropic error term in their discrete Laplacians.

Table 1: Dispersion error tolerance levels where implicit schemes are more computationally efficient than the FCC and IISO1 (or IISO2) explicit schemes, taking into account Jacobi solve with $M = \lceil -\log_{10}(E)/\eta \rceil$. Also shown are the minimum directional cutoff frequencies, $\omega_{k,\text{cutoff}}$ in rad/sample.

$\alpha_1 \in \{1/3, 5/12\}$			more eff. than FCC		more eff. than IISO1	
β	η	$\omega_{k,\text{cutoff}}$	$E = \varepsilon_\text{s}$	$E = \varepsilon_\text{d}$	$E = \varepsilon_\text{s}$	$E = \varepsilon_\text{d}$
0.04650	0.641	0.626π	<1.1%	–	<1.3%	–
0.04345	0.677	0.629π	<0.98%	–	<1.2%	<0.73%
0.04040	0.715	0.632π	<0.92%	<0.56%	<1.1%	<0.67%
0.03735	0.755	0.635π	<0.79%	<0.48%	<0.98%	<0.58%
0.03430	0.799	0.638π	<0.70%	<0.37%	<0.88%	<0.48%
0.03125	0.845	0.641π	<0.53%	<0.28%	<0.70%	<0.36%

greater than $\lceil -\log_{10}(E)/\eta \rceil + 2$ (the residual check is neglected). Table 1 lists the dispersion error tolerances below which the implicit schemes with $\alpha = 1/3$ are more efficient than the FCC and IISO1 (or IISO2) explicit schemes, in terms of point-wise updates required for the iterative solve to converge in single and double precision. As can be seen in the table, one can choose β to give an implicit scheme that is more efficient than the FCC explicit scheme for any dispersion error tolerance that is less than 1.1%. In double precision, the implicit schemes become more favourable when the dispersion error tolerance is less than 0.56%.

Using the same techniques, we could compare the schemes in terms of the spatial grid densities, leading to memory costs required for some dispersion error tolerance level. This relative comparison is similar to what appears in Fig. 3, but the vertical axis would represent the relative efficiency in terms of spatial grid density, and it would be scaled by a factor of 3/4 (on a log scale) to reflect the cubic increase in grid density versus the quartic increase in operations with oversampling of the grid. As such, we simply summarise the main result. In terms of the extra memory storage required for the Jacobi solve (two extra states), the implicit schemes become more efficient than the explicit FCC and IISO1 schemes when the dispersion error tolerance is <3.8% (vs. FCC) or <5.3% (vs. IISO1).

8. NUMERICAL EXPERIMENTS

8.1. Modal frequencies of cubic domain

The known analytical modes of a cubic-shaped room with lossless boundaries provide a simple validation test that can also illustrate some advantages of the implicit schemes. To this end, the low-frequency response of a cubic domain with $\gamma = 0$ and with dimensions (11 m)^3 was simulated using the scheme with $\alpha_1 = 1/3$ in explicit ($\beta = 0$) and implicit forms ($\beta = 1/32$). The first two time-steps, $u(0, \boldsymbol{x})$ and $u(k, \boldsymbol{x})$, were set to a spatial Gaussian with mean (1 m, 2 m, 3 m) (the domain is centered about the origin) and variance 1 m^2. The Courant number was set to λ_max respectively for both schemes and $c = 340$ m/s. To normalise for computational costs (total number of operations), the implicit scheme with $M = 5$ iterations used a coarse grid of size 12x12x12, whereas the explicit scheme used a finer grid of size 21x21x21. The outputs were read at the grid points (4,6,3) and (8,12,6) for the implicit and explicit schemes respectively (counting from one). Spectra of the outputs from these simulations are shown in Fig. 4. As can be seen, the implicit scheme results in a better agreement with the analytical modal frequencies, despite having a coarser spatial grid.

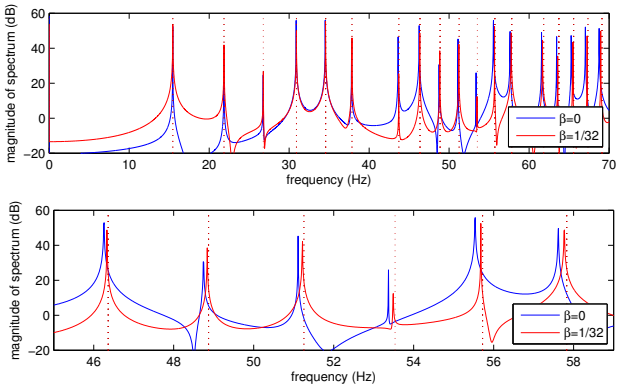


Figure 4: Comparison of low frequency responses for cubic room using IISO1 explicit scheme ($\alpha_1 = 1/3, \beta = 0$) with grid size 21x21x21, and fourth-order implicit scheme ($\alpha_1 = 1/3, \beta = 1/32$) with grid size 12x12x12 and $M = 5$. Dotted lines denote theoretical modal frequencies.

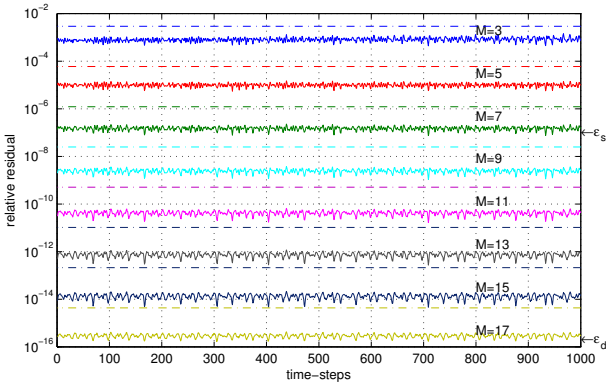


Figure 5: Relative residual from implicit scheme ($\alpha_1 = 1/3, \beta = 1/32$) after M iterations for simulation of cubic domain. Dashed lines denote expected residual, $10^{-0.845M}$. Jagged lines are measured relative residuals. Machine epsilon for single and double precision are marked by arrows.

8.2. Relative residual with fixed number of iterations

It is also worth investigating the relative residual over time using a fixed number of Jacobi iterations. Using the same test case, the relative residuals obtained from conducting simulations with various choices of M are plotted in Fig. 5. As can be seen, the relative residuals (jagged lines) remain smaller in magnitude than the expected residuals with magnitude $10^{-0.845M}$ (dashed lines). In this test case, the limits of single and double precision are effectively reached with seven and 17 iterations respectively.

8.3. Stability in finite precision arithmetic

The stability conditions derived in Section 4 may not be sufficient in practical situations due to unavoidable finite precision effects (round-off error). Single precision may be preferred to double precision since GPU cards tend to have a better peak performance for single precision arithmetic than double, and single precision variables use half of the memory space on the GPU card. However, round-off error in single precision has been known to cause late-time instabilities (after $\mathcal{O}(10^4)$ time-steps) with even the simplest of explicit schemes (SLF) [22], while such instabilities are rarely seen in double precision. A typical room impulse response at 44.1 kHz will require $\mathcal{O}(10^5)$ time-steps to be calculated, so it is important to ensure the long-time stability of these schemes in single precision. These round-off effects have been analysed using the spectral properties of the one-step recursion (state space) matrix in the explicit 27-point schemes [23]. Here, we consider the usual two-step recursion, which does not necessarily encapsulate all round-off errors, but focuses on the spectrum of the Laplacian matrix.

As described in Section 4.2, the two conditions: $\rho(\mathbf{L}) \leq 4\Lambda_{\max}$ and $\mathbf{L} \leq 0$, along with (12), lead to stability of the explicit/implicit schemes. In practice, it is possible that these conditions will not hold in the presence of round-off errors. However, measures can be taken to protect against any consequent instabilities (exponential growth). Linear growth is possible at the stability limit, but such growth is undesirable for room impulse responses. Setting the Courant number slightly below its limit: $\lambda = (1 - \mu)\lambda_{\max}$, for $0 < \mu \lll 1$ prevents such growth ($\mu =$ 1e-4 is a good choice), as well as any exponential growth near the Nyquist caused by round-off errors [23]. To buffer against a violation of the second condition, we can replace $\mathbf{L}$ with $\mathbf{L} - \sigma\mathbf{I}$, for $0 < \sigma \lll 1$, since it follows from Gerschgorin's theorem that $\mathbf{L} - \sigma\mathbf{I} \leq 0$ for σ sufficiently large. This also causes a shift of modal frequencies, but the effect is negligible for $\sigma \ll \omega/c$, and σ should be on the order of 10^{-7}.

To test these counter-measures, the IISO1 explicit scheme ($\alpha_1 = 1/3$) and its fourth-order implicit counterpart ($\beta = 1/32$, $M = 8$) were excited with a Kronecker delta (in space and time) on a grid of size 26x10x10, and run for 10^6 time-steps. The excitation was also DC-filtered [24] to eliminate any unwanted, yet valid, linear drift in the solution.

In Fig. 6a, an exponential drift (DC instability [23]) can be seen; this is caused by round-off error in single precision and is to be corrected through the use of the σ parameter. Fig. 6b shows the effect using a small σ, approximately $2\varepsilon_s$, to correct such an instability (note the scaling on the horizontal axes in Figs. 6a and 6b). The use of $\sigma > 0$ is not necessary in double precision (at least for $\mathcal{O}(10^6)$ time-steps), as seen in Fig. 6c with $\sigma = 0$. Figs. 6d-6f show the fourth-order implicit counterparts using eight iterations. In double precision the implicit scheme is stable for 10^6 time-steps with $M = 8$ and $\sigma = 0$ (Fig. 6f). Lossy boundaries ($\gamma =$ 1e-5) are employed in Figs. 6g-6h, which results in a decay in the responses.

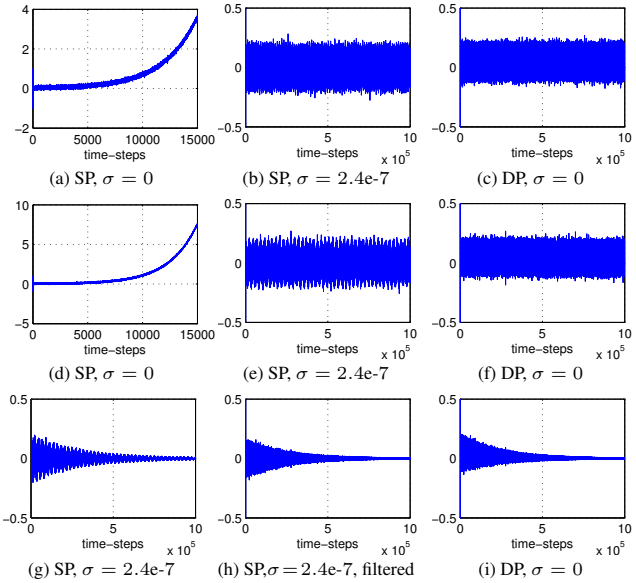
It is important to point out a low-frequency amplitude modulation in Figs. 6b, 6e, and 6g. This is due to the DC mode ($\omega = 0$) being shifted by the effect of σ non-zero. A similar effect arises when a so-called "hard source" is used as an excitation [25]. Here, the oscillation has a normalised frequency of approximately $\sqrt{\sigma\pi}$ rad/sample. The value of σ that will be required in single precision should scale with the duration of the simulation. Thus, for a typical room impulse response, σ should scale with the sample rate, and this low-frequency oscillation should remain inaudible. If desired, the artefact can be removed by applying another DC blocking filter [24] to the output, as seen in Fig. 6h.

9. SIMULATIONS ON GPU

In this section, we present timing results from a basic CUDA implementation of the implicit schemes on a single Nvidia Tesla K20 GPU card. The goal here is not to present speed-ups over single-thread CPU codes, since significant speed-ups have been reported for 27-point explicit schemes in various studies [12, 13, 26, 22]. The interest here is simply to compare GPU implementations of the

Table 2: Timing results from computation of 2000 time-steps on a Tesla K20 GPU card for explicit schemes and implicit schemes using M Jacobi iterations and the compute time increase (CTI) for each implicit scheme over its respective explicit counterpart. The CTIs are expected to be $(M + 2)$ due to the additional SpMVs required by the implicit schemes.

			explicit	implicit, $M = 8$		implicit, $M = 12$	
δ_Δ	(N_x, N_y, N_z)	precision	time (s)	time (s)	CTI	time (s)	CTI
19-point	(640,480,480)	single	50	453	9.06	635	12.7
19-point	(960,640,480)	single	100	894	8.94	1254	12.5
27-point	(640,480,480)	single	75	608	8.11	846	11.3
27-point	(960,640,480)	single	148	1201	8.11	1676	11.3
19-point	(640,480,480)	double	79	801	10.1	1112	14.1
27-point	(640,480,480)	double	89	910	10.2	1242	14.0



(a) SP, $\sigma = 0$ (b) SP, $\sigma =$ 2.4e-7 (c) DP, $\sigma = 0$

(d) SP, $\sigma = 0$ (e) SP, $\sigma =$ 2.4e-7 (f) DP, $\sigma = 0$

(g) SP, $\sigma =$ 2.4e-7 (h) SP,$\sigma =$ 2.4e-7, filtered (i) DP, $\sigma = 0$

Figure 6: Responses from cubic box obtained using explicit/implicit schemes with $\alpha_1 = 1/3$ in single precision (SP) and double precision (DP), with $\lambda = 0.9999\lambda_{\max}$ in each case. Implicit schemes use $M = 8$ iterations. Note that σ is not used in double precision. A DC blocking filter was applied to the output in Fig. 6h.

explicit schemes and their implicit counterparts for a fixed number of Jacobi iterations. Specific details on the GPU implementation will be left out for brevity, but the implementation is similar in nature to those found in [26, 22]. However, it is important to note that the memory bandwidth was maximised by making use of the read-only data cache in the Nvidia Kepler GPU architecture.

Table 2 lists the timing results from computing 2000 time-steps for 19-point and 27-point explicit schemes (the choice of $\boldsymbol{\alpha}$ is not important here) and their implicit counterparts (the choice of $\beta > 0$ is not important) with a fixed number of iterations $M \in \{8, 12\}$. Two different grid sizes were used and the simulations were run in both single and double precision. Results for the larger grid size are only given in single precision due to memory limitations on the GPU card (5 GB).

We expect the implicit schemes to take $M + 2$ times as long as their explicit counterparts due to the extra SpMVs (not taking into account the extra memory bandwidth required). As can be seen in Table 2, the implicit schemes are 10-20% faster than expected in single precision. Meanwhile, in double precision they behave approximately as expected. These variations from the $M + 2$ increase are due to cache effects and memory bandwidth bottlenecks.

10. CONCLUSIONS AND FUTURE WORK

In this study, we have presented 19- and 27-point fourth-order accurate and optimised implicit finite difference schemes for the 3-D wave equation with frequency-independent lossy boundaries on a box-shaped domain. These schemes can be solved using the Jacobi method with a convergence rate of nearly one digit of relative accuracy per iteration. Numerical dispersion was analysed and it was found that the implicit schemes, taking into account the iterative solve, become more computationally efficient than second-order explicit counterparts for situations where the amount of dispersion error that can be tolerated is less than 1%, and exponentially more efficient as this tolerance level approaches zero. These schemes were shown to be stable in finite precision arithmetic for as many as 10^6 time-steps in double precision, as well as in single precision at the cost of introducing inaudible artefacts. Timing results were presented from CUDA implementations run on an Nvidia Tesla K20 GPU card. It was found that the compute times for the implicit schemes scaled as expected with the additional SpMVs required.

Future work will investigate further generalisations for these implicit schemes. The first is to consider a more general form for the implicit scheme where different sets of $\boldsymbol{\alpha}$ parameters are used for the implicit and explicit discrete Laplacian operators, as in [14], providing more free parameters to optimise in order to further minimise numerical dispersion. Another generalisation is to include viscothermal effects, which are necessary for a more detailed model of sound propagation in air [3]. A third generalisation would be to consider these schemes in an unstructured finite volume framework (allowing for the modelling of irregular geometries) with more general (complex) impedance boundary conditions, as in the explicit case [10].

More advanced iterative techniques that are amenable to parallel implementations (Krylov subspace methods) could also be considered; many of which are known, for certain problems, to converge in fewer iterations than the Jacobi method [17]. However, preliminary tests with the system (15), using the myriad iterative methods provided in MATLAB, indicate that while such advanced techniques can converge in fewer iterations, they do not offer substantial improvements in compute times. Ultimately, this is because they require more computation within each iteration (additional SpMVs and residual checks), not to mention additional storage. Finally, an important area of research will be to determine the minimum number of Jacobi iterations required to simultaneously ensure that the residual is inaudible and that stability is maintained for the duration of a given simulation.

Comparative sound examples for the implicit and explicit counterpart schemes will be available for listening at:
`http://www2.ph.ed.ac.uk/~s1164563/dafx14`.

11. REFERENCES

[1] S. Siltanen, T. Lokki, and L. Savioja, "Rays or waves? understanding the strengths and weaknesses of computational room acoustics modeling techniques," in *Proc. Int. Symposium on Room Acoustics*, 2010.

[2] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith III, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[3] P. M. Morse and K. U. Ingard, *Theoretical acoustics*. Princeton University Press, 1968.

[4] G. E. Forsythe and W. R. Wasow, *Finite-difference methods for partial differential equations*. New York: Wiley, 1960.

[5] D. Botteldooren, "Finite-difference time-domain simulation of low-frequency room acoustic problems," *J. Acoustical Society of America*, vol. 98, pp. 3302–3308, 1995.

[6] L. Savioja, T. J. Rinne, and T. Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. Int. Computer Music Conf. (ICMC)*, Danish Institute of Electroacoustic Music, Denmark, 1994, pp. 463–466.

[7] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, 2011.

[8] S. Bilbao, "Wave and scattering methods for the numerical integration of partial differential equations," Ph.D. thesis, Stanford University, 2001.

[9] J. Botts and L. Savioja, "Integrating finite difference schemes for scalar and vector wave equations," in *Proc. IEEE ICASSP*, Vancouver, Canada, 2013, pp. 171–175.

[10] S. Bilbao, "Modeling of complex geometries and boundary conditions in finite difference/finite volume time domain room acoustics simulation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1524–1533, Jul. 2013.

[11] B. Hamilton, "Finite volume perspectives on finite difference schemes and boundary forumulations for wave simulation," in *Proc. Digital Audio Effects (DAFx)*, Erlangen, Germany, Sep. 2014.

[12] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low-and mid-frequency room acoustics," in *Proc. Digital Audio Effects (DAFx)*, vol. 1, Graz, Austria, 2010, p. 75.

[13] C. J. Webb and S. Bilbao, "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity," in *Proc. IEEE ICASSP*, Prague, Czech Republic, 2011, pp. 317–320.

[14] S. Bilbao, "Parameterized families of finite difference schemes for the wave equation," *Numerical Methods for PDEs*, vol. 20, no. 3, pp. 463–480, 2004.

[15] K. Kowalczyk and M. van Walstijn, "A comparison of nonstaggered compact FDTD schemes for the 3D wave equation," in *Proc. IEEE ICASSP*, Dallas, Texas, 2010, pp. 197–200.

[16] A. Gourlay and A. R. Mitchell, "A classification of split difference methods for hyperbolic equations in several space dimensions," *SIAM J. Numerical Analysis*, vol. 6, no. 1, pp. 62–71, 1969.

[17] C. Meyer, *Matrix analysis and applied linear algebra*. Philadelphia, PA: SIAM, 2000, vol. 2.

[18] J. Strikwerda, *Finite difference schemes and partial differential equations*. Philadelphia, PA: SIAM, 2004.

[19] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. Wiley, 2009.

[20] M. van Walstijn and K. Kowalczyk, "On the numerical solution of the 2D wave equation with compact FDTD schemes," in *Proc. Digital Audio Effects (DAFx)*, Espoo, Finland, 2008, pp. 205–212.

[21] B. Hamilton and S. Bilbao, "On finite difference schemes for the 3-D wave equation using non-Cartesian grids," in *Proc. Sound and Music Computing (SMC) Conf.*, Stockholm, Sweden, 2013, pp. 592–599.

[22] C. J. Webb and A. Gray, "Large-scale virtual acoustics simulation at audio rates using three dimensional finite difference time domain and multiple GPUs," in *Proc. Int. Cong. Acoustics (ICA)*, Montréal, Canada, 2013.

[23] J. Botts and L. Savioja, "Spectral and pseudospectral properties of room acoustic finite difference simulation," *IEEE Trans. Audio, Speech, and Language Processing*, 2014, to appear.

[24] J. O. Smith III, *Introduction to digital filters: with audio applications*, 2007, vol. 2.

[25] J. Botts and L. Savioja, "Effects of sources on time-domain finite difference models," *J. Acoustical Society of America*, 2014, to appear.

[26] B. Hamilton and C. J. Webb, "Room acoustics modelling using GPU-accelerated finite difference and finite volume methods on a face-centered cubic grid," in *Proc. Digital Audio Effects (DAFx)*, Maynooth, Ireland, Sep. 2013, pp. 336–343.

12. APPENDIX

12.1. 27-point discrete Laplacian

Using a notation similar to [7] we have:

$$\delta^h_\Delta = \delta_{xx}+\delta_{yy}+\delta_{zz}+a(\delta_{xx}\delta_{yy}+\delta_{xx}\delta_{zz}+\delta_{yy}\delta_{zz})+b(\delta_{xx}\delta_{yy}\delta_{zz})$$

where $a = (\alpha_2 + 2\alpha_3)/4$, $b = \alpha_3/4$, and

$$\delta_{xx}u := u(t, \boldsymbol{x} + \boldsymbol{e}_x h) - 2u(t, \boldsymbol{x}) + u(t, \boldsymbol{x} - \boldsymbol{e}_x h)$$

with the standard unit vector in the x-direction $\boldsymbol{e}_x$. The operators δ_{yy} and δ_{zz} are similarly defined.

12.2. 3-D Laplacian matrix with Neumann conditions

The Laplacian matrix corresponding to the centered Neumann conditions from [7] can be constructed as follows. The 1-D Laplacian matrix with centered Neumann conditions is:

$$\mathbf{D}_N = \begin{bmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{bmatrix}.$$

Let $\mathbf{I}_N$ represent the $N \times N$ identity matrix. Consider a 3-D grid with dimensions $N_x \times N_y \times N_z$ and let it be decomposed into a vector, first into z-planes, then y-rows and x-columns. We construct the matrices

$$\begin{aligned} \mathbf{D}_{xx} &:= \mathbf{I}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{D}_{N_x}, \\ \mathbf{D}_{yy} &:= \mathbf{I}_{N_z} \otimes \mathbf{D}_{N_y} \otimes \mathbf{I}_{N_x}, \\ \mathbf{D}_{zz} &:= \mathbf{D}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{I}_{N_x}, \end{aligned}$$

where $\otimes$ denotes the Kronecker product. The Laplacian matrix of interest can then be written as

$$\begin{aligned} &\mathbf{L} = \mathbf{D}_{xx} + \mathbf{D}_{yy} + \mathbf{D}_{zz} \\ &+ a(\mathbf{D}_{xx}\mathbf{D}_{yy} + \mathbf{D}_{xx}\mathbf{D}_{zz} + \mathbf{D}_{yy}\mathbf{D}_{zz}) + b(\mathbf{D}_{xx}\mathbf{D}_{yy}\mathbf{D}_{zz}). \end{aligned}$$

12.3. Loss matrix

The matrix $\mathbf{Q}$ can be constructed as follows. Let $\mathbf{q}_N$ be the vector: $(1, 0, \ldots, 0, 1)^T$ of length N. We construct the matrices

$$\begin{aligned} \mathbf{Q}_x &:= \mathbf{I}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{q}_{N_x}, \\ \mathbf{Q}_y &:= \mathbf{I}_{N_z} \otimes \mathbf{q}_{N_y} \otimes \mathbf{I}_{N_x}, \\ \mathbf{Q}_z &:= \mathbf{q}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{I}_{N_x}. \end{aligned}$$

Then we have $\mathbf{Q} = \mathbf{Q}_x + \mathbf{Q}_y + \mathbf{Q}_z$.

A PRELIMINARY MODEL FOR THE SYNTHESIS OF SOURCE SPACIOUSNESS

Darragh Pigott,
CSIS,
University of Limerick,
Limerick, Ireland
darragh.pigott@ul.ie

Dr. Jacqueline Walker,
ECE,
University of Limerick,
Limerick, Ireland
jacqueline.walker@ul.ie

ABSTRACT

We present here a basic model for the synthesis of source spaciousness over loudspeaker arrays. This model is based on two experiments carried out to quantify the contribution of early reflections and reverberation to the perception of source spaciousness.

1. INTRODUCTION

The subject of spatial audio covers a vast and wide-ranging array of topics from psychology, acoustics, engineering, mathematics, and computer science. The varied contributions from these different fields make for a fascinating and challenging path towards understanding. One challenge that arises is the exact definition of any particular concept. Our primary concern is the synthesis of circumstances under which a certain perceptual attribute of a reproduced sound field arises in the listener. In particular we are interested in *source spaciousness* i.e. the perceived extent of a sound source in three dimensions.

Spaciousness has been the subject of experiments and studies in the past and there is much to learn from the work of [1]-[4]. One of the drawbacks of the term spaciousness is its use as an everyday term as a descriptor for the sense of space. The lack of a clear definition can lead to ambiguity in discussions about perceptual attributes such as source spaciousness. There are places in the literature where spaciousness is discussed but not defined, and others where a definition is offered which do not correspond to definitions found elsewhere. With this in mind we offer here a concise definition of source spaciousness to remove any possible ambiguity for the purposes of the experiments described below.

1.1. Definitions

In the scientific disciplines of acoustics and psychophysics there is a tendency to define spaciousness in terms of its physical correlates [5]. In some cases the term spaciousness is used as a synonym for Auditory Source Width (ASW) [6]. Griesinger opts for a more intuitive definition of spaciousness to mean the impression of a large and enveloping space [7].

Since we are using the definition to relate a concept to a group of potentially inexperienced listeners, we have opted for a more descriptive definition that describes the perceptual attributes of the sound as the three dimensional extent of the perceived source.

Source spaciousness is the perceived extent of a sound source in three dimensions. It can be expressed as a combination of source width, source depth, and source height. Width describes the extent of the perceived source from left to right, depth describes the source extent from front to back, and height is the extent from bottom to top.

This definition accommodates an extension of the sound source such that the boundaries of the source can expand to include the listener "within" the sound. Such a situation may lead to the need for terms such as source envelopment and source engulfment as special cases of listener envelopment (LEV) [2] and engulfment [8].

With the range of definitions used for the term spaciousness we have to tread carefully and state that we are referring to the work of others only in as much as it reflects on the work presented here, that is to say, we are using the definition of source spaciousness provided above even when we refer to the results of others who may themselves be using the term spaciousness to mean something else.

1.2. Past Experiments/Results

In the area of concert hall acoustics, source spaciousness is treated as a contributing component of an all-encompassing perceptual attribute referred to as Spatial Impression [1], [9]-[11]. The three dimensional nature of a sound image is described in [5] as the subjective effect of early reflections. *"As the lateral reflection level is increased, the source appears to broaden and the music gains body and fullness"*.

The importance of the frequency content of early reflections to source spaciousness is reported in [5] to the degree that it contributes to the source broadening of the image, with the effect being most prominent around the 1kHz range. Blauert and Lindemann reported on the effect of various frequencies had on both the width and depth of a perceived auditory image [1]. Early reflections made up of primarily low frequencies were attributed to the cause of an increase in depth while the presence of higher frequencies resulted in the lateral expansion of the image.

Since the introduction of elevated speakers into the standard reproduction systems is a relatively new development, experiments covering the perception of height as a perceptual attribute are fewer in number relative to the number of experiments dealing with width and depth.

1.3. Research Question

We know from [1] and [5] that a certain amount of source

spaciousness is determined by the presence of early lateral reflections and low frequency reverberation. We also know that the degree to which each dimension of source spaciousness is affected by a lateral reflection is dependent on the frequency content of the reflection(s). We present here a preliminary model to implement these ideas for a system that can synthesize and control the perception of source spaciousness. The purpose of the following experiments is to (a) quantify the contribution of early reflections and low frequency reverb to the perception of source spaciousness and (b) to quantify the contribution of unique frequency bands to the perception of width, depth and height independently.

2. EXPERIMENTAL SETUP

The experiment was carried out in the Spatialization and Auditory Display Environment (SpADE) at the University of Limerick. A description of the acoustic performance of that space can be found in [12]. Many of the features of the experiments are similar to those found in [1].

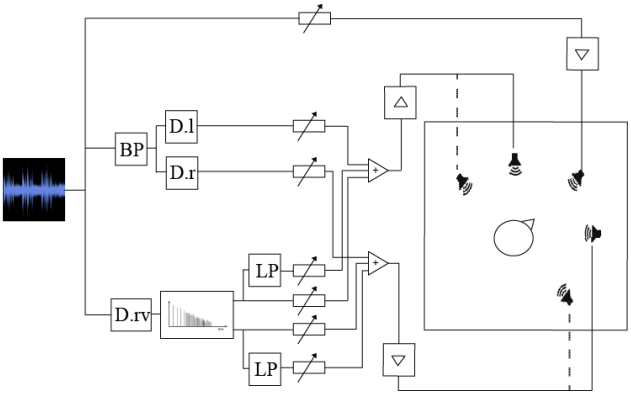


Figure 1: *schematic of the experimental setup*

2.1. Hardware & Software

The speaker setup consisted of 5 Genelec 8030 active near-field monitors positioned 2m from the listening position at angles of 0°, ±45° and ±90°. The direct sound was fed through the centre loudspeaker at 0° along with a reverb signal. The delayed lateral reflections were played back through the speakers to the side with the delayed reverb signal. The parameters of each test signal being examined in each part of the experiment are outlined below.

A reverb signal was created using an EMT 140ST with the reverb time set to 1.75s. This reverb signal was processed with a low pass filter and then mixed with the dry anechoic signal with a delay of 75ms.

Signal processing was applied to the source material in the Max/MSP audio environment. The DSP consisted of gain control, digital delays and 4th-order 24db/oct Chebyshev filters (low-pass and band pass). Each test signal was recorded to disk for use during the experiment to avoid any potential problems with running the signal processing "live". The average Sound Pressure Level (SPL) at the listening area for each of the sound fields presented was 76dB ±2dB.

2.2. Test Signals

The test signals were generated from an anechoic recording of Glinka's Overture, Russlan and Ludmilla, from the Denon Anechoic Orchestral Music Recording CD. The left channel was extracted from the stereo recording and used as source material for both experiments. The spectrum of the opening 15 seconds used for the experiment is shown in Fig 1. Each experiment consists of a direct signal played back from the front loud speaker, along with 2 simulated reflections played back over the left and right loud speakers with an applied delay of 20ms and 30ms respectively. In experiment 1 the speakers at angles ±45° were used while in experiment 2 the speakers at ±90° were used along with the frontal speaker.

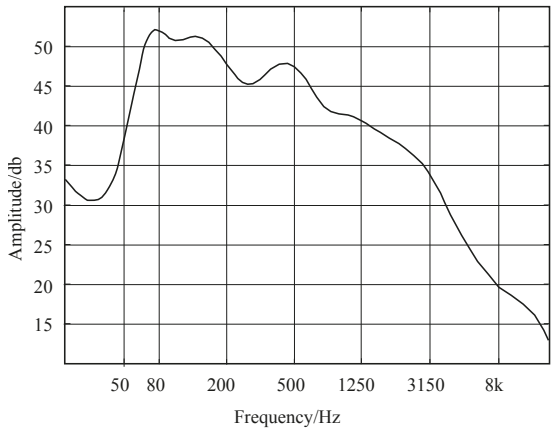


Figure 2: *spectrum of the test signal used*

2.3. Set 1

The parameters for the set of 15 signals in Set 1 are outlined below in Tables 1 & 2. The actual values for each variable were chosen on the basis of their inclusion in [1] where the emphasis was placed on "naturalness" for choosing the parameters outlined below. The sound fields were then arranged into pairs, making 105 pairs for comparison by the participants in the experiment.

Table 1: *variable values for experiment 1*

Cutoff frequency of low pass filtered reverb **fg**	Step +1: 900 Hz Step 0: 650 Hz Step -1: 400 Hz
Level of low pass filtered reverb relative to direct sound **NT**	Step +1: -12 db. Step 0: -14 db. Step -1: -18 db.
Level of early lateral reflections relative to direct sound **S**	Step +1: -3 db. Step 0: -5.6 db. Step -1: -13 db.

Table 2: *Variable values for each test signal in Experiment 1*

Test Signal		A	B	C	D	E	F	G	I	J	K	L	M	N	O	P
Parameter Settings	S	1	1	1	1	1	0	0	0	0	0	-1	-1	-1	-1	-1
	fg	1	-1	1	0	0	-1	1	0	-1	-1	0	0	1	-1	-1
	NT	1	0	-1	-1	1	0	1	-1	1	-1	1	0	0	0	-1

While listening to a pair, it was possible to switch between the sound fields freely, and repetition was allowed. In the first part of the experiment, the subjects were asked to compare the sound fields of the pair and make a judgment as to which was more spacious. Judgments of "no difference" were allowed. Their responses were submitted via a touch screen tablet device via OSC and saved in Max/MSP as a text file.

2.4. Set 2

The parameters for the filters applied to the simulated early reflections of part 2 of the experiment are outlined in Table 3. These sound fields were arranged in pairs resulting in 65 pairs for comparison. For each pair, the subject was asked to make a judgment as to which sound field was (a) wider (b) deeper, and (c) taller of the two. Their responses were in the form of a judgment plus a rating between 1-6 depending on the degree to which one was wider/deeper/taller than the other in each pair. Judgments of 'no difference" were allowed and a rating of 0 was applied to all such responses. During playback it was possible to switch freely between the two sound fields of the pair and repetition was allowed.

Table 3: *variable values for experiment 2*

Test Signal	Bandwidth
1	50 Hz – 80 Hz
2	50 Hz – 200 Hz
3	50 Hz – 500 Hz
4	50 Hz – 1250 Hz
5	50 Hz – 3150 Hz
6	50 Hz – 8000 Hz
7	80 Hz – 20 kHz
8	200 Hz – 20kHz
9	500 Hz – 20 kHz
10	1250 Hz – 20 kHz
11	3150 Hz – 20 kHz
12	8000 Hz – 20 kHz

2.5. Test Subjects

There were 18 participants in total ranging in age between 19 - 28 years old. All were post-graduate students who were studying courses with a strong emphasis on audio and music. Each reported to have normal hearing.

2.6. Pre-Experiment Examples

Prior to the experiment a brief training session was carried out where each subject was presented with several example sound fields with varying degrees of source spaciousness. The definition of source spaciousness was defined as described above and subjects were allowed to make their own judgments of source spaciousness of the example sound fields.

3. RESULTS

3.1. Experiment 1

The participant's responses to part 1 of the experiment were recorded as source spaciousness scores. For each pair under consideration, a 1 was assigned to the sound field judged to be more spacious and a -1 assigned to the sound field judged less spacious. In cases where the elements of the pair were considered to be equally spacious, a value of zero was assigned to both. Using this scoring scheme we can construct a ranking of spaciousness from the data, see Figure 3. The ranking clearly shows the test signals grouped into 3 clusters, each representing a different value for the variable S: level of early reflections.

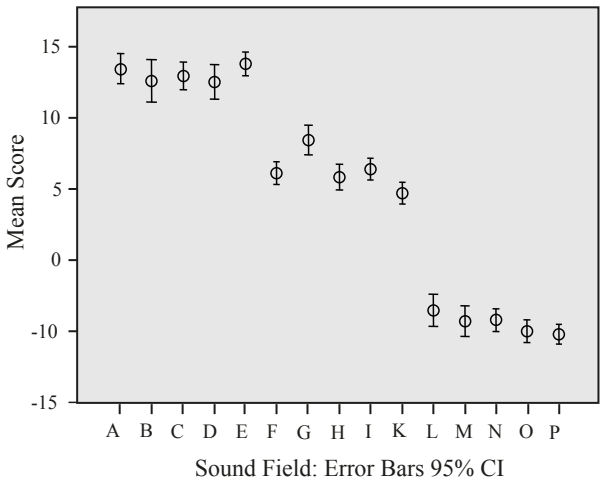


Figure 3: *spaciousness scores for experiment 1*

The existence of a strong relationship between the variable S and the source spaciousness score can be established by visually

Table 4: *Regression model for source spaciousness*

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	12.780	.994		12.851	.000
	Reflections Level	1.668	.031	.944	54.005	.000
	Reverb Filter Cutoff	.004	.001	.095	5.408	.000
	Reverb Level	.202	.053	.067	3.843	.000

a. Dependent Variable: Score

inspecting the chart in Figure 3 and the contents of Table 2. A standard multiple regression was performed between the spaciousness score for each sound field as the dependent variable and the variables S, fg and NT as the independent variables.

An analysis of the effect of the variables fg and NT on source spaciousness revealed little correlation between either variable and the score variable (correlation of .23 & .01 respectively). We also found that the contributions to the end score of the independent variables fg and NT were quite small (Standardized Coefficients of 0.09 and 0.06 respectively). It was proposed that the filter cut-off frequency had influence over the perceived source spaciousness only in as much as it affected the overall energy in the reverberation signal. A new variable was introduced that was the measured peak RMS level of the reverberation signal. Three level groups were identified, and the sound fields were given a new variable with value of -30db, -35db, or -45db according to the measured reverb level R.

This proved to slightly decrease the overall apparent contribution of the reverb signal to the perception of source spaciousness in the analysis. Although the difference is minor it leaves the question open as to whether there is an effect on source spaciousness by varying the frequency of a low cut filter applied to the reverb signal.

The number of cases submitted to analysis was 270, which is a sufficient amount to qualify as suitable for regression analysis [13]. No outliers were found with criteria for Mahalanobis distance set to $p < 0.001$.

Table 4 shows the unstandardized and standardized coefficients for the analysis along with the t value and significance levels. The R, R^2, and adjusted R^2 values for the model are 0.96, 0.92 and 0.92 respectively. This high value for R^2 signifies how dominant the level of early reflections is in determining source spaciousness.

As expected, the primary contributing variable for the spaciousness score is the level of the early reflections. The variation of the reverb signal does have and effect on the result but its significance is negligible in comparison to that of S. When we control for S we found that the effect of the reverb signal on the score was dependent on S. At extreme levels of S, the contribution was minimized, presumably because of the dominance of S. However the effect on the result caused by the reverb became more pronounced when S was in the middle of its range. This effect increased by a factor of 3 compared to its effect at the higher and lower values for S.

3.2. Experiment 2

The focus of the second experiment was on quantifying the contribution of various frequency bands to each of the three dimensions of source spaciousness i.e. source width, source depth and source height. Participants were asked to judge which test signal gave the impression of a wider, deeper and taller source. Compiling the scores in a similar way as we did in experiment 1, the ranking for each dimension is shown in Figures 4,5, & 6.

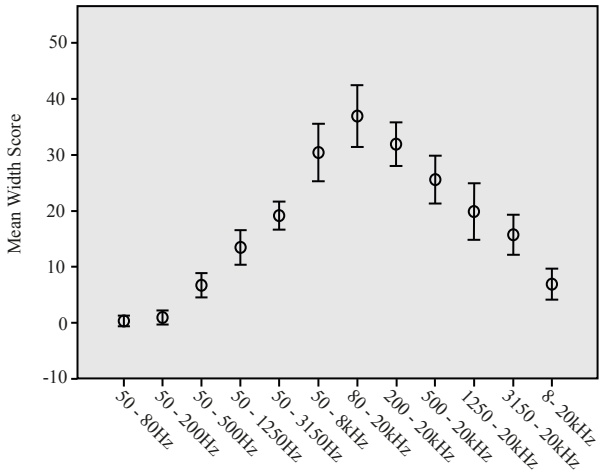


Figure 4: *width scores. Error Bars 95% CI.*

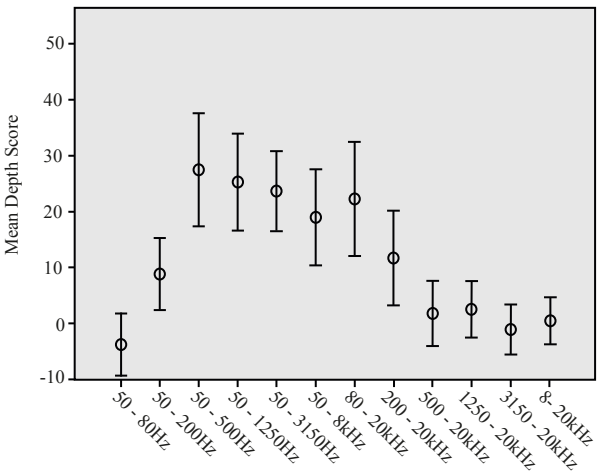


Figure 5: *depth scores: Error Bars 95% CI*

Table 5: *Regression model for width score*

Model		Unstandardized Coefficients		Standardized Co-efficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	.583	.186		3.135	.002
	fb 200 500	1.819	.279	.294	6.518	.000
	fb 500 1250	1.500	.322	.242	4.654	.000
	fb_1250_3150	1.278	.322	.206	3.964	.000
	fb_3150_8000	2.056	.322	.332	6.377	.000
	fb 8000 20k	1.597	.279	.258	5.722	.000

a. Dependent Variable: Width Score

As there is little energy in the source material between 50 Hz and 80 Hz, we cannot conclude much about the effect of energy in that region on the source spaciousness. Looking at Figure 4 we can see that all frequency components contribute to the perceived width of the source. Figure 5 indicates that the depth of the perceived source is determined by frequencies below 500 Hz. The presence of energy at frequencies above 500 Hz adds nothing to the perception of depth and may in fact reduce the effect.

The perceived height of the source is determined according to the presence of frequencies above 1250 Hz. The frequencies between 1250 and 8000 Hz contribute the most to the perception of height in the experimental setup. When the higher frequencies (>1250) are present, the addition of any energy in the range below 1250Hz has little effect on the perception of height. However in the absence of energy in the upper range of the frequency spectrum, the lower frequencies may increase the height of the perceived source.

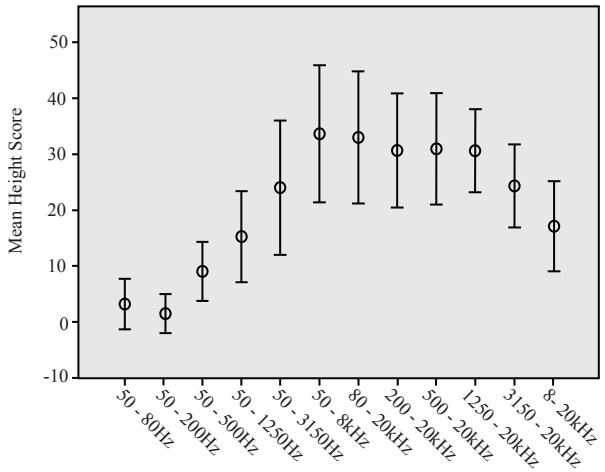


Figure 6: *height scores: Error Bars 95% CI.*

3.3. Models of Width, Depth, and Height.

The variables defining the test signals in experiment 2 were coded into non-overlapping frequency bands. If a frequency band is present as a reflection in a signal it is assigned a value of 1, otherwise it is 0. To determine the contributions from each frequency band to the perception of width depth and height we employed a standard multiple regression with the scores as the dependent variable and the frequency band variables of the early reflections as the independent variables.

After some exploratory analysis we found that the maximum number of independent variables contributing to the perceptual attribute source width is 5. With 216 cases submitted to the regression, the criterion for ratio of cases to independent variables is satisfied. No outliers were found.

The amount of variation in the width score is accounted for by the five frequency bands is shown in Table 5. The frequency range below 200Hz did not make any significant contribution to the width score. The model in Table 5 accounts for 81% of the variance in width score.

The variation in depth score is accounted for by the two frequency bands that make up the range between 80 Hz to 500 Hz. The coefficients for the depth regression are shown in table 6. The contribution of these frequency bands accounts for 39% of the total variation in depth score.

The results of the regression analysis with height as the dependent variable are summarized in Table 7.

We have found that the frequency content of the early reflections accompanying a direct signal have a significant influence on the perception of source spaciousness in terms of the width, depth and height of the perceived source. This confirms the results found in [1] and [5] although there is some disagreement over the exact frequency band which can be said to influence each of the dimensions.

4. SOURCE SPACIOUSNESS MODEL

Based on the results of the experiments presented above, we have devised an equation to represent a linear model of source spaciousness.

$$SS = \sum_{i=1}^{n} (\alpha_{w_i} + \alpha_{d_i} + \alpha_{h_i}) G_i + I_r \alpha_r G_r \qquad (1)$$

where G_i and G_r are the gain of the i^{th} frequency band of the simulated early reflections and the reverb signal respectively. α_{w_i}, α_{d_i}, and α_{h_i} are the regression coefficients from the linear approximations for perceived width, depth, and, height respectively. I_r is the scaling factor applied to the effect of the reverb due to the value of S. α_r is the reverb coefficient from the regression applied to the result of experiment 1.

Table 6: *Regression model for depth score*

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	2.352	.237		9.934	.000
	fb_80_200	1.889	.428	.317	4.410	.000
	fb_200_500	2.167	.428	.364	5.059	.000

a. Dependent Variable: Depth Score

Table 7: *Regression model for height score*

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	1.863	.244		7.635	.000
	fb 1250 3150	1.896	.423	.343	4.485	.000
	fb_3150_8000	1.250	.535	.226	2.338	.020
	fb_8000_20k	.590	.423	.107	1.396	.164

a. Dependent Variable: Height Score

The first term of (1) represents the contribution of early reflections while the second term accounts for the reverberation signal. Although we found there to be minimal effect of the reverb signal on the perception of source spaciousness, we kept this term in the equation to allow for potential future developments involving a reverb signal.

The overall content of the model is based on experiment 1 while the details of the filters applied to the early reflections to control for perceived width, height and depth independently is derived from the results of experiment 2. According to our findings, source spaciousness is a three dimensional spatial attribute that can be described in terms of width, depth, and height.

5. CONCLUSION AND FUTURE WORK

We have presented here a preliminary model for source spaciousness that is to serve as a starting point for the development of a more comprehensive study of this perceptual attribute. While changes in the width are well accounted for by the variables included in the experiments, the other two dimensions are less affected. Future experiments could potentially seek to get a more detailed picture at how the frequency spectrum of early reflections affects the perceptual attribute. The inclusion of elevated loudspeakers for the simulation of source height will also be investigated.

6. REFERENCES

[1] J. Blauert and W. Lindemann, "Auditory spaciousness: Some further psychoacoustic analyses", *The Journal of the Acoustical Society of America*, vol. 80, p. 533, 1986.

[2] M. Morimoto, K. Iida, and K. Sakagami, "The role of reflections from behind the listener in spatial impression", *Applied Acoustics*, vol. 62, no. 2, pp. 109–124, 2001.

[3] D. Griesinger, "General overview of spatial impression, envelopment, localization, and externalization", *AES 15th International Conference: Audio, Acoustics & Small Spaces, 1998*.

[4] J. Berg and F. Rumsey, "Systematic evaluation of perceived spatial quality", *AES 24th International Conference on Multi-channel Audio, 2003*.

[5] M. Barron and A. H. Marshall, "Spatial impression due to early lateral reflections in concert halls: the derivation of a physical measure", *Journal of Sound and Vibration*, vol. 77, no. 2, pp. 211–232, 1981.

[6] D. L. Valente, S. A. Myrbeck, and J. Braasch, "Matching Perceived Auditory Width to the Visual Image of a Performing Ensemble in Contrasting Multi-Modal Environments", *AES 127th Convention, New York, 2009.*

[7] D. Griesinger, "Spaciousness and envelopment in musical acoustics", *Audio Engineering Society, Preprints,* 1996.

[8] R. Sazdov, G. Paine, and K. Stevens, "Perceptual Investigation into envelopment, spatial clarity, and engulfment in reproduced multi-channel audio", *AES 31st International Conference, London, UK, 2007*.

[9] T. Hidaka, L. L. Beranek, and T. Okano, "Interaural cross‐correlation, lateral fraction, and low and high‐frequency sound levels as measures of acoustical quality in concert halls," *The Journal of the Acoustical Society of America*, vol. 98, p. 988, 1995.

[10] M. Barron, "Late lateral energy fractions and the envelopment question in concert halls", *Applied Acoustics*, vol. 62, no. 2, pp. 185–202, 2001.

[11] M. Barron, "The subjective effects of first reflections in concert halls--The need for lateral reflections", *Journal of Sound and Vibration*, vol. 15, no. 4, pp. 475–494, 1971.

[12] M. Ronan, D. Pigott, and R. Sazdov, “Configuring SpADE: Practical Considerations Influencing the Design of a 3D Multi-Channel Listening Environment”, *2nd Convocation of ISSTC, Cork, 2012*.

[13] B. G. Tabachnick and L. S. Fidell, "Using Multivariate Statistics.", Allyn & Bacon, 2001.

LOW FREQUENCY GROUP DELAY EQUALIZATION OF VENTED BOXES USING DIGITAL CORRECTION FILTERS

Stephan Herzog,
Department of Digital Signal Processing
University of Kaiserslautern
Kaiserslautern, Germany
herzog@eit.uni-kl.de

Marcel Hilsamer,
Department of Digital Signal Processing
University of Kaiserslautern
Kaiserslautern, Germany
hilsamer@eit.uni-kl.de

ABSTRACT

In this paper methods to determine the group delay of vented boxes and techniques for the design of filters for group delay equalization are presented. First the transfer function and the related group delay are explained. Then it is shown how the group delay can be computed or approximated for a certain alignment of the box. Furthermore it is shown how to derive the required parameters of the transfer function from a simple electrical measurement of the box, which allows the determination of the group delay without knowledge of the box design parameters. Two strategies for the design and implementation of digital correction filters are shown where one approach allows for a real-time adjustability of the delay. Finally, the performance with a real speaker is evaluated.

1. INTRODUCTION

Vented boxes have been in use for a long time. Their theory was described the first time to a great extent by Thiele in [1] and [2]. Later Small refined the theory further [3]-[4]. Both authors provided a mathematical description of a vented loudspeaker that allowed for a systematic design and an assessment of the transfer characteristics, which was not the case before. Later on, Bullock [5] streamlined the design procedure and made the data provided by Thiele and Small more practically usable.
The advantage of vented boxes w.r.t. closed or dipole speakers is their enhanced bass response. Their drawback is an increased group delay at low frequencies, which among other effects, can lead to the perception of a "muddy", "boomy" or "slow" bass.
These deficiencies at low frequencies are not the only phase errors of loudspeakers. In general, modern speakers are multi-way systems and the multiple ways are separated by a crossover, which can be implemented as a passive, an active analogue or an active digital system. Ideally, the output of the paths add up to a constant frequency response. A crossover is made of filters which provide the desired frequency division, but also introduce unwanted phase shifts and hence group delay errors. Additionally, the placement of the speakers relative to each other can introduce time-alignment errors. The significance and audibility of these phase or group delay errors is subject to ongoing research and discussions, see [6], [7], [8], [9] for example. Time-alignment correction using group delay equalization is proposed by [10] and [11]. For example [12] and [13] propose the correction of phase distortion with allpass filters.
Most of the present work deals with higher frequencies and equalization in the low-frequency region is rarely discussed. Linkwitz [14] states that it is not a trivial task, since a lot of delay is needed at higher frequencies of the spectrum. The authors of [15] focus on the phase correction at higher frequencies and remind that the low frequency sound is not perceived independently of the characteristics of the listening room. This is of course true but not limited to the low frequency range and a loudspeaker as ideal as possible is desirable.
In this paper we will focus on the equalization of group delay errors that are introduced by the driver-enclosure-system in the low-frequency range.

2. FUNDAMENTALS OF VENTED BOXES

A vented box is a loudspeaker enclosure with an additional opening called a vent or port, which is usually made of a tube.
The behaviour of a vented box can be described as a fourth order highpass system. The box itself is a resonator with the air in the box volume acting as a spring and the air in the port behaving as a mass, which together are a resonating mass-spring-system. Such a system is also known as a Helmholtz resonator.
The resonator helps the chassis to reproduce low frequency bass but has the disadvantage that the onset and offset of its oscillation is somewhat delayed to the driver signal. At the box resonance frequency the sound output is nearly solely coming from the port and thus has the group delay of the resonator. For frequencies higher than the port resonance frequency, the sound output from the driver and the port are mixed and at higher frequencies the sound from the driver dominates. Fig.1 illustrates the typical behaviour of a vented box derived from a LTspice simulation [16]. The group delay in this example exhibits a maximum of about 18 ms slightly below the port resonance frequency of about 34 Hz.

2.1. Transfer function of a vented box

The sound pressure frequency response of a vented box can be expressed by the general system function of a fourth order highpass:

$$G_v(s) = \frac{\left(\frac{s}{\omega_0}\right)^4}{1 + a_1\left(\frac{s}{\omega_0}\right) + a_2\left(\frac{s}{\omega_0}\right)^2 + a_3\left(\frac{s}{\omega_0}\right)^3 + \left(\frac{s}{\omega_0}\right)^4} \tag{1}$$

There are also higher order systems possible that make use of additional electrical filters to shape the low-end frequency response. These assisted designs are not considered here, since they are quite unusual. The coefficients of the system function $G_v(s)$ are related to the design parameters of the vented speaker and are defined by

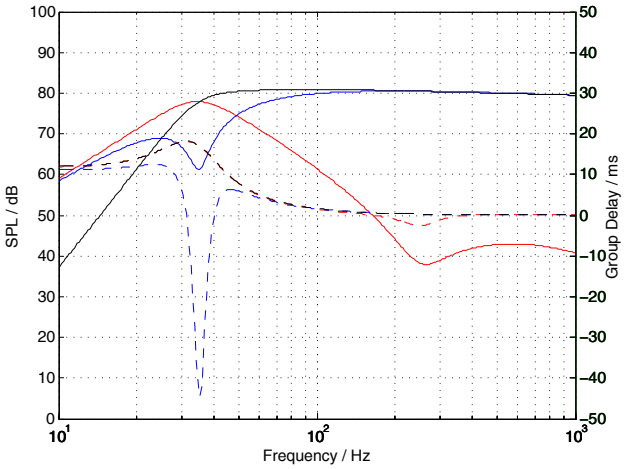


Figure 1: *Spice simulation of vented box showing magnitude (—) and group delay (- -) responses of the driver (blue), the port (red) and the combined output (black).*

the relationships

$$a_1 = \frac{1}{Q_l\sqrt{h}} + \frac{\sqrt{h}}{Q_{ts}} \tag{2}$$

$$a_2 = \frac{\alpha+1}{h} + h + \frac{1}{Q_l Q_{ts}} \tag{3}$$

and

$$a_3 = \frac{1}{Q_{ts}\sqrt{h}} + \frac{\sqrt{h}}{Q_l}. \tag{4}$$

$\alpha = V_{as}/V_B$ is the system compliance ratio. It describes the ratio of the compliance of the air in the box V_B to the compliance of the low-frequency driver V_{as}. $h = f_b/f_s$ is the tuning ratio, which is the ratio of the free-air resonance of the driver f_s to the resonance frequency of the box f_b. Both α and h are determined in the box design process to meet specific requirements. Q_l is the quality factor of the enclosure and depends on the construction of the box with regard to losses and air tightness. For a medium sized box with slight damping at the inner walls $Q_l = 7$ can be assumed. Q_{ts} is the total quality factor of the driver including mechanical and electrical characteristics and additionally resistive contributions from the crossover [17].
Depending on the values of the coefficients, the response of a box is classified as a certain alignment. The choice of an alignment depends on the desired frequency response and thus has an influence on group delay at low frequencies. Not all alignments are possible with all drivers depending on their parameters.
The alignment is usually derived from the magnitude squared function setting $s = j\omega$ and $\widehat{\omega} = \omega/\omega_0$, where ω_0 is the corner frequency of the highpass as

$$|G_v(j\omega)|^2 = \frac{\widehat{\omega}^8}{1 + A_1\,\widehat{\omega}^2 + A_2\,\widehat{\omega}^4 + A_3\,\widehat{\omega}^6 + \widehat{\omega}^8} \tag{5}$$

with

$$A_1 = a_1^2 - 2a_2\,, \quad A_2 = 2 + a_2^2 - 2a_1a_3\,, \quad A_3 = a_3^2 - 2a_2 \tag{6}$$

The box design parameters are then computed from A_1, A_2 and A_3.

2.2. B4 alignment (fourth order Butterworth)

For this alignment the transfer function corresponds to that of a fourth order Butterworth highpass. It is characterized by $A_1 = A_2 = A_3 = 0$. The transfer function of a fourth order Butterworth highpass is

$$B4(s) = \frac{s^4}{(s^2 + \sqrt{2-\sqrt{2}}s + 1)(s^2 + \sqrt{2+\sqrt{2}}s + 1)} \tag{7}$$

If we are interested in the group delay response of a Butterworth highpass we can look at the slightly more simple transfer function of a Butterworth lowpass which has the same group delay characteristics. Replacing s as in the previous section it can be expressed as

$$B4_{LP}(j\omega) = \frac{1}{1 + 2.613\,j\widehat{\omega} - 3.414\,\widehat{\omega}^2 - 2.613\,j\widehat{\omega}^3 + \widehat{\omega}^4}. \tag{8}$$

The general expression for the phase of the fourth order Butterworth lowpass filter is then

$$\beta = -\arctan\left(\frac{2.6131\,\widehat{\omega} - 2.6131\,\widehat{\omega}^3}{\widehat{\omega}^4 - 3.4142\,\widehat{\omega}^2 + 1}\right) \tag{9}$$

from which the group delay can be calculated as $\tau_g = -\mathrm{d}\beta/\mathrm{d}\omega$. The general expression for the group delay of a fourth order Butterworth filter (highpass or lowpass) can be found in eq. (20) in the appendix.
As can be seen in Fig.1, the maximum group delay occurs roughly at the cabinet resonance frequency. The group delay at resonance frequency is easily obtained by setting $\omega = \omega_0$ as

$$\tau_{max} \approx \tau(\omega = \omega_0) = \frac{3.695517}{\omega_0} = \frac{0.58816}{f_0}. \tag{10}$$

Hence, the approximate group delay maximum for a B4 alignment of interest can be determined quite simply. Another point of the group delay response can be estimated. The limit for $\omega \to 0$ is

$$\tau_g(0) = \frac{2.6131}{\omega_0} = \frac{0.4159}{f_0}. \tag{11}$$

With the knowledge of these two points of a group delay response, it is already possible to design a group delay equalizer. However, it has to be known that the box has a B4 alignment and the value f_0 is needed. Furthermore, the B4 alignment is only usable for drivers with $Q_{ts} \approx 0.4$, hence not all vented loudspeakers are designed using a B4 alignment.

2.3. Other alignments

Beside the B4 alignment there also exist further alignments like QB3 (quasi third order Butterworth), (S)C4 ((sub) fourth order Chebychev) and some more [18]. Their transfer functions include further design parameters that can be chosen to obtain desired response characteristics. Hence the determination of the group delay function for these type of filters is not possible in general.
Furthermore, a vented box is usually designed using one of these known alignments, but this is not strictly necessary, since the coefficients of the transfer function $G_v(s)$ can be set to any desired value as long as the design parameters as box size, tuning frequency and driver parameters allow.
Consequently, a more general approach is required to be able to equalize every loudspeaker.

3. DETERMINATION OF LOUDSPEAKER GROUP DELAY

If the alignment of the box is not known or the box is not designed corresponding to one of the known alignments, the parameters of the transfer function $G_v(s)$ can be determined by a measurement.

3.1. Acoustic measurement

The frequency response of any speaker can of course be measured by an acoustical measurement using an appropriate test system. However such a measurement requires a suitable (measurement-) microphone, a microphone preamp and software for signal generation and analysis. Such equipment is not always available, e.g. in a home environment.
Furthermore the results of an acoustical measurement depend heavily on the measurement room. Noise, reflections and standing waves can have a big influence on the results. Especially noise and reflections can cause peaks or ripple in the magnitude as well as in the phase response. Since the group delay is the derivative of the phase w.r.t. frequency, these unwanted disturbances influence the group delay measurement significantly.

3.2. Impedance measurement

The low-frequency behaviour of a loudspeaker can also be determined with an electrical impedance measurement [2]. In this case only a sine-generator, a voltmeter and an amperemeter are necessary. In many cases it should be possible to use standard multimeters, since even simple ones are dedicated to make measurements at 50 Hz and hence in the low audio frequency range. Furthermore, only the frequencies of three extreme values of the impedance response are needed, not the impedance values themselves. We have compared high-precision TRMS (HP 34401A) and simple non-TRMS multimeters without significant differences in this application.
From an impedance measurement the system compliance ratio α, the tuning factor h and Q_{ts} of the driver can be computed. With the knowledge of these values and an assumption for the box quality factor Q_l, the transfer function $G_v(s)$ and thus the low-frequency transfer characteristic of the loudspeaker is completely defined. In

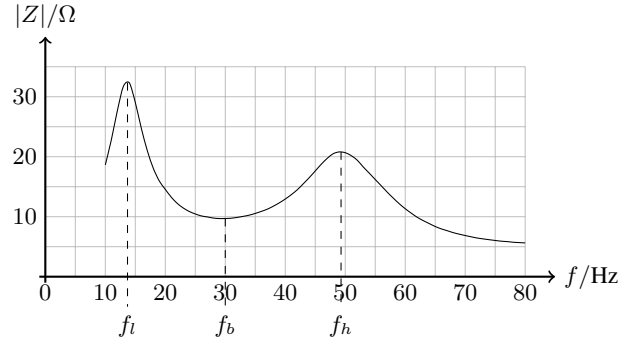


Figure 2: *Impedance curve of a vented box*

Fig.2 a typical impedance for a vented box can be seen. There are three frequencies of interest where f_b is the enclosure resonance frequency and a minimum of the impedance occurs. f_l is the frequency of the impedance maximum below f_b, and f_h is the frequency of the maximum above f_b. The resonant frequency of the speaker in the box f_{sb} can be computed as $f_{sb} = (f_h f_l)/f_b$. An alternative measurement method which requires blocking of the port is described in [19].
From the results of the impedance measurement, according to [4] the compliance ratio α can be computed as

$$\alpha = \frac{(f_h^2 - f_b^2)(f_b^2 - f_l^2)}{f_h^2\, f_l^2}$$

Furthermore the tuning factor can be computed from the impedance measurement as

$$h = \frac{f_b}{f_s} \approx \frac{f_b}{f_{sb}}. \tag{12}$$

The free air resonance frequency f_s normally deviates only slightly from the resonance frequency of the built-in speaker f_{sb} [19], hence the influence of this approximation on the group delay will also be small.
With the knowledge of Q_{ts} all coefficients of the general transfer function can be determined. Bullock [5] gives an approximate formula for the total driver quality factor as

$$Q_{ts} = \left(\frac{1}{20\,\alpha}\right)^{\frac{1}{3.3}}. \tag{13}$$

For the box quality factor Q_l, a value of $Q_l = 7$ can be assumed. A slightly larger value could be applied for smaller boxes and a smaller value for larger ones.
With the knowledge of α, h, Q_l and Q_{ts} the coefficients of the transfer function a_1, a_2 and a_3 of the loudspeaker can be computed using equations (2) - (4). Hence the transfer function $G_v(s)$ and the resulting magnitude, phase and group delay responses can be determined independently of the used alignment. If the measurement is made directly at the terminals of a complete loudspeaker, the influence of the crossover network, which mainly influences Q_{ts} is already included in the results.
An advantage of this method is, that it directly yields a parametric description of the transfer characteristics, as only the three frequencies f_l, f_b and f_h have to be determined. Hence no smoothing is required as it would be the case with a direct acoustic measurement.

4. CORRECTION FILTER DESIGN

If the group delay is known, the next step is the design of the correction filter for compensation of the vented-box group delay at low frequencies. Because of the very small ratio of the lower cut-off frequency f_0 to the sampling frequency f_s, very long filters are needed in the case of FIR-filters to obtain a satisfactory frequency resolution. IIR-filters can work with a significantly lower amount of coefficients but will have high demands on the precision of the coefficients necessary for this task.
Consequently, if a suitable filter has been designed, the filtering process may also require a high resolution, i.e. a powerful processor in the case of a long FIR-filter or high precision in the case of an IIR-filter.
The computation of the filter coefficients can be challenging due to the above reasons. For example an optimization based method as described in [20] can be applied to design an allpass filter that approximates the phase response of the loudspeaker. However, this method may not directly give good results or stable filters in our

application. This is due to the fact that the phase is only approximated in a very narrow frequency band, for which the numerical conditions become an issue. Furthermore a suitable phase offset, which is not directly included in the optimization problem has to be chosen to ensure good approximation of the phase. We will show two alternative methods to design a correction filter.
The required equalization filter must have a negative group delay in a certain frequency range to "speed up" the signal or must introduce additional delay, to "slow down" signals in the complementary frequency range. In the second case an additional delay will be introduced into the signal path, which has to be considered, e.g. in live applications or audio/video synchronous tasks.

4.1. Equalization filters with negative group delay

The use of such filters is not directly possible, because filters showing negative group delay have a high-pass magnitude response [21]. The frequency range, in which the negative group delay occurs is then in the stopband of the filters. This would attenuate frequencies in the desired low frequency range and therefore would need an additional equalization (amplification) which would result in a poor signal-to-noise-ratio.
The use of allpass filters with negative group delay (which would be the filters with the desired characteristic in our application) is not possible. These filters are not stable because their poles would be located outside the unit circle.

4.2. Equalization filters with inverse delay

Such a filter should increase the group delay at all frequencies except the ones near resonance frequency of the cabinet, which could be achieved using allpass filters. This means that a large filter order has to be used to obtain low ripple in the group delay response for higher frequencies. Furthermore the fact, that the required delay can become quite high at typical audio sampling frequencies (some 1000 samples) the Q of the group delay for one allpass is very large since the poles resp. zeros have to be very close to the unit circle. This further increases the filter order needed to obtain a low ripple in the group delay response.
Two alternative methods to design such a filter are shown in the following.

4.2.1. FIR filter with unit magnitude response and inverse phase response

A frequency response function with a constant magnitude and arbitrary phase can be designed in the frequency domain directly. As a starting point, the transfer function of the loudspeaker $G_v(s)$ can be transformed to the discrete time domain via the bilinear transform to obtain $G_v(z)$. Then the impulse response $h_1(n)$ of this IIR filter can be computed for a desired number N of samples. The response can then be transformed to the frequency-domain using a discrete Fourier transform (DFT). In the next step, in the frequency domain the magnitudes can be set to an arbitrary value, e.g. unity if only a phase equalization is required. If the phase has to be equalized to exactly cancel the original phase and no magnitude equalization is desired, this is the only modification needed in the frequency domain. To ensure real coefficients of the filter in the time domain, it has to be ensured that the spectral values of a length N filter satisfy the relation

$$H_1(k) = H_1^*(N-k), \qquad k \neq 0. \tag{14}$$

After a transformation back to the time domain via an inverse DFT, we obtain the impulse response of the FIR filter having only the phase response of the speaker and unity gain for all frequencies. To obtain the final equalization filter with inverse group delay w.r.t the original, the impulse response has to be time reversed.
The disadvantage of this approach is the resulting computationally expensive long FIR-filter that finally does the group delay equalization. This can make a real time implementation e.g. on an embedded DSP-system difficult. A computation of the convolution in the frequency domain using overlap-add or overlap-save schemes would reduce the effort significantly, but requires quite long Fast Fourier Transforms (FFTs) which require more memory, increase the delay due to block processing and can decrease the precision on fixed-point systems.
An advantage of this method is, that more correction can be designed into this filter, e.g. magnitude equalization or highpass filtering for driver protection without increasing the computational effort of the filtering process. The data for magnitude equalization could be obtained via an acoustic measurement whereas the incorporation of predefined functions like subsonic filters would not require additional measurements.
A correction filter as described can be designed using MATLAB. Filters that are not directly based on the modelled frequency response of the speaker but can be tuned manually or selected using presets can be designed with a tool like rePhase [22].

4.2.2. Direct design of an allpass with the same group delay as the speaker and computation as a time-reversed IIR-filter

In this approach we first design an allpass-filter with a group delay approximating that of the loudspeaker. Mainly, we want to compensate for the delay introduced by the cabinet, which is a resonator and hence a second order system, whereas the whole speaker is modelled as a fourth order system. Hence, the approach is to assume that it is sufficient to design a resonator as a two-pole filter

$$H_R(z) = \frac{b_0}{(1 - re^{-j\omega_0}z^{-1})(1 + re^{j\omega_0}z^{-1})} \tag{15}$$

to mimic the group delay response of the box.
Two parameters, the radius r and the angles $\pm\omega_0$ of a pole pair $p_{1,2}$ have to be determined. This can be done based directly on parameters of the original (measured) group delay response. In this approach the resonance frequency of the box determines the angle of the poles of the correction filter. The radius of the poles determines the rate of change of the phase at the pole frequency and thus the maximum of the group delay.
The maximum magnitude response of a resonator which corresponds to the maximum group delay does not directly occur at the pole frequency ω_0 but also depends on the pole radius r and occurs at the frequency

$$\omega_r = \arccos\left(\frac{1+r^2}{2r}\cos\omega_0\right) \tag{16}$$

[23]. This shift of the resonance frequency is due to the fact that we have poles at positive and negative frequencies to get real coefficients, i.e. a two-pole system for a single resonance. The pole at negative frequencies also has an influence on the response evaluated on the positive frequency axis in the upper half of the unit circle and vice versa. For values of r close to unity, the pole of the respective half-plane dominates and the dependency of the resonance frequency from the second pole can be neglected and thus

$\omega_r \approx \omega_0$. Hence we set the pole angles to

$$\omega_0 = \pm 2\pi \frac{f_b}{f_s}. \tag{17}$$

The required pole radius r can be derived from the phase response of the resonator. Here also both poles influence the desired group delay value, which makes the relation quite complicated. The expression for the group delay at ω_0 dependent on r is given in eq. (21) in the appendix. The expression in eq. (21) is quite unhandy and not easily to solve for r. A method for the determination of the required value of r is to compute it iteratively. A suitable starting point are the pole radii of the original transfer function $G_v(z)$.
In addition to the poles of the resonator, two zeros $z_{1,2}$ have to be added to obtain an allpass-system with a constant magnitude response. With the two zeros

$$z_{1,2} = \frac{1}{p_{1,2}^*} \tag{18}$$

we obtain the resulting transfer function of the allpass filter as

$$H_2(z) = \frac{(1 - \frac{1}{r}e^{j\omega_0}z^{-1})(1 + \frac{1}{r}e^{-j\omega_0}z^{-1})}{(1 - re^{-j\omega_0}z^{-1})(1 + re^{j\omega_0}z^{-1})}. \tag{19}$$

The zeros compensate the magnitude and add an additional delay of the same amount as that of the poles. Since $\omega_r = \omega_0$ is fixed, the poles have to be complex conjugates and the zeros directly result from the poles, r is the only parameter to be adjusted for the whole allpass equalization filter. Fig. 3 shows a pole-zero plot corresponding to the application example in the next section which shows the dimension of ω_0 and r.

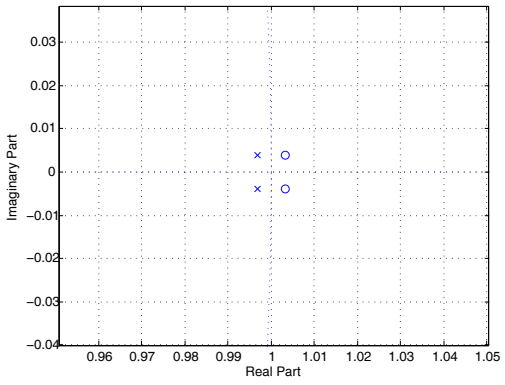


Figure 3: *Poles and zeros of the correction filter before time-reversal*

We now have a second-order recursive filter which approximates the group delay of the loudspeaker. To equalize the speaker, it has to be time reversed, which would normally lead to an unstable filter. This method for the design of the correction filter is not as exact as the approximation using a long FIR-filter on the basis of the measured group delay described in the section above. The advantage of this method lies in the significantly reduced computational effort required to run the filter in real-time.
The time-reversed low-order allpass $H_2(z)$ can be realized efficiently as an IIR-filter using the structure proposed in [24]. The

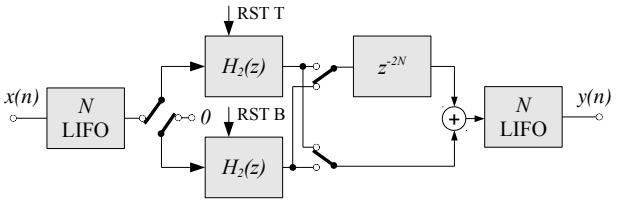


Figure 4: *Block diagram of time-reversed filter implementation from [24]*

block diagram of this filter is shown in Fig.4. Using this method, data is buffered for a number of N samples using a Last In First Out (LIFO) buffer. The output of the buffer is a time-reversed version of the input signal and sent through the second order allpass filter, which requires only 5 multiplies and 6 additions per output sample. Due to the long time constants it may be necessary to implement the filter in double precision which would increase the computation time roughly by a factor of four, which is still much less than a FIR implementation requires. The result is then again time-reversed by a second length-N LIFO-buffer and given to the output. To account for adjacent blocks an overlap-add scheme is applied. Computing the equalization filter as a time-reversed IIR-filter significantly reduces the required computational effort compared to an FIR implementation. However, there is no free lunch and the drawback is, that the delay is increased to $2N$ samples and the memory requirements to $4N$ samples. This would allow to run the equalization filter on quite simple platforms, provided, that they have enough memory. The additional delay may not pose a problem if just a music playback situation is considered.
Another advantage of this approach is, that the delay of the equalization filter can be changed quite easily just by changing pole radius r and re-computing the 5 filter coefficients of $H_2(z)$. The computation of an inverse DFT and time inversion is avoided. This would allow to implement an adjustable delay on an embedded system.
Both approaches, the FIR-filter and the time-reversed IIR-filter use a truncated impulse response of a recursive system as a correction filter. The required length N of this impulse response is of course dependent on the sampling frequency f_s and should be chosen to provide a minimum frequency resolution $\Delta f = f_s/N$ of about 5 Hz. This results in a value of $N \geq 8820$ for $f_s = 44.1$ kHz.
When using an FIR-filter this would mean 8820 multiply and accumulate (MAC) operations per output sample in contrast to about 50 operations and some overhead for the buffering operations when using the time-reversed IIR approach in double precision.

5. APPLICATION EXAMPLE

The following examples show the application of the proposed correction technique to a commercial HiFi loudspeaker (JBL TI5000). This speaker shows an electrical impedance at the loudspeaker terminals as shown in Fig.2. The three frequencies of interest for this speaker are $f_l = 13.8$ Hz, $f_b = 30$ Hz and $f_h = 49$ Hz. The computed total driver quality factor is $Q_{ts} = 0.31$ and the resonance frequency $f_{sb} = 22.6$ Hz. These values are in good accordance with the data given by the manufacturer with $f_b = 30$ Hz, $f_s = 24$ Hz and $Q_{ts} = 0.29$. From the measured frequency values the additional parameters are computed as $\alpha = 2.32$ and $h = 1.33$, which are reasonable values for a QB3 alignment.

With these values the coefficients of the transfer function $G_v(s)$ can be computed. The original magnitude and group delay responses of the speaker as computed from the data of the impedance measurement are shown in Fig.5. The maximum value of the group

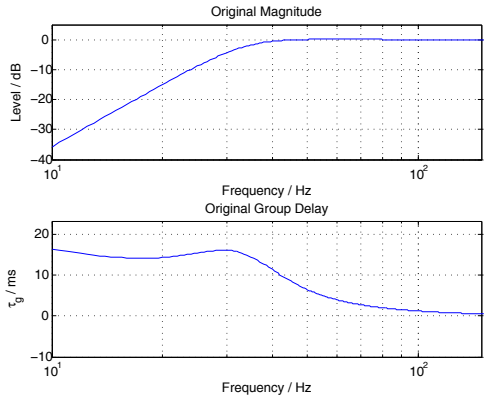


Figure 5: *Original Magnitude and group delay response of the loudspeaker*

delay τ_{gmax} is about 16.5 ms at a frequency of 29.5 Hz, which is close to the measured cabinet resonance frequency of 30 Hz. The -3 dB corner frequency of the system is at about 32.5 Hz.
The resulting pole radii of the discrete-time fourth-order highpass $G_v(z)$ are: 0.99803, 0.99803, 0.99853 and 0.99454. A correction filter has been designed as described in 4.2.2. The pole angles are chosen as $\omega_0 = 2\pi(f_b/f_s)$ with $f_b = 30$ Hz and $f_s = 44.1$ kHz and the pole radii r were determined iteratively as $r = 0.9968$. This leads to the pole-zero configuration as shown in Fig. 3.
In Fig.6 the frequency response of the time-reversed correction filter is shown. The group delay correction is not exact, as expected because the box is a fourth order system and the correction filter a second order system and only models the cabinet resonance. Furthermore the magnitude is unity because of the additional zeros placed at the mirror position of the poles. For evaluation of

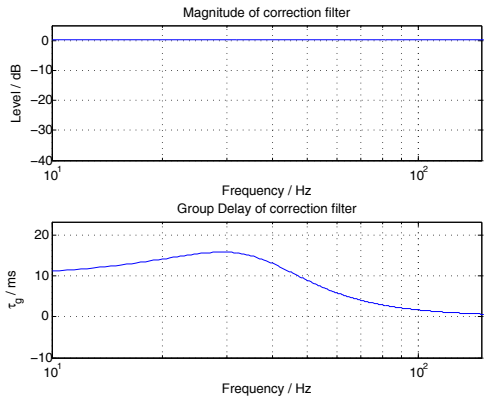


Figure 6: *Magnitude and group delay response of the correction filter before time-reversal*

the equalization performance, the group delay of the correction filter has been subtracted from the original group delay of the loudspeaker. The result is shown in Fig.7. The correction filter does not affect the magnitude response of the speaker but reduces the group delay error of the box in the audible frequency range significantly. The group delay error is about 4 ms at a frequency of 10 Hz, where the magnitude is already at about -35 dB w.r.t. the passband and about -2.4 ms at a frequency of 47 Hz. This error of -2.4 ms is much smaller than the original group delay of 9.3 ms at this frequency before equalization. In [15] it is stated that a delay below 1-2 ms will practically never be noticed and 3-5 ms errors are safe for most program material. At the cabinet resonance frequency of 30 Hz, the group delay is zero as expected.

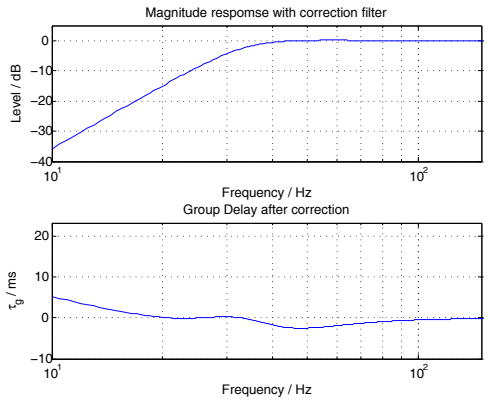


Figure 7: *Magnitude and group delay response of the corrected speaker*

The performance of the filter can be fine tuned by manually adjusting ω_0 and r to further reduce the errors or to adjust the equalization to personal preferences.

5.1. Results

The result has been evaluated in an informal listening test, where the correction was clearly audible for all participants. The low-frequency reproduction gets tighter and more defined. Rhythmic instruments like bassdrums have a better coherence of bass and subbass frequencies and thus are fusing more into one sound. Due to the change introduced by the equalization, the resulting sound is also a little unusual since the listener is in most cases used to listening to uncorrected speakers for a long time.
Another observation is, that the crestfactor of the output signal of the correction filter can change due to the phase shifts in the low-frequency range. To avoid clipping, the level of the output signal may have to be reduced or limited according to the capabilities of the signal processing system.

6. CONCLUSIONS

A way to determine the transfer function and thus the group delay of a vented box in a simple applicable way via an electrical impedance measurement has been shown. The group delay deficiencies of the speaker can be equalized with an FIR-filter, into which further equalization can be incorporated. This method can give very accurate equalization but is computational demanding.
The correction filter can also be designed as a time-reversed allpass by choosing the appropriate resonance frequency and pole radii of

a second order resonator whose magnitude is then corrected with additional zeros. This approach does not account for all sources of unwanted group delay and but delivers good results. Furthermore it allows for a parametric filter design and thus an implementation of a simple real-time control of the delay. Additionally, it reduces the computational load of the filtering process significantly with the cost of introducing some additional delay into the signal path.

7. REFERENCES

[1] Neville Thiele, "Loudspeakers in vented boxes: Part 1," *J. Audio Eng. Soc*, vol. 19, no. 5, pp. 382–392, 1971.

[2] Neville Thiele, "Loudspeakers in vented boxes: Part 2," *J. Audio Eng. Soc*, vol. 19, no. 6, pp. 471–483, 1971.

[3] Richard H. Small, "Vented-box loudspeaker systems–part 1: Small-signal analysis," *J. Audio Eng. Soc*, vol. 21, no. 5, pp. 363–372, 1973.

[4] Richard H. Small, "Vented-box loudspeaker systems-part 4: Appendices," *J. Audio Eng. Soc*, vol. 21, no. 8, pp. 635–639, 1973.

[5] Robert M. Bullock and Robert White, *Bullock on boxes*, Old Colony Sound Laboratory, 1991.

[6] J. Blauert and P. Laws, "Group delay distortions in electroacoustical systems," *Journal of the Acoustical Society of America*, vol. 63, pp. 1478–1483, 1978.

[7] Richard Greenfield and Malcolm J. Hawksford, "The audibility of loudspeaker phase distortion," in *Audio Engineering Society Convention 88*, Mar 1990.

[8] Günter J. Krauss, "On the audibility of group delay distortion at low frequencies," in *Audio Engineering Society Convention 88*, Mar 1990.

[9] Véronique Adam, "Amplitude and phase synthesis of loudspeaker systems," in *Audio Engineering Society Convention 108*, Feb 2000.

[10] Sunil Bharitkar, Tom Holman, and Chris Kyriakakis, "Time-alignment of multi-way speakers with group delay equalization - I," in *Audio Engineering Society Convention 124*, May 2008.

[11] Shintaro Hosoi, Hiroyuki Hamada, and Nobuo Kameyama, "An improvement in sound quality of lfe by flattening group delay," in *Audio Engineering Society Convention 116*, May 2004.

[12] Neville Thiele, "Phase considerations in loudspeaker systems," in *Audio Engineering Society Convention 110*, May 2001.

[13] Veronique Adam and Sebastien Benz, "Correction of crossover phase distortion using reversed time all-pass IIR filter," in *Audio Engineering Society Convention 122*, May 2007.

[14] Siegfried Linkwitz, "Issues in loudspeaker design - 1," http://www.linkwitzlab.com/frontiers.htm.

[15] Matti Karjalainen, Esa Piirilä, Antti Järvinen, and Jyri Huopaniemi, "Comparison of loudspeaker equalization methods based on dsp techniques," *J. Audio Eng. Soc*, vol. 47, no. 1/2, pp. 14–31, 1999.

[16] Jr. W. Marshall Leach, "Vented-box file," http://users.ece.gatech.edu/ mleach/ece4445/index.html.

[17] Jr. W. Marshall Leach, *Introduction to Electroacoustics and Audio Amplifier Design*, Kendall / Hunt Publishing Company, 2003.

[18] Vance Dickason, *Loudspeaker Design Cookbook*, Audio Amateur Pubns, 2005.

[19] Joseph D'Appolito, *Lautsprecher-Messtechnik*, Elektor-Verlag Aachen, 1999.

[20] Markus Lang and Timo I. Laakso, "Simple and robust method for the design of allpass filters using least-squares phase error criterion," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 41, pp. 40–48, 1994.

[21] Hans W. Schüßler, *Digitale Signalverarbeitung 2*, Springer Verlag, Berlin Heidelberg, 2010.

[22] "rephase," http://sourceforge.net/projects/rephase/.

[23] John G. Proakis and Dimitris G. Manolakis, *Digital Signal Processing*, Pearson Prentice Hall, 2007.

[24] S.R. Powell and P.M. Chau, "A technique for realizing linear phase IIR filters," *IEEE transactions on signal processing*, vol. 39(11), pp. 2425–2435, 1991.

8. APPENDIX

The general expression for the group delay of a fourth order Butterworth filter (highpass or lowpass) is

$$\tau_{gB4}(\omega) = \frac{9.28986\,\omega_0\,\omega^6 + 3.84786\,{\omega_0}^3\,\omega^4 + 3.84786\,{\omega_0}^5\,\omega^2 + 9.28986\,{\omega_0}^7}{3.55511\,\omega^8 - 0.000386\,{\omega_0}^2\,\omega^6 + 0.000635\,{\omega_0}^4\,\omega^4 - 0.000386\,{\omega_0}^6\,\omega^2 + 3.55511\,{\omega_0}^8}. \tag{20}$$

The expression for the group delay introduced by a pole pair at $\pm\omega_0$ with both poles having the radius r is

$$\tau_g(r) = -\frac{3r^3\sin^2(2\omega_0) + \left(6r^3\cos^2(\omega_0) - 2r^2\right)\cos(2\omega_0) + \left(2r - 4r^2\right)\cos^2(\omega_0) - 2r^4}{2r^3\sin^2(2\omega_0) + \left(4r^3\cos^2(\omega_0) - 2r^2\right)\cos(2\omega_0) + \left(4r - 4\,r^2\right)\cos^2(\omega_0) - r^4 - 1}. \tag{21}$$

EXPLORING THE VECTORED TIME VARIANT COMB FILTER

Vesa Norilo, *

Centre for Music & Technology,
Sibelius Academy, University of Arts
Helsinki, Finland
vnorilo@siba.fi

ABSTRACT

This paper presents the time variant vectored comb filter. It is an extension of the feedback delay network to time variant and nonlinear domains. Effects such as chorus and flanger, tap delay and pitch shifter are examined in the context of the feedback scheme. Efficient implementation of a stateless vectorizable LFO for modulation purposes is presented, along with a recursive formulation of the Hadamard matrix multiplication. The time variant comb filter is examined in various effect settings, and presented with source code and sound examples.

1. INTRODUCTION

A feedback delay network is a well established for method for implementing efficient synthetic reverberators. The algorithm is a simple yet elegant generalization of the comb filter; the signal and filter parameters are vectored and the feedback attenuation becomes a matrix multiplication.

Different extensions to the comb filter are also ubiquitous. The extensive design space of modulation–delay effects can be seen as comb filter variants. This leads into an intriguing possibility of further generalization, the vectored modulation delay.

This paper explores the addition of vectored delay time and amplitude modulation to the FDN. Effects resembling modulation delay staples such as chorus and flanger are examined and extended. Since all these effects are just parametrizations of the vectored time variant comb filter, various hybrids are also presented.

The fundamentals of feedback delay networks are presented first, in Section 2, *Background*. The generalization into Vectored Time Variant Comb Filters is discussed in Section 3, *Exploring the Design Space*. This section discusses the implementation and applications of the effect as well as efficient implementation of the modulation structure on vector hardware. A summary of the paper is given in Section 5, *Conclusion*.

2. BACKGROUND

The standard comb filter is shown in Figure 1. For high filter orders, it will be perceived as an echo effect. Lower order filters that result in very fast echoes are perceived as frequency response coloration. Comb filters are ubiquitous, especially in artificial reverberation. The traditional design by Schröder[1] employs a bank of these filters, tuned to generate a series of decaying echoes resembling the diffuse reverberation field.

* This work was supported by MuTri Doctoral School, Sibelius Academy, University of Arts Helsinki

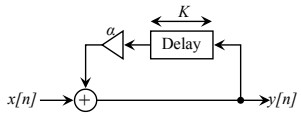


Figure 1: Comb Filter

The seminal work on feedback delay networks for artificial reverberation was done by Gerzon in the 1970s[2]. Since then, the algorithm has become a staple of synthetic reverberation. The overall schematic is similar to the comb filter: the delay and feedback coefficient are vectorized, the feedback gain stage becomes a matrix multiplication.

In contrast to the comb filter bank, each delay line feeds back into several or even all the other delay lines, giving FDN the property of an echo density that increases over time. Real acoustic spaces exhibit a similar property, unlike the constant echo density comb filter bank.

The exact nature of the FDN sound field depends on the properties of the feedback matrix. Much of the research since its discovery has been on tuning the counterintuitive algorithm. Seminal work on the subject has been done by Jot[3]. Rocchesso and Smith present important techniques and constraints for the feedback matrix design, as well as equivalences to classes of digital waveguide networks [4].

Time varying variants of the simple comb filter are also widely used. An overview of these modulation delay effects is in the literature[5]. The contribution of this paper is to explore the combination of these: the vectored, time variant comb filter, and to demonstrate an efficient implementation on SIMD hardware.

3. EXPLORING THE DESIGN SPACE

The example implementation of the vectored time variant comb filter is designed to explore the possibilities of delay and amplitude modulation of significant depth. Typically, modulation techniques in the context of feedback delay networks have been used to break the modes of the reverberator. The analogy to vectored comb filters suggests the possibility of vector chorus, vector flanger and even complicated doubler type effects. Since these are just parametrizations of the filter, hybrid effects combining features of several effects are also potentially interesting.

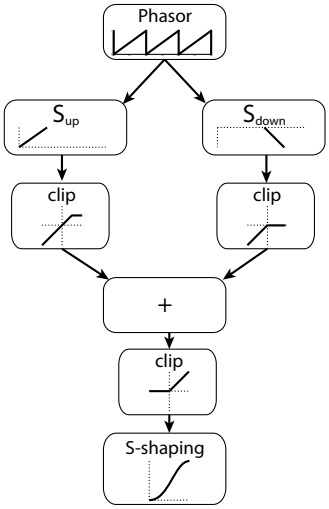


Figure 2: Overview of the waveshaping LFO

An 8-dimensional vectored time variant comb filter is implemented for the purposes of this study. The filter consists of a delay bank with loss filters and a feedback matrix. Two LFOs are provided per delay, one for delay time modulation and one for amplitude modulation.

This design should be easily adaptable for efficient processing on common SIMD units which tend to be 4 or 8 units wide at the time this article was written. The reference implementation can optionally use the Intel AVX instruction set to run most of the comb filter on a parallel SIMD code path. It should be easily adaptable to most similar vector architectures.

3.1. Implementation

3.1.1. Vectorized LFO

This section presents an LFO algorithm capable of producing control waveforms of triangle and square variety, with adjustable symmetry and slopes for ramps and pulses as well. All waveforms can be continuously morphed between linear and pseudosinusoid shape. The oscillator is designed for control signals and is not band limited.

The algorithm is designed for modern hardware and vector processing, which essentially preclude the use of nondeterministic code path or memory access. Wavetables and branch logic are thus out of question. The algorithm is a pure function waveshaper that acts on a simple phasor. Stateful or stateless phasors can be chosen according to the target hardware.

The waveshaper is presented as a cascade of stages following the phasor producing a periodic ramp in the range $[0, 1]$. An overview is given in Figure 2.

The triangle/ramp/square base shape is accomplished by two linear functions and three clipping stages. The base waveform is parametrized by three degrees of freedom, (x_1, x_2, x_3), as shown in Figure 3. The linear functions for S_{up} and S_{down} follow trivially from these points and are given in Equations 1 and 2. For the linear segments to be defined, $x_1 > 0 \wedge x_2 > x_1 \wedge x_3 > x_2$. How small the deltas can be depends on the numerical characteristics of the target hardware.

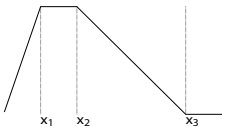


Figure 3: LFO degrees of freedom

The waveform is combined by clipping S_{up} below one and S_{down} below zero. Summing these and clipping above zero yields the final waveform in the unipolar range of $[0, 1]$. S_{down} should be computed in the form $k(x - x_2)$ to preserve numerical precision near zero – the section that will actually be used.

A pseudo-sinusoid waveform can be accomplished by a further waveshaping polynomial (Equation 3). This shaping turns the linear segments in the LFO into S-shape curves that are continuous in the first derivative when applied to a triangle-like wave. A continuous control parameter from linear to pseudo-sinusoid segment can be introduced. All in all, the pseudo-sinusoid shape morphing roughly doubles the computational complexity of the LFO. The S-curve is potentially useful for all of the waveforms: triangle, skewed pulse and ramp.

$$S_{up}(x) = \frac{x}{x_2 - x_1} \tag{1}$$

$$S_{down}(x) = \frac{x - x_2}{x_2 - x_3} \tag{2}$$

$$h(x) = 6(\frac{x^2}{2} - \frac{x^3}{3}) \tag{3}$$

3.1.2. Delay and Filter Bank

The delay and attenuation filter banks used in the effect are straightforward. The filter bank is based on the standard first order loss filter. The delays are implemented as circular buffers.

The one pole filter bank is an easy fit for vector hardware. The same can not be said for the delay bank, due to non-uniform ring buffers. This leads to the memory access pattern requiring a scatter/gather idiom.

The modulation of delay lines makes the signal path nonlinear. This undermines the canonical stability criteria for feedback delay networks. The described modulation is attenuation rather than boost, so an unstable situation is not expected. However, the attenuation effect of amplitude modulation is unpredictable and program dependent. That is why a manual adjustment of feedback beyond 100%, is provided per delay line, with total stability guaranteed by an additional stage for soft saturation.

3.1.3. Feedback Matrix

As in reverberators, a lossless feedback matrix is the starting point. Such matrices are unitary. For the purpose of this study, the orthogonal Hadamard matrix with computationally beneficial features is used. The matrix is generated by taking a N-fold Kronecker product of seed matrices and scaling for orthogonality, as shown in Equation 4. This matrix caters for a network of 2^N delay lines.

$$H_n = \frac{1}{\sqrt{2^N}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{\otimes N} \tag{4}$$

Table 1: Permute–flip–add sequence for 8×8 Hadamard matrix

permute a	0	0	2	2	4	4	6	6
permute b	1	1	3	3	5	5	7	7
sign	+	-	+	-	+	-	+	-
permute a	0	1	0	1	4	5	4	5
permute b	2	3	2	3	6	7	6	7
sign	+	+	-	-	+	+	-	-
permute a	0	1	2	3	0	1	2	3
permute b	4	5	6	7	4	5	6	7
sign	+	+	+	+	-	-	-	-

In the case of two delay lines ($N = 1$), the matrix computation trivially results in a vector containing their sum and difference. A larger feedback matrix can be constructed by computing pairwise feedback vectors, then recursively combining pairs of them by concatenating the vector sum and difference. Each level of recursion corresponds to a Kronecker product, doubling the number of diffuse feedback channels. This algorithm results in $N2^N$ additions, or by the number of delay lines, $nlog_2(n)$ additions as noted in the literature for FDN reverberators[6]. The matrix scaling coefficient $\frac{1}{\sqrt{2^N}}$ can be integrated into the delay line loss filters.

The feedback matrix is also amenable to SIMD computation. Each Kronecker product can be reduced to a vectored permute, sign flip and addition. An example with $N = 3$ is demonstrated in Table 1. Three products are shown. The permute rows a and b correspond to element indices for the left and right hand side of the addition; the sign row denotes sign flips for the right hand side. For architectures with a vector width of 2^N, the entire feedback matrix can be computed in $4N$ vector operations, corresponding to two permutes, sign flip (*xor*) and addition per Kronecker product.

Alternatively, the Hadamard matrix could be vectorized as a time-parallel computation in block processing. This choice could be considered as it saves the permute operations described above; however, it is less appealing due to the matrix appearing in a feedback loop of a modulation delay. The various techniques to work around the latency of such an algorithm would likely cost more than the simple permutation instructions, both in terms of compute efficiency and algorithmic complexity.

3.1.4. Control Surface and Parameter Mapping

The internal parameter set used for each delay line in the effect is shown in Table 2. A one to one mapping from the internal parameter set to a user interface is not likely very attractive. For eight delay lines, the interface would contain 144 parameters. Macro controls would be more useful; this should be studied in the future.

3.2. Applications and Qualitative Evaluation

This section briefly discusses some of the creative possibilities of using the effect described in this study. The evaluations are the subjective impressions of the author; they shouldn't be read as scientific results. For a more detailed perspective, please refer to the example code and sound files that are available at the code repository specified at the conclusion of this paper.

Table 2: Time Variant Vectored Comb parameter set

parameter	**unit**	**description**
delay	ms	delay time
in gain	dB	input signal to delay line
out gain	dB	delay line to output signal
out pan	linear	stereo panorama
tone	linear	loss filter to feedback matrix
fb gain	%	delay line to feedback matrix
LFO rate	Hz	
DM depth	ms	delay time modulation
DM offset	linear	DM phase offset
DM shape	$4\times$ linear	$x_1, x_2, x_3, shape$
AM depth	linear	amplitude modulation
AM offset	linear	AM phase offset
AM shape	$4\times$ linear	$x_1, x_2, x_3, shape$

3.2.1. Vector Chorus–Flanger

The vectored chorus–flanger revolves around delay times and delay modulation depth of $0 - 40$ ms and LFO rates in the range of $0.1 - 5$ Hz. Adding feedback creates a flanger-like moving resonance effect, but the tonal color is a lot more complicated as the feedback network system has a large number of poles.

With longer delay times and feedback, the effect acquires spring reverb characteristics, especially with faster LFO rates.

Complex stereo imaging can be achieved by using variations of similar settings on multiple delay lines and panning them across the image.

3.2.2. Multitap Delay

By using delay times a lot longer than those in a diffuse field reverberator, a sparse multitap delay effect is created. What is especially interesting is the echo density escalation over time. The sparse echoes gradually become a diffuse tail. The effect is capable of generating interesting transitions from percussive textures to static ones.

3.2.3. Pitch Shifter

By using ramp-shape delay time modulation together with phase shifted triangular amplitude modulation results in a simple pitch shifter. The ramp modulator adjusts momentary playback speed, while the triangular envelope is aligned to hide the discontiuity in the ramp. An even overrall amplitude can be attained by using overlapping shifters with orthogonal phase shifts.

The complex feedback is the distinguishing feature from standard slicer shifters. The effect is less useful as a plain transposition, as an infinite number of high order transpositions are generated by the feedback, but can result in extremely full ensemble thickening effects with subtle pitch shift factors.

3.2.4. Hybrid Effects

Interesting combinations of effects can be realized by mixing delay line settings from several of the above categories. Reverb-like settings together with pitch shifter and chorus effects appear to be the most immediately useful.

3.2.5. Semi-Stable Self Oscillation

By utilizing very short delay times in the range of $0 - 20$ ms and feedbacks in excess of 100%, a self-oscillating network can be created. Soft saturation in the feedback loop prevents the blow-up and introduces both harmonics and non-harmonic aliasing frequencies. The tonalities due to complex feedback paths are interesting, but the pitch is quite hard to predict and control.

4. PERFORMANCE EVALUATION

4.1. LFO

The performance of the LFO reference implementation was measured by accumulating its output over $100\,000\,000$ sample frames to ensure timing accuracy. The accumulator is in place to prevent dead code optimization by the compiler. A vectorized LFO with eight independent waveforms was measured. The test program was compiled with Microsoft Visual Studio 2013, with AVX architecture and the fast floating point model enabled. The measurement was run on Windows 7 with the dual core Intel i5-3317U CPU clocked at 1.70GHz.

The vectorized LFO was able to produce 1.14286×10^9 output frames of 8 discrete signals per second – roughly 16 CPU cycles per frame, or 2 cycles per sample. This translates to a real time CPU core utilization of 0.0039% per modulation signal on the machine the measurement was performed on, when processed at 44.1kHz.

4.2. Time Variant Vectored Comb Filter

The entire effect consists of the following modules:

1. delay bank (8)
2. modulation LFOs (16)
3. loss filter bank (8)
4. feedback matrix (8×8)
5. input and output routing matrices

The loss filter, modulation LFOs and all gain and summation stages trivially vectorize to SIMD code. The feedback matrix is also fully vectorized, as explained in Section 3.1.3. The modulation delay bank remains scalar, as the requisite scatter/gather operations defeat the purpose of vectorization on current hardware.

In a test harness like the one described in 4.1, the entire comb filter implementation produced 3.57×10^6 stereophonic output frames per second. This translates to a real time CPU core utilization of 1.23% at 44.1kHz. Roughly 80% of the time is spent in the scalar delay line bank. This suggests that the additional computational load from a comprehensive feedback matrix, in contrast to a plain modulation delay bank, is far from prohibitive in the context of suitable vector hardware.

5. CONCLUSION

This paper examined the extension of feedback delay networks into the realm of modulation delay effects. Efficient vectorized implementation of the parallel modulation structure and a diffusive feedback matrix were demonstrated.

The generalized time variant vectored comb filter is interesting in the sense that it is a superset of a large number of delay-based effects. It offers musically relevant and divergent possibilities when complicated feedback structures are used. In particular, hybrids between spatial and ensemble effects offer novel sounds. Continuous morphing from one effect state to another is also easily attainable.

Possible future work could involve a deeper investigation of the feedback matrix. The current implementation uses a fixed feedback matrix for maximum efficiency. Control over the diffusion between the submatrices of the Hadamard tree could be especially interesting, as it could be seen as a way to isolate or combine sections of the network. The impact of advancing scatter/gather implementations could be interesting in improving the performance of the scalar delay bank. The user interface is also an open question: the exposal of the extensive parameter set via a higher level control surface could increase the viability of the effect from the end user point of view.

A reference implementation of the effect in C++, along with sound examples, is available on Bitbucket under the MIT license, at `https://bitbucket.org/vnorilo`.

6. ACKNOWLEDGEMENTS

The work of the author was supported by the MuTri doctoral school, Sibelius Academy, University of Arts Helsinki. The vector image in Figure 1 is sourced from WikiMedia under the creative commons public domain license.

7. REFERENCES

[1] M R Schroeder, "Digital Simulation of Sound Transmission in Reverberant Spaces," *Journal of the Acoustical Society of America*, vol. 45, no. 1, pp. 303, 1969.

[2] M A Gerzon, "Unitary (energy-preserving) multichannel networks with feedback," *Electronics Letters*, vol. 12, no. 11, pp. 278–279, 1976.

[3] Jean-Marc Jot and Antoine Chaigne, "Digital Delay Networks for Designing Artificial Reverberators," in *the 90th AES Convention*, 1991, vol. 3030, p. preprint no. 3030.

[4] Davide Rocchesso and Julius O Smith, "Circulant and elliptic feedback delay networks for artificial reverberation," *Speech and Audio Processing, IEEE Transactions on*, vol. 5, no. 1, pp. 51–63, 1997.

[5] P Dutilleux, M Holters, Sascha Disch, and Udo Zölzer, "Filters and delays," in *DAFX: Digital Audio Effects*, Udo Zölzer, Ed., pp. 47–82. John Wiley & Sons,Inc., 2nd edition, 2011.

[6] Davide Rocchesso, "Maximally Diffusive Yet Efficient Feedback Delay Networks for Artificial Reverberation," *IEEE Signal Processing Letters*, vol. 4, no. 9, pp. 252–255, 1997.

TIME-VARYING FILTERS FOR MUSICAL APPLICATIONS

Aaron Wishnick

iZotope, Inc.,
Cambridge, Massachusetts, USA
awishnick@izotope.com

ABSTRACT

A variety of methods are available for implementing time-varying digital filters for musical applications. The considerations for musical applications differ from those of other applications, such as speech coding. This domain requires realtime parametric control of a filter such as an equalizer, allowing parameters to vary each sample, e.g. by user interaction, a low-frequency oscillator (LFO), or an envelope. It is desirable to find a filter structure that is time-varying stable, artifact-free, computationally efficient, easily supports arbitrary filter shapes, and yields sensible intermediate filter shapes when interpolating coefficients. It is proposed to use the state variable filter (SVF) for this purpose. A novel proof of its stable time-varying behavior is presented. Equations are derived for matching common equalizer filter shapes, as well as any z-domain transfer function, making the SVF suitable for efficiently implementing any recursive filter. The SVF is compared to state of the art filter structures in an objective evaluation and a subjective listening test. The results confirm that the SVF has good audio quality, while supporting the aforementioned advantageous qualities in a time-varying digital filter for music. They also show that a class of time-varying filter techniques useful for speech coding are unsuitable for musical applications.

1. INTRODUCTION

In digital audio effects, filters are rarely time-invariant. A filter is time-variant if it has a user-controllable parameter. A time-variant filter is also a useful building block for an effect such as a phaser or filter controlled by an LFO or envelope. For these applications, it is important that the filter remain stable, and that the time-varying behavior not introduce perceptible artifacts. Here, we focus on realtime musical applications, where parameters may be varied every sample, as with an LFO, and it should be computationally efficient to do so. An ideal method will allow implementation of any filter shape. As the parameter changes may be smoothed, the transfer functions resulting from the intermediate coefficients should maintain a similar magnitude response to the shapes being interpolated. This study is restricted to second-order recursive filters because higher order filters are typically decomposed into second-order sections.

The choice of filter structure has a large influence on time-varying behavior, including whether the filter will remain stable. Even stable filters can still produce objectionable artifacts, as will be shown in Sec. 6.

In order to implement time-varying filters, given a desired transfer function, one option is to select a time-varying stable filter structure, and configure this structure to realize the transfer function. Another option is to use a time-varying unstable structure such as Direct-Form II transposed, and stabilize it.

A variety of approaches have been proposed to improve time-varying behavior. One category of methods is transient suppression [1] [2] [3], and another is stabilization [4] [5]. These are discussed in more detail in Sec. 2.

One filter structure that is often used to realize realtime, per-sample time-varying behavior is the state variable filter (SVF). Empirically, it is known to remain stable and artifact-free, but these properties have not previously been proven. A proof of time-varying stability will be shown here. By taking the output of this filter from different nodes, it is possible to obtain second-order lowpass, bandpass, or highpass filters. The SVF also maps intuitively to common audio equalization filters, providing independent control over frequency and resonance, which results in a low computational burden. Due to this relation, directly interpolating SVF coefficients also tends to result in sensible intermediate filter shapes, unlike some other structures [6]. Here we will also show how to choose SVF coefficients to realize any desired transfer function. Thus, the SVF satisfies the desired qualities of a time-varying filter for musical applications.

Prior approaches to time-varying filtering are reviewed in Sec. 2. In Sec. 3, we review the proposed digital implementation of the SVF. In Sec. 4, we derive formulas for using the SVF to realize some common filter types for audio equalization, and generally, any second-order z-domain transfer function. This allows the SVF to be easily used to implement any digital filter. In Sec. 5, the time-varying stability of this structure is proven. In Sec. 6, the time-varying behavior is compared with state of the art methods in an objective evaluation of DC response, as well as a subjective listening test, which confirms its good audio quality, and suggests criteria for perceptually good time-varying behavior.

2. PRIOR WORK

A variety of filter structures have been studied in the time-varying case, e.g. [5] [4]. Structures such as Direct-Form II, lattice, and normalized ladder are not necessarily stable when coefficients are changed. Coupled form, also known as normal form or Gold and Rader [7] [8], is stable in the time-varying case. Stability here refers to bounded-input/bounded-output (BIBO) stability [5], meaning that the output of the filter will be bounded so long as the input is bounded.

In addition to these structures, there are several methods for stabilizing a filter or eliminating transients from it. Consider a change from state space matrix $\mathbf{S_1}$ to $\mathbf{S_2}$ at time $n = m$. Let $y_1[n]$ be the output when filtering the entire signal with $\mathbf{S_1}$ and $y_2[n]$ be the output when using $\mathbf{S_2}$. The *output switching* model [1] [3] has the ideal response of

$$y[n] = \begin{cases} y_1[n] & : n < m \\ y_2[n] & : n >= m \end{cases} \quad (1)$$

2.1. Transient Minimization

Transient minimization techniques consider the *transient signal*, defined as the difference between the actual output signal, and the output switching model from equation (1). Transient minimization techniques decrease this transient signal.

Zetterberg and Zhang [1] propose a method, motivated by LPC-based speech coding, that realizes equation (1). It works by recomputing the state vector, but this requires the entire input signal to achieve this, making it unsuitable for realtime use. Välimäki and Laakso [3] propose an approximation to Zetterberg-Zhang which only requires a finite signal history, allowing realtime usage. However, this method is designed for sparsely occurring coefficient changes. Supporting audio-rate coefficient changes, e.g. when modulating a filter with a LFO, would require many filters running at once, making it computationally prohibitive for this application.

Rabenstein [2] uses an intermediate set of coefficients, which minimizes the variance of the transient signal. This method is also intended for coefficient changes that are spaced far apart in time.

2.2. Stabilization

While transient minimization deals with correcting a filter's output, stabilization allows use of a filter structure that is ordinarily not stable when time-varying, by forcing it to stay stable.

Rabenstein and Czarnach [4] present a method of transforming the state vector to stabilize any filter structure. It works by relating a filter in its state space structure to the coupled form. This can be performed every sample, making it suitable for audio-rate coefficient changes. It can be incorporated into the coefficient matrix, allowing the filtering operation to incur no additional cost, while making coefficient computation more costly.

3. THE STATE VARIABLE FILTER

3.1. State Space Form

The continuous-time state variable filter in state space form [9] has the differential equation

$$\begin{aligned} \dot{x_1} &= u - 2Rx_1 - x_2 \\ \dot{x_2} &= x_1 \end{aligned} \tag{2}$$

where x_1, x_2 are the state variables, and u is the input to the filter. The $\dot{x}$ superscript indicates a time derivative. The parameter R controls the resonance, and this continuous-time formulation places the center frequency at unity, i.e. it is normalized. It will be useful to render this system in matrix form, so that

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3}$$

In this case, we have

$$\mathbf{A} = \begin{bmatrix} -2R & -1 \\ 1 & 0 \end{bmatrix} \tag{4a}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T \tag{4b}$$

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \tag{4c}$$

$$\mathbf{u} = \begin{bmatrix} u \end{bmatrix} \tag{4d}$$

3.2. Bilinear Transform

We will apply the bilinear transform to obtain a discrete-time filter. This is equivalent to trapezoidal integration, preserves stability, and maps the entire continuous frequency axis to the discrete-time frequency axis [10] [11].

In audio signal processing literature, cases are encountered where the application of the bilinear transform to a continuous-time filter results in a delay-free loop: a feedback loop that contains no delay elements, where the state at time n appears to depend on itself instantaneously. For example, Smith [9] and Dutilleux [12] both remark that the bilinear transform cannot be used with the SVF for this reason. The Chamberlin filter structure is another discretization of the SVF, using Forward Euler and Backward Euler integrators [13] [9] [14], but this structure becomes unstable for some parameters.

These difference equations are actually implementable with some extra computation. The K-Method [10] [15] is an algebraic method for discretizing and solving systems in state space form, and Zavalishin [16] presents a graphical method that is equivalent, which is also applied to the SVF.

The K-Method involves writing a difference equation for the integrator to be used and then substituting the system to be simulated in state space form into that difference equation. Delay free loops are handled by solving the resulting system, which will be linear in this case.

3.3. Discretization

We apply the K-Method using a Direct-Form II transposed (TDF-II) trapezoidal integrator [11], which is the same integrator as used in [16]. This form is canonical with respect to delay. Introducing $\mathbf{s}$ as the state vector, the TDF-II trapezoidal integrator update rule is

$$\mathbf{x_n} = g\dot{\mathbf{x}}_\mathbf{n} + \mathbf{s_{n-1}} \tag{5a}$$

$$\mathbf{s_n} = \mathbf{s_{n-1}} + 2g\dot{\mathbf{x}}_\mathbf{n} \tag{5b}$$

where the coefficient g is chosen to map a specific analog frequency $w_a = 2\pi f_a$ to a digital frequency $w_c = 2\pi f_c$, at a sampling rate $f_s = \frac{1}{T}$, known as *prewarping* [11]:

$$g = \frac{\tan(\pi T f_c)}{w_a} \tag{6}$$

We can substitute the state space formulation from equation (3) into the integration and update rules from equation (5) to discretize an arbitrary continuous-time state space system. This is similar to how the K-Method is used in [10] and [15], except that we use the TDF-II realization of the trapezoidal integrator, instead of DF-I. Solving for $\mathbf{x_n}$ and $\mathbf{s_n}$, we have

$$\mathbf{H} = (\mathbf{I} - g\mathbf{A})^{-1} \tag{7a}$$

$$\mathbf{x_n} = g\mathbf{H}\mathbf{B}\mathbf{u_n} + \mathbf{H}\mathbf{s_{n-1}} \tag{7b}$$

$$\mathbf{s_n} = \mathbf{s_{n-1}} + 2g\mathbf{A}\mathbf{x_n} + 2g\mathbf{B}\mathbf{u_n} \tag{7c}$$

This is how the K-Method handles delay-free loops: upon substituting (3) into (5), $\mathbf{x_n}$ appears on both sides of the equation. The matrix inverse $\mathbf{H}$ is used to solve this linear system, making it explicit in $\mathbf{x_n}$.

Now we use (7) to implement the SVF. By substituting the SVF state space matrices from equation (4) into this TDF-II trapezoidal integration rule (7), we obtain a discrete-time realization of the SVF:

$$\mathbf{H} = \frac{1}{g^2 + 2Rg + 1} \begin{bmatrix} 1 & -g \\ g & 2Rg + 1 \end{bmatrix} \tag{8a}$$

$$\mathbf{x_n} = g\mathbf{H} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u_n} + \mathbf{H}\mathbf{s_{n-1}} \tag{8b}$$

$$\mathbf{s_n} = \mathbf{s_{n-1}} + 2g \begin{bmatrix} -2R & -1 \\ 1 & 0 \end{bmatrix} \mathbf{x_n} + 2g \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u_n} \tag{8c}$$

By expanding the individual expressions for $x_1[n]$ and $x_2[n]$, the elements of $\mathbf{x}$, it can be verified that this is the same as the discrete-time model of the SVF in [16], where $x_1 = y_{BP}$ and $x_2 = y_{LP}$. The filter can now be implemented by computing equations (8) in order.

3.4. Alternate Implementation

If the filter is implemented with equations (8), the integrator outputs are directly available as the elements of $\mathbf{x_n}$, but additional algebra is required if the integrator inputs $\dot{\mathbf{x}}_\mathbf{n}$ are desired.

It is possible to realize the same filter topology by first computing $\dot{\mathbf{x}}_\mathbf{n}$ as an intermediate variable. A general form can be found by substituting equation (7b) into equation (3):

$$\dot{\mathbf{x}}_\mathbf{n} = (g\mathbf{AHB}+\mathbf{B})\mathbf{u_n}+\mathbf{AHs_{n-1}} \tag{9}$$

If this alternate realization is used, first equation (9) is computed, and then the generic TDF-II trapezoidal integration rule from equation (5) is used. Note that equation (5) does not depend on the specific state space matrices $\mathbf{A}$ or $\mathbf{B}$, only on the integrator gain g.

This realization will be convenient for the next section, where $\dot{x}_1$ will be needed. It can be verified that $\dot{x}_1$ is the same as y_{HP} from [16].

4. REALIZING OTHER FILTER TYPES

One useful property of the SVF is that various transfer functions can be obtained by taking the output from different nodes, as demonstrated in [12] and [16]. Most directly, x_1 is a bandpass filter, x_2 is a lowpass filter, and $\dot{x}_1$ is a highpass filter:

$$H_{HP}(s) = \frac{s^2}{s^2+2Rs+1} \tag{10a}$$

$$H_{BP}(s) = \frac{s}{s^2+2Rs+1} \tag{10b}$$

$$H_{LP}(s) = \frac{1}{s^2+2Rs+1} \tag{10c}$$

When the filter is digitally implemented, $x_1[n]$ and $x_2[n]$ are available as elements of the state vector $\mathbf{x_n}$. If the alternate implementation from equation (9) is used, $\dot{x}_1[n]$ is immediately available as well, otherwise it can be computed from equation (2).

Zavalishin [16] presents some ways of combining these outputs to produce other filter types, such as band-shelving, notch, and allpass. However, difficulty is noted in producing other shapes, such as low- and high-shelf filters. Here we will demonstrate how to obtain these shapes, as well as others that are useful for audio equalization.

4.1. Filters for Audio Equalization

Some common filter types are presented in [17]. To enable broad applicability of the SVF, we will show how to implement some of these filters. The technique presented here is suitable to be used with other s-domain filter design methods, e.g. [18]. The filter types that we will implement are presented as continuous-time transfer functions with unity cutoff in Table 1.

These filters share some parameters: Q controls the filter resonance, and $A = 10^{G/40}$ controls the gain, where G is the gain in decibels.

Table 1: *Filters for audio equalization.*

Type	Transfer Function
Lowpass	$H(s) = \frac{1}{s^2+1/Qs+1}$
Bandpass	$H(s) = \frac{s}{s^2+1/Qs+1}$
Highpass	$H(s) = \frac{s^2}{s^2+1/Qs+1}$
Peaking	$H(s) = \frac{s^2+A/Qs+1}{s^2+1/AQs+1}$
Low Shelf	$H(s) = A\frac{s^2+\sqrt{A}/Qs+A}{As^2+\sqrt{A}/Qs+1}$
High Shelf	$H(s) = A\frac{As^2+\sqrt{A}/Qs+1}{s^2+\sqrt{A}/Qs+A}$

The general strategy is to write the desired transfer function as a linear combination of the lowpass, bandpass, and highpass transfer functions from equation (10). This requires adjusting the resonance parameter R and the trapezoidal integrator coefficient g to scale the filter to the correct frequency. The general form is

$$H(s) = c_{HP}H_{HP}(ks)+c_{BP}H_{BP}(ks)+c_{LP}H_{LP}(ks) \tag{11}$$

The lowpass, bandpass, and highpass filters can be obtained trivially by picking $R = 1/2Q$. For the rest of the filters, it is necessary to solve for R, and possibly to use frequency scaling as in [16]. Frequency scaling maps an analog frequency w_a to the digital frequency w_c, using equation (6), with $w_a = k$. The SVF denominator can be made equal to the target transfer function by manipulating R and k in this way, and then the numerator can be matched by choosing c_{HP}, c_{BP}, and c_{LP}.

This strategy is used to generate filter coefficients, which are displayed in Table 2. In addition to the filter type, each filter is controlled by the critical frequency w_c, the resonance Q, and possibly the gain A. For the SVF, compute $2R$ and k according to the table, and use k to compute the integrator gain g from equation (6).

Then, process a sample through the filter, and combine the signals using the gains c_{HP}, c_{BP}, and c_{LP} to form the output:

$$y[n] = c_{HP}\dot{x}_1[n]+c_{BP}x_1[n]+c_{LP}x_2[n] \tag{12}$$

Table 2: *Filter coefficients*

Type	$2R$	k	c_{HP}	c_{BP}	c_{LP}
Lowpass	$1/Q$	1	0	0	1
Bandpass	$1/Q$	1	0	1	0
Highpass	$1/Q$	1	1	0	0
Peaking	$1/Q$	1	1	A/Q	1
Low Shelf	$1/Q$	$\sqrt{A}$	1	A/Q	A^2
High Shelf	$1/Q$	$1/\sqrt{A}$	A^2	A/Q	1

4.2. Arbitrary Digital Filters

In the previous section, common audio equalization filters, designed by applying the bilinear transform to a continuous-time transfer function with unity cutoff, were matched with the SVF. This technique is generally applicable when given a continuous-time transfer function. There are a variety of other representations available for a digital filter, but some filter design methods, e.g.

those of Berchin [19] and Christensen [20] operate directly in the digital domain, yielding transfer function coefficients. Though the filter was discretized with the bilinear transform, it can be used to realize any second-order filter. Therefore, we now derive SVF filter coefficients for arbitrary second-order digital filters, to enable use of these techniques.

4.2.1. Discrete-Time SVF Transfer Functions

First we must derive discrete-time transfer functions for the SVF. This can be done by substituting $s \leftarrow \frac{1}{g}\frac{1-z^{-1}}{1+z^{-1}}$ for each of the transfer functions from equations (10), in order to apply the bilinear transform in the z-domain, or with the Z-transform. Solving, we find

$$H_{HP}(z) = \frac{1-2z^{-1}+z^{-2}}{1+g^2+2Rg+(2g^2-2)z^{-1}+(1+g^2-2Rg)z^{-2}} \tag{13a}$$

$$H_{BP}(z) = \frac{g-gz^{-2}}{1+g^2+2Rg+(2g^2-2)z^{-1}+(1+g^2-2Rg)z^{-2}} \tag{13b}$$

$$H_{LP}(z) = \frac{g^2+2g^2z^{-1}+g^2z^{-2}}{1+g^2+2Rg+(2g^2-2)z^{-1}+(1+g^2-2Rg)z^{-2}} \tag{13c}$$

4.2.2. Matching a Z-domain Transfer Function

Now that we have the discrete-time SVF transfer functions, we want to choose coefficients to match a given second-order digital filter. Without loss of generality, let that filter be specified as

$$H(z) = \frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}} \tag{14}$$

Like equation (11), we will write the desired filter as a linear combination of the three transfer functions (13):

$$H(z) = c_{HP}H_{HP}(z)+c_{BP}H_{BP}(z)+c_{LP}H_{LP}(z) \tag{15}$$

Thus, there are five coefficients as input, and five degrees of freedom to match that filter: g, R, c_{HP}, c_{BP}, and c_{LP}. To find these parameters, we normalize the SVF transfer functions (13) by dividing the numerators and denominators by $1+g^2+2Rg$, and then set equation (15) equal to (14). This system can be solved by setting all the coefficients of z, as well as the constant terms in the numerator, equal. Solving for positive g and R, we find

$$g = \frac{\sqrt{-1-a_1-a_2}}{\sqrt{-1+a_1-a_2}} \tag{16a}$$

$$R = \frac{a_2-1}{\sqrt{-1-a_1-a_2}\sqrt{-1+a_1-a_2}} \tag{16b}$$

$$c_{HP} = \frac{b_0-b_1+b_2}{1-a_1+a_2} \tag{16c}$$

$$c_{BP} = -\frac{2(b_0-b_2)}{\sqrt{-1-a_1-a_2}\sqrt{-1+a_1-a_2}} \tag{16d}$$

$$c_{LP} = \frac{b_0+b_1+b_2}{1+a_1+a_2} \tag{16e}$$

Note that it is possible for both square roots to be purely imaginary, but the imaginary parts will cancel when they are multiplied or divided, yielding real numbers. Using these coefficients, it is possible to design a digital filter using any design method, decompose it into second-order sections, as noted in [16], and then realize the filter using the SVF. This allows the time-varying stability and artifact-free behavior of the SVF to be used for any digital filter.

5. STABILITY

Stability is more complex in the time-variant case. This topic is treated thoroughly by Laroche [5]. To summarize, a time-variant filter that has the coefficients of a stable time-invariant filter at each point in time may still be unstable. There are stricter criteria for time-variant filters, two of which are presented in [5]. Here we will prove the stability of the TDF-II realization of the SVF.

5.1. Transition Matrix

The stability criteria apply only to the state transition matrix, which describes the linear contribution of the state vector from time $n-1$ to time n. To derive this matrix, solve for $\mathbf{s}_n$ in terms of $\mathbf{s}_{n-1}$ by substituting equation (7b) into equation (7c):

$$\mathbf{s_n} = \mathbf{P}\mathbf{s_{n-1}} + (2g^2\mathbf{AHB}+2g\mathbf{B})\mathbf{u_n} \tag{17}$$

$$\mathbf{P} = (\mathbf{I}+2g\mathbf{AH}) \tag{18}$$

where $\mathbf{P}$ is the state transition matrix. Next, substitute in the SVF matrices from (4) to find the SVF state transition matrix.

$$\mathbf{P} = \frac{1}{g^2+2Rg+1}\begin{bmatrix} 1-g^2-2Rg & -2g \\ 2g & 1-g^2+2Rg \end{bmatrix} \tag{19}$$

Criterion 1 presented in [5] immediately fails for this matrix. The criterion is that there exist a real constant $0 \le \gamma < 1$ such that $\|\mathbf{P}\| \le \gamma$, where the standard Euclidean matrix norm is used. Assuming $g > 0$ and $R > 0$, it can be seen that $\|\mathbf{P}\| = 1$.

Instead, we must use Criterion 2 from [5], which requires a change of basis matrix $\mathbf{T}$. Then, the criterion is that there exist a real constant $0 \le \gamma < 1$ such that $\|\mathbf{TPT}^{-1}\| \le \gamma$. This approach was attempted but not completed in [21].

5.2. Change of Basis

Pick a change of basis matrix of the form $\mathbf{T} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$. We will show that it is possible to pick $k > 0$ such that g and R can take on an arbitrarily large range. This work is done with the aid of a computer algebra system, Mathematica version 8.0.1.0 (Wolfram Research, Inc.; 2011), and some intermediate results will be omitted for brevity. Throughout, the assumptions $g > 0$, $R > 0$, and $k > 0$ will be used.

First, solve for $\|\mathbf{TPT}^{-1}\|$. The resulting expression is very long, so it is omitted here. To simplify, make the substitutions $\alpha = \frac{\sqrt{16+4k^2+k^4}}{2k}$ and $\beta = \frac{4+k^2}{2k}$, and solve for the stable region of parameter values g and R, and coefficient k, where $\|\mathbf{TPT}^{-1}\| < 1$. This stable region is the union of the following inequalities:

$$(g \le \beta-\alpha \wedge \tfrac{k}{2} < R < \beta) \tag{19a}$$

$$(\beta-\alpha < g < 1 \wedge (\frac{k}{2} < R < \frac{g^2+1}{2g} \vee \frac{g^2+1}{2g} < R < \beta)) \tag{19b}$$

$$(g = 1 \wedge (\tfrac{k}{2} < R < 1 \vee 1 < R < \tfrac{k^2+2}{2k} \vee \tfrac{k^2+2}{2k} < R < \beta)) \tag{19c}$$

$$(1 < g < \alpha+\beta \wedge (\tfrac{k}{2} < R < \tfrac{g^2+1}{2g} \vee \tfrac{g^2+1}{2g} < R < \beta)) \tag{19d}$$

$$(g = \alpha+\beta \wedge \tfrac{k}{2} < R < \tfrac{g^2+1}{2g}) \tag{19e}$$

$$(g > \alpha+\beta \wedge \tfrac{k}{2} < R < \beta) \tag{19f}$$

$$(\beta-\alpha < g < \alpha+\beta \wedge R = \tfrac{g^2+1}{2g}) \tag{19g}$$

$$(g = 1 \wedge R = \tfrac{k^2+2}{2k}) \tag{19h}$$

Next, we want to show that for any choice of g_{min}, g_{max}, R_{min}, and R_{max}, the region $0 < g_{min} < g < g_{max}$, $0 < R_{min} < R < R_{max}$ is included in these inequalities, so that the filter is always stable. Split the inequalities into three cases: $g < 1$, $g = 1$, and $g > 1$.

For $g > 1$, consider the union of (19d) and (19g). Inspection of $\beta - \alpha$ reveals that it attains a maximum of $2 - \sqrt{3}$ at $k = 2$. Therefore we have $\beta - \alpha < 1$, so the union contains the region

$$1 < g < \alpha + \beta \wedge \frac{k}{2} < R < \beta \tag{20}$$

For $g < 1$, consider the union of (19b), (19a), and (19g). For $k > 0$, we have $\alpha + \beta > 1$, so this union contains

$$0 < g < 1 \wedge \frac{k}{2} < R < \beta \tag{21}$$

Finally, for $g = 1$, use (19c), (19h), and (19g). Substituting $g = 1$ into (19g) and combining, we find that the union of these three inequalities contains

$$g = 1 \wedge \frac{k}{2} < R < \beta \tag{22}$$

Combining (20), (21), and (22), we see that the filter is stable in the region

$$0 < g < \alpha + \beta \wedge \frac{k}{2} < R < \beta \tag{23}$$

Note that equations (19e) and (19f) have not been used; it will be seen that stability where $g \geq \alpha + \beta$ is unnecessary.

To prove stability over the entire range of parameters, note that as $k \to 0$, $\alpha + \beta \to \infty$, and $\beta \to \infty$. Therefore, for any choice of $g_{min} > 0$, g_{max}, $R_{min} > 0$, and R_{max}, it is possible to choose a $k > 0$ to simultaneously satisfy $\alpha + \beta > g_{max}$, $\frac{k}{2} < R_{min}$, and $\beta > R_{max}$. In other words, the parameters g and R can be allowed to vary over an arbitrarily large range, and the filter will remain stable in the time-variant sense, by Criterion 2 of [5].

6. EXPERIMENTS

The SVF is compared to state of the art time-varying filter structures in objective and subjective tests, in order to evaluate its quality with respect to artifacts. The code, audio files, and data associated with the experiments are available online[1].

Both tests are composed of five trials. In each trial, a different filter shape is used, with a single discontinuous parameter change. Within each trial, different filter structures are compared. In this way, the effects of parameter changes on different filter structures can be evaluated. A constant gain was applied across all excerpts within each trial, to normalize peak levels. The filters for each trial are listed in Table 3.

Table 3: *Filter parameters for each trial*

Trial	Filter	Frequency (Hz)	Q	Gain (dB)
1	Lowpass	80 to 120	6	n/a
2	Lowpass	100	0.6 to 4	n/a
3	Peaking	80 to 120	6	4
4	Peaking	100	6	-4 to 4
5	Peaking	120	0.6 to 4	4

[1] `https://github.com/iZotope/time_varying_filters_paper`

The filter structures compared are Direct-Form II (DF2), coupled form (GR), SVF, SVF using Rabenstein's transient minimization [2] (SVFR), SVF using Rabenstein and Czarnach's stabilization [4] (SVFRC), TDF-II (TDF2), TDF-II using Rabenstein and Czarnach's stabilization (TDF2RC), and output switching (ZZ). Note that the SVF is already stable, but transient minimization is used to evaluate the perceptual impact of this technique, and stabilization is used for comparison against the stabilized TDF-II structure. Although DF-II and TDF-II are not stable, and Zetterberg-Zhang is not suitable for realtime, continuously varying parameters, they are included as points of comparison. Välimäki-Laakso is not included because it is an approximation to Zetterberg-Zhang, so the results are likely to be similar.

6.1. Objective Evaluation

One possible way to objectively evaluate the quality of a filter when parameters vary with time is to analyze the response during steady state DC. The response can be measured by supplying constant DC to a filter's input until the filter reaches a steady state, and then changing the parameters instantaneously while continuing to pass DC. If the DC gain does not change, then there should be no change in the output, which corresponds to the output switching model. Note that if the gain does change, output switching may not be the perceptually best ideal. This will be shown in the subjective evaluation.

This evaluation method was mentioned by Berners [22], and a plot demonstrating it is present at [23], in order to show the suitability of a particular filter structure.

Fig. 1 compares the SVF used as a lowpass filter against a DF-II realization of the same transfer function. It can be seen that the SVF performs ideally, while the DF-II realization exhibits a large transient followed by ripple until it settles back into a steady state.

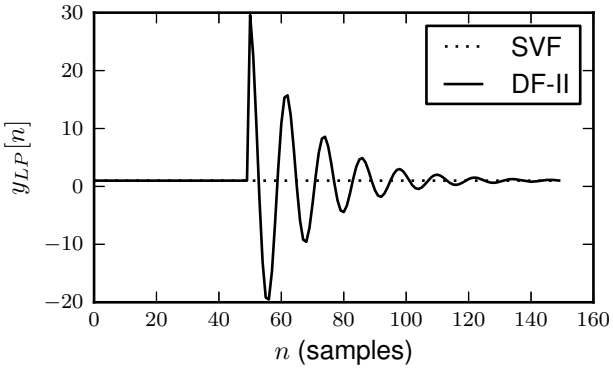


Figure 1: *Comparison between lowpass state variable filter and Direct-Form II topology of steady state DC response when parameters are changed. At sample 50, the parameters are changed from* $g = 0.0458$ *and* $R = 0.4545$ *to* $g = 0.2679$ *and* $R = 0.1111$.

For each filter output, the ℓ_2 norm of the difference from the ideal DC response was computed after the parameter change. The errors are displayed in Table 4. Zetterberg-Zhang is omitted because it passes this test by definition.

As can be seen, the SVF and the stabilized SVF are the only structures besides Zetterberg-Zhang that perform perfectly in this test, with no deviation from the ideal. Transient minimization ac-

Table 4: *Objective test results with DC stimulus. The reported values are the ℓ_2 norms in decibels of the error as compared to the output switching model. Lower values are better, and $-\infty$ is ideal.*

#	DF2	GR	SVFR	SVFRC	SVF	TDF2RC	TDF2
1	18	10	15	$-\infty$	$-\infty$	-36	-9
2	-25	13	13	$-\infty$	$-\infty$	18	18
3	3	-5	0	$-\infty$	$-\infty$	-3	-2
4	-65	-27	-12	$-\infty$	$-\infty$	7	7
5	-42	-4	-3	$-\infty$	$-\infty$	17	15

tually worsens the response of the SVF. Other filter structures perform acceptably for some trials and poorly for others. This ideal DC response can be proven to hold for the SVF over all parameters.

First, let us determine the filter's state, $\mathbf{s_n}$, in a steady DC state, where the input $\mathbf{u_n} = \begin{bmatrix} k \end{bmatrix}$ for all n, where k is the magnitude of the DC signal. Because the output at x_1 is a bandpass filter, and x_2 is a lowpass filter, we know $\mathbf{x_n} = \begin{bmatrix} 0 & k \end{bmatrix}^T$, since the lowpass filter passes DC, and the bandpass filter rejects it. Substituting the values of $\mathbf{u_n}$ and $\mathbf{x_n}$ into equation (7b), we find that

$$\mathbf{s_n} = \begin{bmatrix} 0 & k \end{bmatrix}^T \tag{24}$$

For DC input, both $\mathbf{x_n}$ and $\mathbf{s_n}$ are independent of the filter parameters. Therefore, time-varying parameters do not cause switch-time transients in the DC response of the filter's state, which proves the observed behavior.

6.2. Subjective Evaluation

Listening to the transient elimination methods in a musical context suggests that for use on musical signals, with rapid parameter changes, output switching is the wrong goal. A sinusoidal input can be used to illustrate: an instantaneous change in filter parameters corresponds to an instantaneous change in the amplitude and phase of the signal. The spectrum centered at the point of coefficient change reveals high sidebands, which are audible as an impulsive "click", shown in Fig. 2.

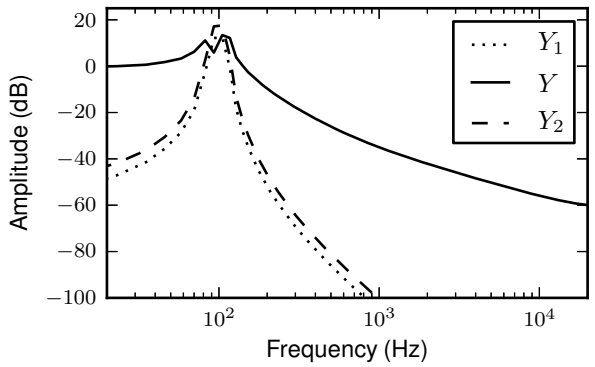


Figure 2: *Spectrum of filtered 100 Hz sinusoid during parameter change with output switching, using 85 ms Hann window (Y). Spectrum of the signal before (Y_1) and after (Y_2) shown for reference.*

Output switching removes the transients that result from the state vector reacting to a change in coefficients. However, it also emphasizes discontinuous changes in filter parameters, resulting in this click. Apparently, the transient caused by a filter structure when its parameters are changed can smooth the change out, reducing these sidebands. This is simply a case of differing goals: Zetterberg and Zhang were motivated by LPC-based speech coding, while here, we consider musical applications.

To better understand the impact of these transients, and to compare the SVF to other solutions, we have subjectively evaluated different filter structures, and schemes of transient elimination and stabilization. A 100 Hz sinusoid at 48 kHz was chosen as the test signal, because it masks the transient very little, allowing artifacts to be easily heard.

6.2.1. Experimental Setup

To evaluate the time-varying response of these filter structures, we performed listening tests using the MUSHRA method [24]. The test was performed with 21 subjects, all of whom have experience playing and recording music, and many of whom perform critical listening professionally. Subjects listened with headphones in a quiet room. They were asked to rank the excerpts in quality, in terms of how unpleasant they found any artifacts they might hear.

In addition to the filter structures, a high-quality reference and hidden low-quality anchor were included. The reference is made by applying a gain envelope corresponding to the gain of the filter at each point in time, smoothed with a 10 millisecond Hann filter kernel. The anchor is made using the unsmoothed gain envelope, with an impulse added when the coefficients change. The amplitude of the impulse is three times greater than the maximum filter gain applied.

6.2.2. Listening Test Results

Fig. 3 displays the average MUSHRA scores over all trials. Table 5 presents the MUSHRA scores separated by trial, so that performance can be compared across filter shapes.

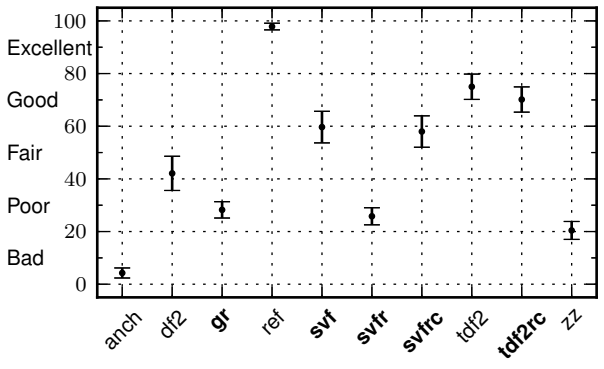


Figure 3: *Average MUSHRA scores and 95% confidence intervals for each filter structure, over all trials. Filter structures that are time-varying stable and support efficient per-sample coefficient update are in bold.*

The results confirm the SVF's good performance, in comparison to other filter structures. The output switching filter consistently received scores ranging from "bad" to "poor". However, of the time-varying stable filters, the stabilized TDF-II structure (TDF2RC) has the best score.

Table 5: *Average MUSHRA scores for each filter structure, separated by trial.*

#	DF2	GR	SVF	SVFR	SVFRC	TDF2	TDF2RC	ZZ
1	18	29	63	20	60	69	63	18
2	99	27	96	22	95	70	64	7
3	26	28	72	23	71	77	71	13
4	43	34	43	37	41	95	94	38
5	25	24	25	27	23	64	59	26

The efficacy of Rabenstein and Czarnach's method of stabilization is confirmed. For both the SVF and TDF-II structures, this state vector adjustment causes only a small decrease in scores. This technique need not be applied to an already stable filter such as the SVF, as it decreases the quality without providing any benefit. Interestingly, this method is derived by transforming a system into the Gold and Rader structure, which received significantly worse scores.

Choice of filter structure is a trade-off: if interpolation of parameters is needed, or if there are performance constraints, the SVF may be a better choice, as interpolating TDF-II coefficients can give less sensible intermediate transfer functions, and the method of Rabenstein and Czarnach requires several more trigonometric function evaluations. On the other hand, if interpolation is not necessary, stabilized TDF-II may be a better generic choice. Though the TDF-II performed well in these listening tests, recall that it performed poorly in the objective test of DC response, while the SVF had an ideal response.

The per-trial scores in Table 5 also suggest that choice of filter structure may depend on the type of transfer function being implemented, and what parameters will be modulated. For example, the SVF performs better than stabilized TDF-II for both lowpass filter trials, and more or less the same when the peaking filter frequency is changed, but significantly worse when the peaking filter resonance or gain are changed.

The transient minimization methods (Zetterberg-Zhang and SVF with Rabenstein's method) both achieve their stated goals, yet they received scores of "poor". This confirms the hypothesis that output switching is the wrong goal in this musical context. The peak signal levels are decreased, and Rabenstein's method successfully decreases the variance of the transient signal in the SVFR excerpts. However, the results indicate that transient minimization degrades the quality of the SVF. Fig. 4 shows that the transient signal and the MUSHRA scores are essentially uncorrelated. While transient minimization may be useful for applications such as speech coding and synthesis [3], it appears to be undesirable for equalization of musical signals.

If transient minimization is not a desirable criteria for musical time-varying filters, what is? Sideband energy appears to be negatively correlated with MUSHRA scores, with Pearson's $r = -0.59$ and $p = 7.02 \times 10^{-5}$, as can be seen in Fig. 5. This is a crude psychoacoustic measure, but perhaps it would be possible to design an optimal time-varying structure by minimization of sideband energy, rather than variance of the transient signal.

7. CONCLUSIONS AND FUTURE WORK

In this paper, the problem of choosing a structure suitable for digital filtering in a musical context with per-sample time-varying parameters has been addressed. In this problem domain, important qualities include support for arbitrary transfer functions, computational efficiency, zero-latency realtime implementation, and good

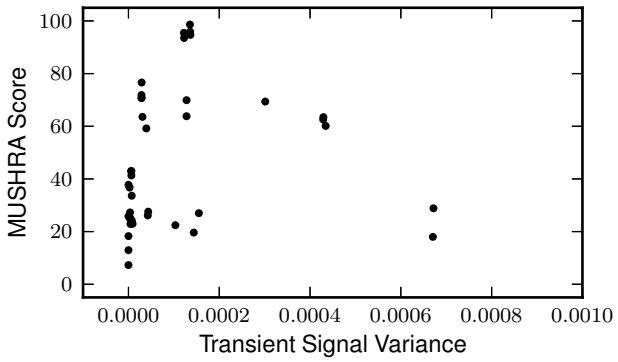


Figure 4: *Scatter plot of transient signal variance versus MUSHRA score, excluding anchor and reference, showing little correlation. Pearson's* $r = 0.11$, $p = 0.48$.

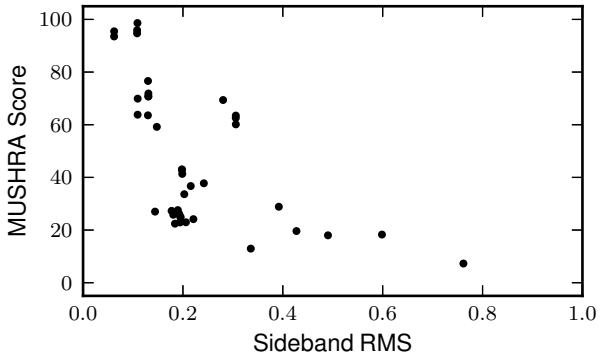


Figure 5: *Scatter plot of sideband energy versus MUSHRA score, excluding anchor and reference, showing correlation. Sideband energy was measured using an 85 ms Hann window centered around the parameter change, by computing the magnitude spectrum, removing the energy inside one equivalent rectangular band (ERB) [25] at 100 Hz, and computing the RMS of the remaining signal.*

intermediate filter shapes when interpolating parameters. The SVF discretized with the TDF-II bilinear integrator was reviewed and proposed as a good general purpose solution to this problem.

In order to make the SVF useful for this purpose, equations were derived for implementing common audio equalization filters, as well as any z-domain transfer function, and its time-varying stability was also proven for the first time. These results allow the SVF to be applied to this problem domain.

The audio quality of the SVF during parameter changes was evaluated in both objective and subjective tests. In the objective test, the SVF was the only filter structure supporting realtime per-sample parameter changes that was found to have an ideal DC response. The results of the subjective listening test confirmed that the SVF performs well, though the stabilized TDF-II performed better, on average. The results also indicated that different structures perform differently depending on the transfer function being realized.

The listening test results also revealed that output switching, i.e., eliminating the transient response of a filter, is not desirable in musical applications. The sideband energy was proposed as one

measure of quality, with low sidebands being most desirable.

Some considerations for the SVF can be drawn by considering the experimental results alongside the presented theory. For example, as the SVF was proven to respond instantaneously to changes in DC gain, and these abrupt discontinuities were shown to cause perceived artifacts, it may be desirable to smooth the changes in coefficients c_{HP}, c_{BP}, and c_{LP}, which set the filter's zeros. In fact, Table 2 shows that the peaking filter's zeros are affected by both gain and resonance, which corresponds with the findings in Table 5: that the SVF performs the worst when the peaking filter gain or resonance are changed. It appears that changes in these three coefficients is responsible for audible artifacts, while the transient response resulting from changes in the poles is perceptually pleasant.

Future research could concentrate on schemes for improving subjective quality in musical contexts. For example, as Rabenstein [2] derived intermediate coefficients to minimize transient signals, perhaps perceptually important factors such as sideband energy could be minimized. Another potential area of research is further perceptual evaluation of a greater variety of structures and transfer functions, using other musical stimuli. From such experiments, it might be possible to determine mathematical criteria relevant to perceived quality as alternatives to transient minimization.

8. REFERENCES

[1] L. H. Zetterberg and Q. Zhang, "Elimination of transients in adaptive filters with application to speech coding," *Signal Processing*, vol. 15, no. 4, pp. 419–428, December 1988.

[2] R. Rabenstein, "Minimization of transient signals in recursive time-varying digital filters," *Circuits, Systems, and Signal Processing*, vol. 7, no. 3, pp. 345–359, 1988.

[3] V. Välimäki and T. I. Laakso, "Suppression of transients in variable recursive digital filters with a novel and efficient cancellation method," *IEEE Transactions on Signal Processing*, vol. 46, no. 12, pp. 3408–3414, December 1998.

[4] R. Rabenstein and R. Czarnach, "Stability of recursive time-varying digital filters by state vector transformation," *Signal Processing*, vol. 8, pp. 75–92, 1985.

[5] J. Laroche, "On the stability of time-varying recursive filters," *J. Audio Eng. Soc.*, vol. 55, no. 6, pp. 460–471, June 2007.

[6] J. Laroche, "Using resonant filters for the synthesis of time-varying sinusoids," in *105th AES Convention*, California, USA, Sep. 26-29, 1998.

[7] C. M. Gold and B. Rader, "Effects of parameter quantization on the poles of a digital filter," *Proceedings of the IEEE*, vol. 55, no. 5, pp. 688–689, May 1967.

[8] C. W. Barnes, "Roundoff noise and overflow in normal digital filters," *IEEE Transactions on Circuits and Systems*, vol. 26, no. 3, pp. 154–159, March 1979.

[9] J. O. Smith, "Digital State-Variable Filters," Available at https://ccrma.stanford.edu/~jos/svf/svf.html, accessed March 06, 2014.

[10] D. T. Yeh, "Digital implementation of musical distortion circuits by analysis and simulation," M.S. thesis, Stanford University, 2009.

[11] J. O. Smith, *Introduction to Digital Filters with Audio Applications*, W3K Publishing, second edition, 2008.

[12] P. Dutilleux, "Simple to operate digital time varying filters," in *Preprint 86th AES Convention*, Hamburg, Germany, Mar. 7-10, 1989, pp. 1–25.

[13] H. Chamberlin, *Musical Applications of Microprocessors*, Hayden Books, second edition, 1985.

[14] D. Wise, "The modified Chamberlin and Zölzer filter structures," in *Proc of the 9th Int. Conference on Digital Audio Effects*, Montreal, Canada, Sep. 18-20, 2006, pp. 53–56.

[15] G. Borin, G. De Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 597–605, Sep 2000.

[16] V. Zavalishin, "The Art of VA Filter Design," Available at http://www.native-instruments.com/fileadmin/ni_media/downloads/pdf/VAFilterDesign_1.0.3.pdf, accessed March 06, 2014.

[17] R. Bristow-Johnson, "Cookbook formulae for audio EQ biquad filter coefficients," Available at http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt, accessed March 06, 2014.

[18] S. J. Orfanidis, "Digital parametric equalizer design with prescribed nyquist-frequency gain," *J. Audio Eng. Soc.*, vol. 45, no. 6, pp. 444–455, June 1997.

[19] G. Berchin, "Precise filter design," *IEEE Signal Processing Magazine*, vol. 24, no. 1, pp. 137–139, Jan 2007.

[20] K. B. Christensen, "A generalization of the biquadratic parametric equalizer," in *Proc. 115th Audio Eng. Soc.*, New York, USA, Oct. 10-13, 2003.

[21] R. Bencina, "Time Varying BIBO Stability Analysis of Trapezoidal integrated optimised SVF v2," music-dsp mailing list, Nov. 2013, Available at http://www.mail-archive.com/music-dsp@music.columbia.edu/msg02467.html.

[22] D. Berners, "Analog Circuit Emulation for Plug-In Design," in *Audio Eng. Soc. Master Class M1*, New York, USA, Oct. 9, 2009.

[23] D. Berners, "The Inner Workings of the Moog Multimode Filter," Available at http://www.uaudio.com/blog/moog-multimode-filter-design/, accessed March 08, 2014.

[24] ITU, "ITU-R BS.1534-1: Method for the subjective assessment of intermediate quality level of coding systems," 2003.

[25] B. C. J. Moore, "Frequency analysis and masking," in *Hearing*, B. C. J. Moore, Ed., pp. 161–205. Academic Press, San Diego, California, 1995.

[26] R. Bristow-Johnson, "The Equivalence of Various Methods of Computing Biquad Coefficients for Audio Parametric Equalizers," Available at http://thesounddesign.com/MIO/EQ-Coefficients.pdf, accessed March 06, 2014.

PERCEPTUAL LINEAR FILTERS: LOW-ORDER ARMA APPROXIMATION FOR SOUND SYNTHESIS

Rémi Mignot[*] *, Vesa Välimäki*

Aalto University,
Department of Signal Processing and Acoustics
Otakaari 5A, 02150 Espoo, Finland

ABSTRACT

This paper deals with the approximation of a given frequency response by a low-order linear ARMA filter (Auto-Regressive Moving Average). The aim of this work is the audio synthesis, then to improve the perceptual quality, a criterion based on human listening is defined and minimized. Two complementary approaches are proposed here for solving this non-linear and non-convex problem: first, a weighted version of the Iterative Prefiltering, second, an adaptation of the Gauss-Newton method. This algorithm is adapted to guarantee the causality/stability of the obtained filter, and eventually its minimum phase property. The benefit of the new method is illustrated and evaluated.

1. INTRODUCTION

The goal of this paper is the approximation of a given frequency response by a low-order linear ARMA filter (Auto-Regressive Moving Average), with a high sampling rate, $F_s \geq 44.1$ kHz. The context of this work is the low-cost sound synthesis of musical tones using the *Source-Filter* principle which consists of the filtering of an excitation signal. Then, because the aim is an audio application, the obtained filter must be as close as possible to the original one in a perceptual sense, rather than using a physical or signal-based criterion.

It is known that in a general case a spectral envelope has a sparser representation with an ARMA model than a purely AR or MA model. It is especially the case for nasal speech, and for musical instruments. For example, even if an ARMA(q, p) filter and an AR(q+p) filter have approximately the same complexity for the time simulation, the ARMA modeling will be more efficient in most of the cases. Some ARMA approximations exist, cf. e.g.: Prony's method [1], Shanks's method [2], the Iterative Prefiltering [3], Durbin's method [4] or the Inverse Linear Prediction [5] (or cf. e.g. [6] for a partial review). Nevertheless, with these methods the cost function is adapted to facilitate the algorithm, and is never adapted to the perception.

A usual idea is to adapt the model to the frequency resolution of the ear. In [7, 8, 9] a warped frequency scale is used to fit the Bark scale, cf. [10, 11], and a warped AR filter is obtained. Unfortunately, first we have shown in [12] that for low-orders, the warped modeling is not satisfying in a perceptual sense. This observation can be explained because the optimization criterion is not fully perceptually based. Moreover, the time-domain implementation of the warped AR filter is two or three times more expensive than a linear AR filter with the same order, cf. e.g. [8].

[*] This work is funded by the Marie Curie Action project ESUS 299781.

In this work, we propose to directly estimate a linear ARMA filter on the linear frequency scale using the minimization of a perceptually-based criterion. In the context of the Source-Filter principle, the target frequency response is obtained by a spectral envelope estimation of an original sound, which can be periodic. This estimation can be done by the DAP method of [13], the True Envelope of [14, 15], or the True Discrete Cepstrum of [16]. Note that it is also possible to use a post-processing, MTELPC [17] or PCF [18], which provide a "quasi-perceptual" pre-smoothing. These points are not detailed in this work.

This paper is organized as follows: in Sec. 2, the ARMA model is given, and the perceptually-based criterion is defined step by step in Sec. 3. Then, the two parts of the algorithm are given in Sec. 4. Section 5 gives one practical example, and presents a perceptual comparison of the proposed method with other standard methods. Finally, section 6 concludes this paper and gives some perspectives.

2. MODEL

Given a complex frequency response $H(f)$, where f is the frequency in [Hz], this work deals with its approximation by the following ARMA(Q, P) filter

$$\widetilde{H}(z) = \frac{B(z)}{A(z)} = \frac{b_0 + \sum_{q=1}^{Q} b_q z^{-q}}{1 + \sum_{p=1}^{P} a_p z^{-p}}, \qquad (1)$$

where Q and P are the orders of the numerator B and the denominator A respectively. z is the complex variable of the z-transform, which is $z = \mathrm{e}^{j2\pi f/F_s}$ on the unit circle, with f the frequency variable and F_s the sampling rate in [Hz]. The polynomial coefficients b_q and a_p are the variables to optimize.

3. PERCEPTUAL CRITERION

3.1. First criterion

Let us define the following criterion which provides a distance between the target $H(f)$ and the model $\widetilde{H}(f)$:

$$\mathcal{C}_1 = \int_0^{\frac{F_s}{2}} \frac{\left[\sigma(H(f), f) - \sigma(\widetilde{H}(f), f)\right]^2}{\sigma(H(f), f)^2} \, M(df). \qquad (2)$$

This cost function is perceptually meaningful because of the following reasons.

Loudness conversion First, the function $\sigma(X, f)$ is the conversion of the (physical) sound pressure level X in pascals [Pa], to the (perceptual) loudness in sones, depending on the frequency f. The conversion σ is here calculated with the consecutive conversions: $\sigma(X, f) = s(\ell(\delta(X), f))$, where $X_{db} = \delta(X) = 20\log_{10}(|X|/p_0)$ is the standard scale in [dB SPL], with $p_0 = 2\times 10^{-5}$ Pa the reference sound level, $L_p = \ell(X_{db}, f)$ is the conversion from the decibel scale to the phon scale, relative to the equal loudness curves cf. e.g. [19, 20], and $L_s = s(L_p) = 2^{(L_p-40)/10}$ is the conversion to the sone scale, cf. e.g. [21].

Frequency scale Second, the measure $M(df)$ takes the frequency resolution of the ear into account, as the standard warping mentioned earlier. With $m(f)$ the conversion from the linear frequency scale in [Hz] to any warped scale, we write $M(df) = dm(f) = m'(f)df$. For example, with the Mel scale of [22], $m(f) = 2595\log_{10}(1 + f/700)$.

Relative error Third, note that the sone and the phon scales are respectively linear and logarithmic scales in the loudness domain, such as the pascal and the decibel scales in the sound level domain respectively. Then, the relative error is computed in (2) in order to take into account the logarithmic sensitivity of the ear. Remark that it would be also possible to directly define $\mathcal{C}_1$ with the absolute error in the phon scale, logarithmic, but it is equivalent up to the first order and the denominator will be used in next section.

3.2. Modified criterion

For a numerical computation, first a new version of $\mathcal{C}_1$ is derived using a discrete sum. Second, the loudness conversion is simplified using a first-order limited development of $\sigma(\widetilde{H}, f)$ around H. Then $\sigma(\widetilde{H}, f) \approx \sigma(H, f)+\sigma'(H, f)(|\widetilde{H}|-|H|)$ with $\sigma'(X, f) = \partial\sigma(X, f)/\partial|X|$, and the criterion becomes:

$$\mathcal{C}_2 = \sum_{m=1}^{M} \frac{\left(|H_m| - |\widetilde{H}_m|\right)^2 \sigma'(H_m, f_m)^2}{\sigma(H_m, f_m)^2}\, m'(f_m), \qquad (3)$$

where the frequencies f_m uniformly sample the range $[0, F_s/2]$ and $H_m = H(f_m)$. Note that in this work σ and its derivative are computed using the analytical expression of [23].

If the phase of the target response H is known, we can replace $(|H_m| - |\widetilde{H}_m|)^2$ by $|H_m - \widetilde{H}_m|^2$. This actually simplifies the optimization procedure and facilitates the convergence. Note that, only knowing $|H|$, its phase can be recovered assuming a minimum phase system, cf. e.g. [24].

Since a sound with a level below the auditory threshold is imperceptible in principle, the function σ is not defined below this threshold which corresponds to 0 phon. Then with $X_0(f)$ the auditory threshold in pascals, such that $\sigma(X_0(f), f) = s(0) = 2^{-4}$ sons, we define the saturated function

$$\underline{\sigma}(X, f) = \begin{cases} \sigma(X, f) & \text{if } |X| \geq X_0(f) \\ 2^{-4} & \text{if } |X| < X_0(f) \end{cases} \qquad (4)$$

and the saturated derivative $\underline{\sigma}'$ in the same way. Finally, the criterion to minimize is written as

$$\mathcal{C} = \sum_{m=1}^{M} \left|H_m - \widetilde{H}_m\right|^2 W_m^2 \qquad (5)$$

$$\text{with} \quad W_m = \frac{\underline{\sigma}'(H_m, f_m)}{\underline{\sigma}(H_m, f_m)} \sqrt{m'(f_m)}. \qquad (6)$$

In consequence, the criterion $\mathcal{C}$ is just the weighted squared sum of the error, with a weight W_m which takes into account the sensitivity of the ear to the frequencies via $m(f)$, to the sound level via σ, and to the auditory threshold via the "saturated" $\underline{\sigma}$.

3.3. Remarks

Because most of the time the sensitivity of the recording device is not available, a possible way to adapt the unscaled recording sound to the pascal scale is just by applying a gain which gives the desired sound level. For example $X_{db} = 70$ dB SPL is a normal level for a single musical instrument.

In (4), $X_0(f)$ is the absolute auditory threshold. It is also possible to combine it with the simultaneous masking threshold, cf. e.g. [25], calculated from the target response $H(f)$. Nevertheless, this strategy seems hazardous because $H(f)$ and $\widetilde{H}(f)$ are not "concrete" spectra, but "abstract" spectral envelopes.

4. OPTIMIZATION ALGORITHM

With an ARMA modeling $\widetilde{H} = B/A$, the minimization of (5) is not trivial because the error is non-linear with the coefficients a_p of the denominator A and this optimization problem is not convex. In this section two complementary iterative algorithms are proposed to minimize the cost function $\mathcal{C}$. The first approach is based on the *Iterative Prefiltering* of [3]. It is referred as the Mode 1 because its result is used as initialization of the second one, the Mode 2, which is based on the Gauss-Newton algorithm, cf. e.g. [26].

4.1. Mode 1: Weighted Iterative Prefiltering

Instead of optimizing a non-linear problem, the Iterative Prefiltering method, initially proposed in [3], consists in iteratively solving linear sub-problems using the Least Mean Square optimization (LMS). For that, the criterion $\mathcal{C}$ is modified at every iteration using the previous estimation.

4.1.1. Secondary criterion

With A' the estimated denominator of the previous iteration, the multiplication of the error $e_m := (H_m - B_m/A_m)W_m$ of (5) by A_m/A'_m, leads to the secondary criterion which follows

$$\mathcal{C}' := \sum_{m=1}^{M} \left| A_m \frac{H_m W_m}{A'_m} - B_m \frac{W_m}{A'_m} \right|^2. \qquad (7)$$

Since A' is known, the new defined error is linear with the parameters a_p and b_q, and the minimization of $\mathcal{C}'$ can be solved using the standard LMS. This procedure is equivalent to the Iterative Prefiltering method of Steiglitz and McBride, cf. [3, 27], with an additional frequency weight $W(f)$. It is important to note that at the convergence, if it happens, A/A' goes toward 1, consequently the secondary criterion $\mathcal{C}'$ gets closer to the primary criterion $\mathcal{C}$.

4.1.2. Linear optimization

In (7), $\mathcal{C}'$ is given in the frequency domain, but considering the Hermitian symmetry of H, $\widetilde{H}$, and W, and using the Parseval theorem, we can write it in the discrete time domain to avoid complex numbers. Whereas the computation of h_n, the time response of H, does not cause any issue, the direct inverse Fourier transform of W

makes a non-causal response because W is real. Nevertheless, $\mathcal{C}'$ is invariant by adding a phase to W, then to avoid time aliasing, we define w_n as the minimum phase solution of W, cf. e.g. [24].

With $y := (h * w)/A'$ and $x := w/A'$, where the symbol $*$ denotes the convolution product and $./A'$ denotes the prefiltering by the AR filter $1/A'$, the secondary criterion $\mathcal{C}'$ is written

$$\mathcal{C}' = \frac{1}{2}\sum_{n=0}^{N-1}\Big(y_n + \sum_{p=1}^{P} a_p y_{n-p} - \sum_{q=0}^{Q} b_q x_{n-q}\Big)^2, \qquad (8)$$

Note that, even if the computations of w and $(h * w)$ may be quite expensive, they are done only once before the first iteration.

Then, for $n \in [1, N]$, $p \in [1, P]$ and $q \in [1, Q+1]$, and with the matrix transpose $.^T$, we define the column vectors Y and μ such that $Y_n = y_{n-1}$ and $\mu = [a_1, \dots a_P, b_0, b_1, \dots b_Q]^T$, and we define the block matrix $\Phi = [-\Phi_y, \Phi_x]$, with the Toeplitz matrices $\Phi_y[n,p] = y_{n-1-p}$ and $\Phi_x[n,q] = x_{n-q}$. Note that considering causal signals, $y_n = 0$ and $x_n = 0$ for $n < 0$.

Consequently, the matrix form of the secondary criterion is: $\mathcal{C}' = \frac{1}{2}(Y - \Phi\mu)^T(Y - \Phi\mu)$, and if Φ is full rank, the optimal solution in the LMS sense is given by solving the linear problem $(\Phi^T\Phi)\mu = (\Phi^T Y)$, which can be written, cf. e.g. [26],

$$\mu = \big(\Phi^T\Phi\big)^{-1}\Phi^T Y = \Phi^\dagger Y. \qquad (9)$$

As it is implicitly mentioned in [3], at the first iteration, we simply choose $A' = 1$. Note that without weight W, at the first iteration $x_n = \delta_n$, the Dirac distribution, and the first estimated B and A are the solutions of Prony's method.

4.1.3. Properties

Remark that the positions of the roots of A and B are not ensured to be inside the unit circle, which means that the causality/stability and the minimum phase property cannot be controled. Even if this problem occurs rarely if the target H checks these properties, it may be overcome by testing the desired properties at every iteration, using the Jury criterion for example [28], and by recomputing the LMS solution with lower orders, P and Q. This strategy usually leads to good properties, but with eventually a worse $\mathcal{C}'$.

As mentioned in [3], the convergence of this iterative procedure is not guaranteed. Nevertheless, we observed in every experiment an efficient decrease in the criterion $\mathcal{C}$ and we observed the convergence of the coefficients of A. Unfortunately, first, some conditioning problems usually appear after some iterations, when $\Phi^T\Phi$ is numerically singular, and second, even if $\mathcal{C}'$ get closer to $\mathcal{C}$, the partial derivatives of $\mathcal{C}'$ are different from those of $\mathcal{C}$, which explains why this algorithm usually does not converge to a local minimum in the sense of $\mathcal{C}$.

In [3], a second iterative procedure, the respective Mode 2, has been proposed to improve the estimation of the first one. This point is not detailed here, we refer the interested readers to [3]. In favorable cases, this new mode converges to the closest local minimum, but again, the convergence is not guaranteed, and may diverge if its initial value is far from a local minimum. Moreover, in our experiments, some conditioning problems may still appear. Finally, as with the Mode 1, the causality/stability, and the minimum phase property, of the obtained filter cannot be clearly guaranteed.

In the next section, we proposed another Mode 2 which is based on the Gauss-Newton algorithm. First, this method has a better convergence, second, the conditionning is efficiently improved, and third, this approach can guarantee the causality/stability and the minimum phase property of the estimated filter.

4.2. Mode 2: Non-linear optimization

We propose in this section an adaptation of the iterative Gauss-Newton algorithm, cf. e.g. [26], with constraints for the causality/stability of the filter, and eventually its minimum phase property. Compared to the standard gradient descent, its convergence is usually faster, and it avoids the successive 1D optimizations along the direction of maximal descent.

4.2.1. Gauss-Newton algorithm

Newton's algorithm is based on a second-order limited development of the criterion. Starting from an initial parametrization of the model, the parameters are iteratively updated by the optimum solution of the quadratic form given by the limited development around the previous parameters. If the cost function is quadratic, the algorithm converges in one step, and if it is not quadratic but sufficiently regular, it naturally converges to the nearest local minimum in some iterations.

With μ^k the column vector collecting the current parameters of the model, the following parameters are given by:

$$\mu^{k+1} = \mu^k - \Omega_{\mathcal{C}}^{-1}(\mu^k)\nabla_{\mathcal{C}}(\mu^k), \qquad (10)$$

with $\nabla_{\mathcal{C}}(\mu)$ the gradient vector and $\Omega_{\mathcal{C}}(\mu)$ the Hessian matrix: $\nabla_{\mathcal{C}}[i] = \partial\mathcal{C}/\partial\mu_i$ and $\Omega_{\mathcal{C}}[i,j] = \partial^2\mathcal{C}/\partial\mu_i\partial\mu_j$.

The Gauss-Newton algorithm differs from the previous one by the approximation of the Hessian matrix. This approximation facilitates the computation and is justified by the fact that the criterion is the squared sum of the magnitude of the error e_m, cf. e.g. [26]. With

$$e_m := (H_m - \widetilde{H}_m)W_m, \qquad (11)$$

the criterion is written $\mathcal{C} = E^H E$, where E is the column vector of the error e_m and $.^H$ is the Hermitian transpose.

Now, defining $J_e(\mu)$ as the Jacobian matrix of E, such that $J_e[m,i] = \partial e_m/\partial\mu_i$, the gradient vector of $\mathcal{C}$ becomes $\nabla_{\mathcal{C}}(\mu) = 2J_e(\mu)^H E(\mu)$, and the approximated Hessian matrix is written $\Omega_{\mathcal{C}}(\mu) = 2J_e(\mu)^H J_e(\mu)$. Consequently

$$\Omega_{\mathcal{C}}^{-1}\nabla_{\mathcal{C}} = \big(J_e^H J_e\big)^{-1} J_e^H E = J_e^\dagger E. \qquad (12)$$

Nevertheless, with (10), the algorithm may diverge in some cases. Then, it is usual to introduce a relaxation factor $\lambda_k \leq 1$, and the algorithm becomes

$$\mu^{k+1} = \mu^k - \lambda_k\Omega_{\mathcal{C}}^{-1}(\mu^k)\nabla_{\mathcal{C}}(\mu^k). \qquad (13)$$

A simple strategy for the choice of λ_k is to successively reduce its value until $\mathcal{C}(\mu^{k+1}) < \mathcal{C}(\mu^k)$. Note that if the Hessian matrix is positive-definite, there always exists a $\lambda_k > 0$ providing a decreasing criterion. Here, we first test $\lambda = 1$ to accelerate the convergence, and we divide it by 2 if $\mathcal{C}$ does not decrease.

4.2.2. Optimization of the ARMA model

Starting from the standard ARMA modeling of (1), to improve the conditioning we introduce a gain g and we force $b_0 = 1$, without loss of generality. The model is then given by $\widetilde{H}(z) = gB(z)/A(z)$, and the parameters to identify are the gain g and the coefficients a_p and b_q of the polynomials $A(z)$ and $B(z)$ respectively, with $p \in [1, P]$ and $q \in [1, Q]$. Moreover, to avoid the singular case in $g = 0$, we do the change of variable $g = \zeta\,\mathrm{e}^\gamma$,

where ζ is the sign of the initial gain, and we optimize γ on $\mathbb{R}$ instead of g.

With $z_m = \mathrm{e}^{j2\pi f_m/F_s}$ and $\mu = [\gamma, a_1, \dots a_P, b_1, \dots b_Q]^T$, the Jacobian matrix $J_e(\mu)$ is given by

$$\begin{cases} \dfrac{\partial e_m}{\partial \gamma} = -\zeta \dfrac{B(z_m)}{A(z_m)} \dfrac{\partial\, \mathrm{e}^{\gamma}}{\partial \gamma} W_m & = -\widetilde{H}(z_m) W_m, \\ \dfrac{\partial e_m}{\partial a_p} = g \dfrac{B(z_m)}{A(z_m)^2} \dfrac{\partial A(z_m)}{\partial a_p} W_m & = z_m^{-p} \dfrac{\widetilde{H}(z_m)}{A(z_m)} W_m, \\ \dfrac{\partial e_m}{\partial b_q} = \dfrac{-g}{A(z_m)} \dfrac{\partial B(z_m)}{\partial b_q} W_m & = z_m^{-q} \dfrac{-g}{A(z_m)} W_m. \end{cases}$$

Remark that in (2) and (5), we only have considered unilateral spectra for $f \in [0, F_s/2]$. Then the solution given by (13) could lead to complex coefficients g, a_p and b_q, with no consideration of the range $[F_s/2, F_s]$. Instead of summing the error on the full range $[0, F_s]$, we prove the equivalence of the following update equation

$$\mu^{k+1} = \mu^k - \lambda_k \mathrm{Re}\{\Omega_{\mathcal{C}}(\mu^k)\}^{-1} \mathrm{Re}\{\nabla_{\mathcal{C}}(\mu^k)\}. \qquad (14)$$

where $\mathrm{Re}\{.\}$ is the real part operator, and where $\Omega_{\mathcal{C}}$ and $\nabla_{\mathcal{C}}$ are still computed on the frequency range $[0, F_s/2]$. This equation can be fastly computed by splitting the real parts and the imaginary parts of J_e and E, cf. (12).

Concerning the causality/stability of the obtained filter, and eventually its minimum phase property, it is necessary to add this constraint in the algorithm. Remind that an ARMA filter is a minimum phase system if and only if both poles and zeros are strictly inside the unit circle. To guarantee the desired property at every iteration, we adapt the choice of the relaxation factor λ_k as it is done in Sec. 4.2.1 for the convergence. To study the location of the roots of the polynomials A and B, we use the Jury stability criterion. Note that at some iterations, at the point μ^k, the local properties of $\mathcal{C}$ may attract the algorithm outside the constraint domain, even if the nearest local minimum is inside. Then, choosing a small λ_k allows to stay inside the domain, and most of the time, from the new position μ^{k+1} the algorithm naturally reconverges to the minimum.

4.2.3. Summary of the algorithm

To summarize the complete algorithm, first the Mode 1 iterations, weighted Iterative Prefiltering, are computed using the initialization $A' = 1$. As mentioned above, even if the Mode 1 usually converges, the primary criterion $\mathcal{C}$ may not be strictly decreasing, which means that with a finite number of iterations, the last result may not be the best one in the sense of $\mathcal{C}$. Then, to initialize the Mode 2, the Gauss-Newton algorithm, among all successive results of the Mode 1, we retain this one which minimizes $\mathcal{C}$. Finally, since the convergence of the Gauss-Newton is well-defined, we can use standard stop criteria. Here the algorithm is stopped when the maximal number of iterations is attained, or when the relative difference of two consecutive criteria is smaller than a threshold defined in [%].

5. EXPERIMENTATIONS

5.1. Illustration

The proposed method, which we call the *Perceptual Linear Filter* (PLF), is illustrated in Fig. 1. Using an Oboe tone, B3 ($\sim$ 247 Hz), first the spectral envelope has been estimated with the True Envelope (TE) of [14] and has been slightly smoothed using the PCF approach of [18]. Then, from the magnitude of the obtained frequency response, the phase has been recovered assuming a minimum phase system, cf. e.g. [24]. Finally, the PLF is computed by the algorithm presented in Sec. 4, Mode 1 and 2, using an ARMA(8,8) model, and is compared to Prony's method with the same orders. All frequency responses are displayed in the dB SPL scale, together with the auditory threshold.

As a general trend, we observe that the PLF method focuses the approximation at the lower frequencies, as the standard warping technique (cf. [8, 9]), but especially to those frequencies where the target response $H(f)$ is above the auditory threshold. We can observe that Prony's estimate $\widetilde{H}_1(f)$ does not fit $H(f)$ around 4.5 kHz, but it fits the last formant after 14 kHz which is imperceptible in principle. On the contrary, thanks to the perceptual weighting $W(f)$, cf. (6), the PLF filter $\widetilde{H}_2(f)$ fits $H(f)$ when it is audible, and it strongly smooths it when it is imperceptible.

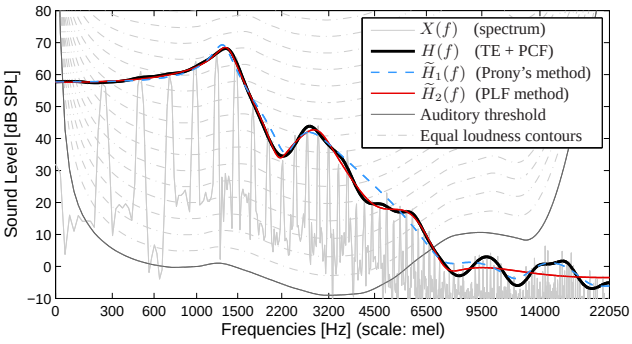


Figure 1: Illustration of the Perceptual Linear Filter (PLF). The orders of the ARMA model are $Q = 8$ and $P = 8$.

5.2. Perceptual evaluation

This section proposes a perceptual evaluation by comparing the PLF method and other methods, using periodic signals imitating instrument sounds. We prefer to perform automatic and objective perceptual tests in order to have an exhaustive evaluation; with many orders, fundamental frequencies, and instruments. A listening test would have required too much time to be done in practice.

First, we define the perceptual measure of the approximation error following some concepts of the PEAQ method, cf. [29, 30]. Then we describe the procedure of the objective tests, and finally the results are presented. Note that to use a neutral evaluation, which does not favor the PLF method, we have to choose an error measure which is as different as possible from the criterion $\mathcal{C}$.

5.2.1. Perceptual Mean Square Error

Let G_m^{ref} and G_m^{test} be the magnitudes in [Pa] of the m-th harmonic of the reference and the test sounds, with the frequencies $f_m = mF_0$ in [Hz] with F_0 the fundamental frequency. Because the approximation is evaluated here, G_m^{ref} and G_m^{test} sample the responses of the target $H(f)$ and the estimate $\widetilde{H}(f)$, at the frequencies f_m.

First, the effect of the middle-ear response is taken into account by multiplying the magnitudes by $\Gamma(f) = 10^{\gamma(f)/20}$ where

$$\gamma(f) = -3.6\left(\frac{f}{1000}\right)^{-0.8} - 0.001\left(\frac{f}{1000}\right)^{4} + 6.5\,\mathrm{e}^{-0.6\left(\frac{f}{1000}-3.3\right)^{2}} \quad (15)$$

The function $\gamma(f)$, which is similar to the middle-ear modeling of [29], is actually the inversion of the auditory threshold modeling given in [31]. Hence, the auditory threshold just corresponds to $\Gamma(f_m)G_m = p_0$ with $p_0 = 2\times10^{-5}$ Pa the reference sound level.

Then, to imitate the auditory system's critical bands, the power of the corrected harmonics, $(\Gamma(f_m)G_m)^2$, is summed by processing a filter bank as done with the PEAQ or the MFCC computation, cf. e.g. [32]. We use here a triangular window with a single overlapping, and 100 filters uniformly spaced in the Bark scale of [10].

Finally, with L_k the outputs of the filter bank, the measure of the perceptual error ε is given by

$$\varepsilon = \frac{1}{K}\left(\sum_{k=1}^{K}\frac{\left(L_k^{\text{ref}} - L_k^{\text{test}}\right)^2}{(L_k^{\text{ref}} + p_0)(L_k^{\text{test}} + p_0)}\right)^{\frac{1}{2}} \quad (16)$$

Here, the auditory threshold is implicitly taken into account because of p_0 which imitates the presence of an inner-ear noise, as with the PEAQ method. Moreover, a relative difference is used here in order to take account for the logarithmic sensitivity of the ear to the sound level. Note that this choice is similar to this one of Sec. 3, but it does not favor the PLF method because all the other methods also minimize a relative error in frequency, as the LPC, cf. [5].

Even if the error measure ε and the criterion $\mathcal{C}$ are based on similar concepts, they are different. This fact allows unbiased results, which does not favor the PLF method.

5.2.2. Experimental procedure

For every half-tone between 220 and 440 Hz, the spectrum envelope of a frame is estimated using the True Envelope of [14]. This frame is chosen around the middle of the sustain part. Then, an accurate AR modeling is done using the TELPC method of [33]. This high-order modeling of the spectral envelope gives the target response $H(f)$.

All tested ARMA methods are computed for the obtained frequency responses H of all half-tones. They provide an ARMA(q,q) approximation in the linear frequency scale. The tested methods are the following:

- **Prony:** The well-known Prony method of [1].
- **StMcB:** The Iterative Prefiltering of Steiglitz and McBride, cf. [3], Mode 1 and 2.
- **WLP*:** The warped LPC modeling, cf. [7, 9], for which the warping factor λ^* is this one which optimally fits the Bark scale, cf. [11]. For $F_s = 44.1$ kHz, $\lambda^* = 0.7564$.
- **WLP.6:** The warped LPC modeling with $\lambda = 0.6$.
- **PLF1:** The Mode 1 of the proposed PLF method, cf. Sec. 4.1.
- **PLF1&2:** The proposed PLF method, Mode 1 and Mode 2 of Secs. 4.1 and 4.2.

Remark that a warped AR(q) filter can be converted in principle into a linear ARMA(q,q) filter. Because the purpose of this paper is the low-cost simulation, we prefer to compare the methods with equal simulation complexity, even if the warped methods have less degrees of freedom.

To cancel the effect of the fundamental frequency, the results of the perceptual tests are printed as a function of the adimensional order $\alpha = q/n_h$, where $n_h = 0.5F_s/F_0$ is the number of harmonics between 0 and $F_s/2$. We tested the adimensional orders $\alpha \in \{0.1, 0.2, 0.3\}$. Table 1 summarizes the used orders q for the lower and the higher fundamental frequencies, 220 and 440 Hz.

	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$
$F_0 = 220$ Hz	10	20	30
$F_0 = 440$ Hz	5	10	15

Table 1: Value of the order q as a function of the adimensional order α and the fundamental frequency F_0, for $F_s = 44.1$ kHz.

For all target $H(f)$ and all approximations $\widetilde{H}(f)$, we derive five harmonic spectra which uniformly sample the associated responses. The fundamental frequencies F_k are chosen on a range of two half-tones around the original fundamental frequency F_0, which means $F_k = F_0\,2^{\frac{k}{2\times12}}$, with $-2 \le k \le 2$. This procedure allows to have a refined evaluation of the response approximation.

Finally, every test spectrum, which samples the approximation $\widetilde{H}(f)$, is compared with its associated reference spectrum, which samples $H(f)$. The perceptual measure of the distance is detailed in Sec. 5.2.1.

5.2.3. Results

The results of the objective evaluation are printed in Fig. 2. The original musical sounds come from the sound database of [34]. Since for all the 13 estimations (half-tones between 220 and 440 Hz), 5 discrete spectra have been synthesized and compared, the mean and the standard deviation of Fig. 2 are computed using 65 computations of the perceptual distance, separately for each method, each tested instrument, and each order α. Here the tested instruments are: clarinet, horn, trumpet and violin; we also tested other sustained instruments, such as: trombone, cello, saxophone, flute, and similar results are obtained.

As a general trend, we observe that the proposed PLF method is among the best methods in all cases, whereas the other methods fail at least once. In consequence, even if the PLF method is not clearly the best method in all cases, it is significantly the more robust. Moreover, comparing with the PLF Mode 1 alone, we observe a slight improvement due to the Mode 2 as expected.

Additionally, the behavior of the warped methods has been already observed in [12] using a listening test. With the optimal warping factor $\lambda^* = 0.7564$, in the sense of [11], for the lower orders the results are sometimes worse than the results of $\lambda = 0.6$. This phenomenon has been explained by analyzing the frequency responses. Indeed, with a strong warping, the high frequencies are compressed around $F_s/2$, and the natural slope of the spectrum becomes stronger in the warped frequency scale. As a result, because of the properties of the LPC, cf. [5], the frequency response at high frequencies is overestimated. One solution is then to reduce the value of λ.

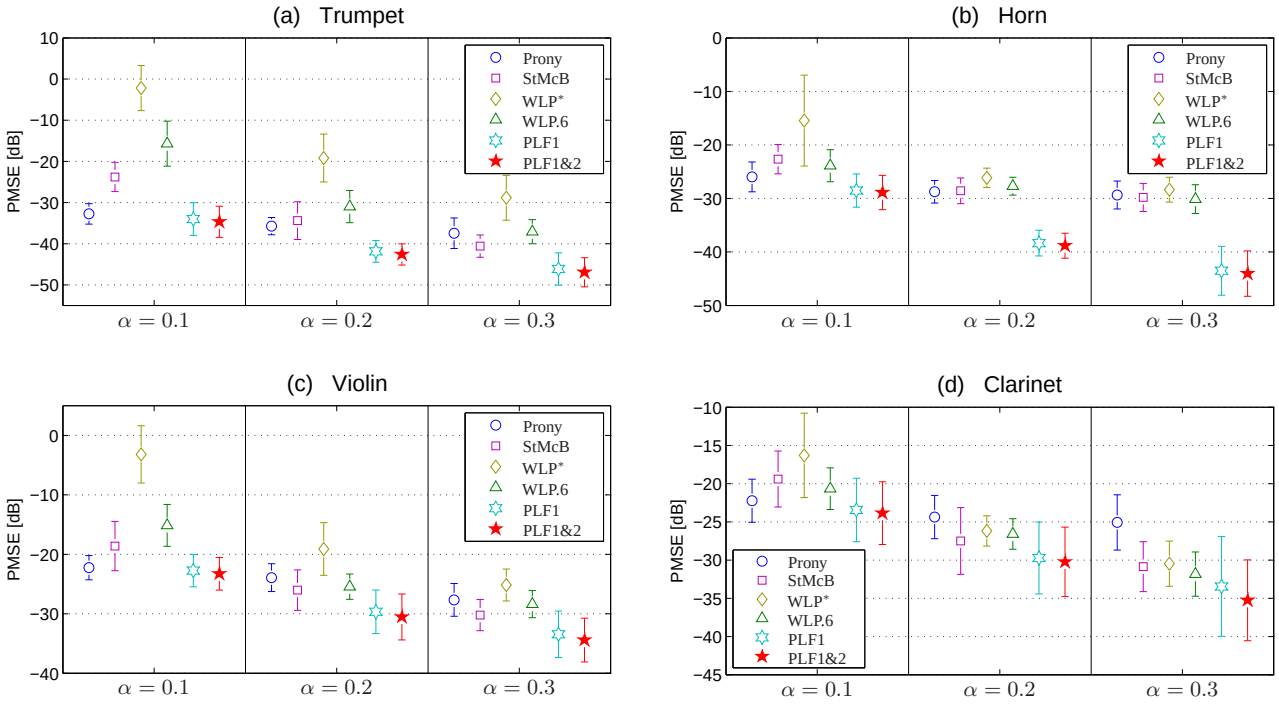


Figure 2: Results of the perceptual evaluation of six ARMA approximation methods, for four instruments and three adimensional orders α. The mean of the perceptual square error (PMSE) is displayed in the decibel scale together with the standard deviation. The tested methods are: Prony's method (Prony), the Iterative Prefiltering Modes 1 and 2 of Steiglitz and McBride (StMcB), the optimal warped LPC (WLP*), the warped LPC with $\lambda = 0.6$ (WLP.6), the PLF method Mode 1 (PLF1) and the PLF method Mode 1 and 2 (PLF1&2).

6. CONCLUSION

In this paper, a novel ARMA approximation for audio signals is presented. It is based on a perceptually meaningful criterion which takes into account the sensitivity of the ear to the frequencies and to sound level via the loudness conversion. The solving algorithm is split into 2 consecutive modes: the first one is a weighted version of the *Iterative Prefiltering* of [3], and the second one is an adaptation of the Gauss-Newton algorithm.

Let's remark that the perceptually-based criterion and the proposed algorithm are two independent contributions of this paper. First the proposed criterion may be optimized using another model or method, second the proposed algorithm can be used with a different frequency weighting. Moreover, even without weighting, for the reasons mentioned earlier (convergence, stability control and conditioning), the proposed Mode 2, is preferable compared to the original Mode 2 of [3].

As illustrated in Fig. 1, this method efficiently focuses the criterion where the original frequency response is audible, and provides less accurate fitting where it is inaudible but with coherent results. A perceptual evaluation is given in Sec. 5.2. Even if the proposed approach does not lead to outstanding results, we notice its stronger robustness. Whereas the other methods may fail in some cases, the PLF method always provides one of the best results.

As a possible improvement of the proposed method, we envisage to apply it with a warped ARMA modeling, cf. [11]. For example, we can notice that a warped ARMA(q,q) filter can be directly converted to an equivalent linear ARMA(q,q) filter. The benefit is to better adapt the model to the criterion $\mathcal{C}$, together with the same number of degrees of freedom, and the same simulation cost. Unfortunately, with a high warping factor λ or high order q, some numerical problems usually occur. In this case, even if the equivalent linear ARMA filter is stable in theory, the finite precision of the floating numbers makes the filter numerically unstable. For example, with $q = 15$, these problems might appear if $\lambda > 0.4$ with the single-precision floating-point.

Unfortunately, this approach may not be suitable in the case of a frame-by-frame analysis-synthesis framework. Indeed, we usually observe strong discontinuities between the estimated spectra of two consecutive frames, which leads to annoying effects. Note that it is also the case in many other ARMA approximation methods. Nevertheless, in the case of the synthesis of a quasi-static spectral envelope, which is under interest in the context of our work, cf. e.g. [35], the proposed PLF method is fully satisfying.

7. REFERENCES

[1] G. Baron de Prony, "Essai expérimental et analytique : sur les lois de la dilatabilité de fluides élastiques et sur celles de la force expansive de la vapeur de l'eau et de la vapeur de l'alkool, à différentes températures," *J. de l'Ecole Polytechnique (Paris)*, vol. 1, no. 2, pp. 24–76, 1795.

[2] J.L. Shanks, "Recursion filters for digital processing," *Geophysics*, vol. 32, no. 1, pp. 33–51, 1967.

[3] K. Steiglitz and L. McBride, "A technique for the identification of linear systems," *IEEE Trans. Automatic Control*, vol. 10, no. 4, pp. 461–464, 1965.

[4] J. Durbin, "The fitting of time-series models," *Revue de l'Institut International de Statistique*, pp. 233–244, 1960.

[5] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, April 1975.

[6] M.H. Hayes, *Statistical Digital Signal Processing and Modeling*, Wiley, 1999, 624 pages.

[7] A. Härmä, "Linear predictive coding with modified filter structures," *IEEE Trans. Speech and Audio Process.*, vol. 9, no. 8, pp. 769–777, 2001.

[8] A. Härmä, M. Karjalainen, L. Savioja, V. Välimäki, U.K. Laine, and J. Huopaniemi, "Frequency-warped signal processing for audio applications," *J. Audio Eng. Soc.*, vol. 48, no. 11, pp. 1011–1031, 2000.

[9] H.W. Strube, "Linear prediction on a warped frequency scale," *J. Acoust. Soc. Amer.*, vol. 68, no. 4, pp. 1071–1076, 1980.

[10] E. Zwicker and E. Terhardt, "Analytical expressions for critical-band rate and critical bandwidth as a function of frequency," *J. Acoust. Soc. Amer.*, vol. 68, pp. 1523, 1980.

[11] J.O. Smith and J.S. Abel, "Bark and ERB bilinear transforms," *IEEE Trans. Speech and Audio Process.*, vol. 7, no. 6, pp. 697–708, 1999.

[12] R. Mignot, H.-M. Lehtonen, and V. Välimäki, "Warped low-order modeling of musical tones," in *Proc. SMC / SMAC*, Stockholm, Sweden, August 2013.

[13] A. El-Jaroudi and J. Makhoul, "Discrete all-pole modeling," *IEEE Trans. Signal Process.*, vol. 39, no. 2, pp. 411–423, 1991.

[14] A. Röbel and X. Rodet, "Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation," in *Proc. Int. Conf. on Digital Audio Effects (DAFx'05)*, Madrid, Spain, September 2005, pp. 30–35.

[15] S. Imai and Y. Abe, "Spectral envelope extraction by improved cepstral method," *Electronics and Communication*, vol. 62-A, no. 4, pp. 10–17, 1979, in Japanese.

[16] R. Mignot and V. Välimäki, "True discrete cepstrum: an accurate and smooth spectral envelope estimation for music processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'14)*, Florence, Italy, May 2014, pp. 7515–7519.

[17] F. Villavicencio, A. Röbel, and X. Rodet, "Extending efficient spectral envelope modeling to Mel-frequency based representation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'08)*, Las Vegas, Nevada, USA, March-April 2008, pp. 1625–1628.

[18] R. Mignot and V. Välimäki, "Perceptual cepstral filters for speech and music processing," in *Proc. IEEE Workshop on Applicat. of Signal Process. to Audio and Acoust. (WASPAA'13)*, New Paltz, NY, USA, October 2013.

[19] D.W. Robinson and R.S. Dadson, "A re-determination of the equal-loudness relations for pure tones," *British Journal of Applied Physics*, vol. 7, no. 5, pp. 166, 1956.

[20] B.C.J. Moore, B.R. Glasberg, and T. Baer, "A model for the prediction of thresholds, loudness, and partial loudness," *J. Audio Eng. Soc.*, vol. 45, no. 4, pp. 224–240, 1997.

[21] S.S. Stevens, "A scale for the measurement of a psychological magnitude: loudness," *Psychological Review*, vol. 43, no. 5, pp. 405, 1936.

[22] D. O'Shaughnessy, *Speech Communication: Human and Machine*, Addison-Wesley, 1987.

[23] "ISO 226 (2003). Contours, Acoustics–Normal Equal-Loudness-Level," *International Organization for Standardization*, Geneva, Switzerland.

[24] A.V. Oppenheim and R.W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 3rd edition, 2009, 1120 pages.

[25] S.A. Gelfand, *Hearing: An Introduction to Psychological and Physiological Acoustics*, Taylor & Francis, 4th edition, 2004, 512 pages.

[26] E. Walter and L. Pronzato, "Identification of parametric models," *Communications and Control Engineering*, 1997, 413 pages.

[27] K. Steiglitz, "On the simultaneous estimation of poles and zeros in speech analysis," *IEEE Trans. Acoust., Speech and Signal Process.*, vol. 25, no. 3, pp. 229–234, 1977.

[28] E.I. Jury, *Inners and Stability of Dynamic Systems*, Wiley-Interscience, New York, 1974, 328 pages.

[29] ITU-R Recommendation BS.1387-1, "Method for Objective Measurements of Perceived Audio Quality," November 2001.

[30] T. Thiede, W.C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J.G. Beerends, and C. Colomes, "PEAQ-The ITU standard for objective measurement of perceived audio quality," *J. Audio Eng. Soc.*, vol. 48, no. 1/2, pp. 3–29, 2000.

[31] E. Terhardt, "Calculating virtual pitch," *Hearing research*, vol. 1, no. 2, pp. 155–182, 1979.

[32] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Process.*, vol. 28, no. 4, pp. 357–366, 1980.

[33] F. Villavicencio, A. Röbel, and X. Rodet, "Improving LPC spectral envelope extraction of voiced speech by True-Envelope estimation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'06)*, Toulouse, France, May 2006, pp. I869–I872.

[34] University of IOWA, Electronic Music Studios, "Musical Instrument Samples," http://theremin.music.uiowa.edu/MIS.html.

[35] R. Mignot and V. Välimäki, "Extended subtractive synthesis of harmonic musical tones," in *136th Audio Engineering Society Convention (AES136)*, Berlin, Germany, April 2014, number 9038.

APPROXIMATIONS FOR ONLINE COMPUTATION OF REDRESSED FREQUENCY WARPED VOCODERS

Gianpaolo Evangelista

Institute for Composition and Electroacoustics
MDW University of Music and Performing Arts
Vienna, Austria
evangelista@mdw.ac.at

ABSTRACT

In recent work, the construction of non-uniform generalized Gabor frames for the time-frequency analysis of signals has been introduced. In particular, while preserving perfect reconstruction, these frames allow for tilings of the time-frequency plane with arbitrary allocation of partially overlapping frequency bands or time intervals.

In a recent paper, the author demonstrated that the construction of such frames can be entirely based on warping operators, which are specified by the required frequency or time warping maps, which, in turn, interpolate the desired frequency or time intervals edges. However, while the online computation of Gabor expansions on non-uniform time intervals presents little or no problem, the computation of Gabor expansions on non-uniform frequency bands requires knowledge of the Fourier transform of the entire signal, which precludes online computation.

In this paper we introduce approximations and ideas for the design of nearly perfect reconstruction analysis and synthesis atoms, which allow for the online computation of time-frequency representations on non-uniform frequency bands.

1. INTRODUCTION

Adapting time-frequency representations, such as the phase vocoder or Short-Time Fourier Transform (STFT), to features of the sound signals or to characteristics of perception, such as glissando, vibrato and 12-tone note system, is a desired goal in the analysis, synthesis and processing and in several contexts ranging from music information retrieval to transformations and special effects.

The STFT's uniform frequency bands can be transformed into non-uniform frequency bands by means of a frequency map, i.e. a monotonically increasing function remapping the frequency axis, as shown in Fig. 1 for adaptation to an equally tempered scale with a constant Q 1/3 octave band splitting.

In a similar way, non-uniform analysis time intervals can be allocated by remapping the time axis of the signal prior to performing uniform time-frequency analysis. The uniform analysis of the time warped signal achieves non-uniform time resolution.

Warping the signal prior to STFT analysis is equivalent to inverse warping the representative elements, i.e. the atoms of the representation.

However, being a time-shift dependent operation, frequency warping disrupts the time organization of signals. Uniform time-frequency analysis of the frequency warped signal results in a frequency dependent distortion of the time axis in the warped time-frequency representation. Similarly, the time warped time-frequency representation shows time dependent distortion of the frequency axis. Thus, warping one variable prior to uniform time-frequency analysis affects the conjugate variable in the representation plane.

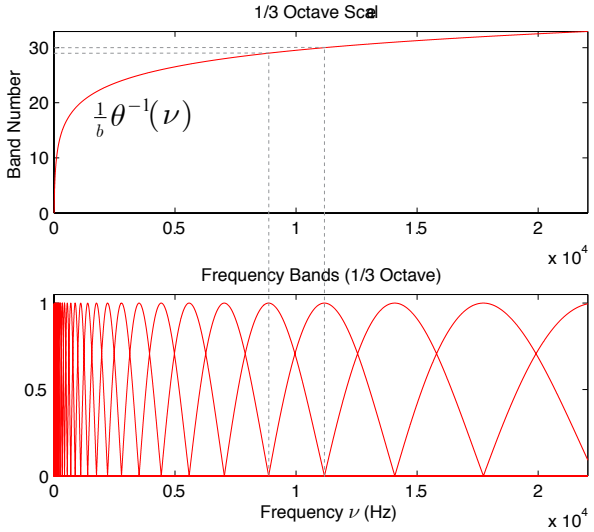


Figure 1: Frequency warping uniform frequency bands according to a $1/3$ of octave scale (top); resulting frequency band characteristics (bottom). Here b is the frequency shift in Hz of the original uniform bands.

In recent work [1, 2, 3], the problem of the construction of flexible frames that allow for arbitrary selection of the frequency bands of their atoms was addressed. In [3] it is shown that the required allocation of generalized Gabor atoms can be specified according to a frequency or time warping map. In [4] the STFT redressing method is introduced, which, with the use of additional warping in time-frequency, shows under which conditions one can have generalized Gabor frames. These conditions are dictated by the interaction of sampling in time-frequency and frequency or time warping operators.

The results in the previously cited work show that arbitrary allocation of the atoms is generally possible in the so called *painless case*, i.e. in the case of finite time support of the windows for arbitrary time interval allocation and of finite frequency support of the

windows for arbitrary frequency band allocation.

Since online computation of the generalized Gabor analysis / synthesis is only possible with finite duration windows, the arbitrary frequency band allocation is not exactly feasible in applications that require real-time, while the arbitrary time interval allocation presents little or no problem.

In this paper, we address the problem of online computation of generalized Gabor analysis with arbitrary frequency band allocation, resorting to approximations that lead to near perfect reconstruction methods.

The paper is organized as follows. In Section 2 we review the concept of applying time and frequency warping to time-frequency representations derived from the continuous time Short-Time Fourier Transform, pointing out the problems introduced by dispersion and resolving them with the redressing method, which involves a further warping operations in the time-frequency domain. In Section 3 we apply the redressing method to frames, which allow for sampled time-frequency analysis and synthesis, and we provide the conditions by which the redressing of dispersion is exact. In Section 4 we introduce approximations suitable for the online computation of redressed frame expansions. In Section 5 we draw our conclusions.

2. REDRESSED WARPED TIME-FREQUENCY

In this section we review concepts that lead to the redressing method for the time alignment of the frequency warped Short-Time Fourier Transform (STFT).

In order to set the notation, the uniform STFT is obtained by applying the operator $\mathcal{S}$ to the signal s:

$$[\mathcal{S}s](\tau,\nu) = \langle s, h_{\tau,\nu}\rangle = \langle s, \mathbf{T}_\tau \mathbf{M}_\nu h_{0,0}\rangle = \int_{-\infty}^{+\infty} s(t)\overline{h_{0,0}(t-\tau)}e^{-j2\pi\nu(t-\tau)}dt, \tag{1}$$

where $\mathbf{T}_\tau s(t) = s(t-\tau)$ is the time-shift operator, $\mathbf{M}_\nu s(t) = e^{j2\pi\nu t}s(t)$ is the modulation operator and the overbar denotes complex conjugation. The operator $\mathcal{S}$ acts over time signals and the frequency ν is considered as a parameter. In (1), the analysis windows

$$h_{\tau,\nu}(t) = [\mathbf{T}_\tau \mathbf{M}_\nu h_{0,0}](t) = h_{0,0}(t-\tau)e^{j2\pi\nu(t-\tau)} \tag{2}$$

are modulated and shifted versions of a unique time window $h_{0,0}$. Their Fourier transforms are related to the Fourier transform of the original window $\hat{h}_{0,0}$ as follows:

$$\hat{h}_{\tau,\nu}(f) = \hat{h}_{0,0}(f-\nu)e^{-j2\pi f\tau}, \tag{3}$$

which are frequency shifted and modulated versions of the Fourier transform of the window $h_{0,0}$.

Since $[\mathcal{S}s](\tau,\nu) = s(\tau) * \overline{h_{0,\nu}(-\tau)}$, where the symbol $*$ denotes convolution, one can rewrite (1) in the frequency domain w.r.t. τ as follows:

$$\widehat{[\mathcal{S}s]}(f,\nu) = \overline{\hat{h}_{0,\nu}(f)}\hat{s}(f) = \overline{\hat{h}_{0,0}(f-\nu)}\hat{s}(f). \tag{4}$$

Non-uniform time-frequency representations can be obtained from uniform ones via time and / or frequency warping, as discussed in Section 2.2, after we formally introduce warping operators in the next section.

2.1. Warped STFT

The warped STFT can be obtained by warping the signal prior to applying the STFT operator. The most general warping operator involves combined time-frequency warping, i.e. time dependent frequency warping or, equivalently, frequency dependent time warping. For the purpose of this paper we consider separable warping, which can be computed by cascading time invariant frequency warping with frequency independent time warping. We mostly focus on pure frequency warping.

A 1D warping operator performs a remapping of the abscissae, as obtained through function composition. A time warping operator $\mathbf{W}_\gamma$ is completely characterized by a function composition operator in the time domain:

$$s_{tw} = \mathbf{W}_\gamma s = s \circ \gamma, \tag{5}$$

where γ is the time warping map and s_{tw} is the time-warped signal. Similarly, a frequency warping operator $\mathbf{W}_{\tilde{\theta}}$ is completely characterized by a function composition operator $\mathbf{W}_\theta$ in the frequency domain:

$$\hat{s}_{fw} = \widehat{\mathbf{W}_{\tilde{\theta}}s} = \hat{\mathbf{W}}_{\tilde{\theta}}\hat{s} = \mathbf{W}_\theta \hat{s} = \hat{s}\circ\theta, \tag{6}$$

where θ is the frequency warping map, which transforms the Fourier transform $\hat{s} = \mathcal{F}s$ of a signal s into the Fourier transform $\hat{s}_{fw} = \mathcal{F}s_{fw}$ of another signal s_{fw}, where $\mathcal{F}$ is the Fourier transform operator and the hat over a symbol denotes the Fourier transformed quantity (signal or operator). We affix the $\tilde{}$ symbol over the map θ as a reminder that the map operates in the frequency domain. Accordingly, we have $\mathbf{W}_{\tilde{\theta}} = \mathcal{F}^{-1}\hat{\mathbf{W}}_{\tilde{\theta}}\mathcal{F} = \mathcal{F}^{-1}\mathbf{W}_\theta\mathcal{F}$.

If the warping map is one-to-one and almost everywhere differentiable then a unitary form of the warping operator can be defined by amplitude scaling, as given by the square root of the derivative of the map (dilation function). For example, a unitary frequency warping operator $\mathbf{U}_{\tilde{\theta}}$ has frequency domain action

$$\hat{s}_{fw}(\nu) = \left[\widehat{\mathbf{U}_{\tilde{\theta}}s}\right](\nu) = \sqrt{\left|\tfrac{d\theta}{d\nu}\right|}\hat{s}(\theta(\nu)), \tag{7}$$

where ν denotes frequency. We assume henceforth that all warping maps are almost everywhere increasing so that the magnitude sign can be dropped from the derivative under the square root.

2.2. Warped Time-Frequency Representations

Remapping signals prior to STFT allows for a reinterpretation of the representation elements: while the organization of the representation (tiling) remains the same, the elements capture different components of the signal. Time warping dilates / shrinks and displaces the characteristic analysis time intervals (resolution and centers) w.r.t. signals. Frequency warping remaps the characteristic analysis frequency bands w.r.t. signals (bandwidths and centers).

Given a frequency warping operator $\mathbf{W}_{\tilde{\theta}}$, the warped STFT is defined through the operator $\mathcal{S}_{\tilde{\theta}}$ as follows

$$[\mathcal{S}_{\tilde{\theta}}s](\tau,\nu) = [\mathcal{S}\mathbf{W}_{\tilde{\theta}}s](\tau,\nu) = \langle \mathbf{W}_{\tilde{\theta}}s, h_{\tau,\nu}\rangle = \left\langle s, \mathbf{W}^\dagger_{\tilde{\theta}}h_{\tau,\nu}\right\rangle, \tag{8}$$

which is indeed a warped version of (1), where $\mathbf{W}^\dagger_{\tilde{\theta}}$ is the adjoint of the warping operator. If the warping operator is unitary then we have $\mathbf{W}^\dagger_{\tilde{\theta}} = \mathbf{W}^{-1}_{\tilde{\theta}} = \mathbf{W}_{\tilde{\theta}^{-1}}$. In that case, warping the signal

prior to STFT is perfectly equivalent to perform STFT analysis with inversely frequency warped windows. The warped STFT is unitarily equivalent to the STFT so that a number of properties concerning conditioning and reconstruction hold [5].

The Fourier transforms of the frequency warped STFT analysis elements are

$$\hat{\hat{h}}_{\tau,\nu}(f) = \left[\widehat{\mathbf{W}_{\tilde{\theta}^{-1}} h_{\tau,\nu}}\right](f) = \sqrt{\tfrac{d\theta^{-1}}{df}}\hat{h}_{0,0}(\theta^{-1}(f)-\nu)e^{-j2\pi\theta^{-1}(f)\tau}, \tag{9}$$

which shows how the analysis elements are obtained from frequency warped modulated windows centered at frequencies $f = \theta(\nu)$. The windows are time-shifted with dispersive delay, where the group delay is $\tau\frac{d\theta^{-1}}{df}$.

Frequency warping generally disrupts the time organization of signals. Indeed, the time-shift operator $\mathbf{T}_\tau$ does not commute with the frequency warping operator:

$$\left[\widehat{\mathbf{W}_{\tilde{\theta}}\mathbf{T}_\tau s}\right](\nu) = \left[\mathbf{W}_\theta\widehat{\mathbf{T}_\tau s}\right](\nu) = e^{-j2\pi\theta(\nu)\tau}\hat{s}(\theta(\nu)), \tag{10}$$

which is different from $\left[\widehat{\mathbf{T}_\tau\mathbf{W}_{\tilde{\theta}}s}\right](\nu) = e^{-j2\pi\nu\tau}\hat{s}(\theta(\nu))$, unless the map θ is the identity map. Thus, an event that starts at time T in the original signal, is dispersed into events starting at times $\phi_d(\nu)T$, where $\phi_d(\nu) = \theta(\nu)/\nu$ is the phase delay of the warping map, which depends on frequency unless the map is linear.

In the applications we would like to produce spectrograms with non-uniform time or frequency resolution but the dispersion introduced by warping results in misalignment and spreading of the time-frequency components in the conjugate variable of the warped one. In the next section we will show how further warping in the time-frequency plane can redress the warped representations.

2.3. Redressing the Warped STFT

To address the problem of realigning the frequency warped STFT $[\mathcal{S}_{\tilde{\theta}}s](\tau,\nu)$, consider its Fourier transform w.r.t. the time variable τ. This can be written in the form (4) by replacing the Fourier transform of the signal with that of the frequency warped signal:

$$\left[\widehat{\mathcal{S}\mathbf{W}_{\tilde{\theta}}s}\right](f,\nu) = \overline{\hat{h}_{0,0}(f-\nu)}\sqrt{\tfrac{d\theta}{df}}\hat{s}(\theta(f)). \tag{11}$$

Recall that f is the frequency variable conjugate to time τ in the time-frequency plane. Performing unitary frequency warping on this variable by means of the inverse frequency map θ^{-1} one obtains:

$$\left[\mathbf{W}_{\tilde{\theta}^{-1}}\widehat{\mathcal{S}\mathbf{W}_{\tilde{\theta}}s}\right](f,\nu) = \overline{\hat{h}_{0,0}(\theta^{-1}(f)-\nu)}\hat{s}(f), \tag{12}$$

where we have used the fact that

$$1 = \tfrac{d\left[\theta(\theta^{-1}(f))\right]}{df} = \tfrac{d\theta}{d\alpha}\big|_{\alpha=\theta^{-1}(f)}\tfrac{d\theta^{-1}}{df}. \tag{13}$$

The redressed frequency warped STFT (12) is again in the form of a time-invariant filtering operation (convolution in time domain) where the filters are frequency warped versions of the modulated windows in (4). As a result, the dispersive delays in the analysis elements (9) are brought back to non-dispersive delays, the Fourier transform of the redressed analysis elements being

$$\hat{\hat{\tilde{h}}}_{\tau,\nu}(f) = \left[\mathbf{T}_\tau\widehat{\mathbf{W}_{\tilde{\theta}}h_{0,\nu}}\right](f) = \hat{h}_{0,0}(\theta^{-1}(f)-\nu)e^{-j2\pi f\tau}. \tag{14}$$

It is possible to interpret (12) as the similarity transformation $\mathbf{W}_{\tilde{\theta}}^\dagger\mathcal{S}\mathbf{W}_{\tilde{\theta}}$ on the STFT operator, which is time-shift covariant.

3. REDRESSED WARPED GABOR FRAMES

In this section we review the definition of Gabor and warped Gabor frames. We would like to apply the same redressing method used in the previous section to counteract dispersion and realign time. However, the Gabor expansion coefficients are time-frequency samples of the STFT so that only a discrete version of time-frequency unwarping can be set forth.

3.1. Gabor frames

Given a window function h and two sampling parameters $a, b > 0$, the set of functions

$$\mathcal{G}(h,a,b) = \{\mathbf{T}_{na}\mathbf{M}_{mb}h : q,n \in \mathbb{Z}\} \tag{15}$$

is called a *Gabor system*. A signal s can be projected over a Gabor system by taking the scalar products $\langle s, \mathbf{T}_{na}\mathbf{M}_{mb}h\rangle$. These are exactly evaluations of the STFT of a signal with window h at the time-frequency grid of points (na, qb). Here we have defined the Gabor system using the same convention as in the definition (1) of the STFT. Usually, Gabor systems are defined with a reverse order of time-shift and frequency modulation operators, i.e. $\{\mathbf{M}_{mb}\mathbf{T}_{na}h : q,n \in \mathbb{Z}\}$. However, the extra phase factors that are introduced to convert from one definition to the other are perfectly irrelevant when establishing properties of the system. Even in the computation the extra phase factors cancel out in the analysis-synthesis algorithm, so they can be ignored.

A sequence of functions $\{\psi_l\}_{l\in I}$ in the Hilbert space $\mathcal{H}$ is called a frame if there exist both positive constant lower and upper bounds A and B, respectively, such that

$$A\|s\|^2 \le \sum_{l\in I}|\langle s,\psi_l\rangle|^2 \le B\|s\|^2 \quad \forall s \in \mathcal{H}, \tag{16}$$

where $\|s\|^2 = \langle s,s\rangle$ is the norm square or total energy of the signal. Frames generate signal expansions, i.e., the signal can be perfectly reconstructed from its projections over the frame.

A Gabor system that is a frame is called a *Gabor frame*. In this case, the signal can be reconstructed from the corresponding samples of the STFT. While not unique, reconstruction can be achieved with the help of a dual frame, which in turn is a Gabor frame generated by a dual window $\tilde{h}$. Perfect reconstruction depends on the choice of the window and the sampling grid. One can show that there exist no Gabor frames when $ab > 1$.

3.2. Warping Gabor frames

From (16) it is easy to see that any unitary operation on a frame results in a new frame with the same frame bounds A and B [5]. In particular, unitary operators can be applied to Gabor frames to obtain new frames. Depending on the operator, the resulting frames are not necessarily of the Gabor type, as the atoms are not generated by shifting and modulating a single window function.

Conceptually, starting from a Gabor frame (analysis) $\{\varphi_{n,q}\}_{q,n\in\mathbb{Z}}$ and dual frame (synthesis) $\{\gamma_{n,q}\}_{n,q\in\mathbb{Z}}$:

$$\begin{aligned}\varphi_{n,q} &= \mathbf{T}_{na}\mathbf{M}_{qb}h\\ \gamma_{n,q} &= \mathbf{T}_{na}\mathbf{M}_{qb}g,\end{aligned} \tag{17}$$

where h and g are dual windows, warped frames can be generated by unitarily warping the signal s prior to analysis and unitarily unwarping it after the synthesis:

$$\begin{aligned}s = \mathbf{U}_{\tilde{\theta}}^{\dagger}\sum_{n,q\in Z}\langle\mathbf{U}_{\tilde{\theta}}s,\varphi_{n,q}\rangle\gamma_{n,q} =\\ \sum_{n,q\in Z}\left\langle s,\mathbf{U}_{\tilde{\theta}}^{\dagger}\varphi_{n,q}\right\rangle\mathbf{U}_{\tilde{\theta}}^{\dagger}\gamma_{n,q},\end{aligned} \tag{18}$$

where $\mathbf{U}_{\tilde{\theta}}$ is a unitary frequency warping operator. Defining the frequency warped frame (analysis) $\{\tilde{\varphi}_{n,q}\}_{q,n\in\mathbb{Z}}$ and dual frame (synthesis) $\{\tilde{\gamma}_{n,q}\}_{n,q\in\mathbb{Z}}$ as follows:

$$\begin{aligned}\tilde{\varphi}_{n,q} &= \mathbf{U}_{\tilde{\theta}}^{\dagger}\varphi_{n,q} = \mathbf{U}_{\tilde{\theta}^{-1}}\mathbf{T}_{na}\mathbf{M}_{qb}h\\ \tilde{\gamma}_{n,q} &= \mathbf{U}_{\tilde{\theta}}^{\dagger}\gamma_{n,q} = \mathbf{U}_{\tilde{\theta}^{-1}}\mathbf{T}_{na}\mathbf{M}_{qb}g,\end{aligned} \tag{19}$$

one obtains the signal expansion

$$s = \sum_{n,q\in Z}\langle s,\tilde{\varphi}_{n,q}\rangle\tilde{\gamma}_{n,q}. \tag{20}$$

Just as Gabor frames can be obtained by uniformly sampling the integral STFT, the warped frames can be obtained as a result of nonuniform sampling in time-frequency. Nonuniform sampling theorems based on a time warping map were introduced in [6] and their adaptation to frequency sampling is immediate. Applications of frequency warping to time-frequency analysis date back to [7]. However, warped Gabor frames suffer from the same problem as the warped STFT: as a result of frequency warping, the time organization of the analysis and synthesis systems is disrupted; the windows are time-shifted with frequency dependent shifts. Indeed the Fourier transforms of the warped Gabor frame elements are

$$\hat{\tilde{\varphi}}_{n,q}(f) = \sqrt{\tfrac{d\theta^{-1}}{df}}\hat{h}(\theta^{-1}(f)-qb)e^{-j2\pi\theta^{-1}(f)na}, \tag{21}$$

which bear frequency dispersive delays. In other words dispersive time samples are produced by the direct application of the warped frame analysis. Similar problems are encountered when time-warping Gabor frames.

The magnitude Fourier transforms $\hat{h}(\theta^{-1}(f)-qb)$ of a set of frequency warped modulated windows corresponding to $1/3$ octave frequency resolution is shown in Fig. 1, together with a scaled version $\frac{1}{b}\theta^{-1}$ of the warping map, which maps warped frequency to fractional band number, i.e., the integer values of $\frac{1}{b}\theta^{-1}$ correspond to the center frequencies of the bands.

3.3. Redressing Warped Gabor Frames

The evaluation of the warped Gabor expansion coefficients

$$\tilde{c}_{n,q} = \langle s,\tilde{\varphi}_{n,q}\rangle \tag{22}$$

is identical to that of a time-frequency sampled warped STFT. In order to redress the frequency warped STFT into a time covariant representation we have introduced additional inverse frequency warping with respect to the time variable τ in the time-frequency plane. However, in the warped Gabor frames (19) this variable is sampled at instants na. Therefore, in order to parallel our warped STFT redressing procedure in the warped Gabor frames case, one can only apply a discrete-time form of frequency warping to the time index n.

It is possible to show [8, 9] that if the discrete-time frequency warping map ϑ is one-to-one and onto $[-\frac{1}{2},+\frac{1}{2}[$, and almost everywhere differentiable there, then the set of sequences

$$\eta_m(n) = \int_{-\frac{1}{2}}^{+\frac{1}{2}}\sqrt{\tfrac{d\vartheta}{d\nu}}e^{j2\pi(n\nu-m\vartheta(\nu))}d\nu, \tag{23}$$

where $n,m\in\mathbb{Z}$, forms an orthonormal basis of $\ell^2(\mathbb{Z})$. These are recognized as generalized Laguerre sequences [10, 11, 12], which are the inverse discrete-time Fourier transforms of warped harmonic complex sinusoids in the frequency domain interval $[-\frac{1}{2},+\frac{1}{2}[$. The map ϑ can be extended over the entire real axis as congruent modulo 1 to a 1-periodic function.

Given a sequence $\{x(n)\}$ in $\ell^2(\mathbb{Z})$, the scalar products

$$\tilde{x}(m) = \langle x,\eta_m\rangle_{\ell^2(\mathbb{Z})} \tag{24}$$

generate another sequence $\{\tilde{x}(m)\}$ in $\ell^2(\mathbb{Z})$, which satisfies

$$\hat{\tilde{x}}(\nu) = \sqrt{\tfrac{d\vartheta^{-1}}{d\nu}}\hat{x}(\vartheta^{-1}(\nu)), \tag{25}$$

where the $\hat{}$ symbol, when applied to sequences, denotes discrete-time Fourier transform. Thus, $\overline{\eta_m(n)}$ defines the nucleus of an inverse unitary frequency warping $\ell^2(\mathbb{Z})$ operator $\mathbf{D}_{\tilde{\vartheta}^{-1}} = \mathbf{D}_{\tilde{\vartheta}}^{\dagger}$. Clearly, the transposed conjugate sequences $\mu_m(n) = \eta_n(m)$ form the nucleus of a unitary frequency warping $\ell^2(\mathbb{Z})$ operator $\mathbf{D}_{\tilde{\vartheta}}$.

In order to limit or eliminate time dispersion in the frequency warped Gabor expansion, one can apply the discrete-time frequency warping operator $\mathbf{D}_{\tilde{\vartheta}^{-1}}$ to the time sequence of expansion coefficients over the warped Gabor frame (22), i.e., with respect to index n. Since the operator is applied only on the time index, for generality, one can include dependency of the map and of the sequences η_n on the frequency index q, which will be useful in the sequel. The new coefficients are obtained as follows:

$$\begin{aligned}\tilde{\tilde{c}}_{n,q} = \left[\mathbf{D}_{\tilde{\vartheta}_q^{-1}}\tilde{c}_{\bullet,q}\right](n) = \sum_{m\in\mathbb{Z}}\overline{\eta_{n,q}(m)}\,\langle s,\tilde{\varphi}_{m,q}\rangle =\\ \left\langle s,\sum_{m\in\mathbb{Z}}\eta_{n,q}(m)\tilde{\varphi}_{m,q}\right\rangle.\end{aligned} \tag{26}$$

In order to reconstruct the signal from the coefficients $\tilde{\tilde{c}}_{n,q}$ one can first recover the coefficients $\tilde{c}_{n,q}$, which stems from the completeness and orthogonality of the set $\{\eta_{n,q}\}_{n\in\mathbb{Z}}$, and then combine them with the dual warped frame elements:

$$s = \sum_{n,q\in\mathbb{Z}}\tilde{c}_{n,q}\tilde{\gamma}_{n,q} = \sum_{n,q\in\mathbb{Z}}\sum_{m\in\mathbb{Z}}\tilde{\tilde{c}}_{m,q}\eta_{m,q}(n)\tilde{\gamma}_{n,q}. \tag{27}$$

Hence, defining the redressed frequency warped Gabor analysis and synthesis frames as follows:

$$\begin{aligned}\tilde{\tilde{\varphi}}_{n,q} &= \mathbf{D}_{\tilde{\vartheta}_q^{-1}}\tilde{\varphi}_{\bullet,q} = \sum_m\eta_{n,q}(m)\tilde{\varphi}_{m,q}\\ \tilde{\tilde{\gamma}}_{n,q} &= \mathbf{D}_{\tilde{\vartheta}_q^{-1}}\tilde{\gamma}_{\bullet,q} = \sum_m\eta_{n,q}(m)\tilde{\gamma}_{m,q},\end{aligned} \tag{28}$$

from (26) and (27) we have:

$$s = \sum_{n,q\in Z} \tilde{\tilde{c}}_{n,q}\tilde{\tilde{\gamma}}_{n,q} = \sum_{n,q\in(Z)} \langle s, \tilde{\tilde{\varphi}}_{n,q}\rangle \tilde{\tilde{\gamma}}_{n,q}. \qquad (29)$$

Indeed, the redressing discrete-time warping transformation is based on an orthonormal and complete expansion in $\ell^2(\mathbb{Z})$, which leads to the unitary equivalence of the redressed warped frames with the warped frames.

Exploiting the periodicity of the discrete-time redressing frequency warping map one can show that the Fourier transforms of the redressed frame is

$$\hat{\tilde{\tilde{\varphi}}}_{n,q}(f) = A(f)\hat{h}(\theta^{-1}(f) - qb)e^{-j2\pi n\vartheta_q(a\theta^{-1}(f))}, \qquad (30)$$

where

$$A(f) = \sqrt{\tfrac{d\theta^{-1}}{df}}\left.\sqrt{\tfrac{d\vartheta_q}{d\nu}}\right|_{\nu=a\theta^{-1}(f)}. \qquad (31)$$

Hence, the effect of the dispersive delays would be counteracted if

$$\vartheta_q(a\theta^{-1}(f)) = d_q f \qquad (32)$$

for any $f \in \mathbb{R}$, where d_q are positive constants controlling the time scale in each frequency band. In this case, the Fourier transforms of the redressed frame elements simply become:

$$\hat{\tilde{\tilde{\varphi}}}_{n,q}(f) = \sqrt{\tfrac{d_q}{a}}\hat{h}(\theta^{-1}(f) - qb)e^{-j2\pi n d_q f}. \qquad (33)$$

Furthermore, if all d_q are identical, all the time samples would be aligned to a uniform time scale throughout frequencies.

However, each map ϑ_q is constrained to be congruent modulo 1 to a 1-periodic function, while the global warping map θ can be arbitrarily selected. Furthermore, having to be one-to-one in each unit interval, the functions ϑ_q can at most experience an increment of 1 there.

The problem of linearizing the phase is illustrated in Fig. 2, where the black curve is the amplitude scaled warping map $d_q\theta(\nu)$ and the gray curve represents the map $\vartheta_q(a\nu)$, which is $1/a$-periodic, both plotted in the abscissa $\nu = \theta^{-1}(f)$. Amplitude scaling the warping map θ allows the values of the map to lie in the range of the discrete-time warping map ϑ_q. The amplitude scaling factors are the new time sampling intervals d_q of the redressed warped Gabor expansion.

If the window h is chosen to have compact support in the frequency domain, which is the so called "painless" case, one can exactly eliminate the dispersive delays with the help of (28). In fact, suppose for simplicity that the bandwidth of the window h is Kb, with K a positive integer, i.e., $\hat{h}(f) = 0$ for $|f| \geq Kb/2$. The choice of the initial sampling interval a allows all the maps $\{\vartheta_q\}_{q\in\mathbb{Z}}$ to be arbitrarily specifiable to match $d_q\theta(\nu)$ in the intervals where the Fourier transforms of the warped modulated windows (warped frame elements) are nonzero. Hence, condition (32) only needs to be satisfied by the map ϑ_q in this interval. Equivalently, we require

$$\vartheta_q(a\nu) = d_q\theta(\nu), \quad (q - \tfrac{K}{2})b < \nu < (q + \tfrac{K}{2})b, \qquad (34)$$

which is possible if on one hand the variation of the argument of the map ϑ_q in (34) satisfies

$$a[(q + \tfrac{K}{2})b - (q - \tfrac{K}{2})b] = Kab \leq 1 \qquad (35)$$

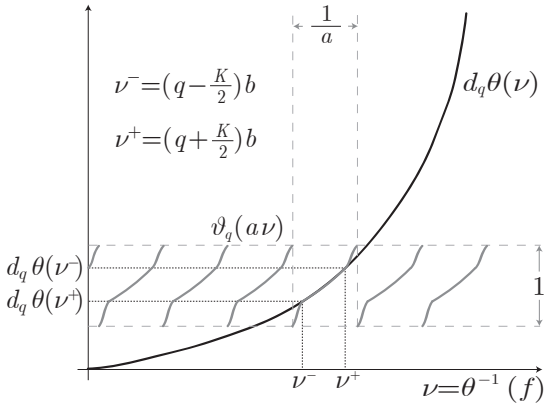


Figure 2: Locally eliminating dispersion by means of discrete-time frequency warping. Black line: curve derived from the original map θ by amplitude scaling. Gray line: discrete-time frequency warping characteristics for local delay linearization.

and, on the other hand, if also the variation of the map ϑ_q over the warped modulated window bandwidth satisfies

$$d_q[\theta((q + \tfrac{K}{2})b) - \theta((q - \tfrac{K}{2})b)] = d_q B_q \leq 1, \qquad (36)$$

where $B_q = \theta((q + \frac{K}{2})b) - \theta((q - \frac{K}{2})b)$ is the full bandwidth of the warped modulated window. The first of these conditions only requires $ab \leq 1/K$, which does not depend on q and can be satisfied assigning sufficient redundancy (oversampling) of the initial Gabor frame. Incidentally, this is the same condition for the original Gabor system to form a frame. A valid choice is $K = 2$, which requires $ab \leq 1/2$. For the second condition, one needs to select $d_q \leq 1/B_q$, as intuitively clear from the sampling theorem. If there is an upper bound B to the bandwidths B_q then one can choose identical $d_q = 1/B$, $q \in \mathbb{Z}$, to satisfy the sampling condition with uniform rates.

In the general case, a perfect time realignment of the components is not guaranteed. By construction, the redressed warped Gabor systems are guaranteed to be frames for any choice of the maps ϑ_q satisfying the stated periodicity conditions, even when the phase is not completely linearized. Locally, within the essential bandwidths of the warped modulated windows it is possible to linearize the phase of the complex exponentials in (30).

4. ONLINE COMPUTATION AND APPROXIMATIONS

The warping map design method to eliminate dispersive sampling in the frequency warped Gabor elements is exact when the elements are compactly supported in the frequency domain. This type of frames are definitely suitable for offline computation using simple and efficient frequency domain techniques [1].

Since the computation of Gabor expansion coefficients is not causal, in online computation one requires the frame elements to have compact support in the time domain. Starting with a finite duration window, one can linearize the phase and choosing suitable sampling parameters, one can eliminate dispersion within the essential bandwidths of the warped modulated windows [4]. However, this is still not sufficient for online computation purposes.

In fact, generally, the modulated frequency warped windows will not have compact support in the time domain even if the original window had this property.

In order to provide an approximation suitable for online computation, one can observe that the window h is narrow band low pass and the warping map is differentiable. Therefore, in the argument of $\hat{h}$ in (30) one can expand $\theta^{-1}(f)$ in Taylor series around the point $\theta(qb)$. Truncating to first order, which corresponds to a local linearization of the warping map within the bandwidth of the window, one obtains:

$$\theta^{-1}(f) \approx qb + \frac{1}{\tau_q}(f - \theta(qb)), \tag{37}$$

where

$$\tau_q = \left.\frac{d\theta}{df}\right|_{f=qb} \tag{38}$$

is the group delay associated to the warping map $\theta(f)$ at frequency $f = qb$. Thus, we have the following approximation:

$$\hat{g}_q(f) = \hat{h}(\theta^{-1}(f) - qb) \approx \hat{h}\left(\tfrac{f-\theta(qb)}{\tau_q}\right) \tag{39}$$

Thus, in this approximation, the window $g_q(t)$ is simply obtained by dilating and modulating the prototype window h, in which the local group delay acts as scaling factor:

$$g_q(t) = \tau_q h(\tau_q t) e^{j2\pi\theta(qb)t}. \tag{40}$$

Hence, if the prototype window has compact support in the time domain, all its approximate warped modulated versions will have compact support.

In order to perform online computations of the redressed frequency warped Gabor expansion, one can start from a prototype window h that has compact support in the time domain, where aliasing is canceled in the time-domain through overlap-add, such as the time-domain cosine window, given by

$$h(t) = \begin{cases} \sqrt{\frac{2b}{R}} \cos\frac{\pi t}{T} & \text{if } -\frac{T}{2} \leqslant t < +\frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

where T is the total duration of the window, $R > 1$ is an integer, b is the frequency sampling interval and we let the time shift parameter $a = T/R$.

In the redressed frame (30) one replaces the warped modulated windows by the scaled windows in (39). Furthermore, one performs redressing in the essential bandwidth and considers uniform time sampling within each analysis band. This requires suitable setting of the time-frequency sampling rates, which we are goind to illustrate for the cosine window example.

The Fourier transform of the cosine window, given by

$$\hat{h}(\nu) = \sqrt{\tfrac{b}{2R}}(\mathrm{sinc}(\nu T - \tfrac{1}{2}) + \mathrm{sinc}(\nu T + \tfrac{1}{2})), \tag{42}$$

is plotted in Fig. 3, from which one can see that the main lobe has bandwidth $3/T = 3/Ra$. Assuming this as the essential bandwidth in which to linearize the phase, in order to satisfy (32) here, one needs to select $R \geq 3$, which is the analogon of (35), and $d_q B_q \leq 1$, which is the analogon of (36), where now $B_q = \theta(qb + \frac{3}{2T}) - \theta(qb - \frac{3}{2T})$.

Concurrently, the parameter T can be selected according to the smallest required essential bandwidth. For example, in the case of a tempered scale warping map, in order to have sufficient

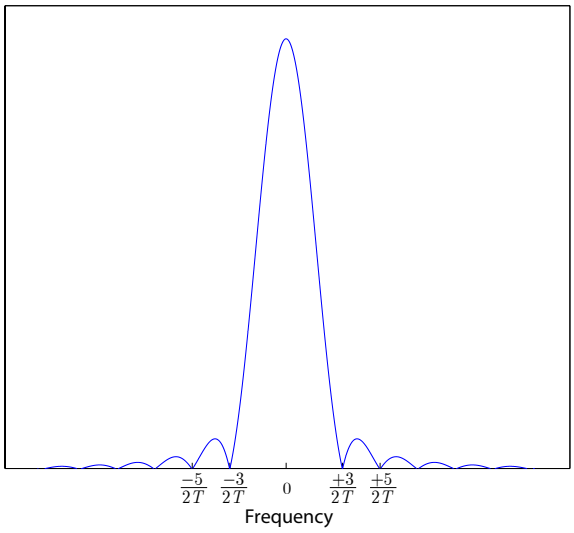


Figure 3: Magnitude Fourier transform of the cosine window.

frequency resolution one can select $\frac{3}{2T} = f_0$, where f_0 is the frequency of the smallest tone to be represented, so that adjacent tones fall away from the main frequency lobe of the window, which gives $a = \frac{3}{2Rf_0}$.

The frequency shift parameter b must be chosen so that $ab \leq 1/R$ for the original Gabor system to be a frame. For $R = 3$ and the chosen value of a, this gives $b \leq 2f_0/3$. However, in practice one would like the tones of the scale to be adequately represented by the warped bands; moreover, narrower bands improve the approximation of the warped modulated windows with the scaled modulated cosine windows. In our examples we chose $b = f_0/3$. The quality of the approximation can be evaluated by comparing the magnitude Fourier transform of the widows, shown in Fig. 4 for the case of 1/3 octave warping map for the centerband frequency of 356.02 Hz. The two modulated windows are shown in the time domain in Fig. 5. One can see that the scaled cosine modulated window closely approximates the warped window on a finite interval, truncating its tails.

As a refinement of the finite length window approximation, one can consider the truncation of the modulated warped windows on a larger interval than that offered by the approximating scaled cosine. The length of the interval can be estimated at 1.5 the support of the approximating cosine window. In this case one can obtain a reconstruction error norm in the order of 10^{-5} of the norm of the signal for the 1/3-octave warping map. Informal perceptual tests show no audible artifacts attached to the approximate analysis and synthesis procedure. A deeper analysis of the approximation error will be the object of a forthcoming paper.

Since the center frequencies of the warped Gabor frames are not equally spaced, the computation of the transform cannot be directly performed by means of the Fast Fourier Transform. Real multirate filterbanks can be designed by combining the complex conjugate channels. Therefore the complexity is linear in the number of samples, where the number of channels is a proportional factor.

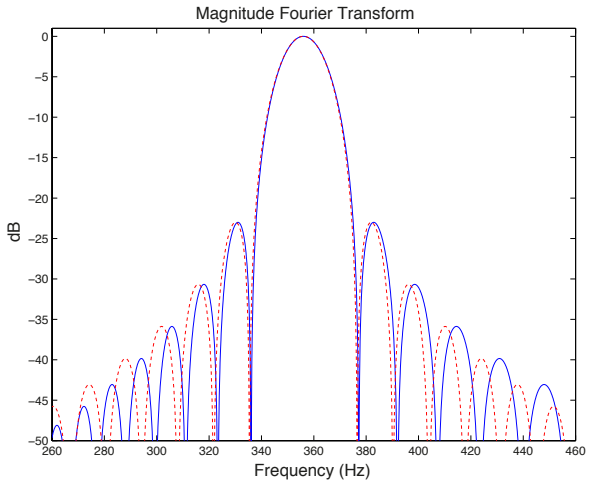


Figure 4: Magnitude Fourier transforms of the warped modulated window (dotted line) and of the approximating modulated scaled cosine window (solid line), calculated for a 1/3 octave time-frequency representation.

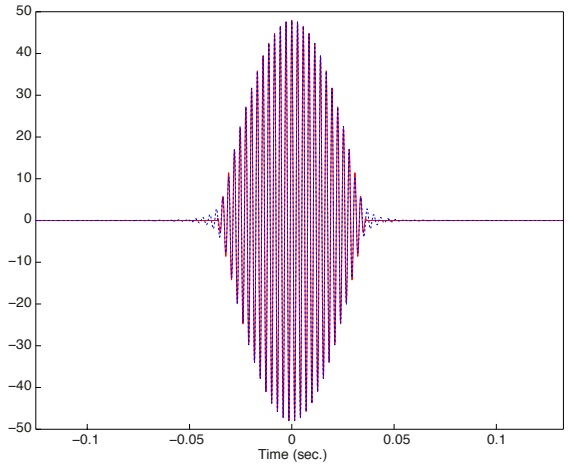


Figure 5: Warped modulated window (dotted line) and of the approximating modulated scaled cosine window (solid line), calculated for a 1/3 octave time-frequency representation.

5. CONCLUSIONS

In this paper, we have introduced approximation methods suitable for the online computation of the analysis and synthesis of time-frequency representations with arbitrary allocation of the frequency bands based on frequency warping. The problems arising from the dispersive sampling introduced by warping are solved by introducing a further warping operation in time-frequency.

The approximation of the frequency warped modulated windows consists in a local linearization of the warping map, which corresponds to time scaling and modulating a prototype window. The effect of dispersion is minimized within the essential bandwidths of the frame elements when these are selected, in order to fulfill causal computational needs, to have compact support in the time domain. A further refinement is obtained by directly truncating the modulated windows on larger intervals than the approximating cosine windows, obtaining higher accuracy in the reconstruction at the cost of larger storage, as the windows can be precomputed offline.

6. REFERENCES

[1] G. A. Velasco, N. Holighaus, M. Dörfler, and T. Grill, "Constructing an invertible constant-Q transform with nonstationary Gabor frames," in *Proceedings of the Digital Audio Effects Conference (DAFx-11)*, Paris, France, 2011, pp. 93–99.

[2] P. Balazs, M. Dörfler, F. Jaillet, N. Holighaus, and G. A. Velasco, "Theory, implementation and applications of nonstationary Gabor Frames," *Journal of Computational and Applied Mathematics*, vol. 236, no. 6, pp. 1481–1496, 2011.

[3] G. Evangelista, M. Dörfler, and E. Matusiak, "Phase vocoders with arbitrary frequency band selection," in *Proceedings of the 9th Sound and Music Computing Conference*, Copenhagen, Denmark, 2012, pp. 442–449.

[4] G. Evangelista, "Warped Frames: dispersive vs. non-dispersive sampling," in *Proceedings of the Sound and Music Computing Conference (SMC-SMAC-2013)*, Stockholm, Sweden, 2013, pp. 553–560.

[5] R. G. Baraniuk and D. L. Jones, "Unitary equivalence : A new twist on signal processing," *IEEE Transactions on Signal Processing*, vol. 43, no. 10, pp. 2269–2282, Oct. 1995.

[6] J. J. Clark, M. Palmer, and P. Lawrence, "A transformation method for the reconstruction of functions from nonuniformly spaced samples," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 5, pp. 1151–1165, 1985.

[7] C. Braccini and A. Oppenheim, "Unequal bandwidth spectral analysis using digital frequency warping," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, pp. 236–244, 1974.

[8] P. W. Broome, "Discrete orthonormal sequences," *Journal of the ACM*, vol. 12, no. 2, pp. 151–168, Apr. 1965.

[9] L. Knockaert, "On Orthonormal Muntz-Laguerre Filters," *IEEE Transactions on Signal Processing*, vol. 49, no. 4, pp. 790 –793, apr 2001.

[10] G. Evangelista, "Dyadic Warped Wavelets," *Advances in Imaging and Electron Physics*, vol. 117, pp. 73–171, Apr. 2001.

[11] G. Evangelista and S. Cavaliere, "Frequency Warped Filter Banks and Wavelet Transform: A Discrete-Time Approach Via Laguerre Expansions," *IEEE Transactions on Signal Processing*, vol. 46, no. 10, pp. 2638–2650, Oct. 1998.

[12] G. Evangelista and S. Cavaliere, "Discrete Frequency Warped Wavelets: Theory and Applications," *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 874–885, Apr. 1998, special issue on Theory and Applications of Filter Banks and Wavelets.

HYBRID REVERBERATION PROCESSOR WITH PERCEPTUAL CONTROL

Thibaut Carpentier, Markus Noisternig, Olivier Warusfel
UMR STMS IRCAM-CNRS-UPMC,
1, place Igor Stravinsky
75004 Paris, France
{tcarpent, noisternig, warusfel}@ircam.fr

ABSTRACT

This paper presents a hybrid reverberation processor, i.e. a real-time audio signal processing unit that combines a convolution reverb for recreating the early reflections of a measured impulse response (IR) with a feedback delay network (FDN) for synthesizing the reverberation tail. The FDN is automatically adjusted so as to match the energy decay profile of the measured IR. Particular attention is given to the transition between the convolution section and the FDN in order to avoid audible artifacts. The proposed reverberation processor offers both computational efficiency and flexible perceptual control over the reverberation effect.

1. INTRODUCTION

During the last decades, two main approaches for digital artificial reverberation processing have been widely used in music and film production [1]: convolution reverbs and delay-network techniques. In this paper, we present and discuss a novel hybrid reverberation processor that combines both methods and overcomes some of the limitations of earlier approaches. A hybrid reverberation effect processor should have the following properties:

- the hybrid reverberation effect should be perceptually indistinguishable from a pure convolution reverb for a given room impulse response;
- the algorithm should fulfill the constraints for real-time audio signal processing;
- the algorithm should be computationally efficient and thus attractive for practical applications;
- the processing method should provide a flexible high-level control over the perceived room effect.

This paper is organized as follows: The remainder of Section 1 briefly discusses the motivation for this study and offers a summary of earlier works and current state of the art methods for reverberation effect processing. Section 2 details the technical aspects of the proposed method. In Section 3, we present the results of a case study. In Section 4, we extend the hybrid reverberator with a perceptual control paradigm. Section 5 discusses some of the limitations of the proposed method and outlines possible future improvements.

1.1. Convolution-based reverberators

The acoustic transfer path between an emitter and a receiver in a room is usually modeled as a linear time-invariant system, which is fully characterized by its impulse response (IR). With this linear model, the room reverberation can be reproduced by convolving an anechoic input signal with the respective room impulse response. This convolution-based "auralization" approach guarantees for an authentic and natural listening experience.

Due to the increase in available processing power and recent advances in the development of computationally efficient low latency algorithms for frequency domain filtering (such as, e.g., the block-partitioned FFT convolution [2] and frequency delay lines [3]), convolution-based reverberation processing became widely applied during the last few decades. However, the computational cost of this method depends on the length of the processed IR. This may become a problem when recreating the reverberation of large concert halls and opera houses, where the length of the IR is typically in the order of a few seconds.

A survey on available convolution-based reverberation rendering software and hardware devices shows that the control over the reverberation effect is, in general, limited to only a few low-level parameters. Typically, the early-to-reverb ratio can be modified by adjusting the gains of the respective time sections of the IR. Often the decay time can be varied too, i.e., either increased or reduced. This can be achieved, for example, by resampling the original IR or by applying an exponentially decaying gain curve to the late reverberation tail. More advanced IR transformations often yield artifacts that result in an unnatural or unpleasant sounding reverb. This clearly limits the range of possible IR transformations in current convolution-based reverberators.

1.2. Parametric reverberators (FDNs)

Jot and Chaigne [4] used feedback delay network (FDN) processing structures for digital reverberation rendering. FDN simulate the statistical properties of the late room reverberation in a computationally efficient way. They are scalable and allow for a continuous tuning of the time and frequency behavior of the room response.

A commonly reported drawback of FDN rendering is the lack of authenticity in the early part of the room response. This is typically linked to transient coloration effects or from insufficient echo and/or modal densities, as it takes some time to build up dense reflection patterns with feedback loop structures.

1.3. Motivation for developing hybrid reverberators

This work aims at developing a hybrid reverberator that combines both convolution processing for the early part of the IR and FDN for late reverberation rendering. The hybridization approach shows several advantages over full convolution processing. Early reflections (ER) typically arrive within less than $50 - 200$ ms. Applying convolution filtering to this part of the IR comes with a low com-

putational cost, while it preserves the naturalness and spectral signature of the room response. The late reverberation decay, which may be several seconds for large rooms, can be accurately modeled with computationally efficient FDNs. The feedback loop structure offers flexible control over the rendering parameters and can be adapted to perceptually-motivated control methods (see Sec. 4 for further details). The two main challenges for the design of such hybrid processor are:

- the estimation of model parameters from the original (e.g., measured) IR for automatic tuning of the FDN;
- to guarantee smooth transitions between the two processing stages (i.e. at the transition between early reflections and reverberation tail) without perceptible artifacts.

1.4. Related works

The idea of combining FIR filter for early reflection modeling with a recursive topology for modeling late reverberation decays dates back to early works on digital artificial reverberation (see e.g. [5–7]), even though actual attempts at hybridization only appeared in the late 2000s.

Stewart [8, 9], for instance, proposed a hybrid reverberator using a 16-channel FDN for generating the late reverberation. This reverberator automatically estimates the FDN parameters from the energy decay relief (EDR). More precisely, the reverberation time (RT) is estimated in each frequency band. The initial spectrum of the FDN is, however, not taken into account. A Hann window assures smooth cross-fading between the concatenated sections (i.e. early reflections and late reverberation) and minimizes perceptible artifacts. Although Stewart et al.'s method is very similar to what is proposed in this paper (see Sec. 2), they only demonstrate that such a hybrid reverberator is viable. To the authors' knowledge, it has never been realized in practice. It should be further noted that this cross-fading approach is not well suited for real-time implementations. The rising edge of the Hann window is applied at the beginning of the late reverberation, which is not possible in real-time.

A similar approach is taken in [10], without providing detailed information on the crossfade in between the two sections. Here, the reverberation decay times are estimated in only two frequency bands and a 16-channel FDN is adjusted to match the original IR at the transition points. No additional spectral shaping is applied to the FDN.

Abel et al. [11] model a plate reverberator with a hybrid processing unit. This method (which is inspired from [8]) first estimates the spectral decay times and then applies them to the FDN. For equalizing the FDN a short FIR filter, which is obtained from a minimum-phase version of the impulse response, is applied to the transition region. The transition between the convolution section and the FDN is accomplished by means of a power-complementary crossfade.

Greenblatt et al. [12] further extended the methods presented in [8] and [11] by improving the window-based crossfade between the convolution and FDN sections. This method allows for any arbitrary window shape and length as it is subtracted from the convolution part.

In [13], Lee et al. generate the ER section using conventional convolution techniques and the late reverberation part with a so-called "switched convolution (SC)" technique. The SC processor consists of a recursive comb filter that is convolved with a short noise segment. The transition between the two processes uses the cross-fading technique developed in [11].

Other works mainly focus on the optimization of the different processes: Heise et al. [14] proposed an optimization strategy for matching the settings of two different audio processors. As a case study they tune an algorithmic reverb so that it mimics a convolution reverb processor. The optimization procedure evaluates the differences between the actual response and the target response on the basis of psychoacoustic features. As a principal measure the euclidian distance between MFCC vectors is applied.

A hybrid reverberation processor with a Moorer structure for the reverb tail is used by Primavera et al. [15–18]. It is based on an iterative optimization algorithm (see [19] for details) to determine the parameters of an IIR filter structure (i.e. delay line lengths, gains, and damping factors) that jointly minimize different cost functions. The cost functions are obtained by comparing the synthesized IR with the real IR in both the time and frequency domain.

Holm-Rasmussen et al. [20] apply linear predictive coding to fit a synthesized reverberation tail to a measured IR. For synthesis sparse FIR filters are used.

Several works investigate different time-frequency representations to estimate the model parameters that best approximate a given room impulse response (see e.g. [21–23] for further details). Most methods apply the short-time Fourier or wavelet transform; the parameters of the filter structures are estimated using the Prony or Steiglitz-McBride method.

2. PROPOSED METHOD

In this paper we propose a method that (automatically) tunes a FDN unit to best approximate the time-frequency response of the original IR. Schroeder (see e.g. [24, 25]) statistically modeled the late reverberation of a room as exponentially decaying Gaussian random process. It is shown in the following that, when applying Schroeder's statistical model, the FDN can be fitted with arbitrary accuracy to both the reverberation decay profile and the initial spectrum of the original IR.

2.1. IR truncation

A time-domain room impulse response can only be Gaussian when a sufficient number of reflections overlap, i.e. when the echo density in a room is sufficiently high enough. The stochastic model for late reverberation is thus only valid for frequencies higher than the Schroeder frequency [24] and times later than the mixing time (t_{mix}). The mixing time determines the transition between the ER and the reverberation tail and thus defines the cross-over point between the convolution section and the FDN section of a hybrid reverberator. Several estimators for the mixing time have been proposed in literature (see e.g. [26–32]), with varying results that, in general, strongly depend on the estimation parameters (e.g. the size of the analysis window). An objective comparison of the performance of these various estimators is beyond the scope of this paper and is postponed to future publications. (Note that Lindau et al. [33] presented a comparative study on the estimation of the perceptual mixing time). For the remainder of this article it is assumed that the mixing time is estimated with sufficient accuracy and that the IR can be modeled as decaying Gaussian random process for times $t \geq t_{mix}$.

2.2. Feedback delay network reverberator

The hybrid reverberation engine presented in this paper is based on IRCAM's parametric reverberation engine, which is part of the sound spatialization software "Spatialisateur" (Spat~). This FDN-based reverberator consists of a "lossless prototype" (i.e. a reverberator with infinite reverberation time that is based on lossless unitary feedback matrix structures) combined with absorptive IIR filters (see [34] for more details). With this processing structure one can achieve arbitrary time and modal densities, low tonal coloration, and independent control of the frequency envelope and decay characteristics [4]. The Spat~ reverberation processor can be controlled by a set of perceptual descriptors (see Sec. 4). These descriptors rely on a simplified model of the IR's time-frequency energy distribution that is reduced to four time segments and three frequency bands (cf. Fig. 1). The Spat~ model separates the IR into three sections: (a) "early" for the very first discrete echoes, (b) "cluster" for the late and more diffuse early reflections with a dense reflection pattern, and (c) "late reverb" for the late reverberation. The cluster is synthesized with multi-tap delay lines feeding a decorrelation unit; the late reverb is generated by a delay-network (that is fed by the output of the cluster section) with typically 4 to 16 feedback channels.

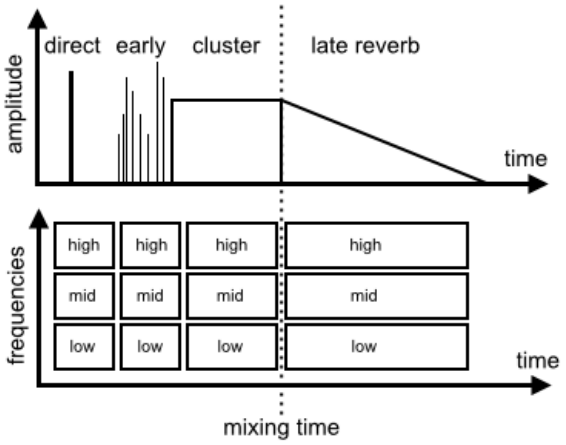


Figure 1: *Time-frequency IR model of the FDN-based reverberator Spat~: echogram (top) and time-frequency distributions (bottom).*

Hybridization requires to adjust the FDN model parameters (i.e. the reverberation profile and the initial frequency spectrum) to the time-frequency envelope of the original IR. This is achieved by analyzing the energy decay relief of the original IR.

2.3. Energy decay relief analysis

The energy decay relief (EDR) is the ensemble average of the time-frequency representation of the reverberation decay after the interruption of the excitation signal (see [35]). It represents the spectral energy density of the IR over time. The EDR is a generalization of Shroeder's energy decay curve (EDC), which allows for a time-frequency representation of the IR. It can be used to accurately estimate the model parameters of exponential reverberation decays. Given an impulse response $h(t)$, the EDR writes:

$$\mathrm{EDR}_h(t,f) = \left| \int_{\tau=t}^{\tau=\infty} h(\tau) e^{-j2\pi f\tau} d\tau \right|^2 . \qquad (1)$$

Eq. (1) can be efficiently computed, e.g., through backward integration of the short-time Fourier spectrum of the impulse response. Following the procedure of [35], the reverberation time $\mathrm{RT}(f)$ can be estimated for any frequency f. Measured impulse responses are usually corrupted by measurement noise, which distorts the computed EDR and results in biased estimates of the decay times. In practice, the analysis of the EDR is restricted to a frequency-dependent time interval in which the hypothesis of exponential energy decay holds.

The absorptive filter g_i in the i^{th} feedback channel of the FDN is then chosen such that the logarithm of its magnitude response is proportional to the delay length and inversely proportional to the reverberation time. With reference to [4] and by neglecting the absorptive filter's phase response, the filter equation writes as

$$20 \cdot \log_{10} \left| g_i(e^{j2\pi f}) \right| = \frac{-60}{\mathrm{RT}(f)} \cdot \tau_i, \qquad (2)$$

where τ_i is the delay length (in seconds) of the i^{th} inner channel. Spat~ implements the absorptive filter g_i as a three-band parametric filter with adjustable crossover frequencies. The estimated $\mathrm{RT}(f)$ is thus averaged and reduced to three frequency bands.

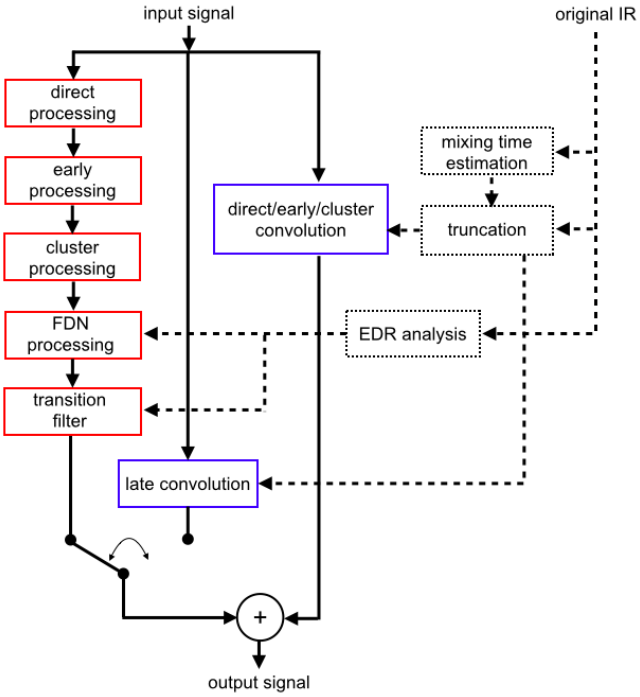


Figure 2: *Hybrid reverberator processing structure. Blocks with dashed-lines indicate offline processing. The "direct/early/cluster convolution" module performs the convolution of the IR truncated to the time interval* $[0 - t_{mix}]$.

2.4. Transition filter (spectral correction filter)

With reference to Jot et al. [4, 26], the EDR is not only characterized by the reverberation time $\mathrm{RT}(f)$ but also by the initial power spectrum $\mathrm{P}(f)$. In theory, the FDN's initial power spectrum is a zero-mean white Gaussian process that is independent of the decay

characteristics. This assumption does not always hold in practice, due to approximations in the derivation of the FDN's correction filter [4] and the FDN-channel lowpass filters that simulate the air absorption of the reflection paths [36]. Note that the air absorption filters are not compensated by the FDN's correction filter.

An additional spectral correction is thus introduced to match the initial spectrum of the FDN with the EDR of the original IR at the mixing time:

$$\text{correction}(f) = \sqrt{\frac{\text{EDR}_h(t_{mix}, f)}{\text{EDR}_{\text{FDN}}(t_{mix}, f)}}. \tag{3}$$

A linear-phase filter is derived from the magnitude response of the spectral correction in Eq. (3) and then applied to the FDN (see Fig. 2). This spectral correction filter guarantees for a smooth and continuous time-frequency envelope of the hybrid reverberation processor at time $t = t_{mix}$.

2.5. Statistical aspects

The modal density D_m (i.e. the average number of modes per Hz) of a FDN with N feedback channels is related to the total length of the delay units by the following equation:

$$D_m = \sum_{k=1}^{N} \tau_k. \tag{4}$$

A crucial requirement for convincing artificial reverberation is to satisfy the assumptions of Schroeder's statistical model, i.e. to operate above the "Schroeder frequency". This condition corresponds to a modal overlap of at least 3:1 (see [35] for details), which is equivalent to

$$D_m \geq \text{RT}_0, \tag{5}$$

where RT_0 denotes the average reverberation time. Eqs. (4) and (5) are used to adjust the delay times of the inner loops of the FDN structure.

2.6. Current real-time implementation

The proposed hybrid reverberator is implemented in C++ and available as an external object (spat.hybrid~) for Max/MSP® as part of the Spat~ package. The external object first loads the IR and then performs the above-mentioned EDR analysis. This initial processing step is performed offline. Once all IR parameters are determined the external object enables the real-time processing of the input audio stream. To ease perceptive comparisons, one can switch between convolution and FDN modeling for the late reverberation tail (cf. Fig. 2). The real-time convolution is implemented as a zero-latency partitioned FFT algorithm adapted from [2]. The N uncorrelated output channels of the FDN can be either summed up to produce a mono output signal, or distributed over several loudspeakers creating a convincing spatial diffuse field out of a mono IR.

3. RESULTS

This section presents the results of a case study applying the above-mentioned algorithms to the IR of a large factory hall. In order to preserve a certain objectivity the IR was taken from a commercial library. The file is about 9 s long and the average decay time is $\text{RT}_0 \approx 4.5$ s. The mixing time is taken as $t_{mix} \approx 200$ ms. The FDN consists of $N = 8$ feedback channels and the crossover frequencies are set to 2.5 and 7 kHz.

Fig. 3 (bottom figure) demonstrates that the EDCs of the original and the hybrid IR are in good agreement; the upper figure compares the frequency-dependent reverberation time before and after applying the hybridization process. For frequencies higher than 1.5 kHz, the decay profile is in good agreement with that of the original IR. However, for lower frequencies an error of approximately $\pm 10\%$ can be observed. This error results from the use of 2^{nd}-order absorptive filters in the FDN loop, which determines the overall shape of the $\text{RT}(f)$ curve. For the given example, the 2^{nd}-order shelving filters cannot approximate the original $\text{RT}(f)$ curve with sufficient accuracy in the low and very high frequency bands.

The choice of 2^{nd}-order filters is motivated by the results of earlier studies on the perceptual characterization of the acoustic quality of concert halls, opera houses, and auditoria. Kahle [37] showed that controlling the reverberation time in three frequency bands covers the full range of perceptual attributes for a large set of room acoustic qualities. First informal listenings tests using the shown case study (among other examples) indirectly confirm these results. Despite the biased $\text{RT}(f)$ depicted in Fig. 3, preliminary results indicate that listeners cannot distinguish sounds processed with the hybrid reverberator from those that have been convolved with the original IR. Anyway, more detailed listening experiments are needed to verify these early results.

The proposed hybrid reverberation processing model does not limit the number of absorption filter frequency bands. Higher-order parametric filters have been successfully implemented, but at the expense of a higher computational cost. The presented three-band filter model provides a good trade off between model accuracy and computational load.

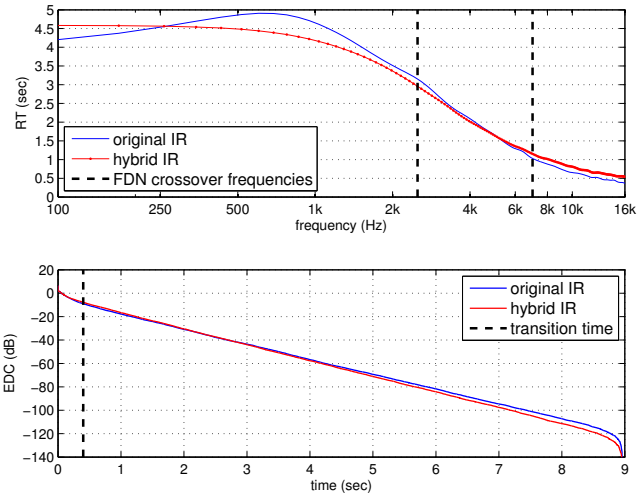


Figure 3: *Estimated reverberation time (top) and energy decay curve (bottom) for both the original and the hybrid IR.*

Fig. 4 (top figure) compares the EDR derived from the original IR with that from the FDN, both evaluated at the transition time, t_{mix}. The high frequency damping in the FDN spectrum (red dotted curve) results from the air absorption filters. Fig. 4 (bottom

figure) depicts the magnitude response of the spectral correction filter that provides a smooth transition in between the convolution and the FDN. We perform a critical band smoothing before the magnitude response is transformed into a 256-taps FIR filter for real-time implementation. The actual length of the spectral correction filter is a tradeoff between the modeling accuracy and the computational efficiency.

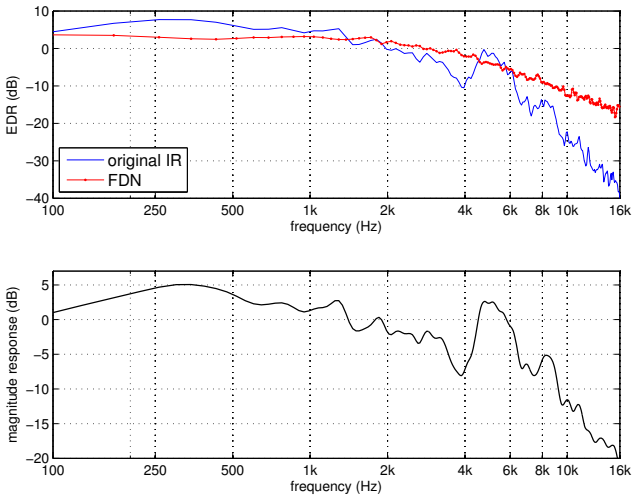


Figure 4: *EDR of the original IR and the FDN at the transition time (top) and the magnitude response of the transition filter (bottom).*

Fig. 5 illustrates the EDR of the original and the hybrid IR. Visual inspection confirms a good agreement between the EDRs. A closer look at the right figure shows that the reverberation time is slightly underestimated, i.e. the RT of the hybrid model is too short in a frequency range from 500 to 1500 Hz. The spectral correction filter shapes the FDN to match the spectrum of the original IR at the transition point. Without this correction filter, the hybrid reverberator would not reproduce the frequency boost around 5000 Hz. In this regard, the proposed processing method clearly outperforms standard FDN implementations.

Informal listening experiments confirm that the hybrid model generates perceptually indistinguishable results for various test signals (e.g. Dirac impulse, percussive sounds, male/female speech, music, etc.). As mentioned above, more detailed listening experiments are needed to verify these early results.

The computational advantage of the proposed hybrid reverberation processor over a pure convoler depends on many parameters, such as the length of the original IR (i.e. the reverberation time), the mixing time, the number of feedback channels in the FDN, the audio I/O latency of the processing environment, and so on. Therefore, it is difficult to draw general conclusions on the cpu load from comparisons with a pure convolution processor. With the different parameter settings that were tested in this case study, we gained about 35% of cpu load compared to an optimized real-time convolution algorithm.

4. PERCEPTUAL CONTROL

In the 1980s and 1990s, IRCAM has undertaken a series of room acoustic measurements and listening tests in different European concert halls. The aim was to establish a set of perceptual descriptors for the acoustic quality of concert halls (see e.g. [37–40]). Multidimensional data analysis (more specifically Individual Differences Scaling analysis, INDSCAL; see [41]), revealed a set of nine mutually independent perceptive descriptors for describing the room acoustic quality. It has been shown that these descriptors correlate well with some objective room acoustic criteria (see [38] for more details). In order to control the room effect along the relevant perceptual dimensions most of the proposed descriptors require both a temporal and spatial weighting; some of them do also require a spectral weighting in order to obtain satisfactory results. An in-depth discussion of the set of descriptors is beyond the scope of this paper. We only give one example to allow for a more general understanding of the perceptual control of the hybrid reverberator. For instance, the "DirE" descriptor refers to the energy of the "temporally extended" direct sound energy and controls the perceived presence of a sound source in a reverberant environment. It is computed from the temporally segmented impulse response as illustrated in Fig. 1. In the following, E_{R0} refers to the estimated energy of the direct sound ($0-20$ms), E_{R1} to the energy of the early reflections ($20-40$ms), E_{R2} to the energy of the cluster (i.e. the late reflections; $40-100$ms), and E_{R3} to the energy of the late reverberation tail (>100ms), respectively. DirE can be computed from these energy estimations as follows:

$$\text{DirE} = E_{R0} + E_{R1} + E_{R2,\text{excess}} + 0.18 \times E_{R2,\text{masked}} \qquad (6)$$

with

$$\begin{aligned} E_{R2,\text{excess}} &= \max(0, E_{R2} - E_{R40}), \\ E_{R2,\text{masked}} &= \min(E_{R2}, E_{R40}), \\ E_{R40} &= E_R|_{[0,40\text{ms}]} = E_{R0} + E_{R1}. \end{aligned}$$

Jullien and Kahle [38–40] have, for instance, shown that the DirE parameter represents well Lochner and Burger's "energy ratio criterion" [42] for the intelligibility of speech. As a result one can control the perceived presence of sound source by controlling the gain of the different time sections of the impulse response. For more details on the perceptive descriptors, please refer to [37–40].

IRCAM's parametric FDN-based reverberator applies the perceptive descriptors in a similar way, and they have been proven useful in many music productions. If we now apply them to the hybrid reverberation process, we have to modify the signal processing structure given in Fig. 2 so that it represents the time-segmented structure given in Fig. 1. The convolution segment (i.e. for the time interval from $t = 0$ to $t = t_{mix}$) is split into three sub-segments corresponding to the sections "direct", "early", and "cluster". The output signals of these subsections are first time aligned with delay lines and then filtered with three-band parametric shelving filters, which are controlled by the perceptual model parameters. Fig. 6 depicts the extended processing model (for simplicity, the data analysis modules are not shown in this figure). When the direct/early/cluster/late filters are flat, the hybrid reverberation unit represents the original IR. When the user manipulates the perceptual factors, the parameters of the filters are updated accordingly. This allows to smoothly modulate the acoustical quality of the IR by navigating along the different perceptual axes. For instance the so-called "source presence" factor controls "DirE" and creates a convincing effect of proximity or remoteness of the sound source by simultaneously adjusting the direct/early/cluster/late levels (E_{R0}, E_{R1}, E_{R2}, E_{R3}) according to the structured model.

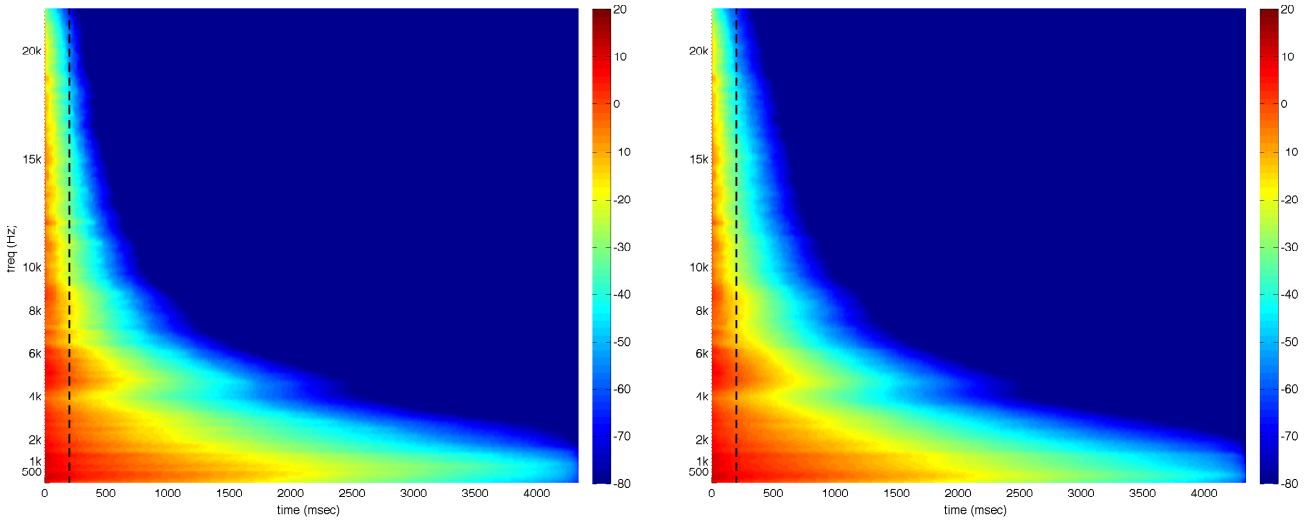


Figure 5: *EDR (in dB) of the original (left) and hybrid (right) IR. The dashed line represents the transition time, $t = t_{mix}$.*

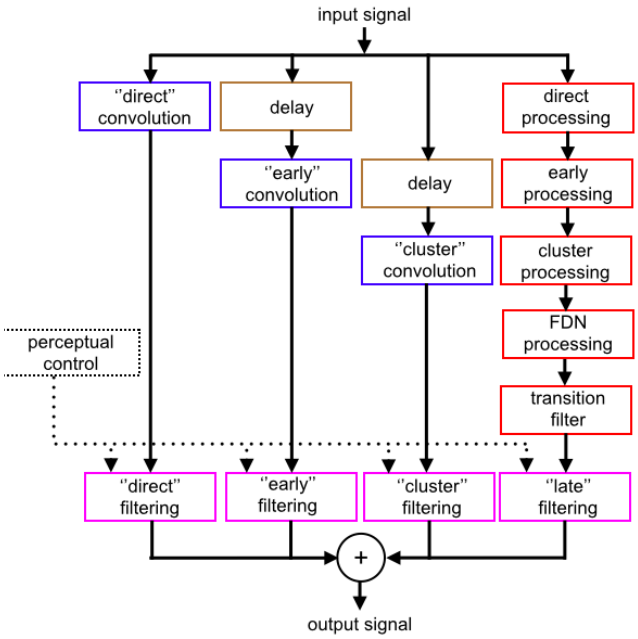


Figure 6: *Process chart of the hybrid reverberator with perceptual control. Blocks in blue correspond to convolution segments. Blocks in red correspond to parametric reverberation. Delay lines (brown blocks) ensure time-alignment of the convolution segments. Blocks in magenta correspond to three-band filters controlled by the perceptual model.*

5. CONCLUSIONS AND PERSPECTIVES

This paper considered both the theory and implementation of a hybrid reverberator that combines convolution processing for early reflections with feedback delay networks for late reverberation rendering. The proposed method first estimates the reverberation time and exponentially decaying envelope in different frequency bands from the original impulse response. These parameters are then used to control the FDN processing. Particular attention is paid to the smooth transition from convolution rendering to FDN processing; the power spectrum is matched at the transition point (given by the mixing time) in each frequency band.

An analysis of different room impulse responses (see also Sections 2 and 3) indicated that three-band shelving filters in each FDN channel may not always succeed to model the late reverberation with sufficient accuracy. The model accuracy strongly depends on the EDR profile of the original IR. Future work will focus on the analysis of the RT(f) curve in order to automatically determine the minimum number of required frequency bands (and corresponding crossover frequencies) to keep the modeling error below a given threshold. Increasing the number of FDN filter bands significantly increases the computational cost. However, with the rapid increase in available processing power real-time implementations may become feasible.

The modal density of a FDN should satisfy Schroeder's suggestions for natural sounding and high quality artificial reverberators (cf. Section 2). A useful extension of the hybrid processor would be estimating the modal density of the original IR to automatically adjust the FDN to these parameters.

The proposed method is based on the stochastic model of late reverberation and thus excludes, e.g., non-exponential decays, flutter echoes, and spring reverbs. Nonetheless it would be possible to extend the technique to IRs exhibiting a double-slope exponential decay; such decay profiles have gained interest in recent years and have been observed in concert halls such as, e.g., the Boston Symphony Hall. Both the EDR analysis and the FDN rendering can be extended to that purpose. Adapting the EDR analysis of [35] to multiple-slope exponential decays do not raise conceptual difficulties. The design of FDNs with multiple decay slopes is currently under investigation.

In this paper we focused on single channel impulse responses for a mono input signal and mono (or multichannel) output sig-

nal(s). A multichannel extension to directional room impulse responses (DRIRs) is currently under development. DRIRs are typically measured with spherical microphone arrays. Preliminary results of multichannel EDR analysis and DRIR denoising have been published in [43, 44]. The proposed methods perform a joint analysis of the EDR of all the microphone cells in order to preserve the spatial coherence between them. Hybrid convolution reverberators operating in the modal domain are used for higher-order Ambisonics rendering.

6. ACKNOWLEDGMENTS

This study was supported in part by the French ANR CONTINT project "Sample Orchestrator 2" (SOR2).

7. REFERENCES

[1] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421 – 1448, July 2012.

[2] W. G. Gardner, "Efficient convolution without input-output delay," in *Proc. of the 97th AES Convention*, San Francisco, CA, Nov. 10-13, 1994.

[3] G. García, "Optimal filter partition for efficient convolution with short input/output delay," in *Proc. of the 113rd AES Convention*, Los Angeles, CA, Oct. 5-8, 2002.

[4] J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Proc. of the 90th AES Convention*, Paris, France, Feb. 19-22, 1991.

[5] M. R. Schroeder, "Digital simulation of sound transmission in reverberant spaces," *J. Acoust. Soc. Amer.*, vol. 45, no. 1, pp. 424 – 431, 1969.

[6] J. A. Moorer, "About this reverberation business," *Computer Music Journal*, vol. 3, no. 2, pp. 13 – 28, 1979.

[7] F. R. Moore, *Elements of Computer Music*, Prentice Hall, Englewood Cliffs, NJ, USA, 1990.

[8] R. Stewart and D. Murphy, "A hybrid artificial reverberation algorithm," in *Proc. 122nd AES Convention*, Vienna, Austria, May 5-8, 2007.

[9] R. Stewart, "Hybrid convolution and filterbank artificial reverberation algorithm using statistical analysis and synthesis," M.S. thesis, University of York, 2006.

[10] R. Stewart and M. Sandler, "Real-time panning convolution reverberation," in *Proc. of the 123rd AES Convention*, New York, NY, Oct. 5-8, 2007.

[11] J. S. Abel, D. P. Berners, and A. Greenblatt, "An emulation of the emt140 plate reverberator using a hybrid reverberator structure," in *Proc. of the 127th AES Convention*, New York, NY, Oct. 9-12, 2009.

[12] A. B. Greenblatt, J. S. Abel, and D. P. Berners, "A hybrid reverberation crossfading technique," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Dallas, TX, Mar. 14-19, 2010, pp. 429 – 432.

[13] K. Sup Lee, N. J. Bryan, and J. S. Abel, "Approximating measured reverberation using a hybrid fixed/switched convolution structure," in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10, 2010.

[14] S. Heise, M. Hlatky, and J. Loviscach, "Automatic adjustment of off-the-shelf reverberation effects," in *Proc. of the 126th AES Convention*, Munich, May 7-10, 2009.

[15] A. Primavera, L. Palestini, S. Cecchi, F. Piazza, and M. Moschetti, "A hybrid approach for real-time room acoustic response simulation," in *Presented at the 128th AES Convention*, London, May 22-25, 2010.

[16] A. Primavera, S. Cecchi, L. Romoli, P. Peretti, and F. Piazza, "An advanced implementation of a digital artificial reveberator," in *Proc. of the 130th AES Convention*, London, May 13-16, 2011.

[17] A. Primavera, S. Cecchi, L. Romoli, P. Peretti, and F. Piazza, "Approximation of real impulse response using IIR structures," in *Proc. of the 19th European Signal Processing Conference*, Aug. 2011.

[18] A. Primavera, M. Gasparini, S. Cecchi, L. Romoli, and F. Piazza, "Hybrid Reverberation Algorithm: a Practical Approach," in *Proc. of the AIA-DAGA Conference*, Merano, Italy, Mar. 18-21, 2013.

[19] G. Constantini and A. Uncini, "Real-time room acoustic response simulation by an iir adaptive filter," *Electronics Letters*, vol. 39, pp. 330 – 332, 2003.

[20] B. Holm-Rasmussen, H.-M. Lehtonen, and V. Välimäki, "A new reverberator based on variable sparsity convolution," in *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*, Maynooth, 2013.

[21] J. C. Sarris and G. E. Cambourakis, "Time frequency analysis and parametric approximation of room impulse responses," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, 2003, vol. 6, pp. 297 – 300.

[22] M. Schonle, N. Fliege, and U. Zolzer, "Parametric approximation of room impulse responses based on wavelet decomposition," in *Applications of Signal Processing to Audio and Acoustics, 1993. Final Program and Paper Summaries., 1993 IEEE Workshop on*, 1993, pp. 68 – 71.

[23] M. Schoenle, N. Fliege, and U. Zoelzer, "Parametric approximation of room impulse responses by multirate systems," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, 1993, vol. 1, pp. 153 – 156.

[24] M. R. Schroeder, "Eigenfrequenzstatistik und Anregungsstatistik in räumen," *Acustica*, , no. 4, pp. 456 – 468, 1954, English translation: M. R. Schroeder, "Statistical Parameters of the Frequency Response Curves of Large Rooms", J. Audio Eng. Society, vol. 35, no. 5, pp. 299 - 306, 1987.

[25] M. R. Schroeder and H. Kuttruff, "On frequency response curves in rooms. comparison of experimental, theoretical and monte-carlo results for the average frequency spacing between maxima," *J. Acoust. Soc. Am.*, , no. 34, pp. 76 – 80, 1962.

[26] J.-D. Polack, "Playing billiards in the concert hall: The mathematical foundations of geometrical room acoustics," *Applied Acoustics*, vol. 38, no. 2–4, pp. 235–244, 1993.

[27] J. S. Abel and P. Huang, "A simple, robust measure of reverberation echo density," in *Proc. of the 121st AES Convention*, San Francisco, CA, Oct. 5-8, 2006.

[28] T. Hidaka, Y. Yamada, and T. Nakagawa, "A new definition of boundary point between early reflections and late reverberation in room impulse responses," *J. Acoust. Soc. Am*, vol. 122, no. 1, pp. 326 – 332, July 2007.

[29] P. Huang and J. S. Abel, "Aspects of reverberation echo density," in *Proc. of the 123rd AES Convention*, New York, NY, Oct. 5-8, 2007.

[30] R. Stewart and M. Sandler, "Statistical measures of early reflections of room impulse responses," in *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10-15, 2007.

[31] G. Defrance, L. Daudet, and J.-D. Polack, "Using matching pursuit for estimating mixing time within room impulse responses," *Acta Acustica united with Acustica*, vol. 95, no. 6, 2009.

[32] G. Defrance and J.-D. Polack, "Estimating the mixing time of concert halls using the eXtensible Fourier Transform," *Applied Acoustics*, vol. 71, no. 9, pp. 777–792, 2010.

[33] A. Lindau, L. Kosanke, and S. Weinzierl, "Perceptual evaluation of model- and signal-based predictors of the mixing time in binaural room impulse responses," *J. Audio Eng. Soc.*, vol. 60, no. 11, pp. 878 – 898, 2012.

[34] J.-M. Jot, "Real-time spatial processing of sounds for music, multimedia and interactive human-computer interfaces," *ACM Multimedia Systems Journal (Special issue on Audio and Multimedia)*, vol. 7, no. 1, pp. 55 – 69, 1997.

[35] J.-M. Jot, L. Cerveau, and O. Warusfel, "Analysis and synthesis of room reverberation based on a statistical time-frequency model," in *Proc. 103rd AES Convention*, New York, NY, Sept. 26-29, 1997.

[36] J. Huopaniemi, L. Savioja, and M. Karjalainen, "Modeling of reflections and air absorption in acoustical spaces – a digital filter design approach," in *IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct 1997.

[37] E. Kahle, *Validation d'un modèle objectif de la perception de la qualité acoustique dans un ensemble de salles de concerts et d'opéras*, Ph.D. thesis, Laboratoire d'Acoustique de l'Université du Maine, Le Mans, 1995.

[38] J.-P. Jullien, "Structured model for the representation and the control of room acoustic quality," in *15th International Congress on Acoustics (ICA)*, Trondheim, Norway, 1995, pp. 517–520.

[39] E. Kahle and J.-P. Jullien, "Some new considerations on the subjective impression of reverberance and its correlation with objective criteria," in *Acoustical Society of America, 127th annual Meeting (Sabine Centennial Symposium)*, Cambridge, MA, USA, June 1994, pp. 239–242.

[40] E. Kahle and J.-P. Jullien, "Subjective listening tests in concert halls: Methodology and results," in *Proc. of the Int. Congr. on Acoustics*, Trondheim, Norway, June 1995, pp. 521 – 524.

[41] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of 'eckart-young' decomposition.," *Psychometrika*, vol. 35, pp. 283 – 319, 1970.

[42] J. P. A. Lochner and J. F. Burger, "The subjective masking of short time delayed echoes by the primary sounds and their contribution to the intelligibility of speech.," *Acustica*, vol. 8, pp. 1 – 10, 1958.

[43] T. Carpentier, T. Szpruch, M. Noisternig, and O. Warusfel, "Parametric control of convolution-based room simulators," in *Proc. of the Int. Symposium on Room Acoustics*, Toronto, Canada, June 9-11, 2013.

[44] M. Noisternig, T. Carpentier, T. Szpruch, and O. Warusfel, "Denoising of directional room impulse responses measured with spherical microphone arrays," in *Proc. of the 38th Annual Convention of Acoustics (DAGA)*, Oldenburg, Germany, March 10-13, 2014.

EXAMINING THE OSCILLATOR WAVEFORM ANIMATION EFFECT

Joseph Timoney [1] and Victor Lazzarini [1],
[1]Sound and Music Technology research group,
NUI Maynooth
Maynooth, Ireland
joseph.timoney@nuim.ie
victor.lazzarini@nuim.ie

Jari Kleimola [2], and Vesa Välimäki [3],
[2]Dept. of Media Technology
[3]Dept. of Signal Processing and Acoustics
Aalto University, Espoo, Finland
jari.kleimola@aalto.fi
vesa.valimaki@aalto.fi

ABSTRACT

An enhancing effect that can be applied to analogue oscillators in subtractive synthesizers is termed Animation, which is an efficient way to create a sound of many closely detuned oscillators playing in unison. This is often referred to as a supersaw oscillator. This paper first explains the operating principle of this effect using a combination of additive and frequency modulation synthesis. The Fourier series will be derived and results will be presented to demonstrate its accuracy. This will then provide new insights into how other more general waveform animation processors can be designed.

1. INTRODUCTION

The modelling of analogue musical equipment using digital techniques has been an area of research that has received considerable attention over the past decade, and is still a very current topic [1]. This field covers the reproduction of Tube amplifiers ([2] and [3]), guitar effects devices ([4] and [5]), spring reverb units [6], analog synthesizer oscillators, both generally in [7] and [8], and in a model specific manner in [9] and [10], and resonant voltage controlled filters ([11], [12], and [13]).

With regard to analog synthesizer oscillators in particular, most of the previous work has focused on the alias-free synthesis of ideal classical waveforms, such as the sawtooth, the triangle, and the rectangular waveforms, see [7], or [14], for example. The reason for this focus on oscillators was simply that digital models of waveforms associated with particular analog synthesizers are more difficult to create because it requires access to such synthesizers in order to make waveform measurements. These can be expensive and difficult to obtain in their vintage versions. The ideal forms of the classic waveform signals have a spectrum that decays about 6 or 12 dB per octave, following the $1/f$ or the $1/f^2$ law (where f denotes frequency), respectively [19].

An early approach was the filtering of the digital impulse train obtained from the summation formula for the cosine series [20]. More recent works have proposed to implement an approximately bandlimited impulse train using a windowed sinc table ([21] and [22]), a feedback delay loop including an allpass filter [24], or a sequence of impulse responses of fractional delay filters [25]. Alternative approaches include the differentiated polynomial waveforms ([26],[27] and [28]), hyperbolic waveshaping [8], Modified FM synthesis[29], polynomial interpolation [30], polynomial transition regions [15] and [18], bandlimited impulse train generation using analog filters [16], and nonlinear phase basis functions [17].

Alongside these oscillator algorithms, other work has focused on enhancing effects that can be applied to them such as Hard Synchronisation ([31] and [32]) and Frequency Modulation ([33] and [34]).

One very interesting effect is described in the literature as Waveform Animation [35]. Animation is a single oscillator effect. It is an enhancement to the traditional non-modular analogue subtractive synthesizers feature of two or three oscillators per voice ([36] and [37]), which has generally held up for digital emulations [38]. The result of the Animation is the production of a deep, thick, pulsing sound. Originally proposed as a technique for modular analog systems it did not appear on synthesizers produced by the major manufacturers who opted for simply adding a unison oscillator option instead ([39] and [40]). More recently this unison oscillator arrangement has become termed as a *Supersaw* [38] or a *Hypersaw* [41]. It became strongly associated with electronic dance music.

Nam *et al*. [25] proposed an implementation of this effect in which several detuned bandlimited impulse trains (BLITs) with appropriate DC offsets are added together and fed through a single leaky integrator. However, this incurs the computational costs of generating multiple waveforms at a small frequency difference from each other. A digital implementation of Waveform animation, however, offers a more efficient alternative for creating this multiple oscillator sound effect than just adding numerous detuned waveforms because it does not result in a corresponding loss in polyphony as groups of oscillators are assigned to each voice.

When it comes to the digital emulation of a particular analog effect there are two choices: either (1) attempt to reproduce a particular analog circuit design directly or (2) to emulate the operation from an algorithmic perspective with tailored digital elements. While the first approach can work very well, it produces an algorithm that is computationally intensive and requires a significant oversampling factor to operate correctly, see [5], [11], [12], and [13]. The second approach is less complex, computationally cheaper, and more flexible, conferring the final implementation with benefits such as having greater polyphony available to the virtual synthesizer. An example of this approach has been presented in [10].

Therefore, in this paper, we will work out the underlying theory of the Waveform Animator oscillator effect from a signals point of view. This will be augmented by a model by which it can be implemented efficiently in modern digital synthesis systems using delay lines. The next section will mention the origins of the effect combined with the theory underlying it.

2. MULTIPLE DETUNED OSCILLATORS

The idea for this sound can be attributed to Risset [42] who de-

veloped it in the late 1960s for some of his compositions. In computer music circles it is sometimes termed the 'Risset Arpeggio' [43]. The intense effect of the detuned sound is due to a complicated beating pattern created among the harmonics of each oscillator. An analytical expression is available that describes this pattern [44]. Assuming a signal with a number of harmonics that has M detuned copies at a spacing of δf_0 between each of them, the beating pattern amplitude of the k^{th} complex harmonic cluster is given by

$$B_k(t) = A_k \frac{\sin(\pi M k \delta f_0 t)}{\sin(\pi k \delta f_0 t)} \qquad (1)$$

where A_k is the amplitude of the k^{th} harmonic.

3. WAVEFORM ANIMATOR

Hutchins proposed the multiphase waveform animator capable of emulating a bank of detuned sawtooth oscillators with a single Voltage Control Oscillator (VCO), by mixing a number of algebraically phase shifted sawtooth waveforms together [35]. The original paper did not show mathematically how this is achieved; rather it was demonstrated in terms of the waveforms it required as it was intended for implementation using a modular analog synthesis system. However, to gain a deeper insight that will assist our digital implementations it is worthwhile to understand the principle of this system fully.

The input to the Animator is a sawtooth with a rising edge of amplitude A. The animator itself consists of a number of channels each controlled by a different triangle wave Low Frequency Oscillator (LFO), whose rate should be less than 2Hz and whose amplitude is smaller than that of the input [35]. A block diagram of one channel that illustrates the principle of the animator is given in Fig. 1. Note that more channels leads to a more intense effect.

In Fig. 1, the input sawtooth and LFO are on the left hand side, there are two subtracting elements, a comparator, and the output appears on the right hand side. This output is a time-varying phase-shifted sawtooth that is then added with the input sawtooth to create the animated effect.

To explain in more detail: subtracting the input from the LFO generates an intermediate waveform. The LFO is very slow in relation to the input so that it is effectively like adding a DC offset to each period of the input wave. Fig. 2 shows this graphically using the relevant waveforms. In Fig. 2 the amplitude of the input sawtooth A = 5.0 and the amplitude of the LFO is 2.0. The result of the operation is that the DC level of the input is altered by a value of 7.0 in this example.

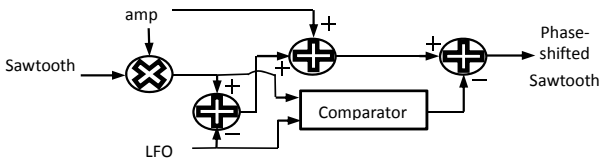


Figure 1. *Block diagram of one channel of the waveform animator.*

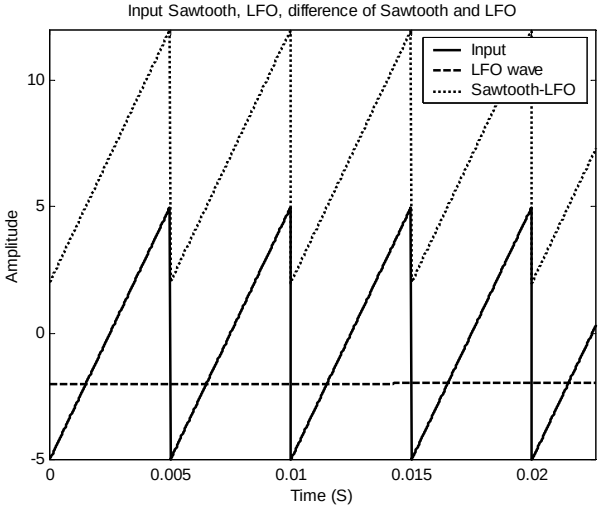


Figure 2. *Input sawtooth (solid line), LFO waveform (dashed line) and difference of the two (dotted line).*

This waveform is fed to a comparator device that is set to emit a pulse when its input is greater than the sawtooth amplitude A, otherwise the output is zero. This results in a PWM waveform whose pulse is on the leading edge and whose pulse width is varying at the rate of the LFO. Further, the amplitude of the pulse is 2A. This PWM wave is then subtracted from the DC-altered sawtooth to produce a time-varying phase-shifted version of the input sawtooth. This is illustrated in Fig. 3. The upper panel shows the generated PWM wave against the comparator input and the lower panel shows the original input sawtooth and its phase shifted version.

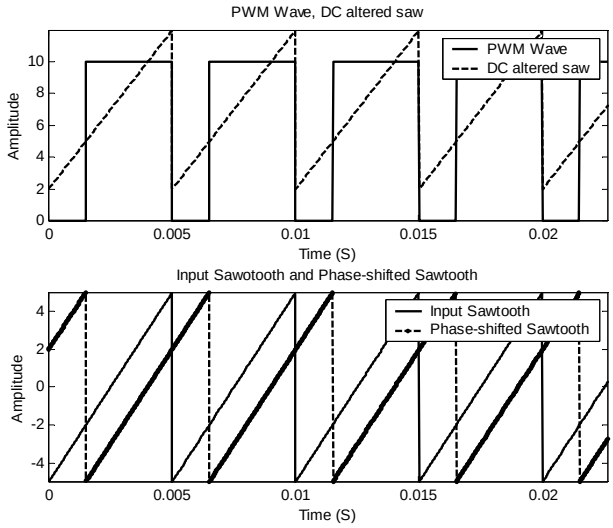


Figure 3. *The DC-altered sawtooth (dashed line) and PWM wave (solid line) at the output of the comparator are given in the upper panel. The input sawtooth (solid line) and the resulting phase-shifted sawtooth (dashed line) are shown in the lower panel.*

To illustrate mathematically what the animator is doing, first assume that we are looking only over a few periods where the

LFO waveform can be regarded as a constant DC level, we can then write the animator output as

$$S_{wa}(t) = -\frac{2A}{\pi}\sum_{k=1}^{\infty}\frac{\sin(2\pi k f_0 t)}{k} + C_{\mathrm{dc}} - P_{zs}(t) \tag{2}$$

where the first term on the rhs of (2) denotes a rising sawtooth, C_{dc} represents the added DC level, and the third term represents the PWM waveform comparator output whose maximum amplitude is 2*A* and minimum value is zero.

The expression for a falling edge, zero-centered, PWM wave of time-varying duty cycle $d(t)$ is [45]

$$P(t) = d(t) + \sum_{k=1}^{\infty}\frac{\sin(2\pi k d(t) - k\,2\pi f_0 t)}{(k\pi)} + \sum_{k=1}^{\infty}\frac{\sin(k\,2\pi f_0 t)}{(k\pi)} \tag{3}$$

Each component of the second term on the rhs of (3) is phase shifted where the phase shift depends both on the duty cycle and increases with increasing frequency because of the factor *k*. To rewrite (3) so that it represents the comparator output correctly it needs to have a leading edge pulse and be scaled in amplitude

$$P_{zs}(t) = -(2A)P(t) + 2A \tag{4}$$

Substituting (3) into (4), and then the result into (2) we can write the animator output as a combination of AC and DC components.

$$S_{wa}(t) = S_{\mathrm{AC}}(t) + S_{\mathrm{DC}}(t) \tag{5}$$

Remembering from (2) that the comparator combines the PWM wave along with the input sawtooth if we concentrate on the AC components of (5) first we have

$$\begin{aligned} S_{AC}(t) = 2A\left(\sum_{k=1}^{\infty}\frac{\sin(2\pi k d(t) - k2\pi f_0 t)}{(k\pi)} + \sum_{k=1}^{\infty}\frac{\sin(k2\pi f_0 t)}{(k\pi)}\right) \\ - 2A\sum_{k=1}^{\infty}\frac{\sin(k2\pi f_0 t)}{k\pi} \end{aligned} \tag{6}$$

which can be written as

$$S_{AC}(t) = -2A\sum_{k=1}^{\infty}\frac{\sin(k2\pi f_0 t - 2\pi k d(t))}{(k\pi)} \tag{7}$$

This gives the equation for a rising sawtooth with a time-varying phase shift. Then, looking at the DC term of (5)

$$S_{DC}(t) = \mathrm{DC} - 2A + 2Ad(t) \tag{8}$$

The term $d(t)$ will be constant within each time period. Thus, for a single time period we can write

$$S_{\mathrm{DC}} \approx \mathrm{DC} - 2A + 2A\hat{d} \tag{9}$$

To show that (9) is zero, we must determine $\hat{d}$ by locating the point of intersection of the LFO waveform with the sawtooth waveform in each period. If we write these as line equations we can use simple geometry to determine the intersection point between the two. For argument's sake, we assume that we are examining the crossing point within the first period of the sawtooth wave. Doing this, the time of their intersection t_p can be expressed as

$$t_p = \frac{A - A_{LFO}}{(2Af_0 - 2A'_{LFO}f_{LFO})} \tag{10}$$

where f_{LFO} is the LFO frequency, $A_{LFO}(t)$ is the time-varying amplitude of the LFO wave and A'_{LFO} is the maximum amplitude it will reach within that one period. The value of the duty cycle for that period will be

$$\hat{d} = \frac{(Af_0 - A_{LFO}f_0)}{(2Af_0 - 2A'_{LFO}f_{LFO})} \tag{11}$$

To further simplify the analysis we assume that within this first period of the sawtooth the amplitude of the triangle wave is constant, i.e.

$$A'_{LFO} = A_{LFO} \tag{12}$$

Then, examining Fig. 2, we can write

$$\mathrm{DC} \approx A + A_{LFO} \tag{13}$$

Substituting (12) into (11) and then combine with (13) in (9) to give

$$S_{\mathrm{DC}} = A + A'_{LFO} - 2A + 2A\frac{(Af_0 - A'_{LFO}f_0)}{(2Af_0 - 2A'_{LFO}f_{LFO})} \tag{14}$$

Next, noting that

$$(2Af_0 >> 2A'_{LFO}f_{LFO}) \tag{15}$$

The third term on the rhs of (14) can then be approximated, and following simple manipulation leads to the expected result

$$S_{\mathrm{DC}} = A + A'_{LFO} - 2A + A - A'_{LFO} = 0 \tag{16}$$

The expression (16) will hold for every period of the input sawtooth. Therefore, the final animator output can be written

$$S_{wa}(t) = -2A\sum_{k=1}^{\infty}\frac{\sin(k2\pi f_0 t - 2\pi k d(t))}{(k\pi)} \tag{17}$$

Examining (17) it can be interpreted as a summation of harmonically related frequency modulated sinusoids where the modulation is the time-varying duty cycle $d(t)$ and the modulation index increases with respect to the harmonic number. This result is very interesting as it means that the bandwidth around each harmonic increases with respect to increasing frequency. This would suggest why the waveform is perceived as being 'animated' as this increasing bandwidth with respect to frequency is similar in effect to adding detuned harmonic waveforms together. Furthermore, the faster the LFO the wider the bandwidth will become. There also should be a relationship between the swing in

the duty cycle with the sideband harmonic magnitudes and ultimately the strength of the effect.

4. ANIMATOR SPECTRAL PROPERTIES

It is worthwhile to investigate the spectral properties of the animated waveform a little further. The time-varying duty cycle signal only changes its value for every new period of the input sawtooth. This means that this modulating duty wave resembles a flat-top multi-level Pulse Amplitude Modulation signal, where the pulse rate is the same as the input sawtooth. However, because it is changing so slowly to simplify the analysis first we can assume that the duty cycle modulation is a shifted and scaled triangle LFO of the form

$$d(t) = \left(\frac{d_{\max} + d_{\min}}{2}\right) - \left(\frac{d_{\max} - d_{\min}}{2}\right) tri(2\pi f_{LFO} t) \qquad (18)$$

where $d_{\max}$ and $d_{\min}$ are the maximum and minimum values of the duty cycle respectively. They are determined by the user choice for A_{LFO}.

We can also write the Fourier series for the triangle wave modulation in (18) as

$$tri(2\pi f_{LFO} t) = \sum_{k_1} \frac{8}{(q\pi)^2} \cos(2\pi q f_{LFO} - \pi),\ q = 1,3,5,\ldots \qquad (19)$$

where q is the harmonic index.

This particular triangle wave will start from its minimum value which is in keeping with the original work [35]. Combining (19) with (18) and then substituting into (17) we can see that we will have a Complex FM waveform [46]. The relative contribution of each harmonic of the LFO to the spectrum of (17) could be computed using this theory. However, it can quickly become complicated if we use many components from the Fourier series in (19). By writing expressions for the modulation indices it is possible to find a way of simplifying the task. Denoting the magnitudes of the modulation indices for each as I_q we can consider the first two significant components (The second harmonic magnitude I_2=0 because it is a triangle wave) we have

$$I_1 = \frac{8}{\pi}(d_{\max} - d_{\min}) \qquad (20)$$

and

$$I_3 = \frac{8}{9\pi}(d_{\max} - d_{\min}) \qquad (21)$$

Noting that $I_1 > 1$ while $I_3 << 1$ (and will be true for all higher modulation indices), this suggests that it would be reasonable to assume that the primary contribution to the modulation of each harmonic in (17) is only the first component with modulation index given by (20) which allows us to rewrite

$$S_{wa}(t) = -2A \sum_{k=1}^{\infty} \frac{\sin(k2\pi f_0 t + kI_1 \cos(2\pi f_{LFO} t) + \vartheta)}{(k\pi)} \qquad (22)$$

where we denote the phase shift $\vartheta = -k\pi(d_{\max} + d_{\min})$.

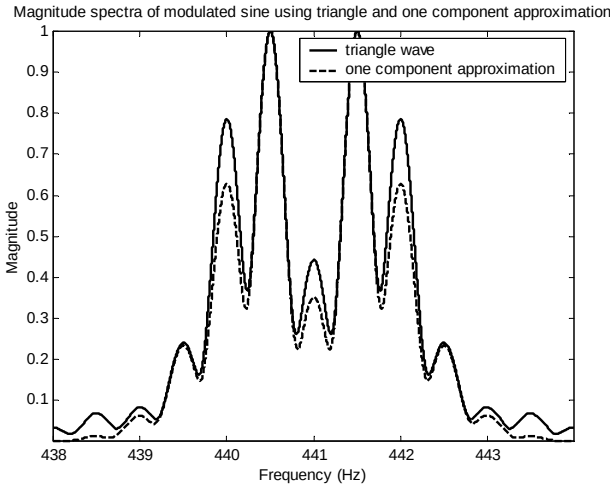


Figure 4. *Comparison of the spectra of a sinewave that has been frequency modulated using the triangle wave LFO of (19) (solid line) versus a one component approximation to it as given by (22) (dashed line).*

To validate the approximation Fig. 4 shows a plot of the spectra of a frequency modulated single sinewave of frequency 441Hz using the modulation function of (18) with frequency 0.5Hz and $d_{\max}$=0.9 and $d_{\min}$=0.1 (solid line) along with the spectrum of the same sinewave but with a sinewave modulation function with modulation index given by (20). From the figure it can be seen that there is an exact match for the magnitude of the first and third order sidebands at 441.5Hz, 440.5Hz, 442.5Hz and 439.5Hz respectively. There is a close match with the second order sidebands at 442Hz and 440Hz. This indicates that the approximation is acceptable.

Once adopting this approximation it is straightforward to have an expression for the magnitude spectrum of (22) around each harmonic

$$|H(kf_0 + nf_{LFO})| = (J_o(kI_1)) \qquad (23)$$

where J_o denotes a Bessel function of order o [47].

With this done it is possible to plot a relationship between the width of the duty cycle (that is, the difference between the maximum and minimum values) versus the magnitude of the first sideband of the modulation. This is helpful when creating an animated waveform as it can be used to decide how to set the amplitude of the triangle wave LFO and to determine the amplitudes of other sawtooths that could be added to the animated wave to get a desired balance between the animated waves and the original. This is similar to the mix function associated with commercial products [38], [41]. An experiment can be run by creating different values for the first modulation index in (20) using different values of duty cycle width. These can then be substituted into (23) to compute the first sideband magnitude (where o = 1). The result of this is given in Fig. 5. It shows that the largest sideband magnitude occurs when the width of the duty cycle is 0.7. This corresponds to a duty cycle maximum of 0.85 and a duty cycle minimum of 0.15. Thus, at this setting most significant level of waveform animation is achieved.

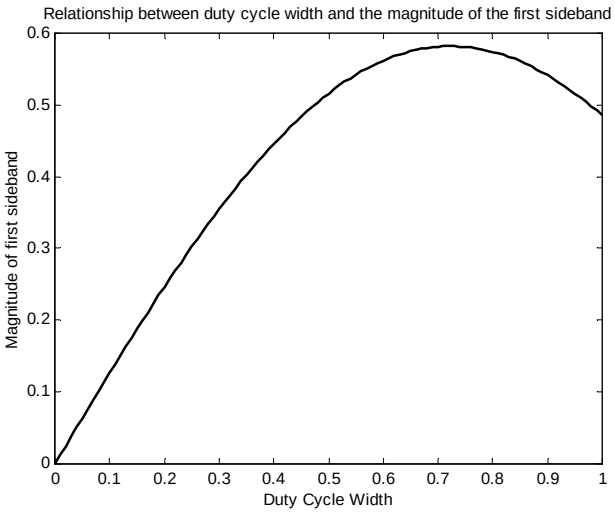


Figure 5. *Relationship between the duty cycle width and first sideband magnitude computed using (20).*

5. ANIMATOR USING A DELAY LINE FILTER

The multiple detuned oscillator effect can also be created by employing a group of delay-line based pitch shifters and a single waveform input [48]. The principle can be seen as an extension of the use of inverse comb filters with time-varying delays. By combining the output of such delay lines with the original signal, we will be able to model the multiple detuned oscillator effect for arbitrary inputs.

The pitch shifter operation is based on a periodic linear change in delay time. The amount of pitch transposition is proportional to the rate of delay change [49]. By modulating a delay line with a signal whose derivative is constant and non-zero, the pitch of the input signal can be shifted. We can define this process for a single up and down transposition pair by the following expressions, where τ is the delay line length, s is the frequency scaling factor (transposition ratio) and $x(t)$ is the input signal:

$$S_{wa,dl}(t) = w(\gamma(t))x(t - D(\gamma(t),\tau)) + w(\gamma(t)+\frac{1}{2})x(t - D(\gamma(t)+\frac{1}{2},\tau)) \quad (24)$$

where the delay modulation signal $\gamma(t)$ is defined as

$$\gamma(t) = (\frac{s-1}{\tau})t - \left\lfloor (\frac{s-1}{\tau})t \right\rfloor \quad (25)$$

The windowing and delay functions in (24), $w(x)$ and $D(x)$ are expressed by

$$w(x) = 0.5 + 0.5\cos(2\pi x) \quad (26)$$

and

$$D(x,d) = xd + \left\lfloor \frac{xd}{d} \right\rfloor d \quad (27)$$

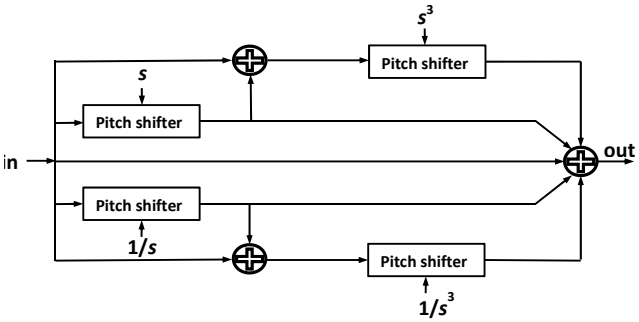


Figure 6. *The pitch-shifter based detuned oscillator effect where s is the transposition factor* $1 + \delta f_0/f_0$.

The windowing is necessary to hide the discontinuities of signal as the delay jumps from 0 to τ. The use of two delay lines is designed to allow crossfading between them, which creates a continuous pitch-shifted signal.

To create a mix of seven signals with slight tuning differences, we can use four pitch shifters arranged as in the block diagram in Fig. 6. The transposition factor s should then be set to $1 + \delta f_0/f_0$, providing a constant-interval spacing between each pitch-shifted copy of the original signal.

An example of the waveform output and the magnitude spectrum is given in Fig. 7. The upper panel shows the waveform over a 9 second period. The envelope of the waveform shows periodic peaks and troughs due to the beating that is occurring between the detuned harmonics. The two plots in the lower panel show the cluster of components around the region of the first harmonic and the second harmonic of the input. As expected the bandwidth around the second harmonic is proportionally wider than that of the first.

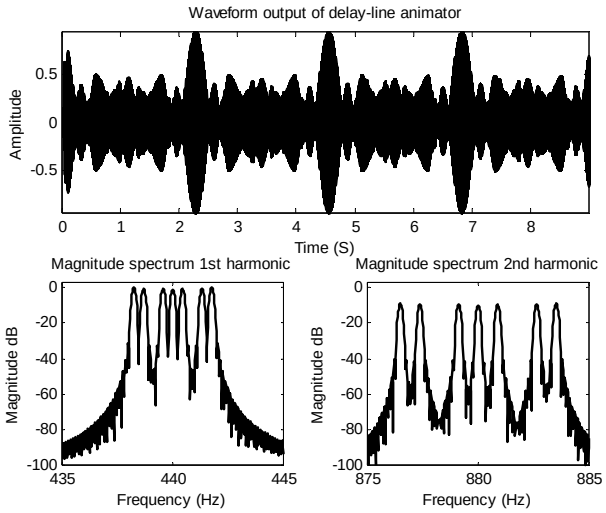


Figure 7. *The pitch-shifter based detuned oscillator effect, where s is the transposition factor*$1 + \delta f_0/f_0$. *The upper panel shows the waveform output and the lower panels shows the magnitude spectrum in the region around the first and second harmonics of the input waveform respectively.*

6. CONCLUSION

This paper has investigated the digital implementation of a Waveform Animator oscillator effect. It has first presented a mathematical analysis that has resulted in expressions for the frequency spectra of this effect, looking in detail at the significance of the modulation components. Secondly, it examined by simulation the relationship between the duty cycle and the degree of animation. Lastly, a delay line based algorithm was then discussed as means to obtain a general model of this effect. Furthermore, if the input to the delay-line model is bandlimited then the output will also be so. Such a model could be incorporated within any synthesis toolkit. It is intended that this work will offer sound designers more insight into alternative approaches for synthesizing 'Supersaw' timbres and also raise their awareness as to the role that frequency modulation plays within these sounds.

7. REFERENCES

[1] J. Pakarinen, V. Välimäki, F. Fontana, V. Lazzarini, and J. Abel, "Recent advances in real-time musical effects, synthesis, and virtual analog models," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, 2011.

[2] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music Journal*, 33(2), Summer 2009, pp. 85-100.

[3] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Trans. Audio, Speech, and Language Processing: Special Issue on Virtual Analog Modeling*, vol. 18, no. 4, pp. 738-746, May 2010.

[4] D. Yeh. and J. O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *Proc. 11th Int. Conf. Digital Audio Effects* (DAFx-08), Espoo, Finland, Sept. 2008.

[5] D. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*. Ph.D. thesis, Stanford University, Stanford, CA, USA, June 2009. https://ccrma.stanford.edu/~dtyeh/papers/pubs.htm

[6] V. Välimäki, J. Parker, and J. S. Abel, "Parametric spring reverberation effect," *J. Audio Eng. Soc.*, vol. 58, no. 7/8, pp. 547–562, July/August 2010.

[7] V. Välimäki and A. Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 116-115, March 2007.

[8] V. Lazzarini and J. Timoney, "New perspectives on distortion synthesis for virtual analogue oscillators," *Computer Music Journal*, vol. 34, no. 1, pp. 28-40, Mar. 2010.

[9] J. Kleimola, V. Lazzarini, J. Timoney, and V. Välimäki, "Phaseshaping oscillator algorithms for musical sound synthesis," in *Proc. 7th Sound and Music Computing Conference*, Barcelona, Spain, July 2010, pp. 94–101.

[10] J. Pekonen, V. Lazzarini, J. Timoney, J. Kleimola, and V. Välimäki, "Discrete-time modelling of the Moog sawtooth oscillator waveform," *EURASIP Journal on Advances in Signal Processing*, 2011.

[11] A. Huovilainen, *Design of a Scalable Polyphony-MIDI Synthesizer*. M.S. thesis, Aalto University, Espoo, Finland, May 2010, http://lib.tkk.fi/Dipl/2010/urn100219.pdf.

[12] T. Helie, "Volterra series and state transformation for real-time simulations of audio circuits including saturations: application to the Moog ladder filter," *IEEE Trans. Audio, Speech and Language Processing*, vol. 18, no. 4, pp. 747-759, May 2010,.

[13] F. Fontana and M. Civolani, "Modeling the EMS VCS3 voltage-controlled filter as a nonlinear filter network," *IEEE Trans. Audio, Speech and Language Processing*, vol. 18, no. 4, pp. 760-772, May 2010.

[14] G. De Sanctis and A. Sarti, "Virtual analog modeling in the wave digital domain," *IEEE Trans. Audio, Speech and Language Processing*, vol. 18, no. 4, pp. 715-727, May 2010.

[15] J. Lane, D. Hoory, E. Martinez, and P. Wang, "Modeling analog synthesis with DSPs," *Computer Music Journal*, vol. 21(4), 1997, pp. 23–41.

[16] D. Ambrits and B. Bank, "Improved polynomial transition regions algorithm for alias-suppressed signal synthesis," in *Proc. Sound and Music Computing Conf.* (SMC 2013), Stockholm, Sweden, pp. 561-568, Aug. 2013.

[17] S. Tassart, "Band-limited impulse train generation using sampled infinite impulse responses of analog filters," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 21, no. 3, pp. 488-497, Mar. 2013.

[18] J. Pekonen and M. Holters, "Nonlinear-phase basis functions in quasi-bandlimited oscillator algorithms," in *Proc. 15th Intl. Conf. Digital Audio Effects* (DAFx-12), York, UK, pp. 261-268, 2012.

[19] J. Kleimola and V. Välimäki, "Reducing aliasing from synthetic audio signals using polynomial transition regions," *IEEE Signal Processing Letters*, vol. 19, no. 2, pp. 67-70, Feb. 2012.

[20] R. Snoman, *The Dance Music Manual*. Focal press, Elsevier, Oxford, UK, 2004.

[21] G. Winham and K. Steiglitz, "Input generators for digital sound synthesis," *J. Acoust. Soc. Amer.*, vol. 47, part 2, pp 665-666, Feb. 1970.

[22] T. Stilson and J. O. Smith, "Alias-free digital synthesis of classic analog waveforms," in *Proc. Int. Computer Music Conf.*, Hong Kong, pp. 332-335, 1996.

[23] T. Stilson, *Efficiently-Variable Non-Oversampled Algorithms in Virtual-Analog Music Synthesis - A Root-Locus Perspective*. Ph.D. thesis, Stanford University, Stanford, CA, USA, June 2006. http://ccrma.stanford.edu/~stilti/papers/Welcome.html.

[24] J. Nam, V. Välimäki, J. S. Abel, and J. O. Smith, "Alias-free virtual analog oscillators using a feedback delay loop," in *Proc. 12th Intl. Conf. Digital Audio Effects* (DAFx-09), Como, Italy, September 1-4, 2009.

[25] J. Nam, V. Välimäki, J. S. Abel, and J. O. Smith, "Efficient antialiasing oscillator algorithms using low-order fractional delay filters," *IEEE Trans. Audio, Speech and Language Processing*, vol. 18, no. 4, pp. 773–785, May 2010.

[26] V. Välimäki, "Discrete-time synthesis of the sawtooth waveform with reduced aliasing," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 214-217, March 2005.

[27] V. Välimäki, J. Nam, J. O. Smith, and J. S. Abel, "Alias-suppressed oscillators based on differentiated polynomial waveforms," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 4, pp. 786–798, May 2010.

[28] V. Välimäki and A. Huovilainen, "Oscillator and filter algorithms for virtual analog synthesis," *Computer Music Journal*, vol. 30, no. 2, pp. 19-31, 2006

[29] J. Timoney, V. Lazzarini, and T. Lysaght, "A modified FM synthesis approach to bandlimited signal generation," in *Proc. 11th Intl. Conf. Digital Audio Effects* (DAFx-08), Espoo, Finland, pp. 27–33, Sept. 1–4, 2008.

[30] V. Välimäki, J. Pekonen and J. Nam, "Perceptually informed synthesis of bandlimited classical waveforms using integrated polynomial interpolation," *J. Acoust. Soc. Amer.*, vol. 131, no. 1, pt. 2, pp. 974-986, Jan. 2012.

[31] E. Brandt, "Hard sync without aliasing," in *Proc. Int. Comp. Music Conf.*, Havana, Cuba, Sept. 17-22, 2001.

[32] J. Timoney, V. Lazzarini, M. Hodgkinson, J. Kleimola, J. Pekonen, and V. Välimäki, "Virtual analog oscillator hard sync: Fourier series and an efficient implementation," in *Proc. 15th Intl. Conf. Digital Audio Effects* (DAFx-12), York, UK, September 17-21, 2012.

[33] D. Lowenfels, "Virtual analog synthesis with a time-varying comb filter," *AES Convention 115*, New York, NY, USA, Oct. 10-13, 2003.

[34] J. Timoney, and V. Lazzarini, "Exponential frequency modulation bandwidth criterion for virtual analog applications," in *Proc. 14th Intl. Conf. Digital Audio Effects* (DAFx-11), Paris, France, Sept. 19-23, 2011.

[35] B. Hutchins, "Analog circuits for sound animation," *J. Audio Eng. Soc.*, vol. 29, no. 11, pp 814-820, Nov. 1981.

[36] M. Russ, *Sound Synthesis and Sampling*. Focal press, Elsevier, Oxford, UK, 2009.

[37] Moog Music Inc., *Minimoog Voyager Old School user's manual*, 2008. HTTP://WWW.BIGBRIAR.COM/MANUALS/OLD_SCHOOL_MANUAL_1_0.PDF.

[38] Roland Corporation, *JP-8000 Owner's Manual*. 1996. [Online]. http://www.rolandus.com.

[39] Roland Corporation, *Jupiter-8 Owner's manual*, 1983. FTP://FTP.ROLAND.CO.UK/PRODUCTSUPPORT/JP-8/01_JP-8_OM.PDF

[40] Sequential Circuits, *Prophet 5 Owner's manual*, 1977 WWW.SUONOELETTRONICO.COM/DOWNLOADS/PROPHET5MANUAL.PDF

[41] Access Music Electronics GMBH, Access TI Quick Start manual, HTTP://WWW.ACCESSMUSIC.DE/PAGE/RENDER/LANG/EN/P/15/DO/SUPPORT_CONTACT_AND_RESOURCES.HTML.

[42] J. C. Risset, "Examples of the musical use of digital audio effects," *J. New Music Research*, vol. 31, no. 2, pp. 93-97, June 2002.

[43] D. Trueman, P. Cook, S. Smallwood and G. Wang, "PLOrK: The Princeton Laptop Orchestra, Year 1," in *Proc. Int. Comp. Music Conf.*, New Orleans, LA, USA, Nov. 6-11, 2006.

[44] W. Hartmann, *Signals, Sound and Sensation*, Springer, USA, 2005.

[45] R. Guinee, "A novel Fourier series simulation tool for pulsewidth modulation (PWM) in pulsed power systems," in *Proc. IEEE 22nd Symp. Fusion Eng.*, Albuquerque, NM, USA, 17-21 June 2007, pp. 1-4.

[46] M. LeBrun, "A Derivation of the Spectrum of FM with a Complex Modulating Wave," *Computer Music Journal*, vol. 1, no. 4, pp. 51-52, Winter 1977.

[47] G. Watson, *A Treatise on the Theory of Bessel Functions*. Cambridge university press, Cambridge, UK, 1995.

[48] K. Bogdanowicz and R. Belcher, "Using multiple processors for real-time audio effects," in *Proc. Audio Eng. Soc. 7th Int. Conf.*, May 1989, pp. 337-342.

[49] S. Disch and U. Zolzer, "Modulation and delay line based digital audio effects," in *Proc. 2nd Int. Conf. Digital Audio Effects* (DAFx-99), Trondheim, Norway, pp. 5–8, Sept. 9–11, 1999.

MULTI-PLAYER MICROTIMING HUMANISATION USING A MULTIVARIATE MARKOV MODEL

Ryan Stables
DMT Lab,
Birmingham City University
Birmingham, UK
ryan.stables@bcu.ac.uk

Satoshi Endo
Institute for Information-Oriented Control
Technische Universität München,
Munich, Germany
s.endo@tum.de

Alan Wing
SYMON Lab,
School of Psychology,
Birmingham University,
Birmingham, UK
a.wing@bham.ac.uk

ABSTRACT

In this paper, we present a model for the modulation of multi-performer microtiming variation in musical groups. This is done using a multivariate Markov model, in which the relationship between players is modelled using an interdependence matrix (α) and a multidimensional state transition matrix (S). This method allows us to generate more natural sounding musical sequences due to the reduction of out-of-phase errors that occur in Gaussian pseudorandom and player-independent probabilistic models. We verify this using subjective listening tests, where we demonstrate that our multivariate model is able to outperform commonly used univariate models at producing human-like microtiming variability. Whilst the participants in our study judged the real time sequences performed by humans to be more natural than the proposed model, we were still able to achieve a mean score of 63.39% naturalness, suggesting microtiming interdependence between players captured in our model significantly enhances the humanisation of group musical sequences.

1. INTRODUCTION

In electronically produced music, humanisation algorithms are often applied to percussive sequences in order to create a more natural sounding expressive performance. This is particularly useful when access to performers or equipment is limited, as events can be programmed onto a quantised grid and then modulated by a music producer, without the requirement for human performance. This process is often applied during the point of music creation from within the digital audio workstation and allows for the incorporation of sampled or synthesised instruments into a piece of music.

One of the main issues with current humanisation systems is that they do not necessarily represent the expressivity exhibited by a human agent, thus the process requires further editing in order to achieve a natural approximation of a human musician. Furthermore, the systems are unable to model the characteristics of group performance when used in a multi-channel environment. These problems are namely due to the fact that the majority of existing humanisation systems modulate the onset locations and respective velocities of an event instantaneously, using a pseudorandom variate, selected from a Gaussian window. Therefore in simulated multi-player performance, phase error is often introduced between the channels. This can actually reduce the naturalness of the performance, rather than enhance it due to perceptually unrealistic cues, generated by multiple instances of the algorithm running in parallel.

1.1. Modelling Microtiming

In this study, we focus specifically on extracting and modulating microtiming offsets in musical performance, this can be defined as the subtraction of an event at time n from a corresponding point on a reference track, as illustrated in Figure 1. Here, the reference grid represents a metronome running in parallel with the performed musical sequence. The challenge of the humanisation algorithm is to then estimate the distribution at $n+1$, written as $P(\theta_{n+1})$. This is usually done independently of all other events in the sequence, based on a distribution centred around the n^{th} grid point, characterised by the parameters μ and σ.

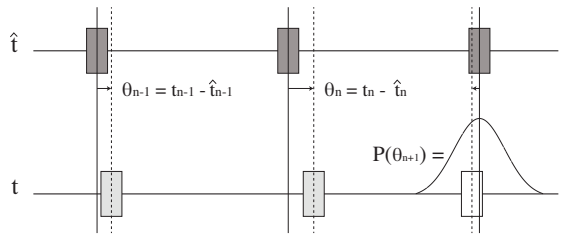


Figure 1: *Representation of a player, following a metronome. The offset measurements from the metronome are shown as θ_n, where $\hat{t}_n$ is the n^{th} metronomic event and t_n is the n^{th} event performed by player 1.*

Attempts have been made in previous studies to increase the naturalness of single-player humanisation systems by incorporating some form of intelligent processing into the variate generation procedure. In [1] for example, fuzzy logic has been used to model strike velocity deviation in a humanisation system, based on subjective rules, derived from domain knowledge. Similarly, microtiming deviation has been modelled using a number of different supervised machine learning techniques by [2]. These techniques are then used to apply the derived microtiming models to quantised sequences, in which they conclude the systems used to model percussive sequences significantly outperform the quantised version, when evaluated for natural expressivity. Microtiming for Brazilian Samba music is also estimated in [3] and [4], using a model based on the extraction of quarter-note patterns using K-Means clustering. Here, it is shown that the degree of expressive timing can be attributed to specific metrical positions, with examples in Samba music. This kind of information is omitted when pseudo-random models are applied, due to the variables being independently distributed for each event.

In previous work by Stables ([5], [6]), it has been shown that the process of stochastic humanisation can be improved using probabilistic temporal models to modulate a quantised sequence, based on microtiming measurements taken from professional musicians. Here, independently distributed variates ($P(\theta_{n+1})$) were replaced by variates that were conditionally dependent in time ($P(\theta_{n+1}|\theta_n)$). In these studies it was shown that the measured sequences exhibited temporal patterns which could be synthesised using finite state machines. In both cases, the empirically developed models were shown to subjectively outperform quantised and Gaussian sequences for both perceived naturalness and musicality.

2. GENERATIVE MULTIVARIATE MODEL

Whilst the models described in section 1.1 work particularly well with single-player sequences, phase error is still introduced in mutichannel performance models due to the lack of inter-performer dependence. This means that when a probabilistic humanisation algorithm is applied to more than one track in a given session, extensive manual correction is often required in order to create a sense of cohesion between the separate channels. It is therefore necessary to consider ways in which a group of musicians can be modelled in parallel, thus preserving the inter-performer timing characteristics of a musical group.

If we make the assumption that the performed musical signals are stylised stochastic processes (as in studies such as [7] and [8]), we can use a Markov chain to estimate a transition through a discrete state-space $Z = \{z_1, z_2, \ldots, z_K\}$, where z_n represents the n^{th} state of the system, providing the sequence being modelled, satisfies the Markov property given in Eq. 1.

$$\begin{aligned} &P(\theta_{n+1} = i_{n+1}|\theta_0 = i_0, \theta_1 = i_1, \ldots, \theta_n = i_n) \\ &= P(\theta_{n+1} = i_{n+1}|\theta_n = i_n) \end{aligned} \tag{1}$$

Here, θ_n represents the n^{th} event and i_n represents the corresponding state. Each state in the model can be described using canonical form representation, consisting of a binary vector of length K, where $\sum_{k=1}^{K} \theta_k = 1$ and $\theta_k \in \{0, 1\}$. For example, in a 5-state model, if the n^{th} event is equal to z_3, we can use the representation $\theta_n = \{0, 0, 1, 0, 0\}^T$. This allows us to define a single-player model using Eq. 2.

$$P(\theta_{n+1}) = S\theta_n \tag{2}$$

Here, S is a state transition matrix (STM), representing the probability of a transition from $\theta_n = i_n$ to $\theta_{n+1} = i_{n+1}$ for $n = \{1, 2, \ldots, N\}$, where N is the number of events in the sequence. We then consider $P(\theta_n)$ to be the Probability Density Function (PDF) representation of θ_n. The canonical form of θ_{n+1} is then calculated using a rejection sampling technique, given here in Eq. 4.

$$(\theta_n)_i = \begin{cases} 1, & i = \beta \\ 0, & i \neq \beta \end{cases} \tag{3}$$

$$\beta = \begin{cases} \gamma_{1,k}, & [\gamma_{1,k}, \gamma_{2,k}] \in P(\theta_n) \\ repeat, & [\gamma_{1,k}, \gamma_{2,k}] \notin P(\theta_n) \end{cases} \tag{4}$$

Where $\gamma_{1,k}$ and $\gamma_{2,k}$ are pair-wise stochastic variables, evaluated against the n^{th} state distribution and β is the state vector index. For situations such as grouped musical performance, in which there are two or more conditionally dependent sequences, we can use a Multivariate Markov Chain (MVMC) model. This consists of the univariate model, estimated across M sequences being performed concurrently, weighted by some measure of inter-player dependence, given in Eq. 5.

$$P(\theta_{n+1}^{(i)}) = \sum_{k=1}^{M} \alpha_{i,k} S^{(i,k)} \theta_n^{(k)} \tag{5}$$

In the multivariate model, $\theta_n^{(i)}$ represents the state distribution of stream i in canonical form and the matrix $S^{(i,k)}$ gives the probability of a transition from the n^{th} state in stream i, to the $(n+1)^{th}$ state in stream k, as demonstrated in Eq. 6. When $k = i$, S represents a standard univariate STM. The weights ($\alpha_{i,k}$) in the model represent the interdependence factor between streams i and k, which can be derived empirically.

2.1. Pulse Approximation

As demonstrated in Figure 1, the estimation of microtiming parameters in the current model relies on an isochronous grid ($\hat{t}$) in order to calculate differentials ($\theta^{(i)}$) at any point in time (n). In single-player streams this model works particularly well if a player has performed the sequence to a click-track as we can use a metronomic grid to approximate $\hat{t}$. However due to the nature of group performance, it is relatively unlikely that the individual performers will follow the same click track, unless the musicians are independently contributing material to the musical piece. This trait is very common in multitrack recording, but less common in group performance. Using a metronomic model, we can represent the grid using Eq. 8.

$$\begin{aligned} &\theta_n^{(i)} = t_n^{(i)} - \hat{t}_n \\ &where, \quad \hat{t}_n = (n-1)\left(\tfrac{60}{\tau}\right) \end{aligned} \tag{8}$$

Where τ represents a measurement of fixed tempo and $t^{(i)}$ is the event generated by the i^{th} performer. In order to adapt this method for group performance, we need to estimate a global representation of tempo within the musical group. We can provide a simplistic model for this by taking the mean of the beat-spacings within each bar, across all players using Eq. 9.

$$S^{(i,k)} = \left\{ \begin{array}{cccc} p(\theta^{(i)} = z_1 | \theta^{(k)} = z_1) & p(\theta^{(i)} = z_2 | \theta^{(k)} = z_1) & \dots & p(\theta^{(i)} = z_K | \theta^{(k)} = z_1) \\ p(\theta^{(i)} = z_1 | \theta^{(k)} = z_2) & p(\theta^{(i)} = z_2 | \theta^{(k)} = z_2) & \dots & p(\theta^{(i)} = z_K | \theta^{(k)} = z_2) \\ \vdots & \vdots & \vdots & \vdots \\ p(\theta^{(i)} = z_1 | \theta^{(k)} = z_K) & p(\theta^{(i)} = z_2 | \theta^{(k)} = z_K) & \dots & p(\theta^{(i)} = z_K | \theta^{(k)} = z_K) \end{array} \right\} \tag{6}$$

$$i = \{1, 2, \dots, M\}, \quad k = \{1, 2, \dots, M\} \tag{7}$$

$$\hat{\tau}_m = \frac{1}{MB} \sum_{i=1}^{M} \sum_{n=1}^{B} (t_n^{'})^{(i)} - (t_{n-1}^{'})^{(i)} \tag{9}$$

Where $t_n^{'}$ represents an event that falls on a beat location and B is the number of beats in the bar. $\hat{\tau}_m$ then represents the estimated tempo for the m^{th} bar. This is now an estimated dynamic measurement of temporal drift and is updated each time a new bar is performed. The micro timing offsets are then subtracted from this grid, using the technique defined in Eq. 8, replacing τ with $\hat{\tau}_m$ and interpolated for n.

2.2. Inter-Player Dependence

We model the interdependence ($\alpha_{i,j}$) amongst performers in the group using lagged cross-correlation, in which player i's stream is lagged by a pre-defined number of events (n) and correlated with the stream of player j. This allows us to estimate the amount of dependence that one player has on another. This technique has been demonstrated by [9] to be optimal at a single event, suggesting that players are highly receptive to short-term variations in accompaniment. This measurement is demonstrated in Eq 10.

$$\alpha_{i,j} = \frac{1}{N} \sum_{k=0}^{N-n-1} \theta_k^{(i)} \theta_{k+n}^{(j)} \tag{10}$$

Where n is a non-negative integer representing the number of events to lag, set to 1 for this application.

3. EXPERIMENT: STRING QUARTET MODELLING

In order to evaluate the performance of the model, we analyse a professional quartet performing an excerpt from the 4^{th} movement of Hayden's String Quartet Op. 74 No. 1 in C-Major, the score for which is given in Figure 2. The quartet consisted of two violins, a viola and a cello, and the excerpt was chosen due to the number of notes being performed concurrently. The quartet have around 12 years experience performing together, and were shown by [9] to follow the lead violin player relatively closely. The excerpt, consisting of 12 bars was performed and recorded 15 times using the same equipment and the musicians were asked to perform using their natural expression. In total, each take contained 48 musical events, all of which were being performed by all members of the quartet at the same metrical positions in the bar.

Each player was recorded using an individual instrument microphone (DPA 4061), positioned on the body of each instrument with a rubber mount in order to reduce bleed in the recordings. The onsets from each player were then extracted using a spectral-flux based technique, and adjusted manually to improve accuracy. To find the microtiming offsets, the pulse was estimated at the beginning of each bar using the method defined in Eq. 9 and the events were subtracted using the technique defined in Eq. 8. The mean tempo for the recordings was found to be 105.0 BPM, with a standard deviation of 6.49. Figure 3 illustrates offset measurements from all 15 takes, with the mean of the results represented in black. Here, deviations are shown across all four performers playing concurrently.

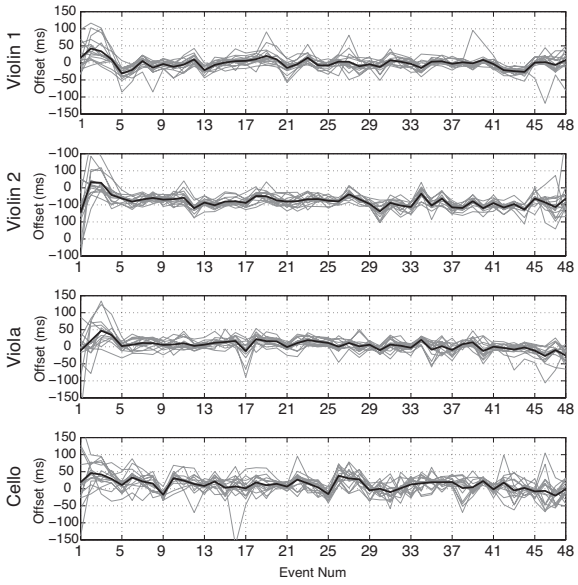


Figure 3: *A graphical representation of the microtiming deviation (θ) for all four performers. the measurements are taken across 15 takes of the same piece with the mean offset indicated in black, the vertical lines represent bar divisions.*

3.1. Subjective Evaluation

To evaluate the perceived naturalness of the model, subjective listening tests were conducted using a MUSHRA-based methodology [10]. The subjects were asked to rank each of the samples with a multi-stimulus interface and provide a rating between 0-100 for how naturally expressive each sample was perceived to be. Participants were informed that the experiments were based on professional musicians and were played an excerpt from a string quartet (not included in the stimuli) before the test began. In total 20 people participated in the experiment, all of whom were all aged between 18-35 and had normal hearing. All participants had some experience in performing or producing music.

The stimuli consisted of 25 versions of the same synthesised polyphonic sequence, the score for which was taken from Haydn's

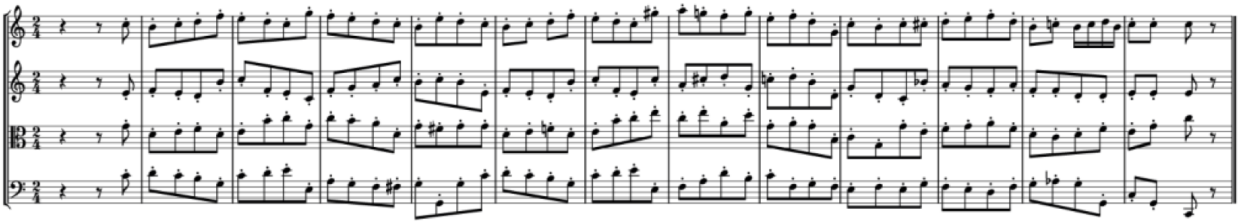


Figure 2: *The score of the excerpt taken from Hayden's Quartet Op. 74 No. 1 in C Major, in which 4 separate instrument parts are shown.*

Quartet Op. 74 and synthesised using a string ensemble pre-set from the Logic Studio 9 plug-in: EXS24 (Apple, CA, USA). The sequences were compiled by generating MIDI note-on messages and importing them into a sequencer. The MIDI was generated using 5 different techniques, these can be categorised as follows[1].

- *Quantised*: The note-on messages were quantised to a fixed grid, thus exhibiting no temporal variation.
- *Gaussian*: Each of the note-on messages were modulated using an independent Gaussian window.
- *MC*: The note-on messages for each channel were modulated using a conditionally independent Markov chain.
- *MVMC*: The note-on messages are modulated using the MVMC model presented in Eq. 5.
- *Human*: The onsets are taken from a dataset of human performers.

In order to isolate microtiming deviation, other parameters such as note-off and velocity were fixed to constant variables. The length of each event was fixed to 1/4-note length and the global tempo was varied across samples, bounded by measurements from the dataset. To control the mean and variance of the micro timing deviations across conditions, the μ and σ parameters used to characterise the distributions in the Gaussian method were derived from the dataset of human performers. This meant that all techniques were able to produce a similar range of θ values.

4. RESULTS

4.1. Performance Analysis

From our observations of a string quartet performing 15 iterations of a 12-bar of a piece in 4/4, we can identify characteristics of the musical group by performing analysis on the data. Firstly, the maximum microtiming deviation was measured to be 198.02ms and the minimum was -202.48ms. Overall the mean was 6.51ms, with a SD of 2.65ms. As the mean tempo was observed to be 105BPM, in 4/4 time signature, the maximum deviation was around 35.4% and the mean deviation was around 1.2% of the inter-onset interval (IOI).

The dependencies between each performer in the group are summarised in Eq. 11 and also shown using boxplots in Figure 4. Both of these diagrams represent the variable α in the model.

$$\alpha = \left\{ \begin{array}{cccc} 0.410 & 0.009 & 0.026 & 0.001 \\ 0.177 & 0.257 & 0.113 & 0.147 \\ 0.217 & 0.203 & 0.320 & 0.175 \\ 0.007 & 0.151 & 0.072 & 0.181 \end{array} \right\} \quad (11)$$

[1]Stimuli can be found at http://www.ryanstables.co.uk/data/dafx14

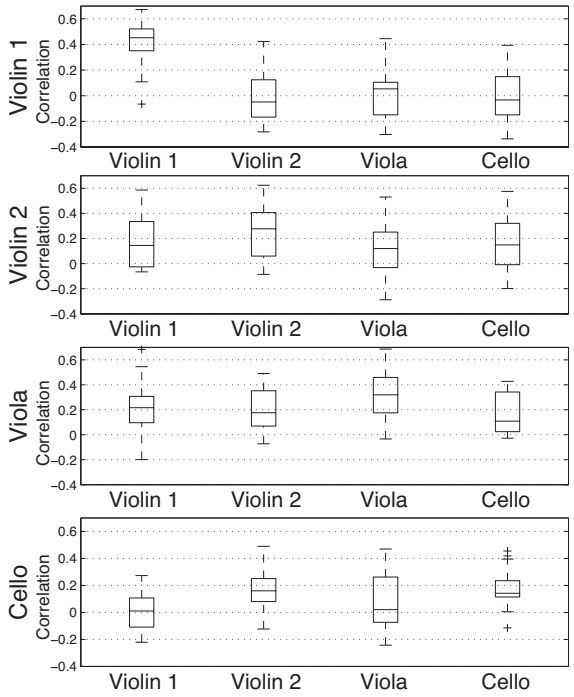


Figure 4: *Boxplot representation of inter-player dependence measured over 15 takes. This is measured using a lagged cross-correlation function.*

Here it is evident that the most highly correlated measurements taken from the data are based on lagged autocorrelation. This promotes the use of Markov chains in musical performance modelling as it suggests there is a strong relationship between an event (x_n) and it's predecessor (x_{n-1}) within the same stream. Generally, the 1^{st} violin has very low correlation scores with the other musicians in the group with a mean of 0.012 and a very high auto-correlation measurement. This suggests that they have adopted the role of lead performer. The other musicians in the group are generally more positively correlated with each other. Here, both the 2nd Violin and the viola player are following the lead violin, whilst the Cello is following the 2nd violin. We can calculate a leadership metric (l_α) for each player by taking the column-wise means, excluding the autocorrelation measurements at cell α_{ij} where $i = j$. This is illustrated in Eq. 12.

$$l_\alpha = \left\{ \begin{array}{cccc} 0.1337 & 0.1210 & 0.0703 & 0.1077 \end{array} \right\} \quad (12)$$

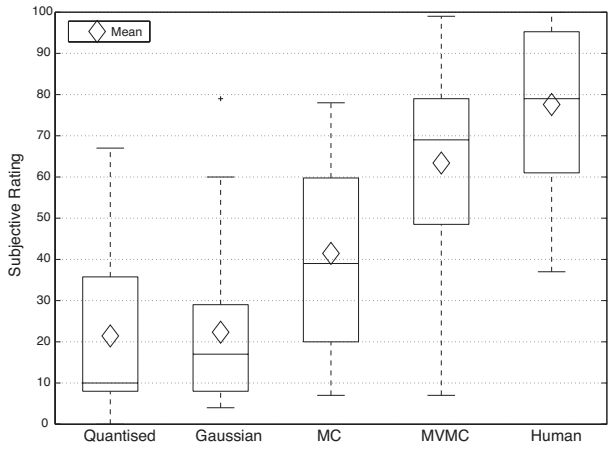


Figure 5: *A boxplot showing subjective listening test results taken from 20 subjects. The stimuli consisted of 5 samples taken from 5 categories (25 in total).*

Here, it is evident that the 1st Violin has the highest degree of leadership, reinforcing the suggestion that the performer has a leading role within the group.

4.2. Model Evaluation

In order to evaluate the naturalness of the model, we performed subjective tests to identify the similarity between the generated sequences and the performed sequences. The results from the subjective tests are illustrated in Figure 5. Here, it is evident that the microtiming sequences sampled from real musicians performed better than any of the synthetic samples with a mean score of 77.59%. The lowest scoring categories were Gaussian and Quantised models, which scored 22.33% and 21.42% respectively. The samples that were generated using the proposed multivariate model scored relatively highly with 63.39%, this was 21.94% higher than the closest category, which was the univariate model suggested in [5]. This result shows that the multivariate model performs slightly less favourably than using onsets taken directly from human performers, however it outperforms all existing methods for univariate modulation.

5. DISCUSSION

5.1. Model Performance

From the analysis of the string quartet, it is evident that the performers all seem to have stronger lagged autocorrelation scores (α_{ij}, where $i = j$), than cross-correlation scores ($i \neq j$). This would suggest that the internal representation of time held by each player takes priority over the external timings of group performance. Whilst these autocorrelation scores are significantly higher than the cross-correlation measurements, the performers still produce a sufficient amount of microtiming offset to cause potentially audible phase errors in the piece. This suggests the model's dependence matrix (α) is a significant factor as both the univariate model and the normally distributed model (with equivalent μ and σ parameters) underperform at producing timing sequences with natural expressivity. Subjectively, the multivariate model tends to produce much more confluent sequences than any of the univariate models running in parallel across multiple channels.

Whilst the subjective listening tests show an increased mean score for the multivariate model, suggesting the model is able to produce realistic musical sequences, there is a much higher variance than in other categories. This means there is uncertainty within the results, with some participants rating the system as low as 7/100. This is acceptable to an extent due to the relative uncertainty in the human samples, however it suggests there is room for improvement due to the inconsistency in results.

5.2. Implementation

Whilst we have demonstrated that the univariate models running in parallel do not perform particularly well for this application, the model allows for the conversion between univariate and multivariate methods by converting α to an identity matrix, imposing conditional independence on all streams. Similarly, we can alter the dependencies in α to change the characteristics of the musical group. If for example, the performance requires the group to closely follow Violin 1, the values in column 1 can be incremented, thus increasing the performers' leadership score (l_α). From an implementation standpoint, this is relatively simple to parameterise as users of the system can input values into the dependence matrix directly or via some mapping function.

Another key aspect to producing natural sounding rhythmic performance is tempo variation. In our listening tests, this was based on existing templates taken from our dataset. In most humanisation systems, this is ignored as control is generally maintained by the host application. For systems that wish to include this attribute, another variable can be added directly to the sum in Eq. 5, derived using the technique defined in Eq. 8. In the performances measured for this study, the tempo variation has a particularly high standard deviation due to the expressive nature of the music. In other genres such as pop-music, this may not be as important due to the prominence of quantisation and click-tracks.

6. CONCLUSION

In this paper, we have presented a model for the synchronous modulation of multiple streams of onsets using a multivariate Markov model. The model derives parameters from a user-defined corpus of multi-performer musical data and probabilistically applies modulation to a group of concurrent sequences. We can estimate the inter-player dependencies using lagged cross-correlation metric and approximate the pulse of the group using the bar-wise mean of all performers. The model is designed to alleviate the phase issues that arise when humanisation algorithms are applied to multiple sequences simultaneously.

We have demonstrated that the model outperforms univariate techniques including an instantaneous pseudorandom model and a Markov chain model applied independently to multiple channels, using data from a string quartet performing Haydn's Quartet Op. 74 No. 1 in C-Major. Through subjective listening tests, we observed an improvement of 21.94% accuracy on the closest synthesized category when measured for naturalness of expression. Whilst this was a significant improvement, sequences derived directly from human agents were still perceived to be more expressive than the model, indicating the importance and complexity of

the interdependence in multi-player musical performance that requires further attention.

7. REFERENCES

[1] L. O'Sullivan and F. Boland, "Towards a Fuzzy Logic Approach To Drum Pattern Humanisation," in *Proc. of the 13th Itl. Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 19-21, 2010.

[2] M. Wright and E. Berdahl, "Towards machine learning of expressive microtiming in Brazilian drumming," in *International Computer Music Conference*, 2006.

[3] Fabien Gouyon, "Microtiming in samba de roda - preliminary experiments with polyphonic audio," in *Simpósio da Sociedade Brasileira de Computação Musical*, 2007.

[4] Luiz Alberto Naveda, Fabien Gouyon, Carlos Guedes, and Marc Leman, "Multidimensional microtiming in samba music," in *12th Brazilian Symposium on Computer Music*. SBCM, 2009, pp. 1–12.

[5] Ryan Stables, Jamie Bullock, and Ian Williams, "Perceptually relevant models for articulation in synthesised drum patterns," in *Audio Engineering Society Convention 131*. Audio Engineering Society, 2011.

[6] Ryan Stables, Cham Athwal, and Rob Cade, "Drum pattern humanization using a recursive bayesian framework," in *Audio Engineering Society Convention 133*, Oct 2012.

[7] Kevin Jones, "Compositional applications of stochastic processes," *Computer Music Journal*, vol. 5, no. 2, pp. 45–61, 1981.

[8] M. Kaliakatsos-Papakostas, M.G. Epitropakis, and M.N. Vrahatis, "Weighted markov chain model for musical composer identification," in *Applications of Evolutionary Computation*. 2011, vol. 6625 of *Lecture Notes in Computer Science*, p. 334–343, Springer.

[9] Alan M Wing, Satoshi Endo, Adrian Bradbury, and Dirk Vorberg, "Optimal feedback correction in string quartet synchronization," *Journal of The Royal Society Interface*, vol. 11, no. 93, pp. 20131125, 2014.

[10] ITURBS Recommendation, "1534-1: Method for the subjective assessment of intermediate quality level of coding systems," *International Telecommunication Union*, 2003.

STREAMING SPECTRAL PROCESSING WITH CONSUMER-LEVEL GRAPHICS PROCESSING UNITS

Victor Lazzarini, John ffitch, Joe Timoney

Depts. of Music and Comp. Sci.,
National University of Ireland
Maynooth, Ireland
vlazzarini@nuim.ie,
jpff@codemist.co.uk,
jtimoney@cs.nuim.ie

Russell Bradford

Dept. of Comp. Sci.,
University of Bath
England
rjb@cs.bath.ac.uk

ABSTRACT

This paper describes the implementation of a streaming spectral processing system for realtime audio in a consumer-level onboard GPU (Graphics Processing Unit) attached to an off-the-shelf laptop computer. It explores the implementation of four processes: standard phase vocoder analysis and synthesis, additive synthesis and the sliding phase vocoder. These were developed under the CUDA development environment as plugins for the Csound 6 audio programming language. Following a detailed exposition of the GPU code, results of performance tests are discussed for each algorithm. They demonstrate that such a system is capable of realtime audio, even under the restrictions imposed by a limited GPU capability.

1. INTRODUCTION

Graphics Processing Units (GPUs) have been used for audio processing in a variety of environments. Typically, they have been employed as co-processors in systems where their massively parallel architectures can be harnessed for signal processing programs[1]. There is now a significant number of reports in the literature demonstrating their use in the implementation of various algorithms, such as Ray Tracing [2], Wave-based Modelling [3], SMS[4], Finite Difference Physical Models[5], to cite but a few.

In this paper we investigate the use of off-the-shelf consumer-level GPUs for the implementation of frequency-domain audio processing. In such a scenario, we do not have a separate dedicated co-processor, but rely solely on the on-board GPU that is also driving the video graphics subsystem. Our goal was to study and implement efficient algorithms that could overcome the limitations of the given hardware and possibly deliver realtime performance. In particular, we are interested in developing applications that can be employed by users without the need for specialised hardware setups. Finally, we also envisage that such implementations can, in a second stage, be applied to dedicated co-processor systems in high-performance computing applications.

The processes implemented in this paper involve separate Phase Vocoder (PV) analysis and synthesis, Additive synthesis from PV data, and a Sliding PV (SPV) algorithm-based frequency domain effect[6].

1.1. Environment and toolset

The chosen environment involved a NVIDIA GT650M GPU, with 1024MB VRAM (see Table 1), running on OSX10.9. The chosen parallel development toolset was CUDA 5.5[7], running in conjuction with the LLVM/Clang C/C++ compiler, with Csound 6.02 as the host for the processing plugins. The choice of environment was dictated by two concerns: a good match for the target hardware, which CUDA is, and on the hosting side, a well-developed environment for testing of audio programs, which is provided by Csound[8] version 6[9].

Table 1: *Some specifications for the target GPU*

cores	384
clock speed	900 MHz
VRAM	1024MB
compute capability	3.0
max threads/block	1024
bandwidth	80 GB/s
multiprocessors	2
cores per multiprocessor	192

2. PROGRAMMING MODEL

The programming model supported by CUDA abstracts the GPU processors into a hierarchy of threads, blocks, and grids. At the lowest level, we have separate threads that can be grouped into blocks. A grid is a collection of blocks.

Each thread in a block can be given a one, two or three dimensional index, to facilitate computation across vectors, matrices or volumes. All threads in a block live on the same multiprocessor, execute in parallel and can share fast memory. Blocks can be scheduled in parallel in separate multiprocessors. There is an upper limit in the number of threads in a block, which depends on the compute capability of the hardware used, and in the case of the GT650M is 1024, as shown in Table 1.

Each thread has a local memory space, and can access shared memory within its block. All threads have also access to global device memory. The fast shared memory is very limited in size,

but has a higher bandwidth and lower latency than global memory. Any memory transfers between host (CPU) and device (GPU) are costly and should be minimised.

Thread execution is grouped in *warps*, which contains 32 threads. For full efficiency, it is advised that all threads in a warp have a single execution path. Divergence via conditional branches will force each branch to be executed serially until these converge back into the same path. For this reason, it is important to minimise divergent conditionals in the GPU code.

The code executed by a thread is provided in a unit called a *kernel*. For all practical purposes, this is a C/C++ function (defined by the CUDA attribute `__global__`) that is designed to run in multiple copies, concurrently. The CUDA programming extensions provide a simple syntax to launch a grid of threads based on a given kernel, with a certain number of blocks and threads per block.

CUDA programming assumes a heterogenous programming model, where a host is responsible for allocating and managing device memory, including data transfers, as well as scheduling the parallel execution on the device. Under this model, serial sections of code, running on the host, are interspersed with parallel ones running in the GPU. This is illustrated by Figure 1.

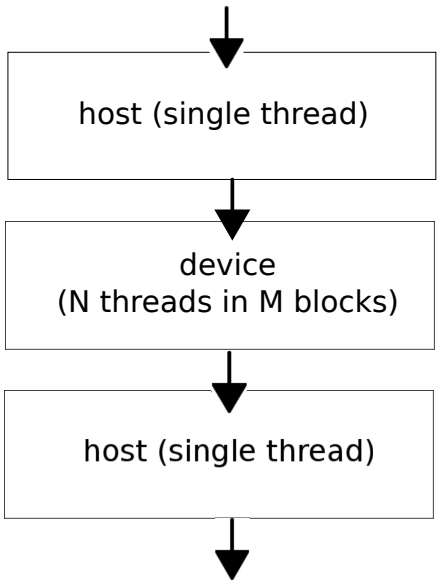


Figure 1: *Heterogenous programming model*

3. STREAMING SPECTRAL PROCESSING

Spectral processing is said to be *streaming* when time domain data is being windowed and transformed in a continuous fashion from an input signal (such as a realtime stream from an analogue-to-digital converter), producing a frequency domain signal of ordered frames at a given rate [10]. This is opposed to the case where all the input data is available for processing at once, and is more restrictive in terms of designing parallel implementations.

Typically, windows will be placed at a constant hopsize, and new output data is produced at a decimated rate, but there are also algorithms for sample-by-sample output, such as the sliding DFT[11], which is used in one of the cases studied. Thus, in the most common cases, we would only need to process spectral data at a reduced rate. This allows us to design a program that will use the GPU as a co-processor to compute the spectral data. The granularity of such process is then set to hopsize samples. This is the basic layout of the code discussed in the following sections.

3.1. Integration with Csound

The code discussed in this paper is hosted in Csound 6 as plugin opcodes (unit generators). Processing is done in vectors of *ksmps* samples, which can be set to any value above 1, with the upper value determined by the analysis hopsize in the case of the standard PV algorithms (no such limit applies to SPV). The PV analysis, synthesis and additive synthesis opcodes work with frequency domain signals (defined by the `fsig` Csound type), as well as the usual time-domain audio (and control signals). Thus, GPU processing is invoked every hopsize samples, in the case of these algorithms. The SPV implementation works solely with audio signals (as it packages analysis, transformation and synthesis in one single unit generator). It operates in fixed-size batches of 512 samples, which provide the best compromise in terms of performance and latency.

3.2. Phase Vocoder Analysis

The steps involved in PV Analysis are detailed in Figure 2 [12]. At the interval of hopsize samples, we window and rotate an input frame of time-domain data and apply a DFT to it. To obtain the PV data in a flexible amplitude + frequency (Hz) format, we then apply a conversion operation that takes the data from rectangular to polar representations and calculates the one-frame phase difference at each bin, then converts it from radians per hopsize samples to cycles per second.

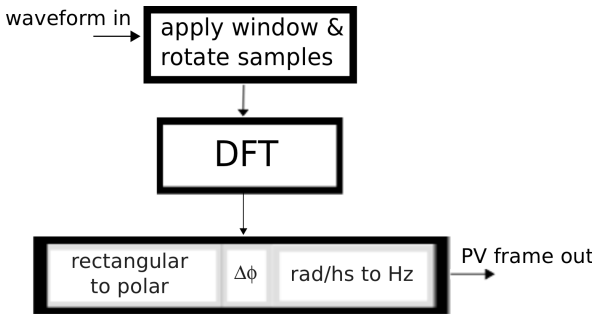


Figure 2: *PV analysis*

These three operations are good candidates for GPU co-processi as they can be parallelised. The window and rotation operations affect each sample separately, so they can be implemented in very simple kernels.

```
__global__ void
rotatewin(float* out, float* in, float *win,
       int N, int offset){
  int k = threadIdx.x +
          blockIdx.x*blockDim.x;
  out[(k+offset)%N] = win[k]*in[k];
}
```

In the code above `k` is the thread index, which is used to access a given sample in the input and output frames, `N` is the DFT size and `offset` is the rotation offset, that depends on the current frame index and the hopsize. The kernel is made transparent to deployment on any number of blocks, so that decision of how many threads per block can be made separately (or even dynamically).

Similarly, the conversion to PV parameters is eminently parallel, dealing with each bin separately. Since 0Hz does not need to be processed, we offset the thread index to start from bin 1. Computation is done in double precision, but PV data is stored as single-precision values following the convention for fsigs in Csound.

```
__device__ double modTwoPi(double x)
{
  x = fmod(x,TWOPI);
  return x <= -PI ? x + TWOPI :
    (x > PI ? x - TWOPI : x);
}

__global__ void
topvs(float* frame, double* oldph,
     double scal, double fac){
  int k = threadIdx.x +
       blockIdx.x*blockDim.x + 1;
  int i = k << 1;
  float re = frame[i], im = frame[i+1];
  float mag = sqrtf(re*re + im*im);
  double phi = atan2(im,re);
  double delta = phi - oldph[k-1];
  oldph[k-1] = phi;
  frame[i] =  mag;
  frame[i+1] = (float)
          ((modTwoPi(delta) + k*scal)*fac);
}
```

Finally, the DFT is implemented using the CUDAFFT library `cufftExecR2C()` function, which is optimised to run on the GPU hardware. Processing from real to complex data is in place, in the format expected by the windowing and rotation, and PV conversion kernels. This simplifies the memory handling, as there is only the need to copy the input data to the GPU and copy the PV output back to the host once. This is the minimum necessary data transfer to and from the device. The location of the frame, waveform, and phase history data is not defined in the kernel code, but in the current implementation, global device memory is used. Shared memory cannot be used for two reasons: in the case of waveform and frame data, it would require access to shared memory pointers by the host which is not available; and in the case of phase history, it would break the unit generator reentrancy condition.

In summary, we have the following sequential steps performed at each hopsize interval:

1. A frame of waveform samples is copied to the device
2. A kernel of N threads running `rotatewin()` is launched
3. DFT is performed with `cufftExecR2C()`
4. A kernel of N/2-1 threads running `topvs()` is launched.
5. A frame of amp + frequency data is copied from the device

Around this GPU-specific code, the host takes care of collecting the input samples into the waveform frames that will be sent to the device, as well as making the output frequency-domain signal available to the rest of Csound.

3.3. Phase Vocoder Synthesis

PV synthesis basically re-trace the steps of analysis in reverse (Figure 3). As before, we have three highly parallel steps. The conversion from PVS parameters into rectangular spectral data is provided by the following kernel:

```
__global__ void
frompvs(float* inframe, double* lastph,
    double scal, double fac) {
  int k = threadIdx.x +
    blockIdx.x*blockDim.x + 1;
  int i = k << 1;
  float mag = inframe[i];
  double delta = (inframe[i+1]
                          - k*scal)*fac;
  double phi = fmod(lastph[k-1]
                       + delta, TWOPI);
  lastph[k-1] = phi;
  inframe[i] =  (float) (mag*cos(phi));
  inframe[i+1] = (float) (mag*sin(phi));
}
```

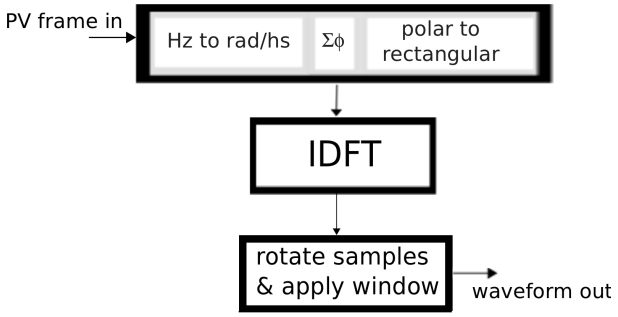


Figure 3: *PV synthesis*

Rotation and windowing is, as in the analysis case, very straightforward:

```
__global__ void
winrotate(float* out, float* in, float *win,
            int blocks, int N, int offset){
  int k = threadIdx.x +
       blockIdx.x*blockDim.x;
  out[k] = win[k]*in[(k+offset)%N];
}
```

The steps involved in PV synthesis are:

1. A frame of PV data is copied to the device
2. A kernel of N/2-1 threads running `frompvs()` is launched.
3. Inverse DFT is performed with `cufftExecC2R()`
4. A kernel of N threads running `winrotate()` is launched
5. A frame of waveform samples is copied from the device

3.4. Additive Synthesis

At face value, additive synthesis appears to be a very suitable technique for GPU implementation, given the fact that it is based on generating independently-computed sinusoidal streams and mixing them together. However, in practice there are some issues that need to be solved, namely

- a suitable sinusoidal oscillator design
- memory use/access

For full-spectrum synthesis using all the analysis data, we use two steps, both involving independent computations, which can be parallelised: an oscillator bank, and the parameter update.

In designing the oscillator that will be used we have to consider in particular the fact that conditionals are very costly in GPU code (as discussed above). A standard table lookup oscillator with floating-point indexing is not very efficient because of the conditional checks and moduli operations for index bounds. An alternative is to use integer indexing and with fast wrap-around using bitmasks. In addition, we have observed that table memory access has an inherent cost (even if the table is loaded to shared memory, which has the fastest access), and direct use of trigonometric functions is about 23% faster.

The most problematic issue with additive synthesis on the GPU is memory access, which can take up a significant amount of the total process time. Additive synthesis, in comparison to PV synthesis, can be relatively memory-hungry. For example, it requires a minimum of N $\times$ H floating-point numbers, where N is the number of bins and H the hopsize. For the full reconstruction of a 2048-point analysis (1024 bins), hopping by 256 samples, we have a 1MB memory requirement for single-precision samples. This memory would need to be accessed twice by the device: for writing by each oscillator, and for reading at a mixdown stage. In this particular case, memory access costs can amount to almost 70% of the total computation time. Reducing the hopsize does not mitigate the problem, because it will increase the number of times a given kernel executes. A solution is to use atomic additions, if these are available and sufficiently fast. In this case, the mix down of each sample can be at the time of the sample generation, and no further memory access is required. In the cases where atomic operations are costly, then we will need to write every partial to memory first, and, in a second sequential step, mix all of them down (in parallel). CUDA offers a very efficient atomic addition for float samples, so we can take advantage of it.

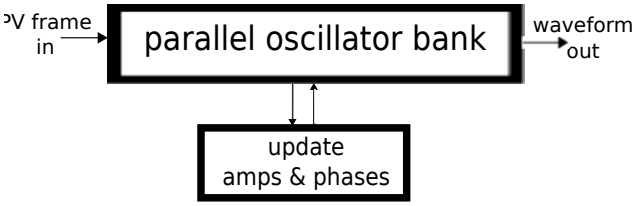


Figure 4: *Additive synthesis*

Of course, is always possible to synthesise a smaller number of bins, which would reduce both memory access and raw computation. In any case, each sample of each partial can be independently calculated (see [13]). For this we can spawn N $\times$ H kernels, each contributing a single sample to their respective partial, in effect parallelising across bins and samples. The additive synthesiser as implemented here is shown on Figure 4. The kernel used compute each sample is shown below:

```
#define MAXNDX ((MYFLT) 0x40000000)
#define PHMASK   0x3FFFFFFF

__global__ void sample(float *out,
  float *frame, float pitch,
  int64_t *ph, float *amps,
  int bins, int vsize, MYFLT sr) {
  int t = (threadIdx.x +
      blockIdx.x*blockDim.x);
  int n =  t%vsize;
  int h = t/vsize;
  int k = h<<1;
  int64_t lph;
  float a = amps[h], ascl = ((float)n)/vsize;
  MYFLT fscal = pitch*MAXNDX/sr;
  lph = (ph[h] + (int64_t)
   (n*round(frame[k+1]*fscal))) & PHMASK;
  a += ascl*(frame[k] - a);
  atomicAdd(&out[n],
           a*sinf((2*PI*lph)/FMAXLEN));
}
```

It takes in single-precision amplitudes and frequencies in a PV-format `frame`, which has been copied from the host into the device, and writes its output sine wave to `out`, For sake of efficiency, we interpolate amplitudes linearly, but frequencies only change at hopsize intervals. Output memory is accessed via an atomic addition. The layout of kernels with respect to bins and samples is shown in 5.

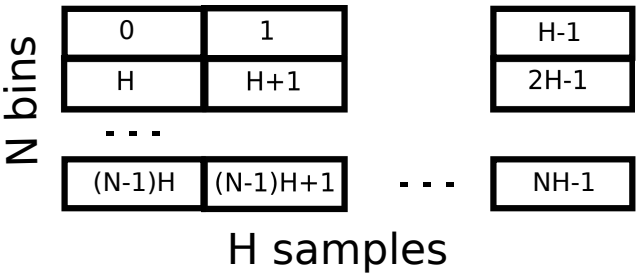


Figure 5: *Layout of kernels for additive synthesis*

The synthesis expression for each kernel is given by

$$s_{hn}(n) = \left\{a_h(t-1) + [a_h(t) - a_h(t-1)]\frac{n}{H}\right\} \times sin(\phi_h(t) + \omega_h(t)n) \quad (1)$$

where h and n are the bin and sample indexes, respectively, H is the hopsize, and t is the time in hopsize samples. The bin amplitudes are found in $a_h(t)$ and $\omega_h(t) = 2\pi\frac{f_h(t)}{f_s}$ are the bin frequencies (with f_s as the sampling rate and f_h the bin frequency in Hz).

A separate second step is needed to update the synthesis parameters for each bin (amplitudes and oscillator phases). The phases $\phi_h(t)$ are updated according to:

$$\phi_h(t+1) = \phi_h(t) + \omega_h(t)H \quad (2)$$

and the amplitudes are updated directly from the input PV frames. This operation is parallel across bins:

```
__global__ void update(float *frame,
     float *amps, int64_t *ph,
     int vsize, MYFLT sr){
```

```
 int h = threadIdx.x
    + blockIdx.x*blockDim.x;
 int k = h << 1, i;
 ph[h]  = (ph[h] + (int64_t)
    (vsize*round(pitch*frame[k+1]*MAXNDX/sr)))
     & PHMASK;
 amps[h] = frame[k];
}
```

The memory transfer from device to host that follows the execution of these kernels is limited to a hopsize vector of floating-point samples.

3.5. Sliding Phase Vocoder

If the hopsize is set to its smallest value, 1, the process can be seen in another way. The advantages and drawbacks are described elsewhere[6], but the algorithm is highly parallel. This can be seen as the extreme case for phase vocoding, but it offers possibilities for use of the GPU architecture.

3.5.1. The Underlying Mathematics

The Discrete Fourier Transform (DST) is defined by the formula

$$F_t(n) = \sum_{j=0}^{N-1} f_{j+t} e^{-2\pi i j n/N} \tag{3}$$

where the PCM-coded input signal is f_t, and $F_t(n)$ are the n frequency (complex) amplitudes for time t, and N is the (assumed) cyclic period of the signal.

If we know the values $F_t(n)$ we can determine $F_{t+1}(n)$:

$$F_{t+1}(n) = \sum_{k=0}^{N-1} f_{k+t+1} e^{-2\pi i k \frac{n}{N}} \tag{4}$$

$$= \sum_{k=1}^{N} f_{k+t} e^{-2\pi i (k-1) \frac{n}{N}} \tag{5}$$

$$= \left(\sum_{k=0}^{N-1} f_{k+t} e^{-2\pi i k \frac{n}{N}} - f_t + f_{t+N} \right) e^{2\pi i \frac{n}{N}} \tag{6}$$

$$= \left(F_t(n) - f_t + f_{t+N}\right) e^{2\pi i \frac{n}{N}} \tag{7}$$

If the values of $F_t(n)$ are kept on the GPU the need for data transfer is much reduced, but we can make use of the transfer of blocks of data for f_t and f_{t+N} at the expense of some latency.

There is however an immediate problem; the window cannot be applied in the time domain. The solution in this case is to apply the window as a frequency-domain convolution. That is to say, it can be applied after the calculation of the DFT as multiplication of the spectral transform of the window. Indeed for cosine-based windows this operation is simple[14].

In the paper by Moorer ([15]) a complicated inverse formula is developed. However it requires N^2 data to be maintained and is clearly impractical, especially on a memory-limited GPU. Instead we use a direct calculation of the definition of the inverse DFT:

$$f_t = \frac{1}{N} \sum_{n=0}^{N-1} F_t(n) e^{2\pi i t n/N} \tag{8}$$

but as we only need consider one value of t for each frame this is more efficient than the formula would suggest. For a single point $t = 0$ this simplifies to

$$\frac{1}{N} \sum_{j=0}^{N-1} F_0(j) \tag{9}$$

3.5.2. Implementation

A GPU-based Csound opcode was developed from the code in [16], where we take an audio input, apply a Transformational FM process[6] and resynthesise it. In this application, the sliding PV allows the unique effect of audio-rate frequency modulation of spectral data. The initialisation function is required to organise CUDA memory for the bin data and the pre-calculate a number of constants (*e.g.* $e^{2\pi i \frac{n}{N}}$). The main processing is done in small vectors of samples and it involves the following steps

1. A vector of samples is copied to the device
2. The sliding DFT is performed (on sample-by-sample basis), in parallel across bins by N/2+1 `slide()` threads (where N is the DFT size).
3. DFT to PV conversion, followed by frequency modification, and finally, PV to DFT conversion is performed by N/2+1 `fmsyn()` threads
4. Reconstruction is performed in parallel across the time-doma samples by vectorsize `reconstruct()` kernels.
5. A vector of samples is copied from the device to the host

4. RESULTS AND DISCUSSION

In this section, we would like to demonstrate that it is possible to execute all of the code discussed in realtime, with low-latency wherever possible, which was the original requirement that we have set out to prove. In testing these conditions, we employ a soundfile as input to the process, and make the requirement for realtime that computation time is less than the duration of input data. For a low-latency condition, we need to have the ratio between computation time and input duration fairly small so that any significant jitter in the computation load is not translated as dropped samples (also known as *xruns*). Tests included running the code to the digital-to-analog converter in realtime using buffer sizes of 128 frames (3ms at $f_s = 44100$) without xruns, which can also characterise a low-latency condition. Timings were taken from the total computation time recorded by Csound, which lumps the serial and parallel code, but since the interest here is the feasibility of the system as whole, this is exactly what we want to measure. The reported times are the average of five runs, but we have observed very little deviation in the individual results. Below, we discuss the individual results for each process.

4.1. PV analysis

The following Csound 6 code was used to test the PV analysis process. It consists of a soundfile input, the GPU-run analysis (`cudanal`) and a standard CPU-based PV synthesis (`pvsynth`, also used to provide a means to check the correctness of the output).

```
/* soundfile input */
asig = diskin:a("flutec3.wav",1,0,1)
/* GPU  PV analysis */
fsig = cudanal(asig,
               ifftsize,
               ihopsize,
               ifftsize, 1)
/* PV synthesis */
asig = pvsynth(fsig)
asig = linenr(asig,0.005,0.01,0.01)
out(asig)
```

Table 2: *GPU PV analysis program times for a 60-sec run.*

(DFT size, hopsize)	time (secs)
(1024, 128)	2.95
(1024, 256)	1.68
(2048, 256)	2.20
(2048, 512)	1.28

For this algorithm, we have observed that the best performance is obtained by maximising the number of threads in a block. Thus we distribute the threads so that they fill the blocks completely, up to the limit of 1024 threads. Since the number of threads is determined by the DFT size, we will be using single blocks for transforms of less than 1024 samples and multiple blocks above this.

The times for a 60-sec run of the program with various combinations of DFT size and hopsize are shown in table 2. This indicates that the best match in terms of performance is that of a 2048 transform every 512 samples. We can assess this as a generally efficient performance, with timings more than 20× faster than the realtime limit.

4.2. PV synthesis

Similarly to above, in order to isolate the performance of the synthesis process, we employ a program that uses an analysis element running in the CPU (`pvsanal`), followed by the GPU synthesis code (`cudasynth`):

```
/* soundfile input */
asig = diskin:a("flutec3.wav",1,0,1)
/* PV analysis */
fsig = pvsanal(asig,
               ifftsize,
               ihopsize,
               ifftsize, 1)
/* GPU PV synthesis */
asig = cudasynth(fsig)
asig = linenr(asig,0.005,0.01,0.01)
   out(asig)
```

Results are shown in table 3. They also indicate a reasonable performance, confirming the best combination of parameters obtained in the analysis tests.

4.3. PV analysis & synthesis

Also interesting is the combination of GPU analysis and synthesis, and following the results above, we can predict that they will be well within realtime capabilities. This is the program used (and the results are shown on Table 4):

Table 3: *GPU PV synthesis program times for a 60-sec run.*

(DFT size, hopsize)	time (secs)
(1024, 128)	3.30
(1024, 256)	1.84
(2048, 256)	2.65
(2048, 512)	1.44

```
/* soundfile input */
asig = diskin:a("flutec3.wav",1,0,1)
/* GPU PV analysis */
fsig = cudanal(asig,
               ifftsize,
               ihopsize,
               ifftsize, 1)
/* GPU PV synthesis */
asig = cudasynth(fsig)
asig = linenr(asig,0.005,0.01,0.01)
out(asig)
```

Table 4: *GPU PV analysis & synthesis program times for a 60-sec run.*

(DFT size, hopsize)	time (secs)
(1024, 128)	4.72
(1024, 256)	2.57
(2048, 256)	3.03
(2048, 512)	1.73
(4096, 512)	1.98
(4096, 1024)	1.20
(8192, 1024)	1.64
(8192, 2048)	1.01
(16384, 2048)	1.38
(16384, 4096)	0.86

These results demonstrate that a full PV analysis/synthesis program can be run quite efficiently on the GPU. However, if we compare it to PV code run sequentially in a high-performance CPU (in this case based on a 2.8GHZ Intel I7 processor), we see that it does not compare too well (Table 5) unless the DFT size is significantly large. Even though the scope of this research is not to draw comparisons between CPU and GPU capabilities for audio processing, it is important to note these results. The major shortcomings of the GPU for the particular processes discussed here are to do with the costs involved in launching kernels in a reasonably fine grain (determined by the hopsize), and memory access. If we examine the sequential steps involving the GPU in the analysis or synthesis code, we will see that these are the most costly portions of the code (with their average computation load):

- memory transfers: 40 - 45%
- FFT: 30 - 35%
- PV parameter conversion: 15 - 20%

None of these issues are particularly avoidable as they are not represented in sections of code that are good candidates for optimisation.

Table 5: *CPU-based PV analysis & synthesis program times for a 60-sec run.*

(DFT size, hopsize)	time (secs)
(1024, 128)	1.24
(1024, 256)	0.69
(2048, 256)	1.28
(2048, 512)	0.70
(4096, 512)	1.34
(4096, 1024)	0.74
(8192, 1024)	1.36
(8192, 2048)	0.75
(16384, 2048)	1.40
(16384, 4096)	0.77

Nevertheless, the results point to the fact that the GPU may be used as a means of freeing up some computation load from the CPU in a multicore/multiprocessor operation scenario. We also anticipate a considerable speedup on more capable hardware, where we are likely to see the GPU outperforming these CPU results.

4.4. Additive synthesis

In general, Additive synthesis does not perform as well as PV synthesis, and in the parallel case there is still a significant difference in performance between the two techniques, even if at times it is the gap is considerably less than in the serial case. The program used for tests is the following (`cudasynth` implements the additive algorithm):

```
asig = diskin:a("flutec3.wav",1,0,1)
fsig = pvsanal(asig, ifftsize, ihopsize,
                        ifftsize, 1)
asig = cudasynth(fsig,1,1,ibins)
asig = linenr(asig,0.001,0.01,0.01)
out(asig)
```

The results are shown in Table 6, with regards to bins and hopsizes (DFT sizes are shown for completion, but they do not influence computation load), with GPU and CPU times side-by-side. A highly optimised serial additive synthesis algorithm was used for this comparison, replacing the `cudasynth` opcode in the listing above.

Overall, the performance is still well within the range for low-latency realtime performance (< 12% of total input duration at the worst case), Comparatively, additive synthesis proves to be a good match for the GPU, especially in the case of full-spectrum reconstruction, and with larger hopsizes. The parallel code performs worse only in the case where the hopsize is small comparatively to the number of bins. This is mostly due to the fact that, in this case, the balance between the parallel load and the process granularity is not ideal. We can observe that this makes an important contribution to the computation cost. The granularity penalty is shown by comparing the cost of calling one grid of 65536 threads (256 bins, 256 hopsize) and two grids containing 32768 threads each (256, 128), where we observe almost 100% slowdown. This shows that GPUs are more suited to larger batches of data, which is not ideal in the streaming processing case.

Table 6: *GPU additive synthesis program times for a 60-sec run.*

(DFT size, bins, hopsize)	GPU time (secs)	CPU time (secs)
(1024, 128, 128)	4.93	3.28
(1024, 128, 256)	3.70	3.01
(1024, 256, 128)	7.20	5.77
(1024, 256, 256)	3.37	5.46
(1024, 512, 256)	4.20	10.76
(2048, 256, 512)	3.04	5.65
(2048, 512, 512)	3.94	10.55
(2048, 1024, 512)	6.87	20.89

These results are very encouraging, and follow other reports of additive synthesis, such as [13], but are not as extremely performant as one might anticipate (the best speed up is of the order of 3) . However the conditions in our case are much more restrictive than in other tests. We have implemented here a fully-flexible general-purpose application of additive synthesis, where we cannot run the processing in large batches, or apply other cost-saving measures that would maximise the GPU processing load. In particular, in order to keep latency and realtime control to a satisfactory minimum, as well as have good reconstruction quality, processing granularity is never bigger than 1/4 DFT size. We also should note that the results obtained in [17] are more in line with the ones reported in this paper.

4.5. Sliding PV

The sliding phase vocoder CUDA opcode (`cudasliding`) combines analysis, frequency scaling and resynthesis. It was tested with the Csound program

```
asig = diskin:a("flutec3.wav",1,0,1)
amod = 1
asig2 = cudasliding(asig,amod,idftsize)
asig = linenr(asig2,0.005,0.01,0.01)
out(asig)
```

and the performance compared with a similar program running solely on the host computer CPU.

Table 7: *GPU and CPU sliding PV program times for a 60-sec run.*

DFT size	GPU time (secs)	CPU time (secs)
512	33.05	68.794
1024	37.98	138.29
2048	54.99	272.33

The results are shown in Table 7. As can be seen that the times using the GPU are within real time, but considerably slower than the standard phase vocoder with GPU support (Table 4). The figures also are slower than reported by [16] on different hardware with more computing capacity. It also suggests that much more work will be needed if the Sliding Constant-Q algorithm[18] that

needs three streams of SDFT to be calculated is to be available for realtime on consumer-level GPUs.

5. CONCLUSIONS

In this paper, we set out to investigate the implementation of streaming spectral processing operations in a consumer-level GPU attached to an off-the-shelf desktop computer under a commonly-used music programming environment, Csound. The full CUDA source code for these unit generators, and a CMake build script, can be found in the Csound git repository:

https://github.com/csound/csound.git

These opcodes are fully integrated into the standard system and are included in the present release (6.03, April 2014).

We have demonstrated that each one of the processes detailed here can be executed in realtime with low latency. The standard algorithms can all generally be executed with good performance, and, among these, additive synthesis is comparatively less efficient, although the parallel version generally outperforms the serial one. With the novel SPV process, we see significant gains, with up to 5× speedup, where the improvements allow the code to be used in realtime. We have identified that the major costs are related to memory transfers from host to device and vice-versa, and device memory access. We believe that this work demonstrates that consumer-level GPU processing can be harnessed for audio applications. In particular a number of novel digital audio effects can be designed to take advantage of the GPU implementation for realtime performance.

6. REFERENCES

[1] L. Savioja, V. Valimaki, and J. O. Smith, "Audio signal processing using graphics processing units," *J. Audio Eng. Soc.*, vol. 59, no. 1, pp. 3–19, 2011.

[2] Niklas Roeber, Ulrich Kaminski, and Maic Masuch, "Ray Acoustics Using Computer Graphics Technology," in *10th Proc. Digital Audio Effects (DAFx-2007)*, Bordeaux, France, 2007.

[3] L Savioja, "Use of GPUs in room acoustic modeling and auralization," in *Proceedings of the International Symposium on Room Acoustics (ISRA 2010)*, Melbourne, Australia, 2010.

[4] P-Y Tsai, T-W Wang, and Su Alvin, "GPU-based Spectral Model Synthesis for Real-time Sound Rendering," in *13th Proc. Digital Audio Effects (DAFx-2010)*, Graz, Austria, 2010.

[5] B Hamilton and C Webb, "Room Acoustics Modelling Using GPU-Accelerated Finite Difference and Finite Volume Methods on a Face-Centered Cubic Grid," in *16th Proc. Digital Audio Effects (DAFx-2013)*, Maynooth, Ireland, 2013.

[6] Russell Bradford, Richard Dobson, and John ffitch, "The Sliding Phase Vocoder," in *Proceedings of the 2007 International Computer Music Conference*, Suvisoft Oy Ltd, Ed. August 2007, vol. II, pp. 449–452, ICMA and Re:New, ISBN 0-9713192-5-1.

[7] "CUDA C Programming Guide," `http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf`, 2014.

[8] Richard J. Boulanger, Ed., *The Csound Book: Tutorials in Software Synthesis and Sound Design*, MIT Press, February 2000.

[9] J. Fitch, V. Lazzarini, S. Yi, Gogins M., and A Cabrera, "The New Developments in Csound 6," in *Proc. Intl. Computer Music Conf.*, Perth, Australia, 2013.

[10] V Lazzarini, J Timoney, and T. Lysaght, "Spectral Signal Processing in Csound 5," in *Proc. Intl. Computer Music Conf.*, New Orleans, USA, 2006.

[11] Russell Bradford, Richard Dobson, and John ffitch, "Sliding is Smoother than Jumping," in *ICMC 2005 free sound*, SuviSoft Oy Ltd, Tampere, Finland, Ed. Escola Superior de Música de Catalunya, 2005, pp. 287–290, `http://www.cs.bath.ac.uk/~jpff/PAPERS/BradfordDobsonffitch05.pdf`.

[12] V. Lazzarini, *The Audio Programming Book*, chapter The Phase Vocoder, pp. 91–122, MIT Press, Cambridge, Massachusets, 2010.

[13] L. Savioja, V. Valimaki, and J. Smith, "Real-time additive synthesis with one million sinusoids using a gpu," in *Proc. 128th AES*, London, UK, 2010.

[14] John ffitch, Richard Dobson, and Russell Bradford, "Sliding DFT for Fun and Musical Profit," in *6th International Linux Audio Conference*, Frank Barknecht and Martin Rumori, Eds., Kunsthochscule für Medien Köln, March 2008, LAC2008, pp. 118–124, Tribun EU, Gorkeho 41, Bruno 602 00, ISBN 978-80-7399-362-7.

[15] James A. Moorer, "Audio in the New Millennium," *J. Audio Eng. Soc.*, vol. 48, no. 5, pp. 490–498, May 2000.

[16] Russell Bradford, John ffitch, and Richard Dobson, "Real-time Sliding Phase Vocoder using a Commodity GPU," in *Proceedings of ICMC2011*. University of Huddersfield and ICMA, August 2011, ICMC, pp. 587–590, ISBN 978-0-9845274-0-3.

[17] A. J. Bianchi and M. Queiroz, "MEASURING THE PERFORMANCE OF REALTIME DSP USING PURE DATA AND GPU," in *Proc. Intl. Computer Music Conf.*, Ljubljana, Slovenia, 2012.

[18] Russell Bradford, Richard Dobson, and John ffitch, "Sliding with a Constant *Q*," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sep 1-4 2008, DAFx08, pp. 363–369, ISBN 978-951-22-9517-3.

A TWO LEVEL MONTAGE APPROACH TO SOUND TEXTURE SYNTHESIS WITH TREATMENT OF UNIQUE EVENTS

Seán O'Leary,
UMR STMS IRCAM - CNRS - UPMC
Paris, France
sean.oleary@ircam.fr

Axel Röbel,
UMR STMS IRCAM - CNRS - UPMC
Paris, France
axel.roebel@ircam.fr

ABSTRACT

In this paper a novel algorithm for sound texture synthesis is presented. The goal of this algorithm is to produce new examples of a given sampled texture, the synthesized textures being of any desired duration. The algorithm is based on a montage approach to synthesis in that the synthesized texture is made up of pieces of the original sample concatenated together in a new sequence. This montage approach preserves both the high level evolution and low level detail of the original texture. Included in the algorithm is a measure of uniqueness, which can be used for the identification of regions in the original texture containing events that are atypical of the texture, and hence avoid their unnatural repetition at the synthesis stage.

1. INTRODUCTION

Sound textures are a class of sounds typically associated with the background of a scene that are somehow repetitive; for example rain, fire, or machinery. It is difficult to define precisely the properties of sound textures. Saint-Arnaud and Popat [1] offered some suggestions towards a working definition. They suggest that sound textures should, in some sense 'exhibit similar characteristics over time'; that is that one short snippet of a texture should exhibit similarities to another. They also suggest a two level description of textures. At the low level atoms of the texture are time localized sound elements, and the higher level describes the distribution of these atoms. They note that while such an atomic model is sometimes relevant to the physics of the texture, e.g. rain, they do not intend it as a general physical description. They give some points summarizing their working definition of sound textures.

1. Sound textures are formed of basic sound elements, or atoms.
2. Atoms occur according to a higher-level pattern, which can be periodic, random, or both.
3. The high-level characteristics must remain the same over long time periods (which implies that there can be no complex message).
4. The high-level pattern must be completely exposed within a few seconds attention span.
5. High-level randomness is also acceptable, as long as there are enough occurrences within the attention span to make a good example of the random properties.

McDermott et al [2, 3] suggest that given the temporal homogeneity of sound textures they can be characterized by time averaged statistics. This approach was inspired by previous work on image textures [4]. This hypothesis was tested by synthesizing various textures by imposing the statistics of a particular texture on a white noise sample. The statistics used described the amplitude envelopes of the textures after being passed through an auditory filterbank. These statistics included the first four moments of the envelopes, cross correlation between envelopes, and some measures relating to the autocorrelation of each envelope. The resulting synthesized sounds were not intended to be perceptually accurate reproductions, rather they were meant to test their hypothesis. They found that the synthesized sound textures could indeed be identified.

These studies give important insights into the requirements of a synthesis algorithm. There are many approaches to sound texture synthesis (for a thorough review of the literature see [5]). Broadly speaking, we can group these methods into model based approaches where the signal is synthesized from model parameters, and sampling or granular approaches where content from the original signal is used in the synthesized signal.

For many applications, such as cinema and computer games, realism of the synthesized sound is paramount. Sampling based methods can bring realism as they contain elements of the target sound. Some previous sampling based algorithms [6, 7] look for points of change to segment texture signals, these segments are then concatenated in a probabilistically determined sequence to produce the synthesized texture. The algorithm of Dubnov et al. [8] uses similarity in history and scale to select sampled wavelet coefficients. Drawbacks of sampling based methods include repetitions of parts of the original signal, difficulty modeling the higher level structure of the texture, and smooth concatenation of the sampled elements.

The proposed algorithm falls into the sampling based category. It looks to exploit regions of similarity in the original texture to inform the sequencing of sampled elements. There are two levels to the synthesis model. Longer term sections, called segments, are used to model the higher level structure of textures. These segments are synthesized from the concatenation of shorter term sections, called atoms. Atoms preserve the local structure of the texture. The sequences of both the segments and atoms are modeled probabilistically, this avoids repetition in the synthesized texture. A new overlap add method is introduced for concatenation. This enables concatenation with short overlap without introducing perceptible modulations.

The paper is organized as follows: Section 2 discusses the relationship of the algorithm to the properties of sound textures outlined in section 1. Section 3 presents the basic algorithm in detail while section 4 extends the algorithm to deal with unique events. Section 5 presents some sound examples. Section 6 presents some conclusions and possible future work.

2. THE RELATIONSHIP OF THE MONTAGE APPROACH TO SOUND TEXTURE PROPERTIES

As stated in the introduction, the montage approach to texture synthesis has two levels; segments and atoms. Segments are used to model the high level structure of the texture. By high level structure we mean features that determine the long term structure of a texture such as quasi-periodicity (e.g. pneumatic drill) or randomness (e.g fire). At the lower level atoms preserve the local structure of the segments.

Segments are modeled after longer sections of the texture. There is not a set length for a segment, rather they have user defined minimum and maximum lengths. The length of each segment is dependent on the selection of its successor. The sequencing of segments is informed by both local similarity for concatenation, and longer term similarity for preserving higher level structure. This sequencing has a probabilistic element to avoid repetition in the higher level structure of the synthesized texture.

These segments are used as templates for the synthesized texture. The segments are synthesized by a process of atom substitution. The original texture is split into atoms. These atoms all have the same user defined duration. For each atom a number of candidates are selected as possible replacements. These candidates are selected from throughout the texture based on the local similarity of the envelopes from an auditory filterbank analysis. This 'envelope matching' preserves the phase of envelope modulations in the synthesized texture. The synthesis of segments consists of substituting each of the original atoms with one of its qualifying candidates (including itself as one of the candidates). The selection of substitutes is probabilistic. This process preserves local structure and introduces new variation over the duration of the segment not present in the original texture. This is to avoid repetition on the atom scale in the synthesized texture.

The algorithm can be considered in terms of the properties of sound textures suggested by Saint-Arnaud and Popat [1] quoted in section 1.

- The presented model synthesizes textures from atoms.
- The high-level pattern of the atoms is preserved by sequencing them according to segments of the original texture. If there is periodicity in the texture it can be reproduced because the atoms will be aligned according to the original texture, this effectively matches the phase of the envelopes. Likewise randomness is maintained by randomizing both the selection of segments from the candidate successors and the choice of atom from the candidates for substitution.
- New high level structure will be introduced due to the sequencing of segments. As the long-term similarity of segments are matched this new structure should be coherent with the original texture.

The algorithm can also be considered in terms of the statistical description of the envelopes suggested by McDermott in [3].

- If the segments are distributed approximately evenly over the duration of the synthesis the moments of the envelopes will be approximately equal to those of the original.
- As the atoms are sampled from the original texture the local synchronicity of the envelope modulations is preserved. This is related to cross correlation of the envelopes in McDermott's texture model.
- The matching of atoms over localized time and frequency, the sequencing of atoms from segments of the original, and the transitions based on history all relate to the autocorrelation of the envelopes; the atom sequencing preserving local modulations and the segment sequencing preserving/synthesi longer term modulations.

3. THE ALGORITHM

In this section the algorithm for analysis and synthesis of textures using the proposed approach is described. After the analysis phase the choices for synthesis are tabulated; all possible segments have candidates for their successors and each atom of the texture has candidates for substitution.

3.1. Analysis

The analysis stage of the montage approach involves finding regions in the texture that are in some way similar - this is necessary both for the selection of candidates for segment succession and the selection of candidates for atom substitution. The first step in the analysis is to represent the signal in a suitable form. As ultimately we are concerned with the perceptual closeness of the synthesized signal to the original a perceptually informed representation of the signal is utilized.

As much of the salient information in textures is contained in the envelopes of the auditory bands [3], a suitable comparison for similarity is taken to be a comparison of the time evolving energy from an auditory filter bank. The short time Fourier transform is a common and suitable processing platform, and so the algorithm will be presented in the context of the STFT.

The STFT is given by:

$$X\left(l,k\right)=\sum_{n=0}^{N-1}x\left(n\right)\omega\left(n-lh\right)e^{\frac{-i2\pi nk}{N}}. \tag{1}$$

Where l is the frame number, k is the frequency bin, N is the analysis window length and h is the hopsize.

Taking the envelopes to be the energies in subbands distributed according to the ERB scale:

$$engEnv_b\left(l\right)=\sum_{k=k_{b1}}^{k_{b2}}\left|X\left(l,k\right)\right|^2H_b\left(k\right). \tag{2}$$

Where k_{b1} is the first bin and k_{b2} is the last bin of the bth band and H is a bank of (frequency domain) band pass filters.

The envelopes then undergo further perceptual processing. The perceived change in loudness with intensity approximately obeys a power law. Hence the envelopes are compressed nonlinearly to simulate this. Each band is also scaled according to the equal loudness curve.

$$env_b=\left(engEnv_b/L\left(f_b\right)\right)^{0.3}. \tag{3}$$

Where L is the loudness curve, f_b is the centre frequency of the bth band and 0.3 is an experimentally determined exponent [9].

This gives a perceptually informed time/frequency representation of the signal sampled at the rate of the STFT analysis. Here we will refer to each time slice of both the STFT and the perceptually processed STFT as a frame.

The next stage in the analysis divides this representation of the signal into atoms. Each atom comprises several analysis frames

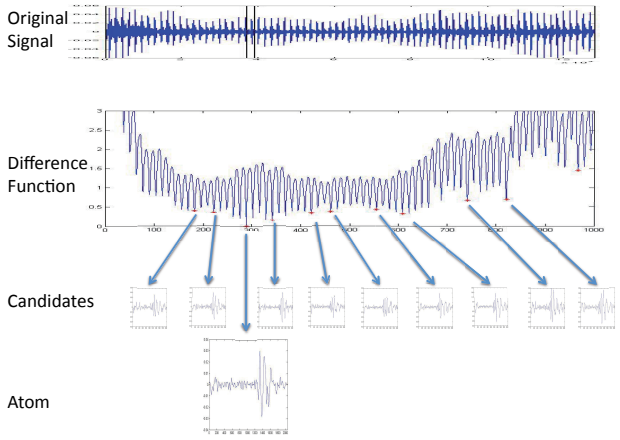


Figure 1: Selection of candidates for an atom

and have a 50% overlap with neighboring atoms. The atoms should be long enough to enable the comparison of envelopes and short enough to ensure enough variation in the synthesized texture. In our example set [10] we use $0.1s$ as the atom duration. This gives us a time/frequency representation of each atom. Each of these atoms undergoes further analysis; looking for similar regions over the duration of the texture.

3.1.1. Candidates for Atom Substitution

For each atom a difference function is created. This difference function gives us a measure of the difference between the atom under consideration and the associated region of the texture. The difference function for the ath atom at the lth frame is given by:

$$d_a(l) = \frac{\sqrt{\sum_{f=0}^{F-1}\sum_{b=1}^{B}\{env_b(l+f) - env_b(aF/2+f)\}^2}}{\sqrt{\sum_{f=0}^{F-1}\sum_{b=1}^{B}\{env_b(l+f)\}^2}}. \quad (4)$$

Where F is the number of frames in an atom and the atoms have a 50% overlap, i.e. an atom hopsize of $F/2$. This difference function is calculated at intervals of a single frame. This difference measure corresponds to the normalized euclidean distance between the auditory envelopes of the atom and the auditory envelopes of the texture from the lth to $l+F-1th$ frame.

A set of substitution candidates for each atom is selected from local minima in the difference function. There is a minimum time distance between selected candidates, dependent on the number of candidates to be selected. This is to ensure that candidates are selected from across the duration of the analyzed texture. This can be important for the selection of segments successors, as the candidates for substitution will also be considered as candidates for segment successors, and for this purpose it is desirable to have candidates spread over the duration of the texture.

An example of a difference function and candidates for a single atom of a texture are shown in Figure 1 for a helicopter sample (available to listen to at [10]). This is a quasi periodic texture, and this example illustrates how periodicity of events can be preserved with this model. Note that the envelope of the candidates is in phase with the envelope of the original atom. It is not necessary to retain the difference function after the analysis of an atom. Once the candidates for substitution are tabulated the difference function can be discarded. The result of the atom analysis is a list of pointers to the addresses in the original STFT of candidates for substitution and a normalized difference value for each of the candidates.

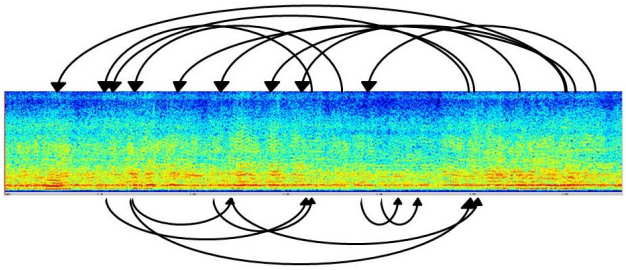


Figure 2: Transition from reading one segment to starting another

3.1.2. Candidates for Segment Successors

During synthesis segment succession occurs by substituting the last atom of the current segment with the beginning of its successor. And so each atom will be considered as a potential end of a segment and its candidates for substitution as a potential beginning for a succeeding segment. For segments, as well as the local similarity from the atom analysis, a longer term comparison is used. This is termed the history for the segment. Hence, a history comparison is also made between each atom and its candidates for substitution. This will be used to judge the possibility of a segment succession at the location of the atom during synthesis. No difference function is created as the history is only calculated for already found atom candidates.

As each segment has a minimum and maximum duration, the succeeding segment will begin between these points (see section 3.2.1). And so in the analysis phase each atom in this range is considered as a possible transition point from the current segment to its successor. This can be considered as a moving window analysis, the window length being the maximum minus the minimum duration of a segment. An example of a subset of possible segment succession points found for a texture is illustrated in Figure 2. For each step in this analysis there are typically many candidates. For example, for a single instance of this analysis if the difference between the minimum and maximum length between transitions is 1.5 seconds, and there are 20 atoms per second and 10 candidates per atom then there are 300 candidate points to consider as possible transition points. Only the succession points with the lowest measured difference are considered, again the selected succession points spanning the duration of the texture. The outcome of the succession analysis is a table of pointers for candidate segment end points for the current segment, associated difference values, and associated starting points for the next segment.

3.2. Synthesis

During synthesis the segment sequence is selected. From this the sequence of atoms is derived. These atoms are concatenated in the STFT domain before inverse Fourier transform and final overlap/add in the time domain are performed.

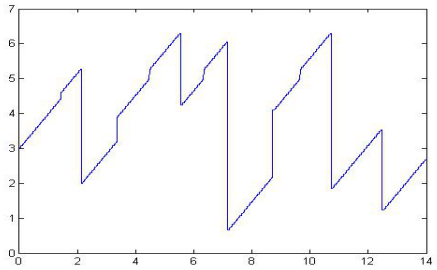


Figure 3: Sequence of transition in synthesized texture vs origin in original texture.

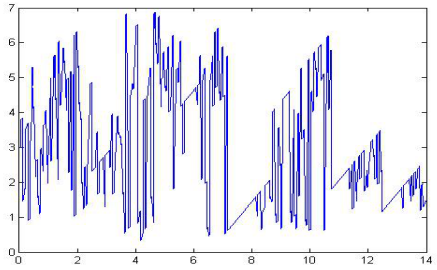


Figure 4: Sequence of atoms in synthesized texture vs origin in original texture.

3.2.1. Sequence Model

Starting from a random point in the original texture the algorithm selects successive segments from the candidates selected during analysis. This high level navigation of the texture acts as a template for the synthesized texture. There are some constraints on the choice of succeeding segments:

1. A segment must be at least a minimum, user defined, length.
2. A segment has a maximum, user defined, length.
3. If the succeeding segment occurs some time before the current segment in the original texture that time must be greater than a user defined minimum (at least equal to the length of the transition 'history').

The first constraint serves two functions: it prevents the synthesized texture from jumping too much and it allows the candidates for succeeding segments to be selected in the analysis phase. The second constraint prevents keeping the same high level structure as the original for long periods. The third prevents repeating parts of the high level structure in rapid succession.

Once a segment successor is selected the duration of the current segment is determined. The atoms for this segment can then be substituted probabilistically with the candidates selected during analysis, each of the qualifying candidates given equal probability of selection. A difference threshold can be used in the selection of atom substitutes. This defines the maximum difference allowed between atoms and possible switches. It was found that taking the median value of the normalized difference of all the candidates for all the atoms was an effective value for thresholding. An example of the sequencing of transitions and substitutions is illustrated in Figure 3 and 4.

The process of segment succession and atom substitution can continue for any desired period of time, producing varied textures which are perceptually similar to the original.

3.2.2. Overlap Add Operation

If we see the atoms as pieces of a jigsaw, the overlap-add operation can be seen to be a way of squeezing in pieces similar to the original into their place. Straightforward overlap-adding of broad band noise leads to modulations due to phase interference. Here a new solution to this problem is proposed. The cross fade of the atoms is done in the STFT domain. The number of frames involved in the cross fade is dependent on the bin number of the DFT (i.e. it is frequency dependent). The cross fade region is taken to be 4 times the inverse of the bin center frequency (i.e. 4 times the period), with a maximum of half the number of frames in an atom and a minimum of a single STFT frame. For bins with an overlap region less than half an atom length the point of maximum cross fade (i.e. 50%) is positioned at the point of least interference. This point is taken to be the point at which the absolute value of the complex difference in the overlap region is minimum.

4. DEALING WITH UNIQUE EVENTS IN THE TEXTURE

Often sampled textures contain local events that are uncharacteristic of the long term texture. Such events can be due to a recording artifact, an unwanted event in the recording, or a unique local event that is part of the process creating the texture. At the synthesis stage it may be desirable to avoid using atoms that contain such unique events as their repetition may be noticeable and artificial sounding in the synthesized texture; highlighting the sampling process and losing the naturalness of the synthesized texture.

Strobl [11] in a study of the concatenative algorithms of [6] and [7] refers to such events as 'disturbing elements', and proposes to identify them manually. Here we propose a method for identifying such elements that is a straightforward and natural extension to the montage approach.

There are two basic steps to this algorithm; 1) identify the unique region and 2) replace it with a qualifying piece of the texture. The replacement step allows the synthesis algorithm described above to remain unchanged.

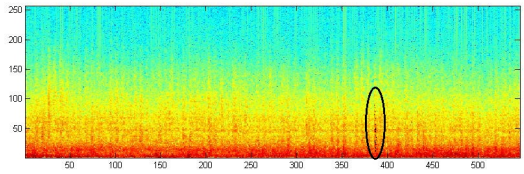
To identify events the difference measure obtained from 4 is utilized as a measure of the uniqueness of atoms. After the initial analysis stage each atom has a number of its closest matches from throughout the texture. The difference between an atom and its best match is taken to be a measure of its uniqueness.

A user defined parameter defines which atoms are to be replaced. This user parameter is a threshold and is stated as percentage of the maximum uniqueness found for the analyzed texture.

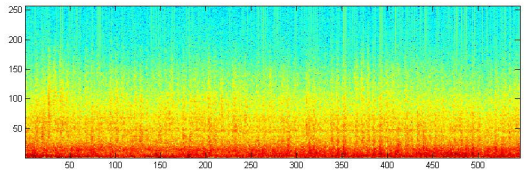
In order to find a region to replace the region selected as unique we again use the difference value defined by 4. Here we use a sum of the difference functions for the atoms adjacent to the selected region. The difference function of the latter atom in the sum is delayed by the appropriate time:

$$d_u(l) = d_{a1}(l) + d_{a2}(l + (w_u + 1)F/2). \quad (5)$$

Where $d_u(l)$ is the difference function used for finding the best match for the uth unique region, d_{a1} is the difference function for the $a1$th atom (the adjacent atom previous to the uth region) and d_{a2} is the difference function for the $a2$th atom (the adjacent atom following the uth region). The minimum of this function

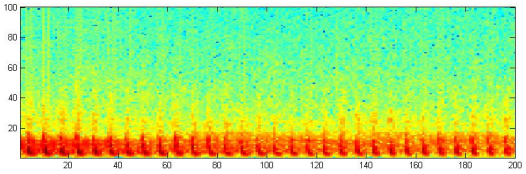


(a) Original with click highlighted.

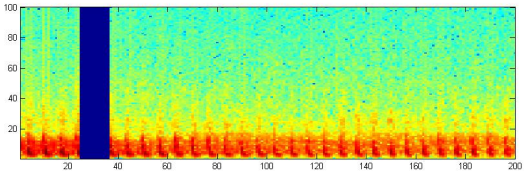


(b) Click identified and replaced.

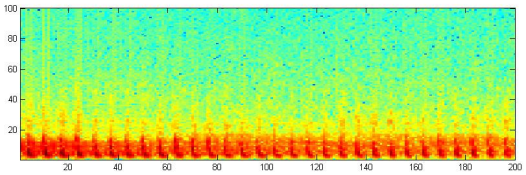
Figure 5: Example of identifying and replacing unique elements in a sampled texture.



(a) Original.



(b) With missing piece.



(c) Missing piece replaced.

Figure 6: Example of replacing a missing or damaged piece of a sampled texture (quasi-periodic helicopter).

gives the closest matching region, according to our measure, to replace the region identified as being unique. Once this region is identified the analysis described in section 3.1.1 is performed for the replacement atoms. Also, reference to the replaced atoms as substitutes for other atoms should be removed. Synthesis can then be performed as described in section 3.2.

An example of this is shown in figure 5. This is a recording of a steam train which contains a single 'click' sound. This 'click' is identified as the most unique region in the texture. For synthesis it is replaced as described above. This method can also be used to repair damaged recordings as is illustrated in figure 6. This illustration shows the spectrograms of helicopter sample, the same sample with a piece deleted, and the sample with the deleted piece replaced using the above method. Note how the approximate period of the events is preserved. The samples used to illustrate this are available at [10].

5. RESULTS

The presented algorithm was used to synthesize both textures containing quasi periodic elements and textures of a more random nature. The synthesized samples are twice the duration of the originals. The original samples were taken from [3]. The details of the synthesis for these sounds are as follows: the atom length was set to 0.1 seconds, the history set to 0.5 secs, and the maximum duration before a new transition set to 2 seconds. 20 candidates were selected for each atom, and 5 candidates selected for each transition. The transition candidates were selected by a simple sum of the normalized distance of the atom (local) difference and difference in histories. These examples can be found at [10].

6. CONCLUSIONS

In this paper an efficient and versatile algorithm for sound texture synthesis was presented. For efficient synthesis the atom and transition candidates can be tabulated from the analysis phase. Synthesis is then a fairly straightforward overlap add procedure in the STFT domain. The algorithm fulfills many requirements of a sound texture synthesis algorithm. At the low level the textures are synthesized from atoms and these atoms are sequenced to model the higher level organization of the original sound texture. Repetitions are avoided by introducing randomness in the sequencing of both the atoms and the segments, and smooth transitions are constructed by taking account of local similarity, longer history and a new overlap/add method.

While there are a number of user defined parameters in this algorithm, these parameters are not abstract, they have a natural relationship with the synthesis.

For the atom analysis the STFT hopsize determines the temporal resolution of the atom analysis, while the atom duration and difference threshold for substitution affect the variation of the timbre of the texture. For the segment sequencing the history length defines the region in which to compare the context of the high level structure, while the minimum and maximum length determine the high level variation.

The synthesis examples ([10]) show that for a large class of textures the synthesis is not extremely sensitive to these parameters. However, if there are extended events or a lot of variation in the original texture it may be beneficial to constrain the variation in the synthesized texture, i.e. lower the difference threshold for atom substitution and extend the history and minimum segment length.

As well as texture synthesis the algorithm has applications to editing textures, such as removing unique events or damaged portions of a sampled texture. The results seem very promising for a wide range of textures; from quasi periodic to random processes.

7. ACKNOWLEDGEMENTS

This research was funded by the French National Research Agency (ANR) as part of the PHYSIS project.

8. REFERENCES

[1] Nicolas Saint-arnaud and Kris Popat, "Analysis and synthesis of sound textures," in *in Readings in Computational Auditory Scene Analysis*, 1995, pp. 125–131.

[2] J H McDermott, A J Oxenham, and E P Simoncelli, "Sound texture synthesis via filter statistics," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA-09)*, New Paltz, NY, Oct 18-21 2009, pp. 297–300, IEEE Signal Processing Society.

[3] J H McDermott and E P Simoncelli, "Sound texture perception via statistics of the auditory periphery: Evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, Sep 2011.

[4] J Portilla and E P Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int'l Journal of Computer Vision*, vol. 40, no. 1, pp. 49–71, December 2000.

[5] Diemo Schwarz, "State of the art in sound texture synthesis," in *Intl. Conf. on Digital Audio Effects (DAFx-11)*, Paris, France, 2011.

[6] R. Hoskinson and D. Pai, "Manipulation and resynthesis with natural grains," in *Proceedings of the International Computer Music Conference*, Havana, Cuba, 2001.

[7] Lie Lu, Liu Wenyin, and Hong-Jiang Zhang, "Audio textures: theory and applications," *Speech and Audio Processing, IEEE Transactions on*, vol. 12, no. 2, pp. 156–167, 2004.

[8] Shlomo Dubnov, Ziv Bar-joseph, Ran El-Yaniv, Dani Lischinski, and Michael Werman, "Synthesis of sound textures by learning and resampling of wavelet trees," 2002.

[9] William M. Hartmann, *Signals, Sound, and Sensation*, Springer, New York, 1998.

[10] Seán O'Leary, "Sound examples," `http://anasynth.ircam.fr/home/english/media/montage-sound-texture-synthesis`.

[11] G. Strobl, "Parametric sound texture generator," M.S. thesis, Technische Universitääat Graz, Austria, 2007.

FAST SIGNAL RECONSTRUCTION FROM MAGNITUDE SPECTROGRAM OF CONTINUOUS WAVELET TRANSFORM BASED ON SPECTROGRAM CONSISTENCY

Tomohiko Nakamura[†] *and Hirokazu Kameoka*[†‡]

[†]Graduate School of Information Science and Technology, The University of Tokyo,
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan.
[‡]NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation,
3-1, Morinosato Wakamiya, Atsugi, Kanagawa, 243-0198, Japan
nakamura@hil.t.u-tokyo.ac.jp, kameoka@hil.t.u-tokyo.ac.jp

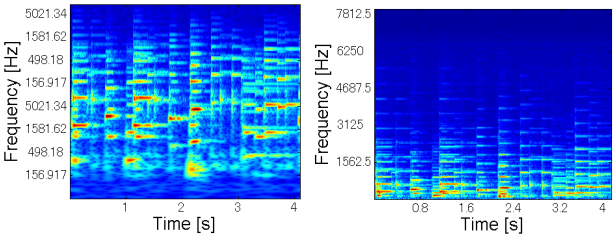
ABSTRACT

The continuous wavelet transform (CWT) can be seen as a filterbank having logarithmic frequency subbands spacing similar to the human auditory system. Thus, to make computers imitate the significant functions of the human auditory system, one promising approach would be to model, analyze and process magnitude spectrograms given by the CWT. To realize this approach, we must be able to convert a processed or modified magnitude CWT spectrogram, which contains no information about the phase, into a time domain signal specifically for those applications in which the aim is to generate audio signals. To this end, this paper proposes a fast algorithm for estimating the phase from a given magnitude CWT spectrogram to reconstruct an audio signal. The experimental results revealed that the proposed algorithm was around 100 times faster than a conventional algorithm, while the reconstructed signals obtained with the proposed algorithm had almost the same audio quality as those obtained with the previous study.

1. INTRODUCTION

The continuous wavelet transform (CWT), also known as the constant-Q transform, is used as a method for time-frequency analysis, which provides a time-frequency representation of a signal with an equal resolution on a log-frequency scale (Fig. 1). The human auditory filterbank is known to have an equal resolution on a log-frequency scale as with the CWT particularly in a high frequency band [1, 2]. Thus, to let computers imitate the significant functions of the human auditory system, one promising approach would be to model, analyze and process spectrograms obtained by the CWT (*CWT spectrogram*). In fact, recent studies (see [3–6]) have shown that multiple fundamental frequency estimation performs very well in the magnitude CWT spectrogram domain. Motivated by this fact, we believe that source separation and sound manipulation can also work well in the magnitude CWT spectrogram domain. However, in order to achieve source separation or sound manipulation, in which the goal is to produce sound, there is a need to reconstruct an appropriate time-domain signal after processing and modifying a magnitude CWT spectrogram. To this end, this paper proposes a method for estimating the phase from a given magnitude CWT spectrogram to reconstruct an audio signal.

The phase estimation algorithm from a magnitude CWT spectrogram has already been proposed by Irino *et al.* [7]. Irino's algorithm consists in iteratively performing the inverse CWT and the CWT followed by replacing the modified magnitude CWT spectrogram with a given magnitude CWT spectrogram. Since the computational speed of the CWT is much slower than the short-time Fourier transform (STFT), this algorithm needs a very long time for computation. In practical situations, the reduction of the computational complexity can be extremely important.



(a) CWT spectrogram. (b) STFT spectrogram.

Figure 1: Examples of the continuous wavelet transform (CWT) and short-time Fourier transform (STFT) spectrograms. While STFT spectrograms have an equal resolution on a linear frequency scale, CWT spectrograms have an equal resolution on a log-frequency scale.

The authors and colleagues have thus far proposed a fast method for estimating the phase from a magnitude STFT spectrogram [8]. When the hop-size is shorter than the frame length, the waveforms in the overlapping segment of consecutive frames must be consistent. This implies the fact that an STFT spectrogram is a redundant representation. Thus, an STFT spectrogram must satisfy a certain condition to ensure that it is associated with a time domain signal. We have referred to this condition as *the consistency condition*. In [8], we have shown that the problem of estimating the phase from a magnitude STFT spectrogram can be formulated as the problem of optimizing the consistency criterion describing how far an arbitrary complex array deviates from this condition.

It became clear that the deviced algorithm is equivalent to the well-known algorithm proposed by Griffin *et al.*, [9]. The formulation derived from the concept of the spectrogram consistency has provided a new insight into the Griffin's algorithm, allowing us to introduce a fast approximate algorithm and give a very intuitive proof of the convergence of the algorithm. Since a CWT spectrogram is also a redundant representation of a signal [10], we may be able to make the best use of the spectrogram consistency concept to develop a fast approximate method for phase estimation from a magnitude CWT spectrogram.

Following the idea proposed in [8], this paper derives an algo-

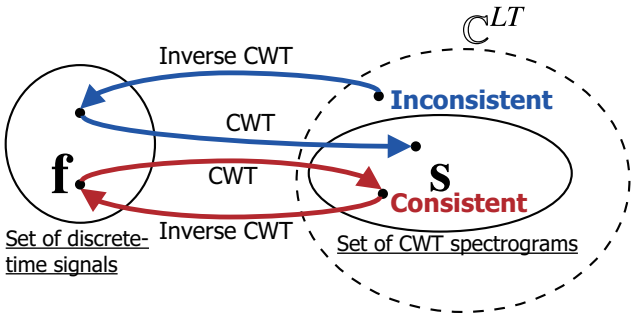


Figure 2: Illustration of the spectrogram consistency concept for the continuous wavelet transform (CWT).

rithm for estimating the phase from a magnitude CWT spectrogram. Sec. 2 formulates the phase estimation problem as an optimization problem based on a consistency condition. Sec. 3 derives an iterative algorithm for phase estimation based on an auxiliary function approach, which turns out to be equivalent to the algorithm proposed by Irino [7]. Our formulation gives a very clear proof of the convergence of the algorithm, though it should be noted that the proof of the convergence has already been given in [10]. Sec. 4 describes a fast approximate method for computing each iterative step of the proposed algorithm.

2. CWT SPECTROGRAM CONSISTENCY

2.1. Consistency condition

The scale parameter of the CWT corresponds to the period (the reciprocal of the center frequency) of the wavelet basis function. Here we consider discretizing the scale parameter such that the center frequencies of the wavelet basis functions are uniformly spaced on a log-frequency scale. Let the indices of the scale parameter and time shift parameter be denoted by $l \in [0, L-1]$ and $t \in [0, T-1]$, respectively, and let the component of a CWT spectrogram associated with the l-th scale parameter $a_l > 0$ (hereafter, the l-th component) be denoted by $\boldsymbol{s}_l = [s_{l,0}, s_{l,1}, \ldots, s_{l,T-1}]^\mathsf{T} \in \mathbb{C}^T$. Given a discrete-time signal $\boldsymbol{f} = [f_0, f_1, \ldots, f_{T-1}]^\mathsf{T} \in \mathcal{F}$ where $\mathcal{F} := \{\boldsymbol{f}' \in \mathbb{C}^T; \sum_t f'_t = 0\}$, its CWT spectrogram $\boldsymbol{s} = [\boldsymbol{s}_0^\mathsf{T}, \boldsymbol{s}_1^\mathsf{T}, \ldots, \boldsymbol{s}_{L-1}^\mathsf{T}]^\mathsf{T} \in \mathbb{C}^{LT}$ is defined as

$$\boldsymbol{s} = W\boldsymbol{f}, \tag{1}$$

where $W \in \mathbb{C}^{LT \times T}$ denotes the CWT matrix, defined as

$$W := \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{L-1} \end{bmatrix}, \quad W_l := \begin{bmatrix} \psi_{l,0} & \psi_{l,1} & \cdots & \psi_{l,T-1} \\ \psi_{l,T-1} & \psi_{l,0} & \cdots & \psi_{l,T-2} \\ \vdots & & \ddots & \vdots \\ \psi_{l,1} & \psi_{l,2} & \cdots & \psi_{l,0} \end{bmatrix}. \tag{2}$$

Here, $\psi_{l,t} := \psi(t\Delta/a_l)/a_l$ is a scaled mother wavelet with the scale of a_l and the time shift of $t\Delta$, where Δ denotes the sampling period of the time shift parameter and $\psi(\cdot) \in \mathbb{C}$ denotes the mother wavelet satisfying the admissibility condition. Each row of $W_l \in \mathbb{C}^{T \times T}$ contains the wavelet basis function of scale a_l with a different time shift parameter. The inverse CWT can be defined by the pseudo-inverse of W:

$$\boldsymbol{f} = W^+\boldsymbol{s}, \quad W^+ := (W^\mathsf{H}W)^{-1}W^\mathsf{H}, \tag{3}$$

where $^\mathsf{H}$ is used to denote the Hermitian transpose. This implicitly means that the inverse CWT is defined as the solution to the following minimization problem:

$$W^+\boldsymbol{s} = \operatorname*{argmin}_{\tilde{\boldsymbol{f}} \in \mathcal{F}} \|\boldsymbol{s} - W\tilde{\boldsymbol{f}}\|_2^2, \tag{4}$$

where $\|\cdot\|_2$ denotes the ℓ^2 norm of a vector.

While the CWT spectrogram of an audio signal (i.e., a complex vector that belongs to the subspace spanned by the column vectors of W) will be mapped to itself by applying the inverse CWT followed by the CWT, a complex vector that does not belong to the subspace will not come back to the same point but will be projected onto the nearest point in the subspace. Thus, we can define a condition for a complex vector to be "consistent" (in the sense that it corresponds to a CWT spectrogram of a signal) as follows:

$$\boldsymbol{0}_{LT} = \boldsymbol{s} - WW^+\boldsymbol{s}, \tag{5}$$

where $\boldsymbol{0}_{LT}$ denotes an LT-dimensional zero vector. It is important to note that when W is replaced with a matrix in which each row is a basis function of the STFT, (5) becomes equivalent to the consistency condition for an STFT spectrogram proposed in [8].

2.2. Phase estimation using spectrogram consistency

When given a magnitude CWT spectrogram $\boldsymbol{a} \in [0, \infty)^{LT}$, we can construct a signal by assigning phase $\boldsymbol{\phi} \in [-\pi, \pi)^{LT}$ to it to obtain a complex spectrogram $\boldsymbol{s}$, and applying the inverse CWT, i.e., $W^+\boldsymbol{s}$. Here, if we assign "inconsistent" phase to the given magnitude spectrogram, the complex spectrogram $\boldsymbol{s}$ will not belong to the signal subspace and so the spectrogram of the constructed signal, $WW^+\boldsymbol{s}$, will be different from $\boldsymbol{s}$. As we want to keep the magnitude spectrogram of the constructed signal consistent with the given magnitude spectrogram, we must find "consistent" phase such that $\boldsymbol{s}$ satisfies the consistency condition.

2.3. Filter bank interpretation

To give a deeper insight into the consistency condition, we focus on the filter bank interpretation of the CWT. The CWT of a signal can be thought of as the output of a filter bank consisting of subband filters whose impulse responses are given by the the scaled mother wavelets. Now, by applying the T-point discrete Fourier transform (DFT) to each block of (5), (5) can be written equivalently as

$$0 = \hat{\mathbf{s}} - \hat{W}\hat{W}^+\hat{\mathbf{s}}, \tag{6}$$

where

$$\hat{W} = \begin{bmatrix} \hat{W}_0 \\ \hat{W}_1 \\ \vdots \\ \hat{W}_{L-1} \end{bmatrix}, \quad \hat{W}_l = F_T W_l F_T^\mathsf{H}, \quad \hat{W}^+ = (\hat{W}^\mathsf{H}\hat{W})^{-1}\hat{W}^\mathsf{H}, \tag{7}$$

$F_T \in \mathbb{C}^{T \times T}$ is the DFT matrix and $\hat{\cdot}$ denotes the DFT of a variable. Since W_l is a circulant matrix, W_l is diagonalized by F_T and F_T^H. The diagonal elements of $\hat{W}_l$ represent the frequency response of the l-th subband filter associated with the scale parameter a_l. The k-th diagonal element of (6) is explicitly written as

$$0 = \hat{s}_{l,k} - \frac{1}{C_k}\sum_{l'} \hat{\psi}_{l,k}\hat{\psi}^*_{l',k}\hat{s}_{l',k}, \tag{8}$$

where $k \in [0, T-1]$ denotes the angular frequency index, C_k is a normalization constant, and * is used to denote the complex conjugate.

If the subbands of the filter bank overlap each other (more precisely, if there exists a pair of channels such that the product of their frequency responses is non-zero at every frequency), i.e. $\forall k, \exists l \neq l', \hat{\psi}_{l,k}\hat{\psi}_{l',k} \neq 0$, (5) becomes a nontrivial condition for a complex vector $\boldsymbol{s} \in \mathbb{C}^{LT}$ to correspond to a consistent CWT spectrogram. Otherwise, all the elements of $\mathbb{C}^{LT}$ trivially satisfy (5), implying that the consistency condition cannot be used as a criterion for phase estimation. Therefore, care must be taken in choosing the quantization intervals of the scale parameter and the type of the mother wavelet function. The Morlet [11], the log-normal wavelet [4] and the wavelets used in the auditory wavelet transform [7] satisfy the above requirement when the quantization intervals of the scale parameter are appropriately chosen. We hereafter assume to use a filter bank that satisfies $\forall k, \exists l \neq l', \hat{\psi}_{l,k}\hat{\psi}_{l',k} \neq 0$.

The requirement for the subbands of the CWT to overlap each other is analogous to the requirement for the short time frames of the STFT to overlap. The consistency condition of STFT spectrograms can be understood as implying that the waveforms within the overlapping segment of consecutive frames must be consistent [8]. The consistency condition of CWT spectrograms, on the other hand, can be interpreted as implying that the outputs of adjacent channels within the overlapping subbands must be consistent.

3. PHASE ESTIMATION BASED ON CWT SPECTROGRAM CONSISTENCY

3.1. Formulation of phase estimation problem

Assume that we are given a magnitude CWT spectrogram, arranged as a non-negative vector $\boldsymbol{a} \in [0, \infty)^{LT}$. We would like to estimate the phase of the given magnitude spectrogram such that it meets the consistency condition. To allow for any vector $\boldsymbol{a} \in [0, \infty)^{LT}$ as the input, we consider finding a phase estimate $\boldsymbol{\phi} \in [-\pi, \pi)^{LT}$ that minimizes the consistency criterion

$$\mathcal{I}(\boldsymbol{\phi}) := \|\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi}) - WW^+\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})\|_2^2, \tag{9}$$

where $\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})$ denotes the estimated CWT spectrogram defined by

$$\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi}) := \boldsymbol{a} \odot \begin{bmatrix} e^{\mathrm{j}\phi_0} \\ e^{\mathrm{j}\phi_1} \\ \vdots \\ e^{\mathrm{j}\phi_{LT-1}} \end{bmatrix}. \tag{10}$$

$\odot$ denotes the element-wise product. $\mathcal{I}(\boldsymbol{\phi})$ describes how far $\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})$ deviates from the consistency condition. Namely, the more consistent $\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})$ becomes, the smaller $\mathcal{I}(\boldsymbol{\phi})$ becomes.

3.2. Iterative algorithm with auxiliary function approach

Unfortunately, the optimization problem of minimizing $\mathcal{I}(\boldsymbol{\phi})$ with respect to $\boldsymbol{\phi}$ is difficult to solve analytically. However, we can invoke the auxiliary function approach to derive an iterative algorithm that searches for the estimate of $\boldsymbol{\phi}$, as with [8]. To apply the auxiliary function approach to the current optimization problem, the first step is to construct an auxiliary function $\mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$ satisfying $\mathcal{I}(\boldsymbol{\phi}) = \min_{\tilde{\boldsymbol{s}}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$. We refer to $\tilde{\boldsymbol{s}}$ as an auxiliary variable. It can then be shown that $\mathcal{I}(\boldsymbol{\phi})$ is non-increasing under the updates $\boldsymbol{\phi} \leftarrow \mathrm{argmin}_{\boldsymbol{\phi}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$ and $\tilde{\boldsymbol{s}} \leftarrow \mathrm{argmin}_{\tilde{\boldsymbol{s}}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$. The proof of this shall be omitted owing to space limitations. Thus, $\mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$ should be designed as a function that can be minimized analytically with respect to $\boldsymbol{\phi}$ and $\tilde{\boldsymbol{s}}$. Such a function can be constructed as follows.

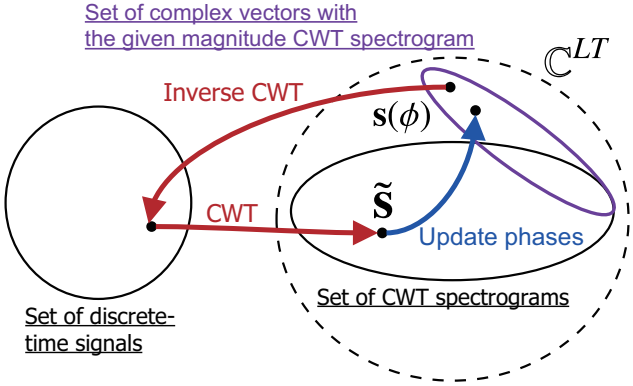


Figure 3: Illustration of the iterative phase estimation algorithm. The red and blue arrows correspond to (14) and (15).

Recall that the operator WW^+ is an orthogonal projection onto the subspace spanned by the column vectors of W and so $WW^+\boldsymbol{s}$ indicates the closest point in the subspace from $\boldsymbol{s}$. Thus, we can show that

$$\mathcal{I}(\boldsymbol{\phi}) = \min_{\tilde{\boldsymbol{f}} \in \mathcal{F}} \|\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi}) - W\tilde{\boldsymbol{f}}\|_2^2 \tag{11}$$

$$= \min_{\tilde{\boldsymbol{s}} \in \mathcal{W}} \|\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi}) - \tilde{\boldsymbol{s}}\|_2^2, \tag{12}$$

where $\mathcal{W}$ denotes the set of consistent CWT spectrograms (the subspace spanned by the column vectors of W). Therefore, we can confirm that

$$\mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}}) := \|\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi}) - \tilde{\boldsymbol{s}}\|_2^2, \quad \tilde{\boldsymbol{s}} \in \mathcal{W}, \tag{13}$$

satisfies $\mathcal{I}(\boldsymbol{\phi}) = \min_{\tilde{\boldsymbol{s}} \in \mathcal{W}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$. (13) can thus be used as an auxiliary function for $\mathcal{I}(\boldsymbol{\phi})$. We can thus monotonically decrease $\mathcal{I}(\boldsymbol{\phi})$ by iteratively performing $\tilde{\boldsymbol{s}} \leftarrow \mathrm{argmin}_{\tilde{\boldsymbol{s}}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$ and $\boldsymbol{\phi} \leftarrow \mathrm{argmin}_{\boldsymbol{\phi}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$. Here, $\tilde{\boldsymbol{s}} \leftarrow \mathrm{argmin}_{\tilde{\boldsymbol{s}}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$ and $\boldsymbol{\phi} \leftarrow \mathrm{argmin}_{\boldsymbol{\phi}} \mathcal{I}^+(\boldsymbol{\phi}, \tilde{\boldsymbol{s}})$ can be written explicitly as

$$\tilde{\boldsymbol{s}} \leftarrow WW^+\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi}), \tag{14}$$

$$\boldsymbol{\phi} \leftarrow \angle\tilde{\boldsymbol{s}}, \tag{15}$$

respectively, where $\angle$ denotes an operator that gives the arguments of the components of a complex vector as a real vector in $[-\pi, \pi)^{LT}$.

(14) means applying the inverse CWT followed by the CWT to $\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})$. Here, when $\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})$ is already a complex vector corresponding to a consistent spectrogram, this update simply becomes $\tilde{\boldsymbol{s}} \leftarrow \boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})$. (15) means replacing the phase estimate $\boldsymbol{\phi}$ with the phase of $\tilde{\boldsymbol{s}}$. A schematic illustration of these updates is shown in Fig. 3. $\mathcal{I}(\boldsymbol{\phi}) = 0$ indicates that $\boldsymbol{s}(\boldsymbol{a}, \boldsymbol{\phi})$ lies in the intersection of the set of consistent CWT spectrograms and the set of complex vectors that are equal to $\boldsymbol{a}$ up to a phase factor.

3.3. Relation to previous work

The present algorithm is equivalent to an algorithm proposed by Irino [7]. In addition, when W is replaced with a matrix in which each row is a basis function of the STFT, the present algorithm becomes equivalent to the phase estimation algorithm for a magnitude STFT spectrogram proposed in [8].

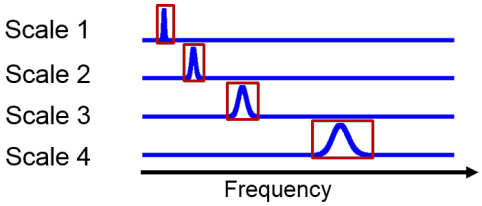


Figure 4: Example of the frequency responses of different subband filters (i.e., the scaled mother wavelets). The mother wavelet is the log-normal wavelet [4].

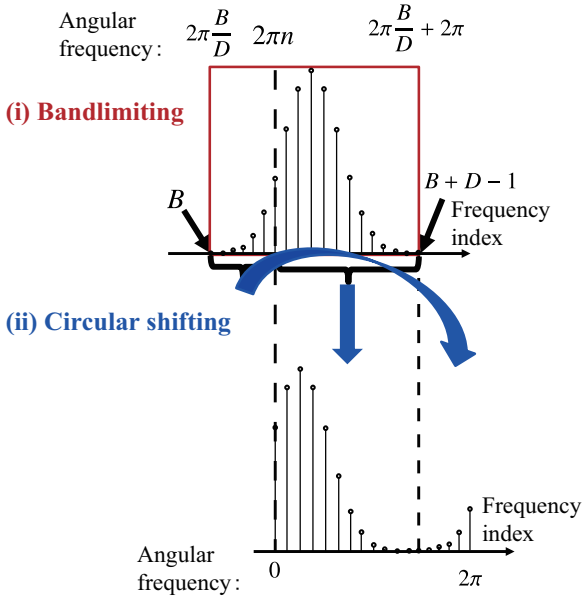


Figure 5: A circularly shifted version of $G_{l,B},\ldots,G_{l,B+D-1}$ [12].

4. FAST PHASE ESTIMATION ALGORITHM

4.1. Fast approximate continuous wavelet transform

The CWT and the inverse CWT are computationally expensive compared to the STFT and the inverse STFT. Here we briefly describe the fast approximate method for computing CWT proposed in [12]. The proposed fast approximate CWT uses the fact that the dominant part of the frequency response of each subband filter is concentrated around its center frequency (as shown in Fig. 4), as is common in many types of mother wavelets including the Morlet and log-normal wavelets [4].

According to the filter bank interpretation of the CWT, the CWT of an input signal, $\boldsymbol{s}_l = [s_{l,0},\ldots,s_{l,T-1}]^{\mathsf{T}} = W_l\boldsymbol{f}$, can be computed by multiplying the DFT of the entire signal, i.e, $\hat{\boldsymbol{f}} = [\hat{f}_0,\ldots,\hat{f}_{T-1}]^{\mathsf{T}} = F_T\boldsymbol{f}$, by the frequency response of the l-th subband, i.e., $\hat{W}_l = \mathrm{diag}(\hat{\psi}_{l,0},\ldots,\hat{\psi}_{l,T-1})$, and then computing the inverse DFT of $\hat{W}_l\hat{\boldsymbol{f}}$. This can be confirmed from

$$\boldsymbol{s}_l = W_l\boldsymbol{f} = F_T^{\mathsf{H}}F_TW_lF_T^{\mathsf{H}}F_T\boldsymbol{f} = F_T^{\mathsf{H}}\hat{W}_l\hat{\boldsymbol{f}}. \tag{16}$$

Note that the second equality follows from the fact that the DFT matrix F_T is a unitary matrix, i.e., $F_T^{\mathsf{H}}F_T = \boldsymbol{I}_T$. Here, if we can assume that the elements of $\{\hat{\psi}_{l,k}\}_k$ are dominant within and near 0 outside the range $k\in[B,B+D-1]$ $(0\le B, 0<D\le T)$, we can approximate $\boldsymbol{s}_l$ reasonably well by using the elements of $\{\hat{\psi}_{l,k}\hat{f}_k\}_k$ only within that range and neglecting the remaining elements. This implies the possibility of computing an approximation of $\boldsymbol{s}_l$ with a lower computational cost.

For simplicity of notation, let us put $G_{l,k} = \hat{\psi}_{l,k}\hat{f}_k$. We are concerned with computing an approximation of the full-band inverse DFT of $G_{l,k}$:

$$s_{l,t} = \sum_{k=0}^{T-1} G_{l,k}e^{j\frac{2\pi kt}{T}}. \tag{17}$$

As mentioned above, $G_{l,0},\ldots,G_{l,T-1}$ can be approximately viewed as a band-limited spectrum. In general, the inverse DFT of a band-limited spectrum can be computed by taking the inverse DFT over the finite support. In the time domain, this process corresponds to downsampling the signal given by the "full-band" inverse DFT. The proposed method uses this idea to approximate the inverse DFT of the full-band spectrum $G_{l,0},\ldots,G_{l,T-1}$. Now, if we choose D such that T/D becomes an integer, we can approximate the downsampled version of $s_{l,t}$ by

$$\tilde{s}_{l,d} = \sum_{k=B}^{B+D-1} G_{l,k}e^{j\frac{2\pi kd}{D}} = \sum_{k=B}^{B+D-1} G_{l,k}e^{j\frac{2\pi k(T/D)d}{T}}. \tag{18}$$

By comparing (17) and (18), we can confirm that

$$s_{l,(T/D)d} \simeq \tilde{s}_{l,d} \quad (d\in[0,D-1]), \tag{19}$$

if we assume $G_{l,k}\simeq 0$ outside the range $k\in[B,B+D-1]$. Since $\tilde{s}_{l,d}$ can be rewritten as

$$\tilde{s}_{l,d} = \sum_{k=0}^{D-1} G_{l,k+B}e^{j(\frac{2\pi k}{D}+2\pi\frac{B}{D})d} = e^{j2\pi\frac{B}{D}d}\sum_{k=0}^{D-1} G_{l,k+B}e^{j\frac{2\pi kd}{D}}, \tag{20}$$

we notice that $\tilde{s}_{l,d}$ can be computed by multiplying the inverse DFT of $G_{l,B},\ldots,G_{l,B+D-1}$ by $e^{j2\pi\frac{B}{D}d}$. Note that this is equivalent to computing the inverse DFT of a circularly shifted version of $G_{l,k}$ (see Fig. 5):

$$\tilde{G}_{l,k} = \begin{cases} G_{l,k+nD} & (k=0,\ldots,B-(n-1)D-1) \\ G_{l,k+(n-1)D} & (k=B-(n-1)D,\ldots,D-1) \end{cases}, \tag{21}$$

where n is an integer such that

$$n-1 < \frac{B}{D} \le n. \tag{22}$$

We consider invoking the fast Fourier transform (FFT) algorithm for computing the inverse DFT and so we assume the size D to be a power of 2. Since $D<T$, the computational cost for computing $\tilde{s}_{l,0},\ldots,\tilde{s}_{l,D-1}$ is obviously lower than that for computing $s_{l,0},\ldots,s_{l,T-1}$.

4.2. Fast phase estimation algorithm

The processes of bandlimiting and circular shifting can be represented by a matrix K:

$$K := \underbrace{\begin{bmatrix} 0_{B_0\times(D-B_0)} & I_{B_0} \\ I_{D-B_0} & 0_{(D-B_0)\times B_0} \end{bmatrix}}_{\text{(ii) Circular shifting}} \underbrace{\begin{bmatrix} 0_{D\times B} & I_D & 0_{D\times(T-D-B)} \end{bmatrix}}_{\text{(i) Bandlimiting}} \tag{23}$$

where I_D and $0_{D\times B}$ are the $D \times D$ identity matrix and the $D \times B$ zero matrix. The downsampled version of $\boldsymbol{s}_l$ obtained with the abovementioned fast approximate CWT can be described as

$$\check{\boldsymbol{s}}_l = F_D^{\mathsf{H}} K \hat{W}_l F_T \boldsymbol{f}. \tag{24}$$

Similarly to the inverse CWT, the fast approximate version of the inverse CWT can be defined by the pseudo-inverse matrix of $F_D^{\mathsf{H}} K \hat{W}_l F_T$. It is important to note that convergence of the phase estimation algorithm in which the CWT and inverse CWT steps are replaced with the fast approximate versions is still guaranteed.

4.3. Time and space complexity

The computational costs for the CWT and the fast approximate CWT mainly depend on the number of the points for the inverse DFT. Since the computational complexity of the full band inverse DFT is $O(T \log_2 T)$, the total computational complexity of the CWT is $O(T \log_2 T + LT \log_2 T)$. By contrast, the computational complexity of the band-limited DFT is $O(D \log_2 D)$ and so the total computational complexity is $O(T \log_2 T + \sum_{l=0}^{L-1} D_l \log_2 D_l)$.

The space complexity of the proposed algorithm is small compared to Irino's algorithm [7]. When the signal length T is long enough, the space complexity depends primarily on the size of the CWT spectrogram. While the size of the CWT spectrogram of Irino's algorithm is LT, that of the proposed algorithm is only $\sum_l D_l$.

5. EXPERIMENTAL EVALUATIONS

5.1. First experiment: Computation time and audio quality

5.1.1. Experimental conditions

To evaluate the computation time and the audio quality of the reconstructed signals by the phase estimation algorithms, we conducted an objective experiment, and compared the proposed algorithm with the Irino's algorithm [7].

We used the magnitude CWT spectrograms of 16kHz-sampled acoustic signals of the 113 male and 115 female speeches in the ATR Japanese speech database A-set [13]. The FFT performs faster for acoustic signals with a power of 2 length than those with the other length, and the used signals were filled by 0 till each length reached a power of 2. Phases were initialized randomly, and both the algorithms were finished at 1000 iterations. As the mother wavelet, we used the log-normal wavelet [4], which is defined in the Fourier-transformed domain:

$$\hat{\psi}(\omega) := \begin{cases} \exp\left(-\dfrac{(\log \omega)^2}{4\sigma^2}\right) & (\omega > 0) \\ 0 & (\omega \geq 0) \end{cases} \tag{25}$$

where ω is an angular frequency and σ is a standard deviation. σ was set at 0.02 and the analysis frequencies ranged 27.5 to 7040 Hz with 20 cents interval (i.e. uniform interval in the log-frequency domain). In the proposed algorithm, we computed the elements within $\pm 3\sigma$ around the central frequencies in the log-frequency domain. The used computer had the Intel Xeon CPU E31245 (3.3 GHz) and a 32 GB RAM.

We employed the perceptual evaluation of speech quality (PESQ) [14] as the evaluation measure for audio quality, which is the world-standard objective evaluation measure for speech quality. It ranges −0.5 to 4.5 and speech quality is higher as the PESQ becomes larger. As an evaluation measure of the computation speed, the computation time per iteration was used.

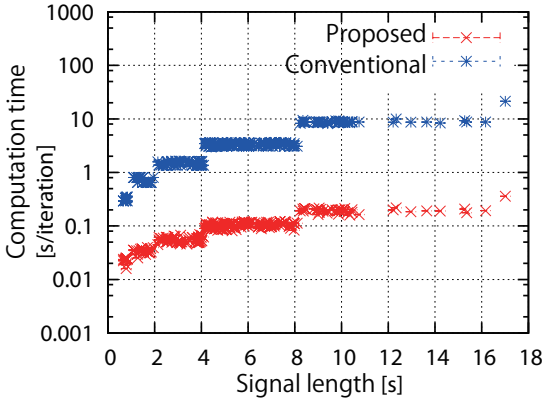


Figure 6: The averaged computation time per iteration with standard errors with respect to various signal lengths.

5.1.2. Results

The averaged PESQ with standard errors were 4.20 ± 0.08 for the Irino's algorithm and 4.1 ± 0.1 for the proposed algorithm. The result indicates that the speech qualities of the reconstructed signals were high enough for practical use. The difference between the Irino's and proposed algorithms was negligible practically[1].

Fig. 6 shows the results for the computation speed with respect to the signal length, since the computational complexity of the algorithms primarily depends on the signal length. The proposed algorithm was around 100 times faster than the Irino's algorithm in the computation time. For example, the averaged computation time per iteration by the Irino's algorithm was around 10 s/iteration for the 15 s signal. In contrast, that by the proposed algorithm was around 0.1 s/iteration.

5.2. Second experiment: Relation between approximation accuracy and audio quality

5.2.1. Experimental conditions

The proposed algorithm includes the approximation, and we next evaluated the relation between the approximation accuracy and the audio quality of the reconstructed signals. We used the 5 s from 30 s of 102 music audio files with 16 kHz sampling frequency in the RWC music genre database [15]. As the mother wavelet, the log-normal wavelet with $\sigma = 0.02$ was chosen. The approximation accuracy of the proposed algorithm corresponds to the calculated range by the downsampling step, and we used the elements within $\pm P\sigma$ ($P = 1, 2, 3, 5$) around the central frequencies in the log-frequency domain. The number of iterations was set at 500 for the proposed algorithm and at 100 for the Irino's algorithm. The used computer had the Intel Core i3-2120 CPU (3.30 GHz) and a 8 GB RAM. The other experimental conditions were the same as in Sec. 5.1.1.

An evaluation measure for audio quality was the objective differential grade (ODG) by the perceptual evaluation of audio qual-

[1] Audio samples are available at `http://hil.t.u-tokyo.ac.jp/~nakamura/demo/fastCWT.html`.

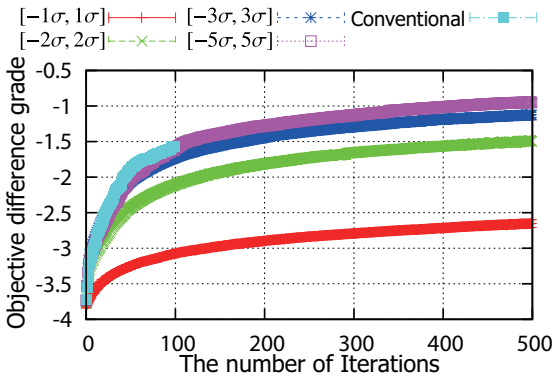


Figure 7: Evolution of the averaged objective difference grades with standard errors by perceptual evaluation of audio quality with respect to the number of iterations for the proposed algorithms with various approximations ($[-P\sigma, P\sigma]$ ($P = 1, 2, 3, 5$)) and the Irino's algorithm [7].

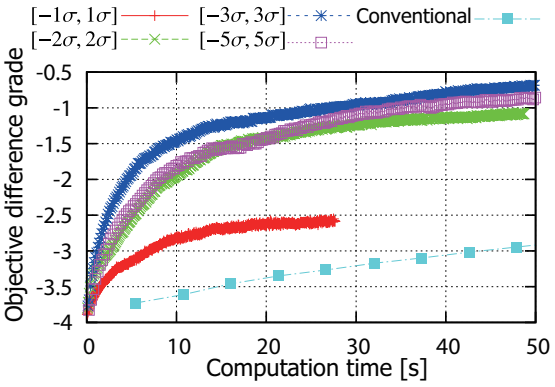


Figure 8: Evolution of the objective difference grades by perceptual evaluation of audio quality with respect to the computation time for the proposed algorithms with various approximations ($[-P\sigma, P\sigma]$ ($P = 1, 2, 3, 5$)) and the Irino's algorithm [7].

ity (PEAQ) [16]. It ranges −4 to 0, and the acoustic quality is higher as the ODG becomes larger.

5.2.2. *Results*

Fig. 7 illustrates the averaged ODGs with standard errors. The ODGs by the proposed algorithms with $P = 3, 5$ were larger than −2.0 after 100 iterations, and the results for $P = 3, 5$ shows high audio quality[2]. The results does not significantly differ from that by the Irino's algorithm in audio quality. We can thus say that the proposed algorithm with around $P \geq 3$ reconstructs the acoustic signals with almost the same audio quality as the Irino's algorithm.

In a viewpoint of the computation speed, the computation time becomes shorter as P is smaller. Fig. 8 shows the result for one of the acoustic signals (RWC-MDB-G-2001 No. 1), and the ODGs by the proposed algorithms quickly become higher than those by

[2]c.f.) When the used audio signals were converted into MPEG-3 files with 160 kbps, the averaged ODGs with standard errors were -3.68 ± 0.03.

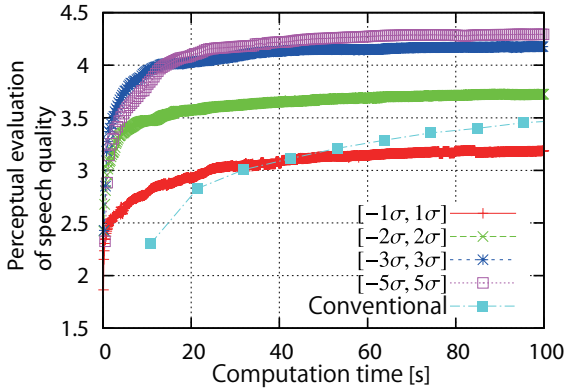


Figure 9: Evolution of the perceptual evaluation of speech quality and the computation time with respect to the proposed algorithms with various approximations ($[-P\sigma, P\sigma]$ ($P = 1, 2, 3, 5$)) and the Irino's algorithm [7].

the Irino's algorithm. The similar result for the speech signal (fafsc110 in the ATR Japanese speech database A-set [13], the 7 s signal) was shown in Fig. 9. Therefore, we conclude that the proposed algorithm with around $P = 3$ provides the reconstructed signals with high audio quality in a reasonable computation time.

5.3. Demonstration of phase estimation

We demonstrate pitch transposition of acoustic signals to confirm effectiveness of the proposed algorithm for sound manipulation. When the analysis frequencies are located uniformly in the log-frequency domain and $D_0 = D_1 = \cdots = D_{L-1}$ in the proposed algorithm, we simply shift the components of the CWT spectrograms to the lower or higher analysis frequency components, and the blank components by the move are filled by zero. However, the shifts cause the mismatches of phases, and the use of the original and zero phases leads to failure of the pitch transposition, hence we need to use the phase estimation for synthesizing the pitch-transposed acoustic signals. By the proposed algorithm, we obtained the synthesized signals [3] as we expected.

6. CONCLUSION

We have proposed a fast and convergence-guaranteed algorithm of the phase estimation by using the fast approximate CWT [12]. The phase estimation problem has been formulated based on the consistency condition, and the iterative algorithm has been derived by applying the auxiliary function method, which is the same as the Irino's algorithm [7]. Furthermore, we show the requirement on scale factors and mother wavelets for the phase estimation by using the consistency condition. The experimental results have shown that the proposed algorithm was about 100 times faster than the algorithm provided in [7]. The audio quality of the reconstructed signals for music and speech data was high enough for practical use, and the difference between the results by the proposed algorithm and the algorithm provided in [7] was negligible.

[3]The synthesized signals are available at `http://hil.t.u-tokyo.ac.jp/~nakamura/demo/fastCWT.html`.

We plan to combine the phase estimation with source separations for magnitude CWT spectrograms for music acoustic signal manipulation such as conversions of chords, keys and scales. To increase convenience, developing the online version of the proposed algorithm is important.

7. REFERENCES

[1] H. Fletcher, "Auditory patterns," *Rev. Mod. Phys.*, vol. 12, pp. 47–61, 1940.

[2] R. D. Patterson, "Auditory filter shape," *J. Acoust. Soc. Am.*, vol. 55, no. 4, pp. 802–809, 2005.

[3] M. N. Schmidt and M. Mørup, "Nonnegative matrix factor 2-D deconvolution for blind single channel source separation," in *Independent Component Analysis and Blind Signal Separation*, pp. 700–707. Springer, 2006.

[4] H. Kameoka, *Statistical Approach to Multipitch Analysis*, Ph.D. thesis, The University of Tokyo, Mar. 2007.

[5] M. Muller, D. P. W. Ellis, A. Klapuri, and G. Richard, "Signal processing for music analysis," *IEEE J. Sel. Topics. Signal Process.*, vol. 5, no. 6, pp. 1088–1110, 2011.

[6] J. P. de León, F. Beltrán, and J. R. Beltrán, "A complex wavelet based fundamental frequency estimator in single-channel polyphonic signals," in *Proc. Digital Audio Effects*, 2013.

[7] T. Irino and H. Kawahara, "Signal reconstruction from modified auditory wavelet transform," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3549–3554, 1993.

[8] J. Le Roux, H. Kameoka, N. Ono, and S. Sagayama, "Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency," in *Proc. Int. Conf. Digital Audio Effects*, Sep. 2010, pp. 397–403.

[9] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, 1984.

[10] D. M. Lopes and P. R. White, "Signal reconstruction from the magnitude or phase of a generalised wavelet transform," in *Proc. Eur. Signal Process. Conf.*, 2000, pp. 2029–2032.

[11] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *B. Am. Meteorol. Soc.*, vol. 79, no. 1, pp. 61–78, 1998.

[12] H. Kameoka, T. Tabaru, T. Nishimoto, and S. Sagayama, "(Patent) Signal processing method and unit," Nov. 2008, in Japanese.

[13] A. Kurematsu, K. Takeda, Y. Sagisaka, S. Katagiri, H. Kuwabara, and K. Shikano, "ATR Japanese speech database as a tool of speech recognition and synthesis," *Speech Commun.*, vol. 9, no. 4, pp. 357–363, 1990.

[14] "ITU-T recommendation P.862, Perceptual evaluation of speech quality (PESQ): an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," Feb. 2001.

[15] M. Goto, "Development of the RWC Music Database," in *Proc. Int. Congress Acoust.*, 2004, pp. l–553–556.

[16] "ITU-T recommendation BS.1387-1, Perceptual evaluation of audio quality (PEAQ): Method for objective measurements of perceived audio quality," Sep. 2001.

NUMERICAL SIMULATION OF STRING/BARRIER COLLISIONS: THE FRETBOARD.

Stefan Bilbao, *

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
sbilbao@staffmail.ed.ac.uk

Alberto Torin

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
s1164558@sms.ed.ac.uk

ABSTRACT

Collisions play a major role in various models of musical instruments; one particularly interesting case is that of the guitar fretboard, the subject of this paper. Here, the string is modelled including effects of tension modulation, and the distributed collision both with the fretboard and individual frets, and including both effects of free string vibration, and under finger-stopped conditions, requiring an additional collision model. In order to handle multiple distributed nonlinearities simultaneously, a finite difference time domain method is developed, with a penalty potential allowing for a convenient model of collision within a Hamiltonian framework, allowing for the construction of stable energy-conserving methods. Implementation details are discussed, and simulation results are presented illustrating a variety of features of such a model.

1. INTRODUCTION

Physical modeling synthesis to date has relied, mainly, on linear models of distributed components, accompanied by pointwise nonlinearities often related to excitation mechanisms (such as, for example, models of the bow, hammer, or lip-reed interaction). See, e.g., [1] for an overview. In the pursuit of more realistic sound synthesis, recent research has focused on inherent nonlinearities in the distributed components themselves, beginning with the introduction of tension modulation effects in strings [2, 3, 4], shock wave effects in acoustic tubes [5], geometric nonlinearities in strings [6], and in 2D systems such as membranes and plates [7]. A distinct form of distributed nonlinearity, and one which is of great significance to models of strings is the contact between a distributed vibrating object with a rigid barrier.

The problem of the string in contact with a rigid barrier has seen research in the realm of musical acoustics for almost a century, going back to early investigations of Indian stringed instruments such as the sitar or tambura [8], and continuing to the present day, particularly using a geometric analysis for barriers of simplified forms [9, 10]. In practical sound synthesis applications, where the barrier may well be of a complex shape, and in musical acoustics investigations, more flexible methods have been employed, including digital waveguides [11, 12, 13, 14, 15], modal techniques [16], and time-stepping methods such as finite difference methods [11, 17, 14, 18].

The particular case of the interaction of a string with a fret, modelled as a lumped barrier element, in order to emulate realistic playing in fretted instruments such as the guitar has been researched by Evangelista [12, 19], which is the case of interest in this paper. Here, a distributed view of the barrier is taken, including frets and the backing fretboard. Finite difference time domain methods are employed, with special attention paid to the problem of numerical stability, which is especially pronounced here, due to the inherently non-smooth form of the collision interaction. To this end, a formalism based upon the use of an added potential, allowing the use of a Hamiltonian framework, but permitting some spurious penetration of the string into the barrier is employed. The action of a stopping finger, in order to simulate finger motion against the fretboard, is also included here. The model here is complementary to that of Evangelista mentioned above, in that here, string motion is taken to be perpendicular to the fretboard—in a full model, both polarizations need to be taken into account. Finger plucking interactions have been described previously—see, e.g., [20]

Section 2 presents a complete model of string vibration in a single polarization, including tension modulation effects, distributed collision against a barrier of arbitrary shape, a plucking excitation, as well as a further collision due to stopping of a finger against the fretboard. An energy analysis completes this section. Section 3 is a concise presentation of finite difference time domain construction, with a discussion of numerical stability, arrived at through an analogous energy analysis, and implementation issues, and in particular a vector nonlinear equation to be solved at each time step. Simulation results, illustrating various features of such a model, are presented in Section 4. Sound synthesis examples are available online at http://www.ness-music.eu

2. STRING MODEL

A model of constrained string vibration may be written in a compact form as

$$\rho\partial_{tt}u = \mathcal{L}[u] + \mathcal{K}[u] + \mathcal{F}_e + \mathcal{F}_c - \mathcal{F}_f \quad (1)$$

Here, $u(x,t)$ is the transverse displacement of a string in a single polarization (assumed here to be perpendicular to a constraining surface, to be described shortly), as a function of time $t \geq 0$ and $x \in \mathcal{D} = [0, L]$, where L is string length when at rest. The string is of linear mass density ρ kg/m, and ∂_{tt} represents double partial differentiation with respect to time t. See Figure 1. Because this model of a string is in a single polarization only, it is thus capable of modelling only string plucks perpendicular to the fretboard—which is a great simplification from the true situation, but one allowing for an analysis of many of the important features of such an instrument.

The linear operator $\mathcal{L}$ is defined, in terms of its action on the function u, as

$$\mathcal{L}[u] = \left(T\partial_{xx} - EI\partial_{xxxx} - 2\sigma_0\rho\partial_t + 2\sigma_1\rho\partial_{txx}\right)u \quad (2)$$

* This work was supported by the European Research Council, under grant number StG-2011-279068-NESS.

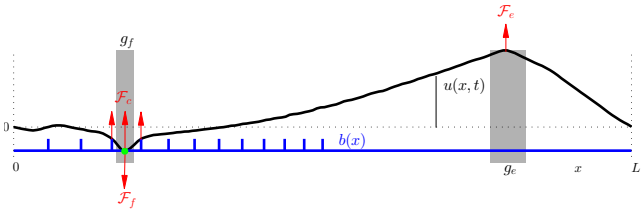


Figure 1: *Diagram of string, of displacement $u(x,t)$, in contact with a barrier $b(x)$, as indicated in blue. An excitation force density $\mathcal{F}_e$ is applied over a distribution g_e, and a force density $\mathcal{F}_f$ is applied by a finger (indicated in green) over a distribution g_f. A collision force density $\mathcal{F}_c$ results at points of contact between the string and barrier.*

and describes the linear dynamics of the string, where partial differentiation with respect to x is indicated by ∂_x. The four terms model, respectively, string tension, stiffness, frequency-independent loss, and frequency-dependent loss. Here, T is string tension, in N, E is Young's modulus, in Pa, I is the string moment of inertia (and equal to $\pi r^4/4$, for a string of circular cross-section and radius r m), and σ_0 and σ_1 are loss parameters, which may be set according to comparison with measured data. Such a linear model is relatively standard in the musical acoustics literature (with some variation in the way in which the frequency-dependent loss terms are modelled [21, 22]).

The nonlinear operator $\mathcal{K}$ is defined as

$$\mathcal{K}[u] = \frac{EA}{2L}\left(\int_{\mathcal{D}} (\partial_x u)^2 \, dx\right) \partial_{xx} u \tag{3}$$

where A is the string cross-sectional area in m^2, and describes effects of tension modulation in the string, giving rise to variations in pitch with excitation amplitude, or pitch glides; such a model is due to Kirchhoff [23] and Carrier [24], and has seen extensive use in sound synthesis applications [2, 3, 4, 25]. This is a particularly simple form of string nonlinearity—more realistic effects, including the generation of phantom partials [26, 6], may be obtained using a complete form which models the coupling between transverse and longitudinal motion in the string.

The final three terms in (1) represent force densities due, respectively, to a plucking action, collision of the string with the fretboard, and the stopping motion of a finger, and will be defined in the following sections.

2.1. Excitation

A relatively simple model of excitation will be employed here, namely that of a force density

$$\mathcal{F}_e = g_e f_e$$

where here, $f_e(t)$ is an applied force in N, and where $g_e(x)$ is a distribution selecting the region of application of the excitation (chosen normalized, with $\int_{\mathcal{D}} g_e \, dx = 1$, and perhaps as a Dirac delta function $g_e(x) = \delta(x - x_e)$, for a plucking point $x = x_e$). In some models of plucking excitation [27], a relatively smooth form of excitation function is employed:

$$f_e(t) = \begin{cases} \frac{f_p}{2}\left(1 - \cos(\pi(t - t_0)/t_p)\right) & t_0 \le t \le t_p \\ 0 & \text{else} \end{cases} \tag{4}$$

This function is characterized by a small set of parameters, namely: start time t_0, duration t_p, and maximum force f_p.

One could go further here and specify a full model of the plucking finger, but as this is not the focus of this paper, and also because in general, the duration of a pluck is extremely short (on the order of 1-10 ms) the simple form above will be employed, as in previous work on guitar synthesis [28]. More involved models are available—see, e.g., [29, 30].

2.2. The Fretboard

The string is assumed to vibrate above a rigid barrier of height $b(x)$—in the case of a fretboard, the function will include the profile of the board itself, as well as pointwise protuberances (the frets themselves). To this end, suppose that the function is of the form $b(x) = b_{back}(x)$, almost everywhere, where $b_{back}(x)$ is a smooth function representing the fretboard itself, in the absence of the frets. At locations x_m, $m = 1, \ldots, N_{fret}$ at which the N_{fret} frets are located, the function takes on the values $b(x_m) = b_{fret}^{(m)}$. See Figure 1.

The force density $\mathcal{F}_c$ acts upwards on the string, and may be defined in terms of a potential density $\Phi_c \ge 0$ as

$$\mathcal{F}_c = \frac{\partial_t \Phi_c}{\partial_t \eta_c} \qquad \text{where} \qquad \eta_c = b - u \tag{5}$$

The potential $\Phi_c(\eta_c)$ here is to be viewed as a penalty density, active whenever and wherever η_c, the difference between the barrier height and string height is positive, implying interpenetration, and thus repelling the string. A useful form of the penalty potential Φ_c is of the form of a power law $\Phi_c = \Phi_{K,\alpha}(\eta_c)$, where, for a value or distribution p

$$\Phi_{K,\alpha}(p) = \frac{K}{\alpha + 1}[p]_+^{\alpha+1} \qquad [p]_+ = \frac{1}{2}(p + |p|)$$

where $K \ge 0$, and $\alpha \ge 1$. In simulation, the degree of interpenetration can be controlled through a proper choice of K and α—see Section 4.2. Note that, under this choice of the potential, $\mathcal{F}_c = K[\eta_c]_+^\alpha$, and so this collision model is of a form similar to that seen in lumped models of impact, such as that of Hertz [31], and commonly used in models of striking action in musical instruments [22, 32]; here, however, it is to be viewed as an approximation to an ideal elastic collision. The form in (5), written in terms of a potential, however, is more useful when it comes to simulation design—see Section 3.

2.3. Finger-stopping

Another separate collision which must be taken into account in a full articulated model of such a stringed instrument is the action of a stopping finger pressing the string against the frets or fretboard. This collision is slightly different from the case of the barrier/string collision described in the previous section, as the finger must be permitted its own dynamics, including damping effects, and is subject to external control. In this case, where the string is assumed to move transverse to the fretboard, rubbing friction effects against the fret are not included—see [12].

For a lumped model of such a finger, the force density $\mathcal{F}_f$, now acting downward on the string from above, may be written as

$$\mathcal{F}_f = g_f f_f$$

Here, $g_f = g_f(x,t)$ is an externally specified function representing the region of contact of the finger with the string at time t, again chosen normalized, with $\int_{\mathcal{D}} g_f\,dx = 1$, and f_f is the force applied to the string, in N. The position of the finger, u_f, may be described by

$$M_f \frac{d^2 u_f}{dt^2} = f_f - f_0$$

where here, M_f is the finger mass, in kg, and where $f_0 = f_0(t)$ is an external force signal supplied by the player.

As in the case of the string/barrier collision, the interaction force f_f depends on a measure η_f of the relative displacement between the string and finger at the stopping location:

$$f_f = \frac{d\Phi_f/dt}{d\eta_f/dt} + \frac{d\eta_f}{dt}\Xi_f \qquad \eta_f = \int_{\mathcal{D}} g_f u\,dx - u_f \quad (6)$$

Here again, $\Phi_f(\eta_f) \geq 0$ is a collision potential—now, however, it is intended to model elastic deformation of the finger under the pressing action; the model here is identical to that of a striking piano hammer, with losses taken into account, and under a continuous excitation force. As in the case of the hammer, a choice of collision potential $\Phi_f = \Phi_{K_f,\alpha_f}(\eta_f)$ is reasonable, where again $K_f \geq 0$ and $\alpha_f \geq 1$. Also modelled here are losses, through a function $\Xi_f(\eta_f) \geq 0$. The model of Hunt and Crossley [33] is appropriate here, with $\Xi_f = \Xi_{K_f,\alpha_f,\beta_f}$, where

$$\Xi_{K_f,\alpha_f,\beta_f}(\eta_f) = \beta_f K_f \frac{d\eta_f}{dt}[\eta_f]_+^{\alpha}$$

for some constant $\beta_f \geq 0$.

2.4. Energy Balance

System (1) includes three separate nonlinearities, due to tension modulation, collision, and finger stopping, as well as non-autonomous time variation due to the finger-stopping distribution g_f, and thus frequency-domain analysis will thus be of virtually no use in designing a numerical method. To this end, it is useful to present an energy balance for the system.

It may be easily verified, through the multiplication of (1) by $\partial_t u$, integrating over the domain $\mathcal{D}$, and employing integration by parts, that the complete model described above satisfies an energy balance of the form

$$\frac{d\mathfrak{H}}{dt} = -\mathfrak{Q} + \mathfrak{P} + \mathfrak{B} \quad (7)$$

where here, at time t, $\mathfrak{H}(t)$ represents the total stored energy of the system, $\mathfrak{Q}(t)$ is total dissipated power, $\mathfrak{P}(t)$ is input power, and $\mathfrak{B}$ represents energy supplied to the string at the boundaries at $x = 0$ and $x = L$.

In particular,

$$\begin{aligned} \mathfrak{H} &= \mathfrak{H}_L + \mathfrak{H}_K + \mathfrak{H}_c + \mathfrak{H}_f \\ \mathfrak{Q} &= \mathfrak{Q}_L + \mathfrak{Q}_f \\ \mathfrak{P} &= \mathfrak{P}_e + \mathfrak{P}_f \end{aligned}$$

where, for the stored energy terms corresponding to linear string vibration, nonlinear string vibration, the collision interaction, and the finger interaction, respectively, one has

$$\begin{aligned} \mathfrak{H}_L &= \int_{\mathcal{D}} \frac{\rho}{2}(\partial_t u)^2 + \frac{T}{2}(\partial_x u)^2 + \frac{EI}{2}(\partial_{xx} u)^2\,dx \\ \mathfrak{H}_K &= \frac{EA}{8L}\left(\int_{\mathcal{D}} \partial_x u\,dx\right)^2 \\ \mathfrak{H}_c &= \int_{\mathcal{D}} \Phi_c\,dx \\ \mathfrak{H}_f &= \frac{M_f}{2}\left(\frac{du_f}{dt}\right)^2 + \Phi_f \end{aligned}$$

For the individual power loss terms $\mathfrak{Q}_L$ and $\mathfrak{Q}_f$ in the string and finger, respectively, one has

$$\begin{aligned} \mathfrak{Q}_L &= \int_{\mathcal{D}} 2\rho\sigma_0(\partial_t u)^2 + 2\rho\sigma_1(\partial_{tx} u)^2\,dx \\ \mathfrak{Q}_f &= \left(\frac{d\eta_f}{dt}\right)^2 \Xi_f(\eta_f) \end{aligned}$$

For the supplied power terms $\mathfrak{P}_e$ and $\mathfrak{P}_f$ from the excitation and stopping finger, respectively, one has

$$\begin{aligned} \mathfrak{P}_e &= f_e \int_{\mathcal{D}} g_e \partial_t u\,dx \\ \mathfrak{P}_f &= f_f \int_{\mathcal{D}} u \partial_t g_f\,dx - \frac{du_f}{dt} f_0 \end{aligned}$$

The boundary power term $\mathfrak{B}$ is given by

$$\begin{aligned} \mathfrak{B} &= \left(T + \frac{EA}{2L}\left(\int_{\mathcal{D}} \partial_x u\,dx\right)^2\right)\partial_t u \partial_x u \\ &\quad - EI(\partial_t u \partial_{xxx} u - \partial_{tx} u \partial_{xx} u) - 2\rho\sigma_1 \partial_t u \partial_{xt} u|_{x=0}^{x=L} \end{aligned}$$

In this study, boundary conditions are chosen as simply supported (i.e., $u = \partial_{xx} u = 0$ at $x = 0$ and $x = L$), and thus $\mathcal{B}$ vanishes identically.

Under unforced conditions (i.e., with no excitation force f_e, no applied finger stopping force f_0, and no time variation of the stopping finger distribution g_f), note that $\mathfrak{H} \geq 0$, and $\mathfrak{Q} \geq 0$, and thus, for all $t \geq 0$

$$\frac{d\mathfrak{H}}{dt} \leq 0 \qquad \longrightarrow \qquad 0 \leq \mathfrak{H}(t) \leq \mathfrak{H}(0)$$

and the system as a whole is dissipative. If, furthermore, loss is not present (i.e., if $\sigma_0 = \sigma_1 = \Xi_f = 0$), then the system is exactly lossless. Such an energy balance serves as a useful design principle in arriving at numerically stable simulation methods. See Section 3.

3. TIME STEPPING METHODS

In this section, the basic techniques underlying the construction of time-domain finite difference schemes are presented, in a condensed vectorized form. For a more expanded treatment of such methods, see, e.g., [34], or, in the context of physical modeling synthesis, [35].

3.1. Grid Functions and Difference Operators

The grid function u_l^n, for integer $n \geq 0$ and $l = 0, \ldots, N$, represents an approximation to the function $u(x,t)$ at time $t = nk$ and $x = lh$. Here, k is the time step (and $f_s = 1/k$ is the sample rate, chosen a priori), and h is the grid spacing, chosen such that it divides the length L evenly as $N = L/h$.

In this case, where the system under study is 1D, and because the boundary conditions are of simple form (that is, simply supported), it is useful to move directly to a vector representation of the state, namely the column vector $\mathbf{u}^n = [u_1^n, \ldots, u_{N-1}^n]^T$. Here, the values u_0^n and u_N^N have been omitted from the vector form, and thus need not be calculated, as they are identically zero—this choice has implications for the matrix representations of various spatial difference operators, as will be described shortly.

For any vector $\mathbf{w}^n$, unit time shifts e_{t+} and e_{t-} are defined as

$$e_{t+}\mathbf{w}^n = \mathbf{w}^{n+1} \qquad e_{t-}\mathbf{w}^n = \mathbf{w}^{n-1}$$

The forward, backward and centered difference approximations to a first time derivative may thus be defined as

$$\delta_{t+} = \frac{e_{t+} - 1}{k} \quad \delta_{t-} = \frac{1 - e_{t-}}{k} \quad \delta_{t\cdot} = \frac{e_{t+} - e_{t-}}{2k} \tag{8}$$

and time averaging operators as

$$\mu_{t+} = \frac{e_{t+} + 1}{2} \quad \mu_{t-} = \frac{1 + e_{t-}}{2} \quad \mu_{t\cdot} = \frac{e_{t+} + e_{t-}}{2} \tag{9}$$

An approximation to a second time derivative follows as

$$\delta_{tt} = \delta_{t+}\delta_{t-} = \frac{e_{t+} - 2 + e_{t-}}{k^2} \tag{10}$$

Forward and backward approximations to spatial differentiation ∂_x, when applied to the grid function $\mathbf{u}^n$, and taking into account the simply supported boundary condition, may be written in matrix form as $\mathbf{D}_{x+}$ and $\mathbf{D}_{x-}$, where $\mathbf{D}_{x+}$ is an $N \times (N-1)$ matrix, and $\mathbf{D}_{x-}$ is $(N-1) \times N$:

$$\mathbf{D}_{x+} = \frac{1}{h}\begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 \end{bmatrix} \qquad \mathbf{D}_{x-} = -\mathbf{D}_{x+}^T$$

where ^{T} indicates the transpose operation.

Approximations to the second and fourth spatial derivative, $\mathbf{D}_{xx}$ and $\mathbf{D}_{xxxx}$ respectively, both $(N-1) \times (N-1)$ matrices, may be written, under simply supported conditions, as

$$\mathbf{D}_{xx} = \mathbf{D}_{x-}\mathbf{D}_{x+} \qquad \mathbf{D}_{xxxx} = \mathbf{D}_{xx}\mathbf{D}_{xx}$$

3.2. Finite Difference Scheme

A finite difference time domain scheme for (1) may then be written, in vector-matrix form, in terms of the grid function $\mathbf{u}^n$, as

$$\rho\delta_{tt}\mathbf{u}^n = \mathfrak{l}[\mathbf{u}^n] + \mathfrak{k}[\mathbf{u}^n] + \mathfrak{f}_e^n + \mathfrak{f}_c^n - \mathfrak{f}_f^n \tag{11}$$

Here, in analogy with definition (2) for the linear operator $\mathcal{L}$, the linear discrete operator $\mathfrak{l}$ is defined as

$$\mathfrak{l}[\mathbf{u}^n] = (T\mathbf{D}_{xx} - EI\mathbf{D}_{xxxx} - 2\sigma_0\rho\delta_{t\cdot} + 2\sigma_1\rho\delta_{t-}\mathbf{D}_{xx})\,\mathbf{u}^n \tag{12}$$

and the nonlinear operator $\mathfrak{k}$ as

$$\mathfrak{k}[\mathbf{u}^n] = \frac{EAh}{2L}\left(\mathbf{D}_{x+}\mathbf{u}^n\right)^T\left(\mu_{t\cdot}\mathbf{D}_{x+}\mathbf{u}^n\right)\mathbf{D}_{xx}\mathbf{u}^n \tag{13}$$

Note the use of the time averaging operator $\mu_{t\cdot}$ in (13) above, necessary in arriving at a stable scheme [36].

3.3. Discrete Force Densities

The discrete force density terms $\mathfrak{f}_e^n$, $\mathfrak{f}_c^n$ and $\mathfrak{f}_f^n$ given in (11) are all $(N-1)$ element column vectors.

The discrete force excitation density $\mathfrak{f}_e^n$ may be written as $\mathfrak{f}_e^n = \mathbf{g}_e f_e^n$ where $\mathbf{g}_e$ corresponds to $g_e(x)$, with $h\mathbf{1}^T\mathbf{g}_e = 1$, where $\mathbf{1}$ is an $N-1$ element column vector consisting of ones, and where f_e^n is sampled from $f_e(t)$, as defined in (4).

The discrete collision force due to the interaction with the barrier $\mathfrak{f}_c^n$ requires a more detailed treatment. Because one would like to model collision between the string and the fretboard at the $N-1$ grid points at which the string is defined, and also at the N_{fret} locations at which the frets themselves are defined (which, in general, do not lie at grid locations), it is useful to write $\mathfrak{f}_c^n = \mathbf{G}_c\mathbf{f}_c^n$, where $\mathbf{f}_c$ is an $N_c = N - 1 + N_{fret}$ element force vector, and $\mathbf{G}_c$ is an $(N-1) \times N_c$ matrix interpolant. In particular, $\mathbf{G}_c = \frac{1}{h}[\mathbf{I}_{N-1}|\mathbf{G}_{fret}]$, where $\mathbf{I}_{N-1}$ is the $(N-1) \times (N-1)$ identity matrix, and where $\mathbf{G}_{fret}$ is an $(N-1) \times N_{fret}$ matrix, the mth column of which is an interpolant to the mth fret location x_m. Any form of interpolant (i.e., bilinear, Lagrangian, etc.) may be employed in this construction.

For the collision itself, one may then write, in analogy with (5),

$$\mathbf{f}_c^n = \frac{\delta_{t\cdot}\Phi_c^n}{\delta_{t\cdot}\eta_c^n} \qquad \eta_c^n = \mathbf{b} - h\mathbf{G}_c^T\mathbf{u}^n \tag{14}$$

in terms of the N_c element vectors Φ_c^n, η_c^n and $\mathbf{b}^n$. This latter vector, representing the barrier profile, may be decomposed as $\mathbf{b} = [\mathbf{b}_{back}^T|\mathbf{b}_{fret}^T]^T$, where $\mathbf{b}_{back}$ is the $N-1$ element column vector consisting of samples of the fretboard profile $b(x)$ at the grid locations, and $\mathbf{b}_{fret}$ is an N_{fret} element column vector consisting of the fret heights $b_{fret}^{(m)}$, $m = 1, \ldots, N_{fret}$. As in the continuous case, a power law potential may be employed, such that $\Phi_c^n = \Phi_{K,\alpha}\left(\eta_c^n\right)$. (Here and henceforth, expressions such as the first in (14) represent a vector resulting from element-by-element division of two vectors.)

The finger force density $\mathfrak{f}_f^n$ may be written as $\mathfrak{f}_f^n = \mu_{t\cdot}\left(\mathbf{g}_f^n\right)f_f^n$, where as in the case of the excitation, $\mathbf{g}_f^n$ is an $N-1$ element normalized column vector—note in particular that it is time-varying, allowing for gestural control of the finger-stopping action. The finger force may be discretized, in analogy with (6), as

$$f_f^n = \frac{\delta_{t\cdot}\Phi_f^n}{\delta_{t\cdot}\eta_f^n} + \delta_{t\cdot}\eta_f^n\Xi_f^n \qquad \eta_f^n = h\left(\mathbf{g}_f^n\right)^T\mathbf{u}^n - u_f^n \tag{15}$$

where $\Phi_f^n = \Phi_{K_f,\alpha_f}(\eta_f^n)$, and where $\Xi_f^n = \Xi_{K_f,\alpha_f,\beta_f}(\eta_f^n)$. Finally, the equation of motion of the finger, in terms of displacement u_f^n may be written as

$$M_f\delta_{tt}u_f^n = f_f^n - f_0^n$$

3.4. Discrete Energy Balance and Stability Conditions

In analogy with the energy balance (7) for the continuous system, a discrete energy balance follows for the scheme presented in Section 3.2:

$$\delta_{t-}\mathfrak{h}^{n+1/2} = -\mathfrak{q}^n + \mathfrak{p}^n + \mathfrak{b}^n \tag{16}$$

where here, $\mathfrak{h}^{n+1/2}$ represents the total stored energy of the system (written here as interleaved with respect to values calculated in the scheme itself), $\mathfrak{q}^n$ is total dissipated power, $\mathfrak{p}^n$ is input power, and $\mathfrak{b}^n$ represents energy supplied to the string at the boundaries at $l = 0$ and $l = N$—in this case, $\mathfrak{b}^n = 0$ by construction, so may be safely ignored in the remainder of this analysis. Here, the various terms may be decomposed as

$$\begin{aligned} \mathfrak{h}^{n+1/2} &= \mathfrak{h}_L^{n+1/2} + \mathfrak{h}_K^{n+1/2} + \mathfrak{h}_c^{n+1/2} + \mathfrak{h}_f^{n+1/2} \\ \mathfrak{q}^n &= \mathfrak{q}_L^n + \mathfrak{q}_f^n \\ \mathfrak{p}^n &= \mathfrak{p}_e^n + \mathfrak{p}_f^n \end{aligned}$$

where, for the stored energy terms corresponding to linear string vibration, nonlinear string vibration, the collision interaction, and the finger interaction, respectively, one has

$$\begin{aligned} \mathfrak{h}_L^{n+1/2} &= \frac{\rho h}{2}|\delta_{t+}\mathbf{u}^n|^2 + \frac{Th}{2}\left(\mathbf{D}_{x+}\mathbf{u}^n\right)^T \mathbf{D}_{x+}\mathbf{u}^{n+1} \\ &\quad + \frac{EIh}{2}\left(\mathbf{D}_{xx}\mathbf{u}^n\right)^T \mathbf{D}_{xx}\mathbf{u}^{n+1} - \frac{\rho\sigma_1 hk}{2}|\delta_{t+}\mathbf{D}_{x+}\mathbf{u}^n|^2 \\ \mathfrak{h}_K^{n+1/2} &= \frac{EAh^2}{8L}\left(\left(\mathbf{D}_{x+}\mathbf{u}^n\right)^T \mathbf{D}_{x+}\mathbf{u}^{n+1}\right)^2 \\ \mathfrak{h}_c^{n+1/2} &= \mathbf{1}^T \mu_{t+}\Phi_c^n \\ \mathfrak{h}_f^{n+1/2} &= \frac{M_f}{2}\left(\delta_{t+}u_f^n\right)^2 + \Phi_f^n \end{aligned}$$

and for the power loss terms,

$$\begin{aligned} \mathfrak{q}_L^n &= 2\rho\sigma_0 h|\delta_{t\cdot}\mathbf{u}^n|^2 + 2\rho\sigma_1 h|\delta_{t\cdot}\mathbf{D}_{x+}\mathbf{u}^n|^2 \\ \mathfrak{q}_f &= \left(\delta_{t\cdot}\eta_f^n\right)^2 \Xi_f^n \end{aligned}$$

For the supplied power terms $\mathfrak{p}_e^n$ and $\mathfrak{p}_f^n$ from the excitation and stopping finger, respectively, one has

$$\begin{aligned} \mathfrak{p}_e^n &= f_e^n h\left(\delta_{t\cdot}\mathbf{u}^n\right)^T \mathbf{g}_e \\ \mathfrak{p}_f^n &= f_f^n h\left(\mu_{t\cdot}\mathbf{u}^n\right)^T \delta_{t\cdot}\mathbf{g}_f^n - \delta_{t\cdot}u_f^n f_0^n \end{aligned}$$

Considering the discrete power balance (16), under unforced conditions (i.e., $\mathfrak{p}_e^n = \mathfrak{p}_f^n = 0$), note that the loss terms $\mathfrak{q}_L^n$ and $\mathfrak{q}_f^n$ are non-negative; the only stored energy term which is not non-negative is that corresponding to the string energy $\mathfrak{h}_L$. It is straightforward to show [35] that under the condition $h \geq h_{min}$, where

$$h_{min}^2 = \frac{k}{2}\left(\frac{Tk}{\rho} + 4\sigma_1 + \sqrt{\left(\frac{Tk}{\rho} + 4\sigma_1\right)^2 + \frac{16EI}{\rho}}\right) \quad (17)$$

the term $\mathfrak{h}_L$ is non-negative; this condition serves as a stability condition for the entire scheme. Again, under lossless conditions (i.e., with $\sigma_0 = \sigma_1 = \Xi^n = 0$), the scheme is numerically lossless. See Section 4.4. Notice that condition (17) is equivalent to that arrived at using von Neumann analysis [34] for the linear string in isolation, though now for the complete system involving multiple nonlinearities.

3.5. Vector-matrix Update Form

In the interest of illustrating how such a scheme may be used in practice, it is useful to rewrite it in a vector-matrix update form as

$$\mathbf{A}^n\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}^n\mathbf{u}^{n-1} + \mathbf{j}_e f_e^n + \mathbf{J}^n\mathbf{f}^n \quad (18)$$

where here, $\mathbf{A}^n$, $\mathbf{B}$ and $\mathbf{C}^n$ are $(N-1)\times(N-1)$ matrices defined as

$$\begin{aligned} \mathbf{A}^n &= (1+\sigma_0 k)\,\mathbf{I}_{N-1} + \left(\mathbf{a}^n\right)\left(\mathbf{a}^n\right)^T \\ \mathbf{B} &= 2\mathbf{I}_{N-1} + \left(\frac{k^2T}{\rho} + 2\sigma_1 k\right)\mathbf{D}_{xx} - \frac{EIk^2}{\rho}\mathbf{D}_{xxxx} \\ \mathbf{C}^n &= (\sigma_0 k - 1)\,\mathbf{I}_{N-1} - \left(\mathbf{a}^n\right)\left(\mathbf{a}^n\right)^T - 2\sigma_1 k\mathbf{D}_{xx} \end{aligned}$$

Due to the tension modulation nonlinearity, $\mathbf{A}^n$ and $\mathbf{C}^n$ are dependent on previously computed state values through the column vector $\mathbf{a}^n$, defined as

$$\mathbf{a}^n = \frac{k}{2}\sqrt{\frac{EAh}{\rho L}}\mathbf{D}_{xx}\mathbf{u}^n$$

The vector $\mathbf{j}_e$ is defined as $\mathbf{j}_e = k^2\mathbf{g}_e/\rho$, and $\mathbf{f}^n = [(\mathbf{f}_c^n)^T\,|f_f^n]^T$ is the consolidation of the contact forces due to the barrier and finger, with the combined matrix $\mathbf{J}^n$ given by $\mathbf{J}^n = k^2\mathbf{G}^n/\rho$, where $\mathbf{G}^n = [\mathbf{G}_c|-\mathbf{g}_f^n]$. Notice that $\mathbf{J}^n$ and $\mathbf{G}^n$ include effects of time variation due to the motion of the stopping finger.

The update form (18) requires the determination of the collision force vector $\mathbf{f}^n$; to this end, it may be rewritten as

$$\mathbf{u}^{n+1} = \mathbf{q}^n + \tilde{\mathbf{J}}^n\mathbf{f}^n \quad (19)$$

where

$$\mathbf{q}^n = \left(\mathbf{A}^n\right)^{-1}\left(\mathbf{B}\mathbf{u}^n + \mathbf{C}^n\mathbf{u}^{n-1} + \mathbf{j}_e f_e^n\right) \quad \tilde{\mathbf{J}}^n = \left(\mathbf{A}^n\right)^{-1}\mathbf{J}$$

Though the calculation of $\mathbf{q}^n$ and $\tilde{\mathbf{J}}^n$ might appear to require the full inversion of a matrix $\mathbf{A}^n$ (or at least a linear system solution), note that $\mathbf{A}^n$ is a rank one perturbation of a scaled identity matrix, and thus the inverse may be written directly, using the Sherman-Morrison-Woodbury formula [37] as

$$\left(\mathbf{A}^n\right)^{-1} = \frac{1}{1+\sigma_0 k}\left(\mathbf{I}_{N-1} - \frac{\left(\mathbf{a}^n\right)\left(\mathbf{a}^n\right)^T}{1+\sigma_0 k + \left(\mathbf{a}^n\right)^T\left(\mathbf{a}^n\right)}\right)$$

which leads to a matrix multiplication with $O(N)$ operations.

3.6. A Nonlinear Equation

Define the set of collision distances η^n as $\eta^n = [(\eta_c^n)^T\,|\eta_f^n]^T$. From the definitions (14) and (15), one then has

$$\eta^n = \begin{bmatrix}\mathbf{b}\\ -u_f^n\end{bmatrix} - h\mathbf{G}^n\mathbf{u}^n$$

From this, one may further define the vector $\mathbf{r}^n = [(\mathbf{r}_c^n)^T\,|r_f^n]^T$ as $\mathbf{r}^n = \eta^{n+1} - \eta^{n-1}$, and $\mathbf{r}^n$ may be written as

$$\mathbf{r}^n = \gamma^n - \mathbf{Z}\mathbf{f}^n - h\left(\left(\mathbf{G}^{n+1}\right)^T\mathbf{u}^{n+1} - \left(\mathbf{G}^{n-1}\right)^T\mathbf{u}^{n-1}\right) \quad (20)$$

where

$$\gamma^n = \begin{bmatrix}\mathbf{0}_{N_c,1}\\ -2\left(u_f^n - u_f^{n-1}\right) + \frac{k^2}{M_f}f_0^n\end{bmatrix}$$

where $\mathbf{0}_{N_c,1}$ is an N_c element column vector, and $\mathbf{Z}$ is an $(N_c+1)\times(N_c+1)$ matrix, all zero, except for a value of k^2/M_f as the entry at the lower right corner.

For the forces, from the definitions (14) and (15), one has

$$\mathbf{f}^n = \mathbf{\Lambda}^n + \mathbf{P}^n\mathbf{r}^n \quad (21)$$

where $\mathbf{\Lambda}^n$ is a diagonal $(N_c+1)\times(N_c+1)$ matrix with diagonal entries given by $\text{diag}\left(\mathbf{\Lambda}^n\right) = [(\lambda_c^n)^T\ |\lambda_f^n]^T$, with

$$\lambda_c^n = \frac{\phi_c\left(\mathbf{r}_c^n + \eta_c^{n-1}\right) - \phi_c\left(\eta_c^{n-1}\right)}{\mathbf{r}_c^n}$$

and

$$\lambda_f^n = \frac{\phi_f\left(r_f^n + \eta_f^{n-1}\right) - \phi_f\left(\eta_f^{n-1}\right)}{r_f^n}$$

and where $\mathbf{P}^n$ is an $(N_c+1)\times(N_c+1)$ matrix, all zero except for a value of $\Xi_f^n/(2k)$ in the lower right hand entry.

Finally, (19), (20) and (21) may be consolidated into a single vector nonlinear equation as

$$\mathbf{Q}^n\mathbf{r}^n + \mathbf{M}^n\Lambda^n + \mathbf{l}^n = \mathbf{0}$$

where

$$\begin{aligned}\mathbf{M}^n &= \mathbf{Z}^n + h\left(\mathbf{G}^{n+1}\right)^T\tilde{\mathbf{J}}^n\\ \mathbf{Q}^n &= \mathbf{I}_{N_c+1} + \mathbf{M}^n\mathbf{P}^n\\ \mathbf{l}^n &= -\gamma^n + h\left(\mathbf{G}^{n+1}\right)^T\mathbf{q}^n - h\left(\mathbf{G}^{n-1}\right)^T\mathbf{u}^{n-1}\end{aligned}$$

Numerically, such an equation may be solved using an iterative method such as, e.g., Newton-Raphson.

4. SIMULATION RESULTS

In this section, various features of simulations for the system described above are explored.

4.1. Visualization: Free Vibration

As a first example, consider a string positioned above a fretboard and a series of 12 frets, under a plucking action—see Figure 2, showing the time evolution of the string profile under different plucking forces. In one case, the string vibration is free from collision, but in the other, it is sufficient to allow for rebounding against the frets, greatly distorting the profile of the string subsequently. It should be noted that under normal lossy conditions, string vibration amplitude is decreased over time, and thus the collision with the fretboard will lead to transients; similarly, stiffness effects in the string lead to dispersion, also decreasing the maximum string displacement after the initial pluck.

4.2. Spurious Penetration

The penalty potential formulation intended to model the rigid collision between string and fretboard allows some unphysical penetration of the string into the fretboard itself. One question which emerges is then: how large is this penetration? For the plucked excitation simulation described in the previous section, the maximum penetration over the length of the string is plotted as a function of time step in Figure 3—in this case, it takes on values under 10^{-9} m, which is definitely acceptable in any acoustics simulation. The degree of penetration may be controlled through the choice of K—the larger it is, the less the penetration, with the side effect that the number of iterations required in Newton's method tends to increase. See Section 5 for more commentary on this point.

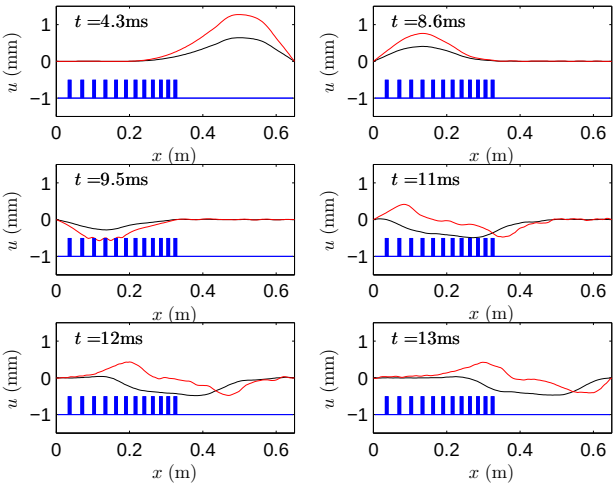


Figure 2: *Time evolution of the profile of a string in contact with a fretboard (in blue), under plucking excitations of different amplitudes—in black, with a maximal excitation of $f_p = 0.5$ N, and in red, with $f_p = 1$ N. In this case, the string is of parameters $L = 0.65$ m, $\rho = 5.25\times10^{-3}$ kg/m, $T = 60$ N, $E = 2\times10^{11}$ Pa, with radius $r = 4.3\times10^{-4}$ m, and loss parameters $\sigma_0 = 1.38$ and $\sigma_1 = 1.25\times10^{-4}$. The barrier collision parameters are $K = 10^{15}$ and $\alpha = 2.3$, and the pluck occurs pointwise at location $x = 0.52$ m. The sample rate is 88.2 kHz.*

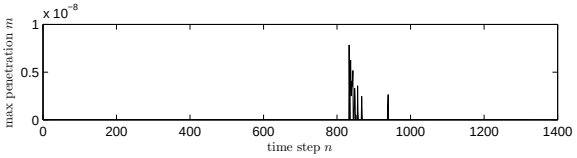


Figure 3: *Maximal penetration, in m, as a function of time step n, for the simulation described in Section 4.1.*

4.3. Visualization: Finger Tap

As a further example, consider the string under the application of a tapping gesture on the fretboard, as illustrated in Figure 4. In this case, the tapping is modelled (crudely!) as an unforced finger with an initial velocity rebounding from the string, accompanied by an intermediate pinning action against the fretboard itself. See Figure 4, illustrating the interaction of the finger with a string backed by a fretboard and a series of 12 frets, with parameters for the string and finger as given in the caption.

4.4. Energy Partition

In this example, the system has been assumed lossless, such that a plot of the energy partition for the system over time may be shown, as in Figure 5 at left; the finger energy is transferred first to the linear and nonlinear energy components of the string, then to the stored energy of the collision, when the string is in contact with the fretboard, and finally fully back to the finger, which rebounds with a speed identical to its initial speed. Notice in

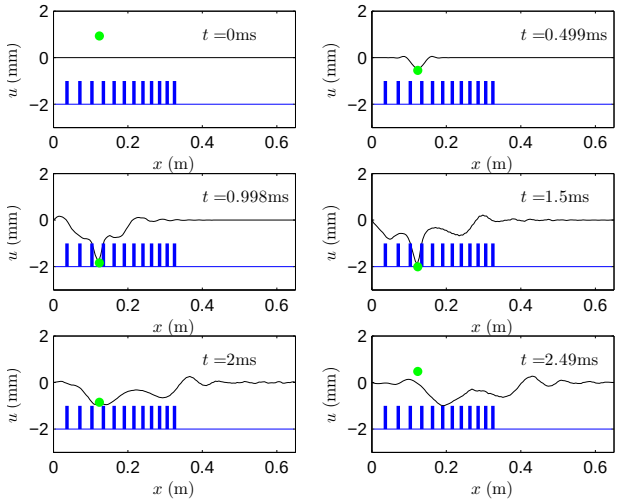


Figure 4: *Collision of an unforced finger (in green), with a string (in black) in contact with a fretboard (in blue). In this case, the string/fretboard parameters are as given in the caption to Figure 2, and the finger is of mass* 5×10^{-3} *kg, and approaches the string with velocity 3 m/s, at a position 0.012 from the end of the string. The finger collision potential parameters are* $K_f = 10^{10}$ *and* $\alpha_f = 2.3$ *and the sample rate is 88.2 kHz.*

particular contact/recontact phenomena visible in the energy of the string/fretboard collision. Energy is conserved to roughly 14 places in this case, as is visible in a plot of the normalized energy variation at right in Figure 5.

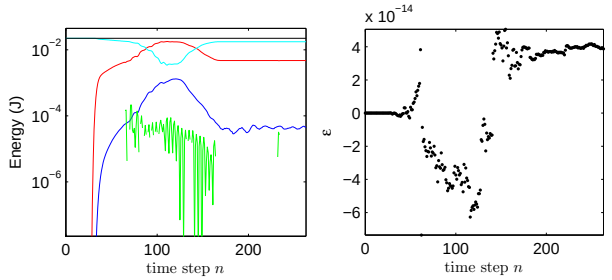


Figure 5: *Left: energy partition for the system of parameters as given in the caption to Figure 4, as a function of time step* n*. Linear string energy* $\mathfrak{h}_L$ *(red), nonlinear string energy* $\mathfrak{h}_K$*, (blue), string/barrier collision energy* $\mathfrak{h}_c$ *(green), finger energy* $\mathfrak{h}_f$ *(cyan) and total energy* $\mathfrak{h}$ *(black). Right: normalized energy variation* $\epsilon = (\mathfrak{h}^{n+1/2} - \mathfrak{h}^{1/2})/\mathfrak{h}^{1/2}$.

4.5. Time-varying Finger Position

As a final example, consider the same system, under the application of a sliding finger stop position—see Figure 6, showing snapshots of the string profile as the finger, under a constant applied force, slides across a single fret, effecting a pitch change. Here, the finger is assumed to act pointwise, at the position as indicated; notice in particular that due to the finite string stiffness, the slope of the string exhibits a strong variation at the fret location, and the minimum may occur at a location slightly shifted from that of the finger.

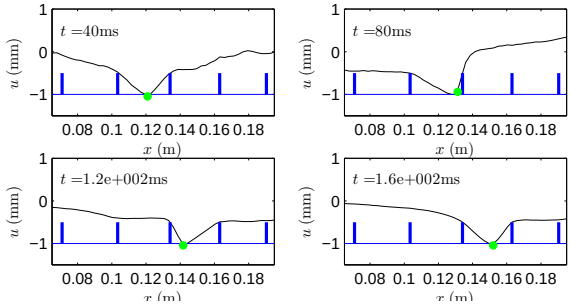


Figure 6: *Time evolution of string profile, for a string/barrier/finger system of parameters as described in the previous sections, where the finger, modelled pointwise, slides over a single fret during a playing gesture.*

5. CONCLUDING REMARKS

This paper is intended as an exploration of various features of string vibration in a more realistic setting, particularly involving the non-trivial contact of various components, including a barrier intended to represent a fretboard. Various features have been neglected here. The most important of these is the modelling of vibration in both polarizations; here, only the polarization transverse to the barrier has been modelled, allowing for an examination in particular of a colliding finger. In the case of excitation in the other polarization, however, a different nonlinear mechanism is required for the finger stopping, which closely resembles that of the bow-string interaction—see [12]. The other important element, not modelled here, is coupling to a body (in the case of, say, an acoustic guitar), and perhaps to the surrounding acoustic space. When such features are included, one is not far from a fully articulated model of a guitar, leaving, then, the enormous problem of gestural control—which is not considered here.

From a numerical point of view, a Hamiltonian potential formulation has been used here in order to arrive at a stable numerical method. As with all such stable methods, this leads to an implicit design in the nonlinear part of the problem (note that the linear part of the scheme, in isolation, remains explicit), and ultimately to a nonlinear vector algebraic equation to be solved at each time step. Though it is possible to show, for very simple systems such as a lumped mass colliding with a rigid barrier [38], and certain extensions to the distributed case [18], that a unique solution exists, in this vector case, a means of showing existence and uniqueness is not immediately forthcoming—meaning that, when an iterative method such as Newton-Raphson is employed it may either (a) not converge, or (b) converge to one solution which may be spurious. Thus an open question, for this and all nontrivial collision problems, is the determination of such uniqueness and existence conditions.

Beyond this basic question, at the level of the iterative solver

employed (in this case, Newton Raphson, but many others are available), there are further issues—one is that, even if existence and uniqueness results are available, convergence of a particular iterative method is not ensured. Another is that, in general, the iterative solver can prove to be something of a bottleneck not merely in terms of the over-all operation count (here, 50 iterations have been employed, for results to machine accuracy, though this can be significantly reduced for audio synthesis), but also in parallel implementations, where reducing the number of iterations (which must be performed serially) is of paramount importance.

6. REFERENCES

[1] J. O. Smith III, *Physical Audio Signal Processing*, Stanford, CA, 2004, Draft version. Available online at `http://ccrma.stanford.edu/~jos/pasp04/`.

[2] V. Välimäki, T. Tolonen, and M. Karjalainen, "Plucked-string synthesis algorithms with tension modulation nonlinearity," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Phoenix, Arizona, March 1999, vol. 2, pp. 977–980.

[3] T. Tolonen, V. Välimäki, and M. Karjalainen, "Modeling of tension modulation nonlinearity in plucked strings," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 3, pp. 300–310, 2000.

[4] S. Bilbao, "Energy-conserving finite difference schemes for tension-modulated strings," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Canada, May 2004, vol. 4, pp. 285–288.

[5] C. Vergez and X. Rodet, "A new algorithm for nonlinear propagation of sound wave: Application to a physical model of a trumpet," *Journal of Signal Processing*, vol. 4, pp. 79–88, 2000.

[6] B. Bank and L. Sujbert, "Generation of longitudinal vibrations in piano strings: From physics to sound synthesis," *J. Acoust. Soc. Am.*, vol. 117, no. 4, pp. 539–557, 2005.

[7] S. Bilbao, "A family of conservative finite difference schemes for the dynamical von Karman plate equations," *Numerical Methods for Partial Differential Equations*, vol. 24, no. 1, pp. 193–216, 2008.

[8] C. Raman, "On some Indian stringed instruments," *Proc. Ind. Assoc. Cult. Sci.*, vol. 7, pp. 29–33, 1922.

[9] J. Kappraff R. Burridge and C. Morshedi, "The sitar string, a vibrating string with a one-sided inelastic constraint," *SIAM J. Appl. Math.*, vol. 42, no. 6, pp. 1231–1251, 1982.

[10] M. Schatzman, "A hyperbolic problem of second order with unilateral constraints: The vibrating string with a concave obstacle," *J. Math. Anal. Appl.*, vol. 73, pp. 138–191, 1980.

[11] A. Krishnaswamy and J. O. Smith III, "Methods for simulating string collisions with rigid spatial obstacles," in *IEEE Workshop on Appl. of Siqnal Processing to Audio and Acoust.*, New Paltz, New York, October 2003, pp. 233–236.

[12] G. Evangelista, "Physical model of the string fret interaction," in *Proc. Int. Conf. Digital Audio Effects*, Paris, France, Sept. 2011, pp. 345–351.

[13] E. Rank and G. Kubin, "A waveguide model for slapbass synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, New Paltz, New York, October 1997, pp. 443–446.

[14] S. Siddiq, "A physical model of the nonlinear sitar string," *Arch. Acoust.*, vol. 37, no. 1, pp. 73–79, 2012.

[15] D. Kartofelev, A. Stulov, H.-M. Lehtonen, and V. Välimäki, "Modeling a vibrating string terminated against a bridge with arbitrary geometry," in *Proc. Stockholm Musical Acoust. Conf.*, Stockholm, Sweden, August 2013.

[16] C. Vyasarayani, S. Birkett, and J. McPhee, "Modeling the dynamics of a vibrating string with a finite distributed unilateral constraint: Application to the sitar," *J. Acoust. Soc. Am.*, vol. 125, no. 6, pp. 3673–3682, 2010.

[17] M. Frontini and L. Gotusso, "Numerical study of the motion of a string vibrating against an obstacle by physical discretization," *Appl. Math. Mod.*, vol. 14, pp. 489–494, 1990.

[18] A. Torin S. Bilbao and V. Chatzioannou, "Numerical modelling of collisions in musical instruments," Under review, Acta Acustica united with Acustica. Submitted version available at http://arxiv.org/abs/1405.2589.

[19] G. Evangelista and F. Eckerholm, "Player instrument interaction models for digital waveguide synthesis of guitar: Touch and collisions," *IEEE Transactions on Audio Speech and Language Processing*, vol. 18, no. 4, pp. 822–832, 2010.

[20] G. Cuzzucoli and V. Lombardo, A physical model of the classical guitar, including the player's touch, *Computer Music Journal*, vol. 23, no. 2, pp. 52–69, 1999.

[21] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," M.S. thesis, University of Illinois, 1969.

[22] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *J. Acoust. Soc. Am.*, vol. 95, no. 2, pp. 1112–1118, 1994.

[23] G. Kirchhoff, *Vorlesungen über Mechanik*, Tauber, Leipzig, 1883.

[24] G. F. Carrier, "On the nonlinear vibration problem of the elastic string," *Quarterly of Applied Mathematics*, vol. 3, pp. 157–165, 1945.

[25] T. Hélie and D. Roze, "Sound synthesis of a nonlinear string using Volterra series," *Journal of Sound and Vibration*, vol. 314, no. 1–2, pp. 275–306, 2008.

[26] H. Conklin, "Generation of partials due to nonlinear mixing in a stringed instrument," *J. Acoust. Soc. Am.*, vol. 105, no. 1, pp. 536–545, 1999.

[27] G. Derveaux, A. Chaigne, P. Joly, and E. Bécache, "Time-domain simulation of a guitar: Model and method," *J. Acoust. Soc. Am.*, vol. 114, no. 6, pp. 3368–3383, 2003.

[28] M. Laurson, C. Erkut, V. Välimäki, and M. Kuuskankare, "Methods for modeling realistic playing in acoustic guitar synthesis," *Computer Music Journal*, vol. 25, no. 3, pp. 38–49, 2001.

[29] J. Woodhouse, "On the synthesis of guitar plucks," *Acta Acustica united with Acustica*, vol. 90, pp. 928–944, 2004.

[30] G. Evangelista and J. O. Smith, "Structurally passive scattering element for modeling guitar pluck action," in *Proceedings of the 13th International Digital Audio Effects Conference*, Graz, Austria, September 2010, pp. 10–17.

[31] G. Horvay and A. Veluswami, "Hertzian impact of two elastic spheres in the presence of surface damping," *Acta Mechanica*, vol. 35, pp. 285–290, 1980.

[32] L. Rhaouti, A. Chaigne, and P. Joly, "Time-domain modeling and numerical simulation of a kettledrum," *J. Acoust. Soc. Am.*, vol. 105, no. 6, pp. 3545–3562, 1999.

[33] K. Hunt and F. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *ASME J. Appl. Mech.*, pp. 440–5, June 1975.

[34] J. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, pp. 1–435, SIAM, Philadelphia, 2004.

[35] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, John Wiley and Sons, Chichester, UK, 2009.

[36] S. Bilbao and J. O. Smith III, "Energy-conserving finite difference schemes for nonlinear strings," *Acta Acustica united with Acustica*, vol. 91, no. 2, pp. 299–311, 2005.

[37] T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[38] V. Chatziioannou and M. van Walstijn, "An energy conserving finite difference scheme for simulation of collisions.," in *Proc. Stockholm Musical Acoust. Conf.*, Stockholm, Sweden, 2013.

AN ENERGY CONSERVING FINITE DIFFERENCE SCHEME FOR THE SIMULATION OF COLLISIONS IN SNARE DRUMS

Alberto Torin[*]
Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
A.Torin@sms.ed.ac.uk

Brian Hamilton
Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
brian.hamilton@ed.ac.uk

Stefan Bilbao
Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
sbilbao@staffmail.ed.ac.uk

ABSTRACT

In this paper, a physics-based model for a snare drum will be discussed, along with its finite difference simulation. The interactions between a mallet and the membrane and between the snares and the membrane will be described as perfectly elastic collisions. A novel numerical scheme for the implementation of collisions will be presented, which allows a complete energy analysis for the whole system. Viscothermal losses will be added to the equation for the 3D wave propagation. Results from simulations and sound examples will be presented.

1. INTRODUCTION

Physics-based simulation of musical instruments is now an active research topic, both for acoustical studies and sound synthesis, and various numerical techniques can now tackle a wide range of complex systems. Percussion instruments, and drums in particular, with their various interacting components, constitute attractive and challenging target problems. From the first attempts at simulating single membranes in 2D, research has rapidly moved towards the simulation of complete instruments (see [1] for a review.) In recent years, a physical model for nonlinear circular membranes with snares has been proposed [2]. Finite difference methods have been employed to model timpani drums [3], snare drums [4] and nonlinear double-headed drums (i.e., tom toms and bass drums) [5].

In this paper, a physics-based simulation of a snare drum will be presented. The model consists of two membranes (batter and carry head), coupled with the surrounding air and connected by a rigid shell. A set of snares (thin metal wires) is placed below the carry head, in contact with it. In the present work, a novel energy conserving scheme for the simulation of collisions between the snares and the resonant membrane will be presented. This constitutes a major improvement with respect to previous attempts [4], for which numerical stability is not guaranteed (and is indeed a problem in implementation.) A similar approach can be adopted for the mallet-membrane interaction, which is included in this model, thus giving an energy conserving scheme for the whole system. When used as a sound synthesis tool, the usual 3D scheme describing the acoustic field produces artefacts that harm the quality of the sound. This problem can be addressed by adopting a more realistic model of 3D wave propagation that includes viscothermal losses.

A major issue broached in this paper is the numerical simulation of collisions, which play an important role in many fields, including engineering and computer graphics, and the literature on the subject is abundant (see [6] for a review). A mainstream approach for collision detection in many applications is the use of penalty-based methods, based on repulsive forces generated by slight interpenetration between the objects. In musical acoustics, many instruments rely on collisions for the production of sound, with an obvious example given by percussions. Several approaches have been used in the past, and in many cases this type of interaction has been modelled as a nonlinear Hertzian force depending on the mutual penetration of the colliding objects [7]. This model has been successfully adopted, e.g., for the simulation of the hammer-string interaction in pianos [8, 9].

For totally elastic collisions, these methods could be considered as unphysical, as they allow interpenetration in otherwise rigid bodies, and simulations of collision without the need for contact forces have been proposed [10]. Nonetheless, penalty-based methods have many advantages, as they offer a mathematically tractable and phenomenologically accurate description of the behaviour of the system, and will therefore be adopted in this study. Furthermore, the maximum penetration allowed can be bounded by choosing suitable values for the coefficients. When it comes to numerical schemes, the risk of instability is always present, and is particularly pronounced for rigid collisions. Energy-based finite difference approaches, which have a long history [11, 12], provide useful analysis tools in this sense but, for nonlinear interactions, existence and uniqueness of a solution are not always guaranteed. For the particular choice of penalty force used in this work, however, a uniqueness result has been proved recently in the case of a mass in contact with a rigid barrier [13].

This paper is organised as follows: a brief description of the

[*] This work was supported by the European Research Council, under grant StG-2011-279068-NESS.

underlying physical model will be given in Sec. 2, while its numerical implementation using finite difference methods will be discussed in Sec. 3. Sec. 4 presents an analysis of the implementation of the collision scheme; finally, some results and sound examples will be shown in Sec. 5.

2. DESCRIPTION OF THE MODEL

The geometry of the snare drum model under consideration is shown in Figure 1. Two circular membranes of equal radius R are positioned within a finite enclosure $\mathcal{V}$ of air, with which they are coupled. They are placed parallel to one another with centres along the z axis, and are defined over regions $\mathcal{M}_b$ at $z = z_b$ and $\mathcal{M}_c$ at $z = z_c$, respectively, with

$$\mathcal{M}_b = \mathcal{M}_c = \left\{(x, y) \,|\, x^2 + y^2 \leq R^2\right\}. \tag{1}$$

A rigid cylindrical cavity connects the membranes, by enclosing the portion of air between them ($z_c \leq z \leq z_b$).

As mentioned before, the important feature of snare drums is the presence of a set of snares in contact with the resonant membrane. Generally, these are 12-15 in number. For the sake of simplicity and to avoid the proliferation of notation, the following analysis will concentrate on a single snare of length L defined over a 1D domain $\mathcal{D}_s$. In implementation, however, it is straightforward to include several snares in the model.

The upper membrane is struck by a mallet, modelled as a lumped object, while the bottom membrane, together with the snare, is set into motion by the air pressure inside the cavity generated by the blow. Absorbing conditions are applied at the walls of the air box $\mathcal{V}$.

As a similar model has been employed already [4], some of the details of the system will be omitted here.

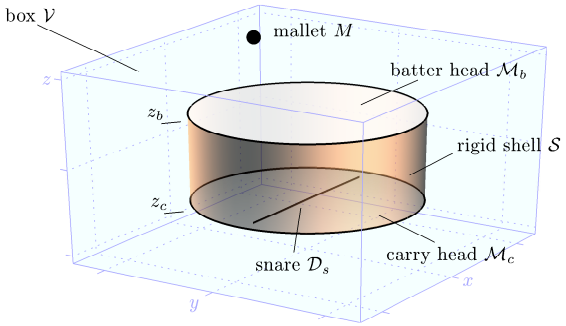


Figure 1: *Geometry of the model.*

2.1. Membranes

Let the index $i = b, c$ identify batter and carry head, respectively. The transverse displacements $w_i = w_i(x, y, t)$ of the membranes at some position $(x, y) \in \mathcal{M}_i$ and time t can be described by lossy wave equations with additional terms due to coupling conditions with the air and external collision forces. Batter and carry membrane equations read, respectively:

$$\rho_b \partial_{tt} w_b = \mathcal{L}_b[w_b] + \mathcal{F}_b^+ + \mathcal{F}_b^- + \mathcal{F}_M, \tag{2}$$

$$\rho_c \partial_{tt} w_c = \mathcal{L}_c[w_c] + \mathcal{F}_c^+ + \mathcal{F}_c^- + \mathcal{F}_s + \mathcal{F}_0 + \mathcal{F}_L. \tag{3}$$

with

$$\mathcal{L}_i[w_i] = T_i \Delta_{2D} w_i - 2\rho_i \sigma_{0,i} \partial_t w_i + 2\rho_i \sigma_{1,i} \Delta_{2D} \partial_t w_i, \tag{4}$$

where $\Delta_{2D} = \partial_{xx} + \partial_{yy}$ is the 2D Laplacian operator in Cartesian coordinates and ∂_t denotes partial time differentiation. $\mathcal{L}_i[w_i]$ groups together the linear terms in the wave equation, while the other terms are the air pressure exerted above ($\mathcal{F}_i^+$) and below ($\mathcal{F}_i^-$) each membrane. The last term $\mathcal{F}_M$ in (2) describes the mallet-membrane interaction. The equation for w_c is almost identical to (2), except for the form of the collision term $\mathcal{F}_s$ and for the presence of two additional terms $\mathcal{F}_0$ and $\mathcal{F}_L$ resolved at the two ends of the string attached to the membrane (see Sec. 2.5). The explicit expression for the coupling and collision terms will be discussed below, while the various physical parameters in (2), (3) and (4) are listed in Table 1. Additional terms, like stiffness or tension modulation nonlinearities, could be easily included in this model. For the sake of simplicity, their discussion is omitted in this work.

At the rim of both membranes, fixed boundary conditions are applied.

Table 1: *List of physical parameters used in this model.*

Membranes ($i = b, c$)	
$w_i(x, y, t)$	membrane displacement (m)
T_i	tension (N/m)
ρ_i	surface density (kg/m^2)
$\sigma_{0,i}$	frequency independent loss coefficient (1/s)
$\sigma_{1,i}$	frequency dependent loss coefficient (m^2/s)
Air	
$\Psi(x, y, z, t)$	acoustic velocity potential (m^2)
c_a	wave speed (m/s)
ρ_a	density (kg/m^3)
σ_a	viscothermal loss coefficient (m)
Mallet	
$z_M(t)$	mallet position (m)
M	mass (kg)
κ_M	stiffness parameter (N/m$^\alpha$)
α	nonlinear exponent
Snare	
$u(\chi, t)$	snare displacement (m)
T_s	tension (N)
ρ_s	linear density (kg/m)
$\sigma_{0,s}$	frequency independent loss coefficient (1/s)
$\sigma_{1,s}$	viscosity coefficient (m^2/s)
κ_s	stiffness parameter (N/m$^\beta$)
β	nonlinear exponent

2.2. Air

In this model, the equation for air propagation adopted is:

$$\partial_{tt} \Psi = c_a^2 \Delta_{3D} \Psi + c_a \sigma_a \Delta_{3D} \partial_t \Psi, \tag{5}$$

where $\Psi(x, y, z, t)$ is an acoustic velocity potential as in [14] and $\Delta_{3D} = \partial_{xx} + \partial_{yy} + \partial_{zz}$ is the 3D Laplacian operator. The coefficient σ_a for viscothermal losses generally depends on various physical parameters, among which temperature and humidity, but values for it generally lie in the range of 10^{-7} to 10^{-6} m [14].

The drum shell $\mathcal{S}$ is modelled as a rigid, reflective boundary encircling the cylindrical region between the membranes. This can be obtained by imposing the normal derivative of Ψ to be zero across the shell:

$$\mathbf{n} \cdot \nabla_{3D} \Psi = 0. \tag{6}$$

where ∇_{3D} is the gradient and $\mathbf{n}$ is the unit vector normal to the shell surface. Absorbing conditions are applied over the boundaries $\partial\mathcal{V}$ of the computational region. In this work, first order Engquist-Majda conditions will be adopted [15], as they are easy to implement within an energy-based framework. Another possibility is the use of PMLs [16].

2.3. Coupling conditions

Coupling conditions between the membranes and air can be obtained by imposing the continuity of pressure and velocity at the interface. In terms of Ψ, these conditions may be written as:

$$\mathcal{F}_i^+ = -\rho_a \lim_{z \to z_i^+} \partial_t \Psi \mid_{\mathcal{M}_i} \qquad \mathcal{F}_i^- = \rho_a \lim_{z \to z_i^-} \partial_t \Psi \mid_{\mathcal{M}_i}, \tag{7}$$

$$\partial_t w_i = - \lim_{z \to z_i^-} \partial_z \Psi \mid_{\mathcal{M}_i} = - \lim_{z \to z_i^+} \partial_z \Psi \mid_{\mathcal{M}_i} . \tag{8}$$

These conditions hold over the membrane regions $\mathcal{M}_b$ and $\mathcal{M}_c$.

2.4. Mallet interaction

The mallet exciting the membrane is modelled as a lumped, but not necessarily point-like object, with mass M and position $z_M(t) \in \mathbb{R}$ measured relatively to z_b. Let the contact region over $\mathcal{M}_b$ be defined by a distribution $g_b(x, y)$, with $\int_{\mathcal{M}_b} g_b = 1$. For a mallet striking the membrane from above, the equation of motion and the collision term appearing in (2) can be written as:

$$M \ddot{z}_M = f_M, \qquad \mathcal{F}_M = -g_b \, f_M, \tag{9}$$

where the dot symbol represents total time differentiation. It is usual in the literature to express the collision force f_M as a power law in terms of the mutual interpenetration η of the two objects [3, 7]:

$$f_M = \kappa_M [\eta]_+^\alpha, \quad \eta = \int_{\mathcal{M}_b} g_b \, w_b dx \, dy - z_M \tag{10}$$

with stiffness parameter $\kappa_M > 0$ and $\alpha > 1$, and which is active only when $\eta > 0$; the symbol $[\,\cdot\,]_+$ is used in this article to indicate the positive part, $[\eta]_+ = (\eta + |\eta|)/2$. Such an approach traces its origins in the work of Hertz at the end of 19th century (see [6] for a historical review.)

An equivalent approach, which leads to an energy conserving numerical scheme (see below), is to express the collision force f_M as the derivative of a potential Φ_M, which again will depend on the average distance η between the mallet and the membrane:

$$f_M = \frac{d\Phi_M}{d\eta} = \frac{\dot{\Phi}_M}{\dot{\eta}}, \qquad \Phi_M = \frac{\kappa_M}{\alpha + 1} [\eta]_+^{\alpha+1}. \tag{11}$$

2.5. Snare

A single snare can be modelled as a 1D string with internal losses and an additional term describing the collisions with the membrane. The equation of motion can thus be written as:

$$\rho_s \partial_{tt} u = T_s \partial_{\chi\chi} u - 2\sigma_{0,s} \partial_t u + 2\sigma_{1,s} \partial_{\chi\chi} \partial_t u - \mathcal{F}_e. \tag{12}$$

Note the change of sign in the collision force density $\mathcal{F}_e$, as in this case the string is striking the membrane from below. A stiffness term could be included as well, without complicating too much the implementation. As before, in order to define a function $G_s(x, y)$ that distributes collisions over the membrane, it is necessary to introduce a two-element affine mapping $\boldsymbol{\pi}(\chi) : \mathcal{D}_s \to \mathcal{M}_s$, from the 1D domain of the snare to the resonant membrane, that projects each point of the string onto the corresponding point on the membrane above it. A natural choice for G_s is

$$G_s(\mathbf{x}, \chi) = \delta^{(2)}(\mathbf{x} - \boldsymbol{\pi}(\chi)), \tag{13}$$

where $\mathbf{x} = (x, y)$ and $\delta^{(2)}$ is a 2D Dirac delta function. The collision density $\mathcal{F}_s$ can thus be written as

$$\mathcal{F}_s = \int_{\mathcal{D}_s} G_s(\mathbf{x}, \chi) \mathcal{F}_e(\chi) d\chi. \tag{14}$$

Analogously to the mallet-membrane case, $\mathcal{F}_e(\chi)$ can be written in terms of a distributed potential $\Phi_s(\chi)$:

$$\mathcal{F}_e(\chi) = \frac{\partial_t \Phi_s}{\partial_t \xi}, \tag{15}$$

with

$$\Phi_s(\chi) = \frac{\kappa_s}{\beta + 1} [\xi]_+^{\beta+1}, \qquad \xi(\chi) = u - \int_{\mathcal{M}_c} G_s \, w_c \, dx \, dy. \tag{16}$$

Once again, note the change in sign in the definition of $\xi(\chi)$ compared to the corresponding quantity η.

The choice of perfectly elastic collisions between the snare and the membrane must be considered only as a starting point for simulation. More refined models that introduce damping in contact forces can be adopted, like that proposed by Hunt and Crossley [17], and could be perceptually important in determining the decay time of the sound. This model requires, however, an experimental investigation of the loss coefficient, which is outside the scope of this paper.

At the end points of the snare, boundary conditions must be carefully analysed. Let $g_0 = \delta^{(2)}(\mathbf{x} - \boldsymbol{\pi}(0))$ be the distribution function from the end of the snare at $\chi = 0$ to the corresponding point on the membrane $\mathcal{M}_c$. When the snare is attached to the membrane, their displacements must be the same at the end point:

$$u|_{\chi=0} = \int_{\mathcal{M}_c} g_0 \, w_c \, dx \, dy. \tag{17}$$

Furthermore, the force density acting on $\mathcal{M}_c$ at the same point can be written as:

$$\mathcal{F}_0 = g_0 \, f_0, \qquad f_0 = T_s \partial_\chi u|_{\chi=0} + 2\rho_s \sigma_{1,s} \partial_{t\chi} u|_{\chi=0}. \tag{18}$$

These expressions can be easily arrived at through energy analysis techniques (see Sec. 2.6). Analogous conditions can be written for the edge at $\chi = L$.

2.6. Energy balance

A thorough analysis of the model presented above by means of frequency methods can be ruled out, given the simultaneous presence of several interacting components with strongly non-linear couplings and irregular geometry. To this end, an alternative approach is given by energy methods.

One way of calculating the energy of the system is to multiply Eqs. (2), (3), (5), (9) and (12) by the first time derivative of the variable on the left side of the equation, and then to integrate over the corresponding domain (e.g., multiply (2) by $\partial_t w_b$ and integrate over $\mathcal{M}_b$, etc.) Using integration by parts leads to an energy balance and to the determination of suitable boundary and coupling conditions, as outlined below.

An energy balance for the whole system can be arrived at by summing the contributions for the various components, and can be written as

$$\frac{d\mathfrak{H}}{dt} = -\mathfrak{Q} + \mathfrak{B}, \tag{19}$$

where $\mathfrak{H} = \mathfrak{H}_b + \mathfrak{H}_c + \mathfrak{H}_M + \mathfrak{H}_s + \mathfrak{H}_a$ is the total energy of the system, $\mathfrak{Q}$ represents all the loss terms and $\mathfrak{B}$ groups together boundary terms. The explicit expressions for the various contributions to $\mathfrak{H}$ are given below:

$$\mathfrak{H}_i = \int_{\mathcal{M}_i} \frac{\rho_i}{2}(\partial_t w_i)^2 + \frac{T_i}{2}|\nabla_{2D} w_b|^2 \, dx\, dy, \quad i = b, c, \tag{20a}$$

$$\mathfrak{H}_a = \int_{\mathcal{V}} \frac{\rho_a}{2c_a^2}(\partial_t \Psi)^2 + \frac{\rho_a}{2}|\nabla_{3D}\Psi|^2 \, dx\, dy\, dz, \tag{20b}$$

$$\mathfrak{H}_M = \frac{M}{2}\dot{z}_M^2 + \Phi_M, \tag{20c}$$

$$\mathfrak{H}_s = \int_{\mathcal{D}_s} \frac{\rho_s}{2}(\partial_t u)^2 + \frac{T_s}{2}(\partial_\chi u)^2 + \Phi_s \, d\chi, \tag{20d}$$

where ∇_{2D} represents the 2D gradient. In order for the scheme to be energy conserving, all the terms in $\mathfrak{B}$ must sum to zero, and it is indeed the case here, while contributions to $\mathfrak{Q}$ come from the loss terms in the membranes' and snare's equations, plus from absorbing conditions over the boundaries of $\mathcal{V}$. It can be shown that each of these individual terms is positive, thus leading to a net dissipation of energy in the system.

3. FINITE DIFFERENCE SCHEMES

In this section, the implementation of the model described above will be carried out using the finite difference method [18].

The discretisation in space of the various components will be performed over different Cartesian grids in 1D, 2D or 3D depending on the dimension of the domain. Time discretisation, instead, will be unique for the entire system, with temporal step $k = 1/F_s$ defined as the inverse of the sampling frequency F_s. Spatial grid steps can be derived in terms of k according to stability conditions analysed below. A one dimensional function, like $u(\chi, t)$ for example, will be approximated by a discrete function u_l^n, over a grid with step h_s (where n and l represent the time and spatial index, respectively.) However, it is very convenient to represent grid functions as column vectors, regardless of their dimensions. If in the 1D case it is obvious how to perform such operation, in the 2D and 3D cases several options are available. On a 2D grid, points will be grouped columnwise along the y axis, while in the 3D the same operation will be applied to successive horizontal slices along the vertical axis for increasing values of z.

Let $\mathbf{u}^n$ be the vectorised form of u_l^n. For such a variable, one can define forward and backward time shift operators as following:

$$e_{t+}\mathbf{u}^n = \mathbf{u}^{n+1}, \qquad e_{t-}\mathbf{u}^n = \mathbf{u}^{n-1}. \tag{21}$$

Time difference and averaging operators can be obtained from combinations of the previous ones, and are listed in Table 2. Space difference operators, when operating on vectors, can be expressed as matrices [18].

Table 2: *List of time difference and averaging operators.*

Time difference operators	
$\delta_{t+} = (e_{t+} - 1)/k$	forward difference
$\delta_{t-} = (1 - e_{t-})/k$	backward difference
$\delta_{t\cdot} = (e_{t+} - e_{t-})/2k$	centred difference
$\delta_{tt} = (e_{t+} - 2 + e_{t-})/k^2$	second difference
Time averaging operators	
$\mu_{t+} = (e_{t+} + 1)/2$	forward average
$\mu_{t-} = (1 + e_{t-})/2$	backward average
$\mu_{t\cdot} = (e_{t+} + e_{t-})/2$	centred average

3.1. Membranes

Let $\mathbf{w}_i^n$ be the discrete approximations in vector form of the membranes' displacements $w_i(x, y, t)$ over grids of spacing h_i, with $i = b, c$. Equations (2) and (3) can be thus discretised as

$$\rho_b \delta_{tt}\mathbf{w}_b^n = \mathfrak{l}_b[\mathbf{w}_b^n] + \mathfrak{f}_b^{+,n} + \mathfrak{f}_b^{-,n} + \mathfrak{f}_M^n, \tag{22}$$

$$\rho_c \delta_{tt}\mathbf{w}_c^n = \mathfrak{l}_c[\mathbf{w}_c^n] + \mathfrak{f}_c^{+,n} + \mathfrak{f}_c^{-,n} + \mathfrak{f}_s^n + \mathfrak{f}_0^n + \mathfrak{f}_L^n. \tag{23}$$

The operator $\mathfrak{l}_i[\mathbf{w}_i^n]$ is the discrete counterpart of (4):

$$\mathfrak{l}_i[\mathbf{w}_i^n] = T_i \mathbf{D}_{\boxplus,i}\mathbf{w}_i^n - 2\rho_i\sigma_{0,i}\delta_{t\cdot}\mathbf{w}_i^n + 2\rho_i\sigma_{1,i}\delta_{t-}\mathbf{D}_{\boxplus,i}\mathbf{w}_i^n \tag{24}$$

where $\mathbf{D}_{\boxplus,i}$ is the matrix form of the 2D Laplacian Δ_{2D}, which is generally different between the two membrane grids.

3.2. Air

Let $\mathbf{\Psi}^n$ be a discrete approximation of $\Psi(x, y, z, t)$ over a 3D grid of spacing h_a. A finite difference approximation for (5) can be written as

$$\delta_{tt}\mathbf{\Psi}^n = c_a^2 \mathbf{D}_{\boxplus\!\!\boxplus}\mathbf{\Psi}^n + c_a\sigma_a\delta_{t-}\mathbf{D}_{\boxplus\!\!\boxplus}\mathbf{\Psi}^n, \tag{25}$$

where $\mathbf{D}_{\boxplus\!\!\boxplus}$ is the matrix representation of the 3D Laplacian Δ_{3D}.

The last term introduces a frequency-dependent loss that increases with frequency. It is critical to include viscothermal losses in this model in order to suppress spurious artefacts that are perceptually very relevant. More will be said about this in Sec. 5.3.

The implementation of boundary conditions over the shell, absorbing conditions over the walls and coupling conditions with the membranes will be omitted, as they have been analysed several times in recent works. The interested reader is referred to [4, 5].

3.3. Mallet

Let z_M^n and f_b^n be the sampled versions at time $t = nk$ of $z_M(t)$ and $f_b(t)$, respectively. Equation (9) becomes:

$$M\delta_{tt}z_M^n = f_b^n, \qquad \mathfrak{f}_b^n = -\mathbf{g}_b f_b^n, \tag{26}$$

where $\mathbf{g}_b$ is a column vector representing the distribution $g_b(x, y)$. Normalisation is obtained by imposing $h_b^2\mathbf{1}^T\mathbf{g}_b = 1$, where $\mathbf{1}^T$ is the transpose of a column vector consisting of ones.

As discussed in Sec. 2.4, f_b^n can be expressed in terms of a discrete potential Φ_M^n:

$$f_b^n = \frac{\delta_{t\cdot}\Phi_M^n}{\delta_{t\cdot}\eta^n}, \qquad \eta^n = h_b^2\mathbf{g}_b^T\mathbf{w}_b^n - z_M^n, \tag{27}$$

where $\Phi_M^n = \Phi_M(\eta^n)$.

3.4. Snare

The displacement $u(\chi, t)$ of the snare can be represented by the vector $\mathbf{u}^n$, over a 1D grid of spacing h_s. Equation (12) can be written as:

$$\rho_s\delta_{tt}\mathbf{u}^n = T_s\mathbf{D}_{\chi\chi}\mathbf{u}^n - 2\rho_s\sigma_{0,s}\delta_{t\cdot}\mathbf{u}^n + \\ + 2\rho_s\sigma_{1,s}\delta_{t-}\mathbf{D}_{\chi\chi}\mathbf{u}^n - \mathfrak{f}_e^n, \tag{28}$$

where $\mathbf{D}_{\chi\chi}$ is the matrix representation of the operator $\partial_{\chi\chi}$. The discrete version $\mathfrak{f}_s$ of the collision force density $\mathcal{F}_s$ in (14) is defined as:

$$\mathfrak{f}_s^n = \mathbf{G}_s\mathfrak{f}_e^n, \tag{29}$$

where $\mathbf{G}_s$ is the matrix form of the linear operator $\int_{\mathcal{D}_s} G_s(\,\cdot\,)\,d\chi$. As in the mallet case, it is possible to express $\mathfrak{f}_e$ in terms of a discrete potential $\mathbf{\Phi}_s^n$

$$\mathfrak{f}_e^n = \frac{\delta_{t\cdot}\mathbf{\Phi}_s^n}{\delta_{t\cdot}\boldsymbol{\xi}^n}, \tag{30}$$

$$\mathbf{\Phi}_s^n = \frac{\kappa_s}{\beta+1}[\boldsymbol{\xi}^n]_+^{\beta+1}, \qquad \boldsymbol{\xi}^n = \mathbf{u}^n - \mathbf{G}_s^T\mathbf{w}_c^n. \tag{31}$$

The vector by vector division in (30) is intended here and in the remainder of the article as an element-by-element operation.

At the end point $l = 0$, continuous boundary conditions (17) and (18) can be discretised as

$$u_0^n = h_c^2\mathbf{g}_0^T\mathbf{w}_c^n, \tag{32a}$$

$$\mathfrak{f}_0^n = \mathbf{g}_0 f_0^n, \quad f_0^n = (T_s\delta_{\chi-} + 2\rho_s\sigma_{1,s}\delta_{t-}\delta_{\chi-})u_0^n, \tag{32b}$$

where $\mathbf{g}_0$ is the discrete approximation of the distribution g_0. Analogous expressions can be found for the other end point. When applied to the grid point u_0^n, the operator $\delta_{\chi-}$ would give:

$$\delta_{\chi-}u_0^n = (u_0^n - u_{-1}^{*,n})/h_s. \tag{33}$$

As $u_{-1}^{*,n}$ lies outside of the 1D grid, it is sometimes called virtual or ghost point (hence the notation *). Equation (32b) must be considered as a formal way of determining suitable update conditions for the scheme (see Sec. 4.2.)

3.5. Energy and Stability

In the numerical case, an energy balance corresponding to (19) can be written as:

$$\delta_{t-}\mathfrak{h}^{n+1/2} = -\mathfrak{q}^n + \mathfrak{b}^n, \tag{34}$$

where $\mathfrak{h}^{n+1/2}$ is the numerical energy of the system at time $(n + 1/2)k$, $\mathfrak{q}^n$ represents losses and $\mathfrak{b}^n$ the boundary terms. As in the continuous case, $\mathfrak{h}^{n+1/2}$ can be written as a sum of the following terms:

$$\mathfrak{h}_i^{n+1/2} = h_i^2\left(\frac{\rho_i}{2}|\delta_{t+}\mathbf{w}_i^n|^2 + \frac{T_i}{2}((\mathbf{D}_{x+}\mathbf{w}_i^n)^T\cdot e_{t+}(\mathbf{D}_{x+}\mathbf{w}_i^n)) \right. \\ \left. + \frac{T_i}{2}((\mathbf{D}_{y+}\mathbf{w}_i^n)^T\cdot e_{t+}(\mathbf{D}_{y+}\mathbf{w}_i^n))\right), \quad i = b, c, \tag{35}$$

$$\mathfrak{h}_a^{n+1/2} = h_a^3\left(\frac{\rho_a}{2c_a^2}|\delta_{t+}\mathbf{\Psi}^n|^2 + \frac{\rho_a}{2}((\mathbf{D}_{x+}\mathbf{\Psi}^n)^T\cdot e_{t+}(\mathbf{D}_{x+}\mathbf{\Psi}^n)) \right. \\ + \frac{\rho_a}{2}((\mathbf{D}_{y+}\mathbf{\Psi}^n)^T\cdot e_{t+}(\mathbf{D}_{y+}\mathbf{\Psi}^n)) \\ \left. + \frac{\rho_a}{2}((\mathbf{D}_{z+}\mathbf{\Psi}^n)^T\cdot e_{t+}(\mathbf{D}_{z+}\mathbf{\Psi}^n))\right), \tag{36}$$

$$\mathfrak{h}_M^{n+1/2} = \frac{M}{2}(\delta_{t+}z_M^n)^2 + \mu_{t+}\Phi_M^n, \tag{37}$$

$$\mathfrak{h}_s^{n+1/2} = h_s\left(\frac{\rho_s}{2}|\delta_{t+}\mathbf{u}^n|^2 + \frac{T_s}{2}((\mathbf{D}_{\chi+}\mathbf{u}^n)^T\cdot e_{t+}(\mathbf{D}_{\chi+}\mathbf{u}^n)) \right. \\ \left. + \mathbf{1}^T\mu_{t+}\mathbf{\Phi}_s^n\right), \tag{38}$$

where $|\cdot|$ denotes the Euclidean norm of a vector, and the various difference matrices represent forward spatial difference operators [19]. It is understood that, in the air term, the z derivative be calculated everywhere but across the two membranes. The boundary term $\mathfrak{b}^n$ is identically zero. When the system is lossless, and reflective conditions are applied over the walls of the box, the total energy $\mathfrak{h}^{n+1/2}$ is conserved to machine accuracy. See Sec. 5.1 for details. Otherwise, energy is monotonically dissipated.

By requiring that all the energy terms be positive, stability conditions for the schemes can be arrived at. For the membranes and snare schemes, the presence of collisions does not alter the usual conditions:

$$h_i^2 \geq 2k^2T_i/\rho_i + 8\sigma_{1,i}k, \quad i = b, c, \tag{39}$$

$$h_s^2 \geq k^2T_s/\rho_s + 4\sigma_{1,s}k. \tag{40}$$

For the air scheme, stability condition depends on σ_a:

$$h_a^2 \geq 3c_a^2k^2 + 6c_a\sigma_a k. \tag{41}$$

4. NUMERICAL IMPLEMENTATION

In this section, the numerical implementation of mallet-membrane and snare-membrane collisions will be discussed. In both cases, it is necessary to solve a nonlinear equation at every time step. Existence and uniqueness of solution will be analysed.

4.1. Mallet-membrane collision

Consider the mallet and batter membrane schemes first. The update for the membrane points included in the distribution $\mathbf{g}_b$ will be coupled to the mallet's position by the collision force f_b^n. When all the terms in (22) and (26) are expanded and air coupling is taken

into account, the update expressions for $\mathbf{w}_b^{n+1}$ and z_M^{n+1} can be schematically written as

$$\mathbf{A}_b \mathbf{w}_b^{n+1} = \boldsymbol{\omega}_b^n[\mathbf{w}_b^n, \mathbf{w}_b^{n-1}, \boldsymbol{\Psi}^n, \boldsymbol{\Psi}^{n-1}] - \frac{k^2}{\rho_b}\mathbf{g}_b f_b^n, \quad (42)$$

$$z_M^{n+1} = \zeta_M^n[z_M^n, z_M^{n-1}] + k^2 f_b^n / M, \quad (43)$$

where $\mathbf{A}_b$ is a symmetric, positive definite matrix due to losses and air coupling, and $\boldsymbol{\omega}_b^n$ and ζ_M^n represent linear combinations of known terms from previous time steps. These two components can be updated by finding η^{n+1} first, then by calculating f_b^n and finally by inserting it in (42) and (43). To this end, start by inverting the system (42) and by multiplying it by $h_b^2 \mathbf{g}_b^T$, then subtract (43). After a brief calculation, it is possible to write a nonlinear equation in $r^n = \eta^{n+1} - \eta^{n-1}$ which must be solved at every time step:

$$r^n + \gamma \frac{\Phi_M(r^n + a^n) - \Phi_M(a^n)}{r^n} + b^n = 0, \quad (44)$$

where

$$\gamma = \frac{h_b^2 \mathbf{g}_b^T \mathbf{A}_b^{-1} \mathbf{g}_b k^2}{\rho_b} + \frac{k^2}{M}, \qquad a^n = \eta^{n-1}, \quad (45a)$$

$$b^n = \zeta^n - h_b^2 \mathbf{g}_b^T \mathbf{A}_b^{-1} \boldsymbol{\omega}_b^n + \eta^{n-1}. \quad (45b)$$

The particular choice of a power law nonlinearity for Φ_M guarantees a unique solution for (44), as has been shown in [13] for a simpler case.

4.2. Snare-membrane collision

The implementation of the snare-membrane interaction is somewhat complicated by the fact that the snare is a distributed object, and by the presence of air coupling and boundary conditions at the end points. As before, Eqs. (23) and (28) can be schematically written as

$$\mathbf{w}_c^{n+1} = \mathbf{A}_c^{-1} \boldsymbol{\omega}_c^n + \frac{k^2}{\rho_c} \mathbf{A}_c^{-1} \left(\mathbf{g}_0 f_0^n + \mathbf{g}_L f_L^n + \mathbf{G}_s \mathfrak{f}_e^n\right) \quad (46)$$

$$\mathbf{u}^{n+1} = \boldsymbol{\upsilon}^n / q_s - k^2 \mathfrak{f}_e^n / (\rho_s q_s), \quad (47)$$

where $\boldsymbol{\omega}_c^n[\mathbf{w}_c^n, \mathbf{w}_c^{n-1}, \boldsymbol{\Psi}^n, \boldsymbol{\Psi}^{n-1}]$ and $\boldsymbol{\upsilon}^n[\mathbf{u}^n, \mathbf{u}^{n-1}]$ depend on known values of the various variables, $\mathbf{A}_c$ is a constant matrix analogous to $\mathbf{A}_b$ and $q_s = 1 + \sigma_{0,s} k$.

One way to proceed in order to solve this system is to start by solving for the pointwise forces f_0 and f_L. Using (32b), it is possible to write

$$u_0^{n+1} = -\frac{k^2}{q_s \rho_s} f_0^n + \frac{1}{q_s} \upsilon_0^n, \quad (48)$$

where the first term on the right hand side replaces derivatives that could not otherwise be calculated, and $f_{s,0}^n = 0$ because the snare and the membrane are attached. By multiplying (46) by $h_c^2 \mathbf{g}_0^T$ and by imposing condition (32a), it is possible to write f_0^n in terms of the (still unknown) force density $\mathfrak{f}_e^n$:

$$f_0^n = \phi_0^n - \boldsymbol{\nu}^T \mathfrak{f}_e^n, \quad (49)$$

with ϕ_0^n combination of known terms and $\boldsymbol{\nu}$ a constant vector. A similar process can be repeated for f_L.

Now, after substituting these expressions for f_0^n and f_L^n back into (46), it is possible to follow the same procedure used for the mallet-membrane case. Multiplying (46) by $h_c^2 \mathbf{G}_s^T$ and subtracting this from (47) leads to a nonlinear equation in the unknown vector $\mathbf{r}^n = \boldsymbol{\xi}^{n+1} - \boldsymbol{\xi}^{n-1}$ formally similar to (44):

$$\mathbf{r}^n + \boldsymbol{\Gamma} \frac{\boldsymbol{\Phi}_s(\mathbf{r}^n + \mathbf{a}^n) - \boldsymbol{\Phi}_s(\mathbf{a}^n)}{\mathbf{r}^n} + \mathbf{b}^n = 0, \quad (50)$$

where $\boldsymbol{\Gamma}$ is a constant, symmetric and positive definite matrix, and $\mathbf{b}^n$ and $\mathbf{a}^n$ depend only on known values. Once this equation is solved, it is possible to calculate $\mathfrak{f}_e^n$, and therefore to update the rest of the scheme explicitly. Uniqueness of a solution in the vector case is guaranteed by the special form of $\boldsymbol{\Gamma}$ [20].

As explained in Sec. 2, the present analysis has concentrated on a single snare for simplicity sake. However, it is straightforward to extend the derivation of the previous section to $N_s > 1$ snares. The grid values for the various snares can be consolidated in a single vector, and expressions like (50) still hold. Conditions involving the end points, instead, will be transformed into vectors of size N_s, and their values will generally be coupled.

5. RESULTS

5.1. Energy conservation

As discussed in Sec. 3.5, the numerical energy of the system can be calculated, and must remain constant to machine accuracy in the lossless case and without absorbing conditions over the walls of $\mathcal{V}$. Figure 2 shows the normalised variations of $\mathfrak{h}$, together with the partition into the various components. Such an energy measure can be extremely useful for debugging purposes, as virtually any error has an impact on the conservation of $\mathfrak{h}$. The drum is excited by a mallet with $M = 0.028$ kg and initial velocity $v = -5$ m/s at $t = 0$ s. Seven snares are included in the model.

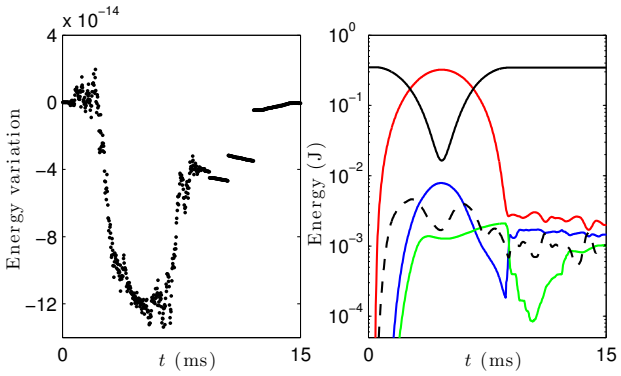


Figure 2: Left: normalised variations of the total energy $\mathfrak{h}$ for a snare drum in the lossless case. Right: contribution to the total energy given by the various components (solid black: mallet, red: upper membrane, blue: lower membrane, dashed black: air, green: snares). The sample rate is 44 100 Hz.

5.2. Evolution of the system

Figure 3 schematically illustrates what happens when the drum is excited with parameters given in the previous section. A positive pressure due to the compression of the membrane is generated inside the cavity, which pushes the lower membrane downwards,

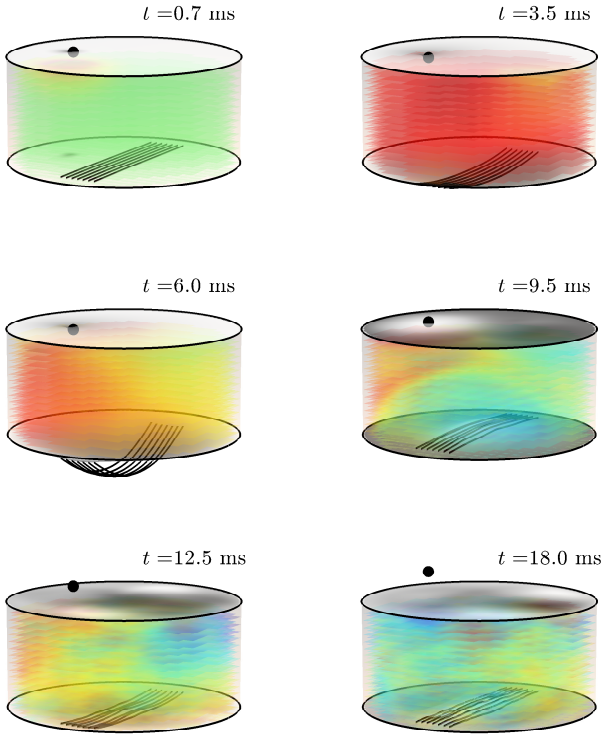


Figure 3: Snapshots of the evolution of the snare drum system at times as indicated. The pressure variations from atmospheric inside the cavity are depicted (green: zero variation, red: positive variation, blue: negative variation.) Displacements have been altered for illustration purposes.

together with the snares. The snares reach their maximum displacement at $t = 6$ ms, when they start to move upwards. At about $t = 9$ ms, the snares hit the membrane almost coherently (notice the pressure wave generated by the impact). At later times, the behaviour of the snares becomes rapidly chaotic.

5.3. A note on dispersion error, ABCs and viscosity

It is well-known for the lossless case ($\sigma_a = 0$ m) that the scheme employed for the 3D air box exhibits significant dispersion error [21]. Dispersion error essentially means that the ideal linear relationship between temporal and spatial frequencies is warped in the finite difference scheme. Numerical wave speed is thus dependent on direction and frequency. In this case, high frequencies tend to lag along the axial directions. While more accurate schemes exist for minimising dispersion, such as interpolated schemes [21], due to the complexity of the full 3D drum embedding presented here and with the goal of presenting a complete energy analysis, such schemes are currently distant options. This 3D scheme is often used with a large oversampling of the grid in order to reduce dispersion error to acceptable levels, such as in [3] where a 24 kHz sampling rate was used for a 700 Hz output. The full audible bandwidth is of interest here so oversampling was not employed, since computational costs rise drastically when reducing the time-step (16x increase for doubling of the sampling rate).

The presence of dispersion error causes some challenges when absorbing boundary conditions (ABCs) are used. The absorbing boundaries employed here are of the first-order Engquist-Majda type:

$$(\partial_t - c_a \mathbf{n} \cdot \vec{\nabla}_{3D})\Psi = 0\,, \quad (x, y) \in \partial\mathcal{V}. \tag{51}$$

The problem that is encountered with this condition (and any ABC for that matter), is that it assumes the wave speed to be constant, but in the finite difference scheme the numerical wave speed is directionally and frequency-dependent [21]. Another problem with condition (51) is that it is less effective for incoming waves that are not normal to the boundary. Ultimately, these two effects combine such that the ABCs only partially absorb incident waves. This can be seen in the spectrogram displayed in Fig. 4a, which refers to the output of a simulation without viscosity in the air ($\sigma_a = 0$ m) and without the cavity or snares. The output was taken along a diagonal above the top membrane and the spectrogram uses a 512 sample Hann window with 75% overlap. It can be seen there is energy which is slow to decay at approximately 8643 Hz. This is in fact the temporal frequency ($0.196F_s$) that experiences the worst dispersion error (approx. 30% error) for axial-directed waves [21]. There is another peak at $0.304F_s$, which is the temporal frequency pertaining to the worst error for side-diagonal directions (approx. 25% error) [21].

When the cavity and snares are added to the simulation there is an increase in mid-frequency energy due to the modes of the cavity and due to the snares activity. A spectrogram for this case is shown in Fig. 4b. In this case, the energy that is slow to decay causes audible 'hiss' and 'ringing' artefacts. Although not presented here, higher-order ABCs (up to fourth order) were also not effective at reducing this effect. Fortunately, viscosity in air has a damping effect that targets high frequencies [14]. A spectrogram from the same listening position, now with $\sigma_a = 2 \times 10^{-6}$ m, is shown in Fig. 4c. It can be seen that the energy in this band of frequencies decays faster than in the lossless case. It was found that this added decay was sufficient to eliminate the audible artefacts.

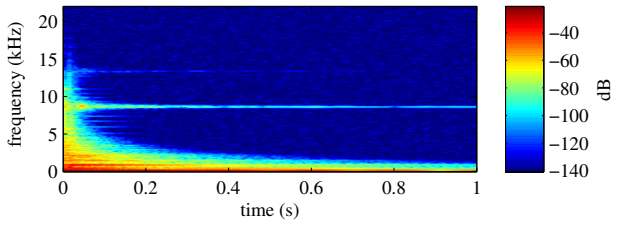
5.4. Sounds and Videos

Sound examples and videos can be found at the author's website:
`www2.ph.ed.ac.uk/~s1164558`

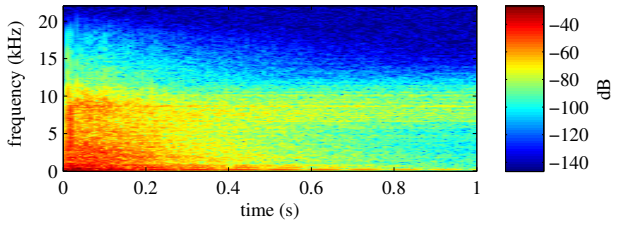
6. FINAL REMARKS

In this paper, a physics-based model of a snare drum has been presented. A novel, energy conserving numerical scheme for the simulation of collisions has been discussed, which can be applied both to the mallet-membrane and to the snare-membrane interactions. This constitutes a major improvement with respect to previous works, as in this case the stability of the numerical scheme can be guaranteed.

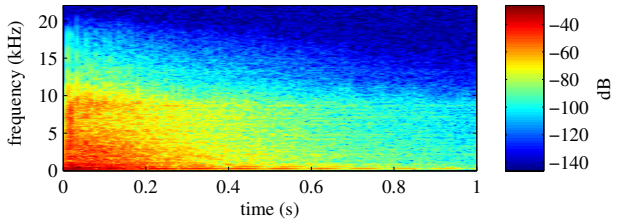
Another problem that has been discussed in this work is the effect of dispersion in the 3D Cartesian scheme in virtual embedding simulations such as this. It has been found that, when high frequencies are created in the model, either by the mallet or by the snares, a slowly attenuating "hiss" is produced, which dominates the spectrogram of the output sound and harms its quality. This problem has been interpreted as dispersion of the 3D scheme exacerbating the proper functioning of absorbing boundary conditions. However, when viscothermal effects are added to the 3D scheme these artefacts are rendered inaudible.



(a) *Without cavity, snares and viscosity*



(b) *With cavity and snares, without viscosity*



(c) *With cavity, snares and viscosity*

Figure 4: *Spectrograms for simulation output.*

A point which has not been mentioned in this work is the computation cost of this model. As discussed in Sec. 4, the collision model presented relies on the solution of a nonlinear equation with the Newton-Raphson method at every time step. If in the case of the mallet this is just a scalar equation, it becomes a challenging problem for the snares-membrane interaction, where a vectorial equation is involved. When a realistic number of snares is included in the numerical model, the dominant part of the code in terms of computation time is the solution of the nonlinear system (50), and not, as one would expect, the update of the 3D field. The former, in fact, requires the iterative solution of a linear system, which in this case is dense. It is, therefore, an intrinsic serial operation. As well known, parallel hardware like GPGPUs can be extremely useful in accelerating the computation of systems with a high degree of parallelisability, and this is becoming a mainstream approach to room acoustics simulation [22]. However, this hardware is not suited for cases like the present one, where operations must be performed in a sequential order. One of the major challenges at the moment is to find alternative methods that could tackle more effectively this problem.

7. REFERENCES

[1] J. A. Laird, *The physical modelling of drums using digital waveguides*, Ph.D. thesis, University of Bristol, 2001.

[2] F. Avanzini and R. Marogna, "A modular physically based approach to the sound synthesis of membrane percussion instruments," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 4, pp. 891–902, 2010.

[3] L. Rhaouti, A. Chaigne, and P. Joly, "Time-domain modeling and numerical simulation of a kettledrum," *The Journal of the Acoustical Society of America*, vol. 105, pp. 3545, 1999.

[4] S. Bilbao, "Time domain simulation and sound synthesis for the snare drum," *The Journal of the Acoustical Society of America*, vol. 131, pp. 914–925, 2012.

[5] A. Torin and S. Bilbao, "Numerical experiments with non-linear double membrane drums," in *Proc. 4th Stockholm Musical Acoustics Conference (SMAC 2013)*, Stockholm, Sweden, 2013.

[6] P. Wriggers, *Computational Contact Mechanics*, Springer-Verlag Berlin Heidelberg, second edition, 2006.

[7] F. Avanzini and D. Rocchesso, "Modeling collision sounds: Non-linear contact force," in *Proc. COST-G6 Conf. Digital Audio Effects (DAFx-01)*, Limerick, Ireland, 2001, pp. 61–66.

[8] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. I. A physical model for a struck string using finite difference methods," *The Journal of the Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

[9] J. Chabassier, *Modélisation et simulation numérique d'un piano par modèles physiques.*, Ph.D. thesis, Ecole Polytechnique X, 2012.

[10] D. Kartofelev, A. Stulov, H.-M. Lehtonen, and V. Välimäki, "Modeling a vibrating string terminated against a bridge with arbitrary geometry," in *Proc. 4th Stockholm Musical Acoustics Conference (SMAC13)*, Stockholm, Sweden, 2013.

[11] D. Greenspan, "Conservative numerical methods for $\ddot{x} = f(x)$," *Journal of Computational Physics*, vol. 56, no. 1, pp. 28–41, 1984.

[12] S. Li and L. Vu-Quoc, "Finite difference calculus invariant structure of a class of algorithms for the nonlinear klein-gordon equation," *SIAM Journal on Numerical Analysis*, vol. 32, no. 6, pp. 1839–1875, 1995.

[13] V. Chatziioannou and M. van Walstijn, "An energy conserving finite difference scheme for simulation of collisions," in *Proc. of the Sound and Music Computing Conference (SMC 2013)*, Stockholm, Sweden, 2013, pp. 584–591.

[14] P. M. Morse and K. U. Ingard, *Theoretical acoustics*, McGraw-Hill Inc., USA, 1986.

[15] B. Engquist and A. Majda, "Absorbing boundary conditions for numerical simulation of waves," *Proceedings of the National Academy of Sciences*, vol. 74, no. 5, pp. 1765–1766, 1977.

[16] J. G. Meloney and K. E. Cummings, "Adaptation of FDTD techniques to acoustic modeling," in *11th Annual Review of Progress in Applied Computational Electromagnetics*, 1995, vol. 2, pp. 724–731.

[17] K. Hunt and F. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *Journal of applied mechanics*, vol. 42, no. 2, pp. 440–445, 1975.

[18] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time dependent problems and difference methods*, Wiley New York, 1995.

[19] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley Publishing, Chichester, UK, 2009.

[20] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Under review*, vol. available online at http://arxiv.org/abs/1405.2589, 2014.

[21] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 1, pp. 34–46, 2011.

[22] C. J. Webb and S. Bilbao, "Computing room acoustics with CUDA-3D FDTD schemes with boundary losses and viscosity," in *Acoustics, Speech and Signal Processing, IEEE International Conference on*, Prague, Czech Republic, 2011, IEEE, pp. 317–320.

PHYSICAL MODELING OF THE MXR PHASE 90 GUITAR EFFECT PEDAL

Felix Eichas, Marco Fink, Martin Holters, Udo Zölzer

Department of Signal Processing and Communications,
Helmut-Schmidt-Universität
Hamburg, Germany
felix.eichas@hsu-hh.de

ABSTRACT

In this study, a famous boxed effect pedal, also called *stompbox*, for electrical guitars is analyzed and simulated. The nodal DK method is used to create a non-linear state-space system with `Matlab` as a physical model for the *MXR Phase 90* guitar effect pedal. A crucial component of the effect are Junction Field Effect Transistors (JFETs) which are used as variable resistors to dynamically vary the phase-shift characteristic of an allpass-filter cascade. So far, virtual analog modeling in the context of audio has mainly been applied to diode-clippers and vacuum tube circuits. This work shows an efficient way of describing the non-linear behavior of JFETs, which are wide-spread in audio devices. To demonstrate the applicability of the proposed physical model, a real-time VST audio plug-in was implemented.

1. INTRODUCTION

Nodal analysis has been widely used to derive non-linear state-space systems from electrical circuits [1–6]. In this work the type of the nodal DK method described in [1] is applied to the circuit of the *MXR Phase 90* phaser effect. The used modeling technique has the advantage that the state-space matrices can directly be calculated from so called *incidence matrices*, which describe the position of each circuit element in relation to the nodes of the circuit, and *diagonal matrices* which contain the values of the corresponding circuit elements.

The research field of virtual analog modeling in the audio context was so far dominated by studies investigating distortion and overdrive effects. Musical distortion circuits, like diode clippers, and various guitar amplifier and effect circuits were analyzed in [3, 6]. In [4] the main focus is on the modeling of vacuum tubes, which can mainly be found in guitar amplifiers. A dynamic filter broadly used in musical environments, known as Dunlops Crybaby Wah-wah effect pedal, is based on a circuit including two bipolar junction transistors (BJTs). The modeling of the BJTs and the simulation of the effect device was reported in [1]. However, the class of modulation effects was not subject of detailed research so far. Therefore, this paper focuses on modeling a time-variant phasing effect and in particular, the usage of JFETs as non-linear variable resistors in audio circuits.

A slightly modified version of the original circuit by *MXR*, which was available as a D.I.Y. effect pedal kit from [7], is used as the reference device. The circuit of the kit was thoroughly analyzed and every circuit element was measured prior to assembling the pedal. The characteristics of every JFET used in the reference circuit have been measured and are used in the implementation, which is of relevance since they strongly influence the tonal behavior of the effect pedal.

In section 2 a brief review on the nodal DK method, the used discretization method, the handling of operational amplifiers and non-linear elements as well as the used non-linear solver is given. Section 3 describes how the phaser operates in general and discusses the circuit of the analog reference device. In section 4 the results of the measurements are evaluated. Section 5 specifies the real-time VST plug-in implementation, whereas section 6 concludes this paper.

2. NODAL DK METHOD

The nodal DK version of [1] has been used in this work to transfer the effect's schematic to its digital emulation. The laws of Kirchoff, namely Kirchoff's current law (KCL) and Kirchoff's voltage law (KVL), are used to design a state-space system of the circuit under test. The state-space matrices are constructed using *incidence matrices* $\mathbf{N_i}$, which describe the connections of the $\mathbf{i}_{\text{th}}$ group of circuit elements, like resistive, capacitive, or inductive elements, to the nodes of the circuit. A further requirement for constructing the state-space system are *diagonal matrices* $\mathbf{G_i}$, containing the values of the circuit elements and the so called *system matrix*

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{N}_R^T \boldsymbol{G}_R \boldsymbol{N}_R + \boldsymbol{N}_x^T \boldsymbol{G}_x \boldsymbol{N}_x & \boldsymbol{N}_{vo} \\ \boldsymbol{N}_{vi} & \boldsymbol{0} \end{pmatrix} \tag{1}$$

resulting from the aforementioned incidence and diagonal matrices. $\mathbf{N_i}$ are $m \times n$ matrices with m as the total number of a certain circuit element (e.g. resistors or capacitors) and n as the total number of nodes in the circuit. As it is common for nodal analysis, the reference node (commonly the ground node) is not numbered. The positive and negative poles of every element are marked by a $(+1)$ and a (-1) in each row. If an element is connected to the reference node, only the pole of the element which is not connected to the reference node is marked by a (± 1) in the incidence matrix.

The $\mathbf{G_i}$ matrices hold the information about the values of each circuit element and the $\mathbf{N_v}$ matrices give the position of voltage sources in the circuit. Note, that the subindices $\mathbf{R}$ and $\mathbf{X}$ refer to resistive and capacitive elements, respectively.

The procedure uses the trapezoidal discretization rule to get the discrete-time approximations for the energy storing elements of the circuit. In the circuit of the phaser there are only capacitors as energy storing elements. The resulting state-space system has the form

$$\mathbf{x}(n) = \mathbf{A}\mathbf{x}(n-1) + \mathbf{B}\mathbf{v}(n) + \mathbf{C}\mathbf{i}_\mathbf{n}(n) \tag{2}$$

$$\mathbf{y}(n) = \mathbf{D}\mathbf{x}(n-1) + \mathbf{E}\mathbf{v}(n) + \mathbf{F}\mathbf{i}_\mathbf{n}(n) \tag{3}$$

$$\mathbf{v}_\mathbf{n}(n) = \mathbf{G}\mathbf{x}(n-1) + \mathbf{H}\mathbf{v}(n) + \mathbf{K}\mathbf{i}_\mathbf{n}(n). \tag{4}$$

The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{H}$, and $\mathbf{K}$ can be computed from $\mathbf{S}, \mathbf{N_i}$, and $\mathbf{G_i}$ (more detailed information on the state-space system is provided in [1]). The actual computation of the state-space system consist of three steps. At first the non-linear relations between the present voltages at the non-linear element $\mathbf{v_n}(n)$ and it's non-linear currents $\mathbf{i_n}(n)$ are obtained through Eq. (4). This is done by an iterative non-linear solver (see section 2.3). Then, the output of the system $\mathbf{y}(n)$ can be computed using Eq. (3). At last the internal states $\mathbf{x}(n)$ of the system, representing the amount of charge in the capacitors, have to be updated using Eq. (2).

2.1. Capacitors

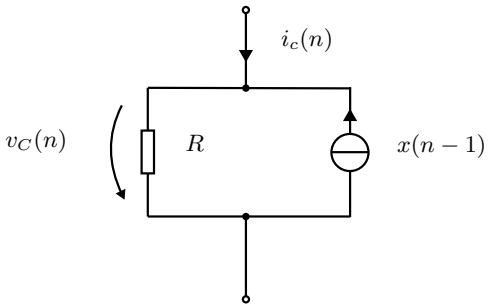


Figure 1: Companion circuit for energy-storing elements of the circuit

From the differential equation, describing the current through a capacitor

$$i_C = C\frac{d\,v_C}{d\,t}, \tag{5}$$

it's discrete-time approximation (using the trapezoidal discretization rule)

$$\frac{1}{2}\left(i_C(n) + i_C(n-1)\right) = \frac{C}{T}\left(v_C(n) - v_C(n-1)\right), \tag{6}$$

and the canonical state

$$x_C(n) = -\frac{2C}{T}v_C(n) - i_C(n) \tag{7}$$

the state-update equation can be formulated

$$x_C(n) = 2\frac{2C}{T}v_C(n) - x_C(n-1). \tag{8}$$

Combining Eq. (7) and Eq. (8) yields

$$i_C(n) = \frac{2C}{T}v_C(n) - x_C(n-1) \tag{9}$$

which describes the companion circuit shown in Fig. 1.
Due to this procedure every capacitor is replaced by a parallel circuit consisting of a resistor $R = \frac{T}{2C}$ and a current source which holds the state information.

2.2. Operational Amplifiers

In this work operational amplifiers are considered to be ideal. They consist of three nodes, describing the inverting input $(-)$, the non-inverting input $(+)$ and the output (out). The ideal operational amplifier (op-amp) is considered as a voltage controlled voltage source, where the voltage difference at the input defines the output voltage $v_{out} = A(v_+ - v_-)$. The input resistance $R_{in} = \infty\,\Omega$ is infinitely large and the output resistance $R_{out} = 0\,\Omega$ infinitely small as illustrated in Fig. 2. A is the open-load amplification factor [5].
Without any op-amp in the circuit the relation between the voltage

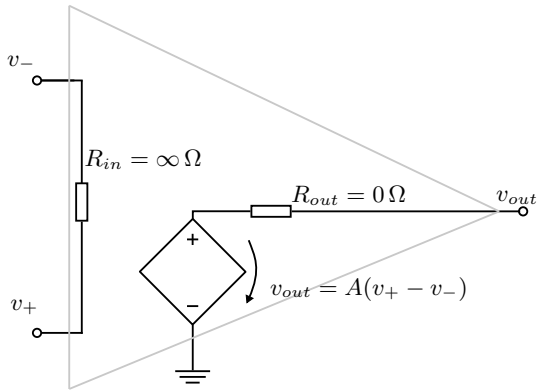


Figure 2: Equivalent circuit diagram for an operational amplifier.

incidence matrices (see section 2) would be

$$\mathbf{N_{vi}} = \mathbf{N_v^T} \tag{10}$$

and

$$\mathbf{N_{vo}} = \mathbf{N_v} \tag{11}$$

with $\mathbf{N_v}$ being of $n \times k$ size. n corresponds to the number of nodes in the circuit while k is the number of voltage sources.
To add an ideal op-amp to the state-space system described in section 2 the only modifications that have to be made are changes in the incidence matrices for the voltage sources $\mathbf{N_v^T}$ and $\mathbf{N_v}$. Therefore, one row is added to the $\mathbf{N_v^T}$ matrix and one column to the $\mathbf{N_v}$ matrix. The additional row in the $\mathbf{N_v^T}$ matrix contains a $(+1)$ at the corresponding node representing the inverting and a (-1) for the non-inverting input of the op-amp respectively

$$\mathbf{r_i} = (\ldots\ 0\ 1\ -1\ 0\ \ldots), \tag{12}$$

which yields

$$\mathbf{N_{vi}} = \begin{pmatrix} N_v^T \\ r_i \end{pmatrix}. \tag{13}$$

These modifications indicate that the inputs of an ideal op-amp have the same potential though they draw no currents.
Since the controlled voltage source describing the op-amps output is connected to the ground node the additional column in the $\mathbf{N_{vo}}$ matrix contains only a $(+1)$ at the corresponding node

$$\mathbf{c_o} = (\ldots\ 0\ 1\ 0\ \ldots)^T \tag{14}$$

leading to

$$\mathbf{N_{vo}} = \begin{pmatrix} \boldsymbol{N_v} & \boldsymbol{c_o} \end{pmatrix}. \tag{15}$$

This describes the voltage source from Fig. 2. By using this technique, an ideal op-amp can be easily integrated in the state-space representation and its behavior is described by the peripheral wiring around the op-amp.

2.3. Non-linear Circuit Elements

A major challenge in virtual analog modeling are non-linear elements. Typically, the non-linear relation between voltages and currents are iteratively solved by minimizing an error function $f(x)$ producing the current iteration's residual e. This costly computation has to be performed for every sample and every non-linear component of the desired system. One way to avoid these complex computations, potentially inhibiting real-time functionality, is to pre-compute the non-linear function for a certain range of input values and store the corresponding results in a lookup table. Certainly, the storing of lookup tables requires a decent amount of memory. Hence, an actual implementation is always subject to a compromise between computational complexity and memory requirements.

The non-linear elements in the *Phase 90* circuit are 4 JFETs

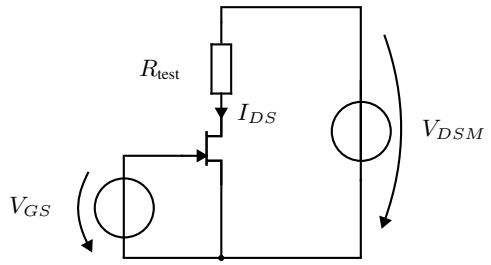


Figure 3: Measurement setup for the JFETs.

which modify the center frequency of the allpass filter cascade and a PNP BJT as a summing amplifier. The input-output characteristic of each JFET was determined by varying the gate-source voltage V_{GS} and drain-source voltage V_{DS} and measuring the drain current I_{DS} of the device. A low-impedance resistor R_{test} was connected in series to the drain-source channel of the JFET for ultra high frequency (UHF) stability as can be seen in Fig. 3. The relevant voltage ranges for $V_{DS} \in [-2\,\text{V}, \ldots, 2\,\text{V}]$ and $V_{GS} \in [-2.8\,\text{V}, \ldots, -1.5\,\text{V}]$ were determined using a *SPICE* simulation of the phaser circuit and from measurements on the reference device.

The results of these measurements were saved in lookup tables which are used in the model to calculate the current-voltage relations for every JFET in the circuit.

2.4. Non-linear Solver

As previously mentioned, the non-linear voltage-current relations are solved by iteratively minimizing an error function. The well known Newton-Raphson method has been used to solve the error function

$$f(\mathbf{v_n}(n)) = \mathbf{G}\mathbf{x}(n-1) + \mathbf{H}\mathbf{v}(n) + \mathbf{K}\mathbf{i}(n) - \mathbf{v_n}(n) = e, \quad (16)$$

which is Eq. (4) in a slightly modified form.

The function is linearized at a certain starting value and the root of the linearization is used as the next starting point until the residual of the error function is smaller than a predefined tolerance

while $|f(\mathbf{v_m})| > \epsilon$ **do**
  $\mathbf{v_{m+1}} = \mathbf{v_m} - b\frac{f(\mathbf{v_m})}{f'(\mathbf{v_m})}$;
end

Algorithm 1: Pseudo code for the Newton-Raphson method.

$f(\mathbf{v_m}) < \epsilon$, as illustrated with algorithm 1, where m denotes the iteration index and the step size b is 1.

For the PNP-BJT a damped approach of this method has been implemented. Due to the exponential functions in the Ebers-Moll-Model, describing the BJT, a step with a size that is too big may lead to a function value which can not be represented by normal floating point arithmetic. Thus the non-linear solver does not converge and this again leads to invalid numbers which can lead to an unstable state-space system. Therefore the step size of the damped method is halved as long as the resulting residual is bigger than the original residual being computed with the previous step size. Listing 2 shows the corresponding pseudo-code for the implemented method

while $|f(\mathbf{v_m})| > \epsilon$ **do**
  $b = 1$;
  $\mathbf{v_{m+1}} = \mathbf{v_m} - b\frac{f(\mathbf{v_m})}{f'(\mathbf{v_m})}$;
  $\mathbf{v_{\tilde{m}}} = \mathbf{v_{m+1}}$;
  $b = \frac{b}{2}$;
  $\mathbf{v_{\tilde{m}+1}} = \mathbf{v_m} - b\frac{f(\mathbf{v_{\tilde{m}}})}{f'(\mathbf{v_{\tilde{m}}})}$;
  while $|f(\mathbf{v_{\tilde{m}+1}})| > |f(\mathbf{v_{m+1}})|$ **do**
    $b = \frac{b}{2}$;
    $\mathbf{v_{\tilde{m}+1}} = \mathbf{v_m} - b\frac{f(\mathbf{v_{\tilde{m}}})}{f'(\mathbf{v_{\tilde{m}}})}$;
  end
  $\mathbf{v_{m+1}} = \mathbf{v_{\tilde{m}+1}}$;
end

Algorithm 2: Pseudo code for the damped Newton-Raphson method.

3. PHASER EFFECT

Phasing is a common modulation effect in audio applications. Many hardware realizations are based on the following idea. The input signal is sent through an allpass filter cascade which leads to a frequency dependent phase shift. The phase-shifted signal is then added to the original signal which introduces phase cancellation and elevation for certain frequencies. Since each first order allpass filter introduces a maximum phase shift of $180\,^\circ$ the amount of resulting spectral notches is half the number of used allpass stages. The center frequency of each allpass filter is varied by a *low frequency oscillator* (LFO) in the same way. This leads to a time variant phase shift in the output signal of the allpass filter cascade causing the phase cancellation to sway back and forth the frequency axis and thus creating the characteristic effect of the phaser.

3.1. Phase90 Circuit

The circuit of the *Phase90* can be divided into functional blocks, like the power supply block, the low frequency oscillator, and the

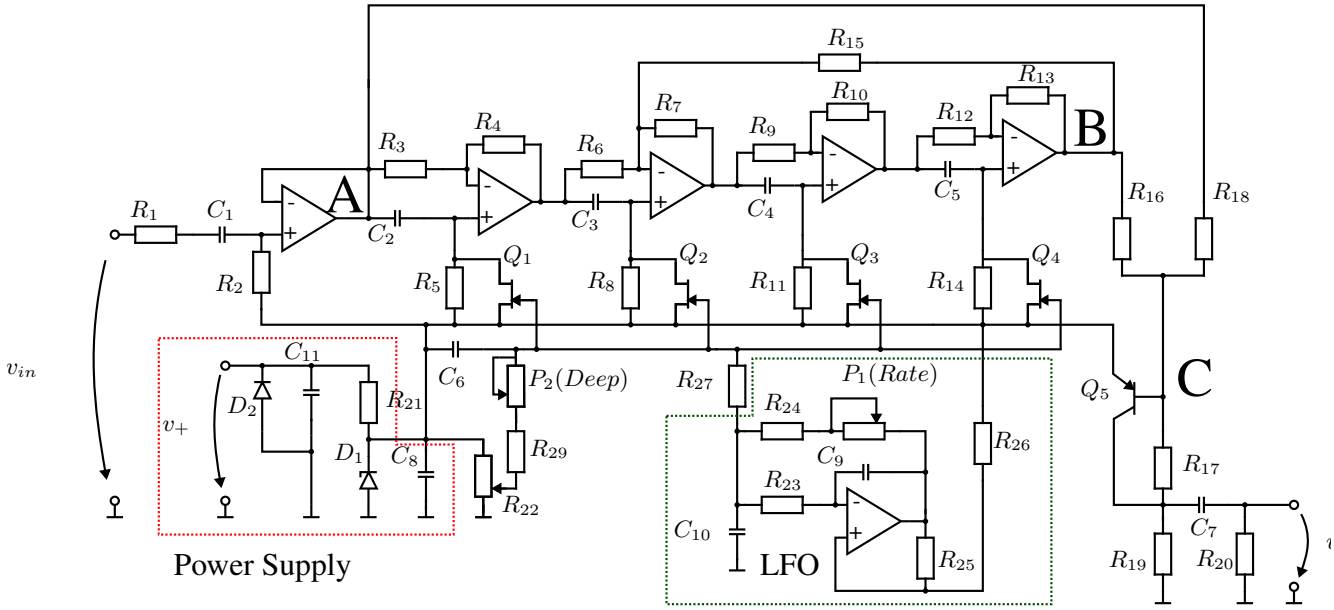


Figure 4: Circuit of the Phase90.

signal processing blocks which actually process the input signal. For physical modeling of audio circuits only the signal processing blocks of the device are of interest. Other blocks can be neglected as their contribution would not affect the tonal qualities of the model but only increase the computational effort.

In Fig. 4 the negligible blocks of the circuit have been marked as *power supply* and *LFO*. The complete power supply block can be replaced by a voltage source with the value of the voltage defined by the zener diode D_1. This voltage will be called V_{lift} since it adds a constant DC offset of about $V_{\text{lift}} \approx 5\,\text{V}$ to the input signal.

Modeling of the *low frequency oscillator* (LFO) is not necessary since the output of the LFO can be described analytically. The LFO can thus be replaced by another voltage source which produces an oscillating voltage V_{OSC} of the same characteristics as the LFO would create.

The signal path of the circuit contains three main blocks. The first one is the *input stage*, which is just a simple voltage follower, lifting the input signal up to V_{lift}. It consists of an op-amp, R_1, R_2 and C_1. The (A) in Fig. 4 marks the output of the input stage.

It is possible to divide the circuit into stages, because the op-amps have a low output resistance. The corresponding nodes are suitable for dividing the signal path of the circuit into blocks. After the *input stage* the signal passes a cascade of four allpass filters, using JFETs $Q_1 - Q_4$ as variable resistors. The continuous change of the resistance R_{JFET} leads to alternating center frequencies

$$f_c = \frac{1}{2\pi(R_{\text{fix}}||R_{\text{JFET}})C_{\text{fix}}} \tag{17}$$

of the allpass filters. R_{fix} are the resistors $[R_5, R_8, R_{11}, R_{14}]$, applied in parallel to the JFETs, and C_{fix} denotes the capacitors $C_{2,\ldots,5}$, that are placed at the positive op-amp input. The output of the allpass-stage is marked in Fig. 4 as (B).

The resistance R_{15} introduces a feedback in the allpass cascade which leads to an amplification of certain frequencies. This resonance introduces a harmonic distortion of the amplified frequencies in the output signal of the *Phase 90*. By including a switch or a potentiometer in this feedback path the resonance can be completely switched off or adjusted by the potentiometer allowing a kind of *tone control*. In 1974 the first version of the *Phase 90* was released. The circuit of this early version did not include the feedback path with resistor R_{15}. The so called *script* version (due to it's script font *MXR* logo) is still popular amongst guitar players and reaches horrendous prices in trade. In the state-space model this change of circuitry can be easily adapted leading to a usage of the *Phase 90* model beyond the limitations of the circuit.

The last stage of the signal path is the output stage. The phase-shifted allpass signal is added to the direct signal at the base of the PNP bipolar junction transistor Q_5 driving the output voltage v_{out} of the phaser. Figure 4 illustrates this process: signal (A) is added to signal (B) at the base of the PNP-BJT Q_5 (C).

4. EVALUATION

This section shall present the undertaken experiments, assessing the model's quality. Besides the illustration of the JFET measurements, the static and dynamic behavior of the *Phase 90* emulation is shown in detail. Additionally, the auditory impression is graded.

4.1. JFET Characteristics

The JFETs were measured as described in section 2.3 and the results of these measurements can be seen in Fig. 5. The absolute value of the drain-current decreases when the gate-source voltage V_{GS} declines thus increasing the channel resistance of the JFET. In [8] or similar literature the functionality of the JFET is described in detail. To reduce computational complexity the relations between voltages and currents are stored as a lookup table for the efficient calculation of the non-linear element in the circuit.

In this circuit and for the given operating point of the JFETs the

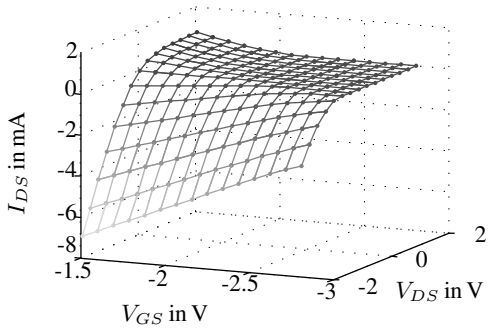


Figure 5: Output characteristic of a single JFET.

gate-source voltage V_{GS} is independent of the drain-source voltage V_{DS} or the drain-current I_{DS}. Since the gate-source voltage is controlled by the LFO it is sufficient to calculate the drain-current drain-source voltage relations subject to the current value of the gate-source voltage which can be computed linearly without the need of a non-linear solver.
Furthermore, it was important to use a matched set of four JFETs with similar characteristics in the reference device. Due to production uncertainties the characteristics of JFETs vary from transistor to transistor. Since all JFETs are driven by the same control voltage an unmatched set of transistors would not have the same operating point and thus the allpass-filters would not be tuned correctly.

4.2. Static Behavior

The connection between resistor R_{27} and R_{24} or capacitor C_{10} respectively was unraveled to disconnect the LFO from the circuit (see Fig. 4) and a voltage source with a constant DC value was applied to this node. A sine wave with the frequency $f_{sin} = 1\,\text{kHz}$ and amplitude of 1 V was fed into the input of the reference device and into the input of the state-space model.
Figure 6 shows the spectra of the output signals for the analog

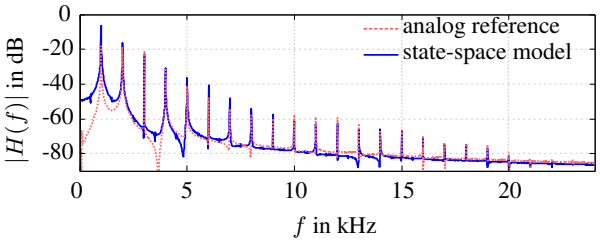
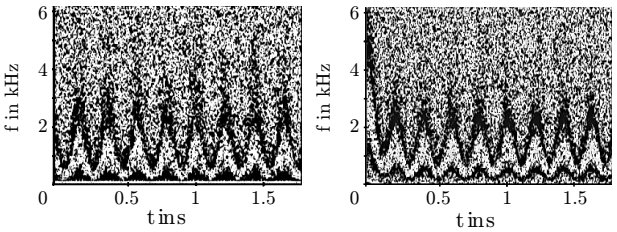


Figure 6: Output spectrum of analog reference device (dashed red line) and state-space model (solid blue line) for a mono-frequent excitation signal with $f_{sin} = 1\,\text{kHz}$ and 1 V amplitude.

reference and the digital model. The model exhibits more signal energy for lower frequencies than the reference device but apart from that the similarity of the distortion behavior of model and system are satisfactory.

4.3. Dynamic Behavior

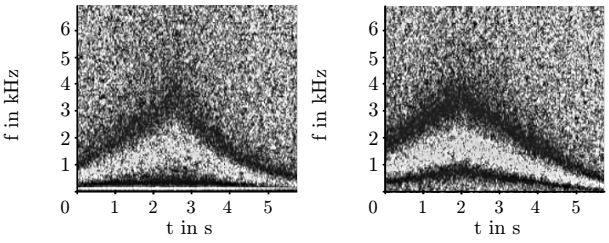
Since the phaser is a time-variant modulation effect the dynamic behavior is of major importance since it strongly influences the auditory impression. To visualize the varying spectral notches, both



(a) Analog Device (b) State-space model

Figure 7: Output of (a) the analog reference device and (b) the state-space model of the phaser for white noise as the input signal and maximum LFO frequency.

systems were fed with white noise. The output was measured using the maximum depth setting and two settings for the speed: (1) the maximum speed and (2) the minimum speed.
As previously mentioned, the amount of spectral notches is half the number of allpass stages. The Phase90 circuit contains 4 stages and hence, two notches can be seen in Fig. 7 and 8 in form of oscillating dark lines, indicating the frequencies that are canceled.
It is noticeable that the analog signal contains AC hum at



(a) Analog Device (b) State-space model

Figure 8: Output of (a) the analog reference device and (b) the state-space model of the phaser for white noise as the input signal and minimum LFO frequency.

$f = 50\,\text{Hz}$. In Fig. 7 (b) the charging process of the capacitor models can be seen for $t < 0.1\,\text{s}$. When the capacitors are fully charged the frequency cancellation occurs at the same frequencies as in the reference device.
Figure 8 indicates that the approximated output voltage of the state-space LFO does not behave exactly like its analog counterpart. The output voltage of the LFO was approximated by a triangular function while the actual LFO output voltage has slightly curved edges and the falling edge is a little longer than the rising edge.
The notch width of the analog reference device seems larger than for the state-space model while the edges of the notches seem to be sharper in the state-space representation. This is probably caused by noisy side-effects occurring in the JFETs but not in their simplified model.

4.4. Auditory Impression

Despite the afore mentioned differences of state-space model and analog reference device the auditory impression of the model is satisfactory. The difference of the LFO signals is barely noticeable for slower oscillation speeds and perceptually diminishes for faster LFO speeds. Listening examples for the reference device, the emulated system, and the emulated *script* version (without resonant feedback) can be found online. See [9] for listening examples.

5. REAL-TIME IMPLEMENTATION

State-space implementations are known to be complex due to the necessity to perform many matrix operations. Hence, a real-time implementation was done to test the real-time capability of the phaser emulation. The authors decided to implement a VST plug-in in C/C++ using the well-known JUCE framework [10] that allows straight-forward implementations by simply extending an automatically generated plug-in template with the actual signal processing code.
The circuit was divided in several processing-blocks and the same was done in the real-time implementation. Every module was implemented in a separate class and each object of a module class is a member of the phaser class. There are 6 modules needed to calculate a sample of the output signal. At first the *input* module, then 4 *allpass* modules and at the end of this chain the *output* module. The second *allpass* module is expanded to contain resistance R_{15} and another input to integrate the feedback path mentioned in section 3.1. Alternatively this stage can also be replaced by an *allpass* module without feedback via the graphical user interface of the plug-in to include the afore mentioned 1974 version of the *Phase 90*. The phaser class offers a public processing function, called from the VST host through the JUCE interface, that processes a block of audio data.
Initially the authors planned to use a wide-spread C++ library for linear algebra [11]. It allows a similarly compact representation of matrix operations as `Matlab`. The compact representation, leading to nice readable and maintainable code, is achieved with the help of massive templating. Unfortunately, the implementation based on the templated code was intolerably slow. Hence, a new class providing functionality for memory allocation, addition, multiplication, and copying of matrices was implemented and used from thereon. The computation of the matrices $\mathbf{A}$ to $\mathbf{K}$ of Eq. (2-4) requires the inversion of the potentially large system matrix $\mathbf{S}$. This inversion was performed using the well-known Linear Algebra Package (LAPACK) [12], implemented in Fortran. Using these tools, allows to realize the plug-in, that is now applicable in manifold applications and running real-time. Nevertheless, this first unoptimized, straight-forward implementation utilizes about 70 % of a single Intel i5 processing core.

6. CONCLUSIONS

In this work the circuit of the *MXR Phase 90* was analyzed and emulated using a state-space implementation. A D.I.Y. clone of the effect was assembled and measured to have a reference device. The measurements focused on measuring the JFETs characteristics, which were saved in a lookup-table to efficiently describe the behavior of the transistors. To realize the state-space representation of the effect, the nodal DK method was used, which is a lucid method to transform an analog circuit into a mathematical model.
A `Matlab` implementation of the state-space model was developed allowing to imprint the tonal characteristics of the effect on pre-recorded `.wav` files. Additionally a VST plug-in of the effect was implemented in C/C++ to allow real-time application of the *Phase 90*.
The model was analyzed and compared to the reference device in the static (non-oscillating) and the dynamic case. The comparison showed that model and reference are not exactly the same but particularly in the dynamic case the results are very satisfactory. The proposed model also has the possibility to expand the effect beyond the limitations of the circuit. The very first 1974 *script* version of the *Phase 90*, without the resonant feedback, can be used as well as todays version.

7. REFERENCES

[1] M. Holters and U. Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal dk method to circuits with variable parts," in *Proc. Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 19-23, 2011, pp. 31–35.

[2] D. Yeh, J.S. Abel, and J.O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects. part i: Theoretical development," in *IEEE Trans. Audio, Speech, and Language Process.*, May 2010, vol. 18, pp. 203–206.

[3] D. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*, Ph.D. thesis, Stanford University, 2009.

[4] K. Dempwolf, *Modellierung analoger Gitarrenverstärker mit digitaler Signalverarbeitung*, Ph.D. thesis, Helmut-Schmidt-Universität, 2012.

[5] J. Macak, *Real-time Digital Simulation of Guitar Amplifiers as Audio Effects*, Ph.D. thesis, Brno University of Technology, 2011.

[6] D. Yeh and J.O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *Proc. Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 1-4, 2008, pp. 19–26.

[7] Musikding, "Online Material to the Phaser Kit," Website, Available online at `http://diy.musikding.de/?p=384` – accessed February 27th 2014.

[8] U. Tietze and C. Schenk, *Halbleiter Schaltungstechnik*, Springer, Heidelberg, Germany, 1986.

[9] "Listening Examples for the Phase 90," Website, Available online at `http://www2.hsu-hh.de/ant/webbox/audio/eichas/dafx14/audio_examples_local_version.html` – accessed March 17th 2014.

[10] "JUCE Cross-Platform C++," Website, Available online at `http://www.juce.com/` – accessed March 14th 2014.

[11] "C++ linear algebra library," Website, Available online at `http://arma.sourceforge.net/` – accessed March 14th 2014.

[12] "Lapack - linear algebra package," Website, Available online at `http://www.netlib.org/lapack/` – accessed March 14th 2014.

A PHYSICALLY-INFORMED, CIRCUIT-BENDABLE, DIGITAL MODEL OF THE ROLAND TR-808 BASS DRUM CIRCUIT

Kurt James Werner, Jonathan S. Abel, Julius O. Smith III,

Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, Stanford, California
[kwerner|abel|jos]@ccrma.stanford.edu

ABSTRACT

We present an analysis of the bass drum circuit from the classic Roland TR-808 Rhythm Composer, based on physical models of the device's many sub-circuits. A digital model based on this analysis (implemented in Cycling 74's `Gen~`) retains the salient features of the original and allows accurate emulation of circuit-bent modifications—complicated behavior that is impossible to capture through black-box modeling or structured sampling. Additionally, this analysis will clear up common misconceptions about the circuit, support the design of further drum machine modifications, and form a foundation for circuit-based musicological inquiry into the history of analog drum machines.

1. INTRODUCTION

When Roland discontinued the TR-808 Rhythm Composer in 1984, it was considered somewhat of a flop—despite significant voice design innovations, disappointing sales and a lukewarm critical reception seemed clear indicators that digitally-sampled drum machines were the future. Ironically, this lack of interest drove second-hand prices down and made it an attractive source of beats for techno and hip-hop producers. It soon became ubiquitous, playing a central role in the development of acid house. More than a decade later, when the Beastie Boys rapped "nothing sounds quite like an 808" [2], no one disagreed.

To this day, the 808 remains a benchmark against which all other analog drum machines are measured. Among all of its voices, perhaps the most influential has been the bass drum, the thumping foundation of countless four-on-the-floor dance beats. It could be tweaked via stock user controls to sound like a fairly realistic kick, or extended beyond recognition to a multi-second-long decaying pseudo-sinusoid with a characteristic sighing pitch. Its clicky attack could cut through a mix, but could be dialed back with a passive tone control.

Despite the significant work that has been done on cloning[1] and emulating[2] the 808 bass drum, there is an almost complete lack of published analyses on the circuit.[3] The history of the 808 is steeped in anecdote, and misinformation about its voice design still abounds. Although the fabric of lore surrounding the design, inception, and use of the 808 lends a richness to its overall mythology, they also give credence to a blithe sort of analog fetishism. The device's ingenious and satisfying properties are often attributed to mere circuit element nonlinearities. In addition to being inaccurate, this mindset directs attention away from a more interesting story. The designers of the 808's voice circuits[4] masterfully blended ingenuity and efficiency, creating circuits with great detail and complexity, but a part count low enough to be amenable to mass manufacture.[5]

The 808 was released just before the development of the MIDI standard (it used Roland's DIN sync protocol). As MIDI gained traction, users and technicians became accustomed to retrofitting the 808 with MIDI capabilities, also making extensive modifications to its voice circuitry.[6] This tradition parallels the development of circuit-bending and other music hardware hacking, and could unfortunately be lost in the process of digitally emulating an 808.[7]

The goals of this research are to partition the 808's bass drum circuit into functional blocks, create a physically-informed analysis of each block, model each block in software, and evaluate the results, paying special attention throughout to analysis of the circuit's behavior in terms of the electrical values of circuit elements (resistors and capacitors). These methods are well-represented in virtual analog literature,[8] but have not previously been used in the analysis of analog drum machine circuits.[9]

[1] Full analog clones such as the AcidLab Miami, clones of individual voices in a modular synth like the Analogue Solutions line of Concussor modules, and new drum machines using simplified 808 circuitry [3] are common (references are representative but far from comprehensive).

[2] The first 808 emulation, Propellerhead's sample-based ReBirth RB-338 [4], was introduced in 1997. Since then, there have been many commercial emulations based on structured sampling and black-box models. Marketing materials for the D16 Group's Nepheton mention circuit modeling. Roland's TR-8 Rhythm Performer, from their upcoming AIRA line, will employ their proprietary Analog Circuit Behavior (ACB) technique, presumably a form of physical modeling.

[3] [5] discusses [1] in the context of designing and building a hardware clone of the bass drum. [6] offers a qualitative description in the context of imitating classic bass drum sounds with other synthesizers. [7], which takes a control systems approach to designing an 808-inspired bass drum synthesizer, is a rare academic treatment.

[4] Roland president Ikutaro Kakehashi names Mr. Nakamura, though also indicates that it was a team effort [8].

[5] Robert Henke writes [9]: "The TR-808 is a piece of art. It's engineering art, it's so beautifully made. If you have an idea of what is going on in the inside, if you look at the circuit diagram, and you see how the unknown Roland engineer was making the best out of super limited technology, it's unbelievable. You look at the circuit diagram like you look at an orchestral score, you think, how on earth did they come up with this idea. It's brilliant, it's a masterpiece."

[6] for instance, Robin Whittle's professional modification work [10]

[7] [11] presents one approach to simulating circuit-bent instruments based solely on digital circuitry.

[8] For instance, [12] collects a representative set of references on modeling classic analog filters, and [13] is a comprehensive treatment of musical distortion circuits.

[9] However, [14] presents a physical and behavioral circuit model of the *digital* E-mu SP-12 sampling drum computer.

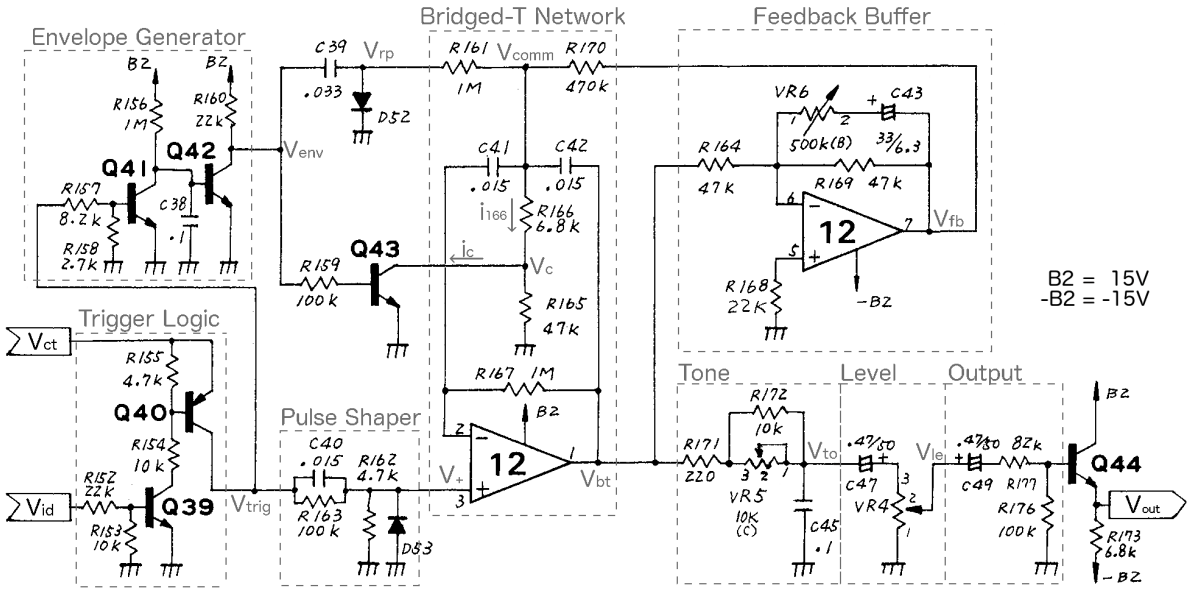


Figure 1: *TR-808 bass drum schematic, blocks marked (adapted from [1]).*

Our analysis of the circuit will clear up a host of common misconceptions about the 808 bass drum, including the role of device nonlinearities, the difference between the bass drum's characteristic pitch sigh and the frequency jump during the attack, and why unmodded bass drums can sound very different from one another.

Drawing on this analysis, we'll propose methods for software modeling of a modified/circuit-bent 808 bass drum circuit. By adopting a physically-informed approach, this work will avoid common pitfalls of software-based analog drum machine emulations, including the "machine gun effect," inaccurate behavior when a new note is triggered before the previous one has died out, and inaccurate behavior under various accent voltages.

In addition to the analysis itself, the primary result of this will be software that implements a circuit-bendable 808 bass drum.

An overview of the circuit is discussed in §2 and an analysis of each part of the the circuit and their interconnections is given in §§3–9. This is followed by a discussion of digital modeling techniques in §10 and results in §11.

2. OVERVIEW

Fig. 1 shows a schematic diagram of the TR-808 bass drum circuit. This schematic labels important nodes and currents, and emphasizes how the circuit can be broken down into blocks: trigger logic (see §3), a pulse shaper (see §4), a bridged-T network (see §5 and 7), a feedback buffer (see §6), an output tone and volume stage (see §9), and an envelope generator with complex behavior (see §8). Fig. 2 shows a block diagram of the digital model of the bass drum circuit. Both figures should be consulted alongside the analysis of each block in the following sections.

A bass drum note is produced when the μPD650C-085 CPU applies a common trigger and (logic high) instrument data to the trigger logic. The resulting 1-ms long pulse is delivered via the pulse shaper to the bridged-T network (a band pass filter), whose ringing produces the core of the bass drum sound. The 1-ms long pulse also activates an envelope generator, which alters the bridged-T's center frequency for the first few milliseconds and supplies a retriggering pulse to the center of the bridged-T network after a few milliseconds. Leakage through the retriggering pulse circuit accounts for "sighing" of the bass drum's pitch.

Certain features of the bass drum sound are user-controllable. The output level is set by variable resistor (potentiometer) VR_4, the tone is set by VR_5, and the length of a bass drum note is controlled via VR_6.

Partitioning the circuit into blocks will serve the triple purpose of greatly simplifying the mathematics of the system, elucidating the design intent of each sub-circuit, and allowing for the design and simulation of "mods" and "bends" that affect the architecture of the bass drum (for instance: disconnecting the pitch sigh or bypassing the tone stage). These partitions are chosen so that they occur where high-input-impedance stages are driven by low-output-impedance stages, where loading effects between blocks are negligible [15].[10]

By favoring a clear analysis that elucidated the design intent, this work supports informed modifications/hacks/bends of the circuit. A more complicated analysis could obscure the logic of the device's construction, with minimal gains in accuracy. Framing the analysis in terms of component values simplifies the simulation of "mods" and "bends" based on component substitution (for instance: making the pitch tunable, extending the decay time, or changing the pitch envelope's timing).

Certain parts of the model assume small-signal conditions and linearity where they may not be strictly present. Op-amps (the bass drum uses a dual μPC4558 [16]) are assumed to be linear, feature zero output impedance, and have an infinite ability to source current. In reality, op-amps that are not designed for rail-to-rail performance (including the μPC4558) experience saturation as their output voltage approaches the power supply rails. Op-amps have a small (but non-zero) output impedance and feature internal protection circuitry to limit the amount of current they can source.

[10] `https://ccrma.stanford.edu/~jos/pasp04/Equivalent_Circuits.html`

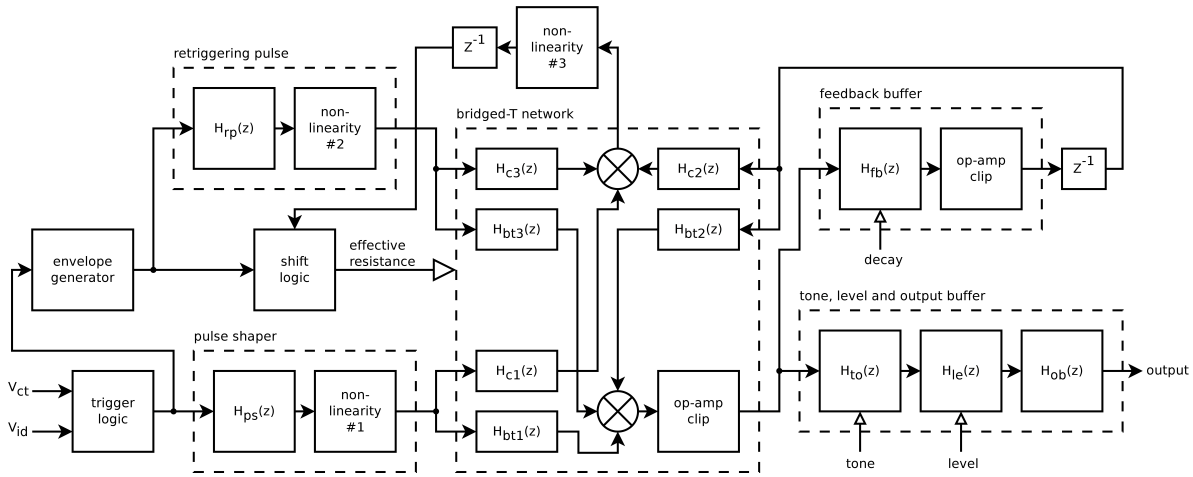


Figure 2: *TR-808 bass drum emulation block diagram.*

3. TRIGGER LOGIC

The CPU controls the timing and amplitude of each sound generator on the 808. A timing signal and accent signal are produced by the CPU, and combined into a common trigger signal (V_{ct}), whose ON voltage is set by a user-controllable global accent level. In general, instrument timing data (unique to each voice) sequenced by the CPU is "ANDed" with V_{ct} to activate individual sound generators.

In the case of the bass drum, the circuit comprised of Q_{39}–Q_{40} and R_{152}–R_{155} "ANDs" the instrument data (V_{id}) with V_{ct}. When V_{id} is present (logic high), a 1-ms long pulse with the same amplitude as V_{ct} (between 4–14 V, depending on VR_3) is passed to the collector of Q_{40}.

4. PULSE SHAPER

The pulse V_{trig} produced by the trigger logic drives a pulse shaper stage, which uses a nonlinear low shelf filter to deliver a shaped pulse to the bridged-T op-amp's inverting input (V_+). This circuit passes the high frequency components of input pulse, while "shaving off" the falling edge of the pulse. Fig. 3 shows the response of the pulse shaper to input pulses with a range of accents.

In the time domain, the output voltage swings immediately high in response to an applied trigger pulse, then smoothly settles down to ${}^{V_{\mathrm{TRIG}}R_{162}}/_{(R_{162}+R_{163})}$. 1 ms later, when the applied pulse returns to ground, the output swings low and then starts to rise smoothly up to ground. It is important to note that the falling edge response is largely independent of the pulse amplitude - each time 0.71 V (approximately one diode drop) develops across D_{53}.

Since it is the edges of the shaped pulse that will kick the bridged-T network into oscillation, this analysis is concerned with the amplitude of each edge. The rising edge has an amplitude equal to the applied trigger voltage V_{TRIG}, and the falling edge has an amplitude approximately equal to ${}^{V_{\mathrm{TRIG}}R_{162}}/_{(R_{162}+R_{163})} + 0.71$.

4.1. Pulse Shaper ODE

An ordinary differential equation (ODE) describing the behavior of the pulse shaper will form a baseline for simulating it. First, recall the equations for the impedances of resistors and capacitors[11] and the Shockley ideal diode equation, which provides a model of the relationship between the current I and voltage V_D of a p–n junction diode:

$$I = I_s \left(e^{\frac{V_D}{V_T}} - 1 \right) , \tag{1}$$

where I_S, the reverse bias saturation current ($\approx 10^{-12}$ A), and V_T, the thermal voltage (≈ 26 mV at room temperature), are properties of the device.

Nodal analysis yields an implicit nonlinear ODE of first order:

$$R_{162}R_{163}C_{40}\left(\frac{\mathrm{d}V_{trig}}{\mathrm{d}t} - \frac{\mathrm{d}V_+}{\mathrm{d}t}\right) - R_{162}V_{trig} + \left(R_{162}+R_{163}\right)V_+ - R_{162}R_{163}I_s\left(e^{\frac{-V_+}{nV_T}} - 1\right) = 0\,. \tag{2}$$

Although it is possible to simulate ODEs like this with numerical methods,[12] good results can be obtained by cascading a linear filter into a memoryless nonlinearity [18]. This can be done by deriving a linear continuous-time transfer function by neglecting the diode, and developing a physically-informed, memoryless nonlinearity to account for the diode's clipping behavior for negative output voltages.

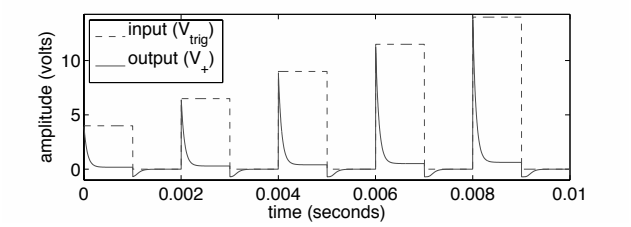


Figure 3: *Pulse shaper behavior under various input pulses.*

[11] $Z_R = R$ and $Z_C = \frac{1}{sC}$, where R is resistance, C is capacitance, and s is the differentiation operator on the complex plane (the "S plane").

[12] [17] provides a good discussion in the context of audio effect simulation.

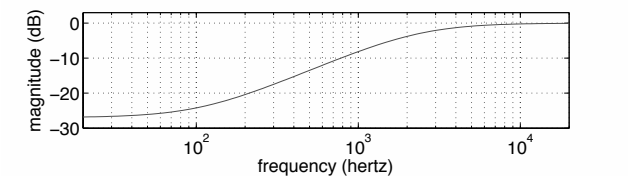


Figure 4: *Pulse shaper shelf filter core, magnitude response.*

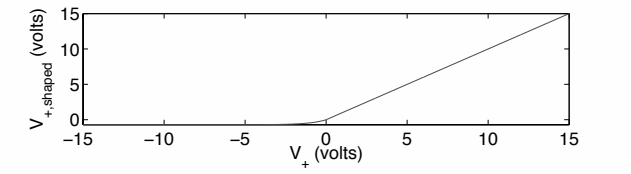


Figure 5: *Pulse shaper diode memoryless nonlinearity.*

An alternate approach to this method would be framing the pulse shaper as a time-variant filter after [19].

4.2. Shelf Filter Core

Neglecting the influence of the diode, the pulse shaper is a passive low shelf filter. Nodal analysis yields a continuous-time transfer function as a fraction of polynomials in s:

$$H_{ps}(s) = \frac{V_+(s)}{V_{trig}(s)} = \frac{\beta_1 s + \beta_0}{\alpha_1 s + \alpha_0}, \tag{3}$$

with coefficients:

$$\begin{aligned}
\beta_1 &= R_{162} R_{163} C_{40} \\
\beta_0 &= R_{162} \\
\alpha_1 &= R_{162} R_{163} C_{40} \\
\alpha_0 &= R_{162} + R_{163}\,.
\end{aligned}$$

Evaluating this transfer function along the $s = j\omega$ axis and taking the magnitude yields the magnitude response of the filter core (shown in Fig. 4).

Note well that the DC response ($s = 0$) of this filter matches the observed response, $R_{162}/(R_{162}+R_{163})$.

4.3. Diode Memoryless Nonlinearity

A memoryless nonlinearity that approximates the diode's function, based on Eqn. (1) is:

$$V_{+,\text{shaped}} = \begin{cases} V_+ & \text{if} \quad V_+ \geq 0 \\ 0.71\left(e^{V_+} - 1\right) & \text{if} \quad V_+ < 0 \end{cases}. \tag{4}$$

As shown in Fig. 5, it will not allow the output voltage to swing lower than approximately one diode drop, and leaves positive output voltages unaffected.

5. BRIDGED-T NETWORK

The bridged-T network is a Zobel network topology which found use in the measurement of high resistances at radio frequencies as early as 1940 [20]. Bridged-T networks in this context were designed using the image-impedance principle to present the same impedance at both their input and output ports. In the 1970s, bridged-T networks found new use in analog drum machines.

When placed in the negative feedback path of an op-amp, a certain type of bridged-T network (capacitive "arms," and resistive "bridge" and path to ground) forms a band pass filter, which can be used to create decaying pseudo-sinusoids in response to impulsive input. This technique was known in the electronics hobbyist community as early as 1979[13] and used by drum machine manufacturers as early as 1976.[14] The 808 represented Roland's first implementation of the bridged-T network, and it was used in every single one of its sound generators in some form.[15]

The bass drum uses the most complicated form of the bridged-T network in the 808. It filters multiple inputs, including the output of the pulse shaper V_+, the output of the feedback buffer V_{fb}, and the retriggering pulse V_{rp} applied via R_{161}. The center frequency of the bridged-T network is subject to modulation (via Q_{43}) by the output of the envelope generator, as well as by leakage through R_{161}.

Ignoring the circuitry that will apply the retriggering pulse and the circuitry which modulates the bridged-T's center frequency, the transfer function $H(s) = V_{bt}(s)/V_+(s)$ is found by defining an intermediate node V_{comm}, and applying nodal analysis. Assuming ideal op-amp behavior, this yields a continuous-time transfer function:

$$H(s) = \frac{V_{bt}(s)}{V_+(s)} = \frac{\beta_2 s^2 + \beta_1 s + \beta_0}{\alpha_2 s^2 + \alpha_1 s + \alpha_0}, \tag{5}$$

with coefficients:

$$\begin{aligned}
\beta_2 &= R_{\text{effective}} R_{167} C_{41} C_{42} \\
\beta_1 &= R_{\text{effective}} C_{41} + R_{167} C_{41} + R_{\text{effective}} C_{42} \\
\beta_0 &= 1 \\
\alpha_2 &= R_{\text{effective}} R_{167} C_{41} C_{42} \\
\alpha_1 &= R_{\text{effective}} (C_{41} + C_{42}) \\
\alpha_0 &= 1\,.
\end{aligned}$$

Evaluating along the $s = j\omega$ axis and taking the magnitude yields a magnitude response $H_{bt1}(s)$ for the simplified bridged-T core, shown in Fig. 6. Respecting superposition (grounding V_{rp} and V_{fb}), the substitution $R_{\text{effective}} = R_{161} \parallel (R_{165} + R_{166}) \parallel R_{170}$ can initially be used. Finding the proper value for $R_{\text{effective}}$ is non-trivial and will be a main focus of the rest of this analysis. Later, the output of the feedback buffer V_{fb} and the source of the retriggering pulse V_{rp} will be treated as second and third inputs to the bridged-T network, and modulations of $R_{\text{effective}}$ via Q_{43} will be considered.

This magnitude response shows a center frequency at ≈ 49.5 Hz,[16] which is close to the entry Roland's "typical and variable" tuning chart (56 Hz) [1] and the sound of a real 808 bass drum.

[13] see: "Rhythm Pattern Generator MM5871" [21, p. 48-49]

[14] It was used in the Korg's mini pops 120 bass drum and snare drum voices. The related twin-T circuit was used in the low and high congas [22].

[15] The TR-808 snare drum, lo/mid/hi tom/congas, and rim shot/clave all use bridged-T networks in similar ways to the bass drum, to create decaying pseudo-sinusoids in response to impulsive input. Bridged-T networks are also used as band pass filters in the remaining voices: handclap, cowbell, cymbal, and open/closed hihat.

[16] [1] provides an equation for the center frequency of a simple bridged-T network: $f_c = 1/\left(2\pi\sqrt{R_{\text{effective}} R_{167} C_{41} C_{42}}\right)$.

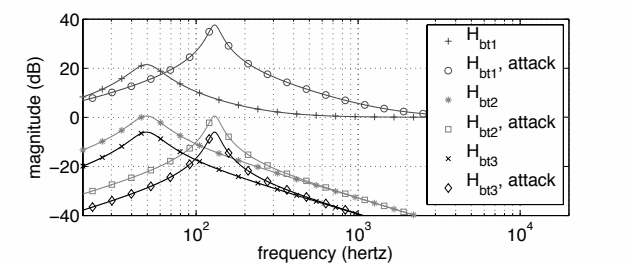


Figure 6: *Magnitude responses of bridged-T network.*

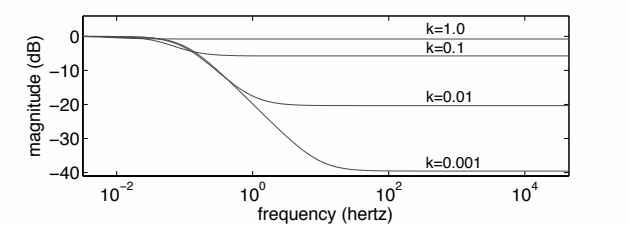


Figure 7: *Family of feedback buffer magnitude responses, with various bass drum decay settings* $k \in [0.001, 1.0]$.

6. FEEDBACK BUFFER STAGE

The length of an 808 bass drum note is user-controllable via potentiometer VR_6, "bass drum decay," which affects the frequency response of the feedback buffer high shelf filter. The output of the bridged-T network is applied to the input of the feedback buffer stage, shaped by its frequency response, and then applied to R_{170}, another input to the bridged-T network. The less that the feedback buffer stage attenuates the signal passing through it, the longer the decay of the 808 note will be.

Assuming an ideal op-amp, there will be no current through, and therefore no voltage across, R_{168}. Nodal analysis yields a continuous-time transfer function:

$$H_{fb}(s) = \frac{V_{fb}(s)}{V_{bt}(s)} = \frac{\beta_1 s + \beta_0}{\alpha_1 s + \alpha_0}, \tag{6}$$

with coefficients:

$$\begin{aligned}\beta_1 &= -R_{169}\mathrm{VR}_6 k C_{43}\\ \beta_0 &= -R_{169}\\ \alpha_1 &= R_{164}\left(R_{169} + \mathrm{VR}_6 k\right) C_{43}\\ \alpha_0 &= R_{164},\end{aligned}$$

where $\mathrm{VR}_6 k$ is the resistance of the decay control with maximum resistance VR_6 and knob position $k \in [0.0, 1.0]$. The magnitude response for the feedback buffer is shown in Fig. 7.

7. BRIDGED-T IN FEEDBACK

When examining the bridged-T network in a feedback configuration, where its input is the output of the feedback buffer section, it will have a different transfer function. This topology is discussed in [23, p. 138].

Defining an intermediate node V_{comm} at the node joining R_{166}, C_{41}, and C_{42}, holding V_- at ground (apropos superposition theorem), applying ideal op-amp assumptions, and applying nodal analysis yields a continuous-time transfer function:

$$H_{bt2}(s) = \frac{V_{bt}(s)}{V_{fb}(s)} = \frac{\beta_1 s}{\alpha_2 s^2 + \alpha_1 s + \alpha_0}, \tag{7}$$

with coefficients:

$$\begin{aligned}\beta_1 &= -R_{\|} R_{167} C_{41}\\ \alpha_2 &= R_{\|} R_{167} C_{41} C_{42}\\ \alpha_1 &= R_{\|} R_{170}\left(C_{41} + C_{42}\right)\\ \alpha_0 &= R_{\|} + R_{170}.\end{aligned}$$

Again, superposition must be respected. $R_{\|}$ is the parallel combination $R_{161} \parallel (R_{165} + R_{166})$. The magnitude response $H_{bt2}(s)$ of the bridged-T in feedback is shown in Fig. 6.

8. FREQUENCY EFFECTS

Discussion of the 808's bass drum often conflates two phenomenon: a brief increase in the center frequency of the bridged-T network by more than an octave during the attack, and the 808's pitch sigh, a subtle modulation caused by leakage through R_{161}.

8.1. Frequency Shift on Attack

In addition to supplying a 1-ms wide pulse to the pulse shaper, the trigger logic also activates an envelope generator comprised of Q_{41}, Q_{42}, and surrounding resistors and capacitors. The output voltage V_{env} of this envelope generator (taken at the collector of Q_{42}) swings quickly up, and doesn't settle back down to ground until approximately 5 ms after the trigger swings low.

While the collector of Q_{42} is high, some current will flow into the base of Q_{43}. The effect of this is that the collector of Q_{43} gets grounded, and the $R_{\mathrm{effective}}$ decreases from $R_{165} + R_{166}$ to R_{166}. So, the transfer functions describing the bridged-T networks behavior will change (labelled "attack"), raising both the Q and the center frequency, as shown in Fig. 6.

Although this brief change of center frequency (≈6 ms, less than a single period at the higher frequency) isn't long enough to be perceived as a pitch shift, it greatly affects the sound of the bass drum's attack, making it "punchier" and "crisper."

A few milliseconds after the start of a note, when the center frequency of the bridged-T network shifts down to its normal position, most of the energy at the normal center frequency will have been attenuated already. To keep the note from experiencing an abrupt jump in amplitude, the circuit composed of C_{39}, R_{161} and D_{52} applies a retriggering pulse to the bridged-T network.

A rough model of this can be obtained by treating the envelope applied at the collector of Q_{42} as an ideal voltage source V_{env} and treating V_{comm} as ground (as shown in Fig. 8). Now, the combination of C_{39}, R_{161} and D_{52} can be viewed as a simple high pass filter with a diode clipper across its output. This can be analyzed just like the pulse shaper from §4.

This retriggering pulse V_{rp} is applied to the bridged-T network through R_{161}. The transfer function $H_{bt3}(s) = {}^{V_{bt}(s)}/_{V_{rp}(s)}$ of this path through the bridged-T network is very similar to the feedback case discussed in §7, save an interchange of of R_{161} and R_{170}. Its magnitude response and the shifted version during the attack are shown in Fig. 6.

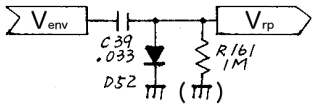


Figure 8: *Retriggering pulse filter.*

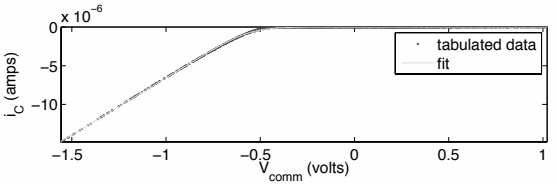


Figure 9: *Fit to memoryless nonlinearity relating V_{comm} and i_C.*

The length of the envelope and the timing of the retriggering pulse can be changed somewhat by changing C_{38}. Changes on R_{156} also affect biasing, and will have more complicated consequences.

8.2. Leakage and Pitch Sigh

Although the center frequency shift during attack is explained in §8.1, this effect is only active for the first few milliseconds of a note. It does not account for the bass drum's characteristic pitch sigh. This sigh is actually a consequence of leakage through R_{161}.

When V_{comm} swings low enough (below about one diode drop below ground), the base of Q_{43} gets lifted up and current flows into the base, causing even more current i_C to be drawn in through its collector. This current changes the effective resistance ($R_{\mathrm{effective}}$, the ratio between V_{comm} and i_{166}, the current drawn through R_{166}), hence altering its transfer function and center frequency.

Although the physics of this part of the circuit are difficult to deal with in closed form, a few observations can lead to a simplified model with good properties. Using SPICE to simulate a stock bass drum note and tabulate V_{comm} and i_C, it is clear that their relationship is reasonably well-approximated by a memoryless nonlinearity, shown in Fig. 9.

This memoryless nonlinearity can be reasonably approximated using a parameterized equation sharing some topological similarity to Eqn. (1):[17]

$$i_C = -\log\left(1 + e^{-\alpha(V_{\mathrm{comm}} - V_0)}\right)\frac{m}{\alpha}\,, \tag{8}$$

where m is the slope of the response in the linear region, V_0 is the voltage offset of the "knee," and α is a parameter that controls the width of the transition region. These parameters are chosen by a three step fitting procedure. First, the method of least squares is used to estimate m and V_0 in the linear region only. These values of m and V_0 are tested with a line search over reasonable α values, again minimizing squared errors. Finally, a fine-grained brute-force search over a small neighborhood surrounding this fit yields some small improvements, and the fit parameters:

$$\begin{aligned}\alpha &= 14.3150\\ V_0 &= -0.5560\\ m &= 1.4765 \cdot 10^{-5}\,.\end{aligned}$$

[17] This is related to the physics of the p–n junction in Q_{43}, not D_{52}.

To get $R_{\mathrm{effective}}$ in terms of just V_{comm}, i_c's effect on $R_{\mathrm{effective}}$ must be taken into account. Considering the entire $R_{\mathrm{effective}}$ branch of the bridged-T network (including i_C), defining an intermediate note V_C at the collector of Q_{43}, applying KCL, and rearranging yields:

$$R_{\mathrm{effective}} = \frac{V_{\mathrm{comm}} R_{166}\left(R_{165} + R_{166}\right)}{V_{\mathrm{comm}}\left(R_{165} + R_{166}\right) - R_{165}\left(V_{\mathrm{comm}} - R_{166} i_C\right)}\,. \tag{9}$$

Note that, when $i_C = 0$, this reduces down to the trivial $R_{165} + R_{166}$, and that as i_C increases, $R_{\mathrm{effective}}$ goes down. A low enough V_{comm} leads to a higher center frequency.

To apply this insight to the previously-derived transfer functions, the series combination of $R_{165} + R_{166}$ should be replaced by $R_{\mathrm{effective}}$.

Furthermore, to change $R_{\mathrm{effective}}$ in terms of V_{comm}, the model must keep track of V_{comm} as the sum of each of the bridged-T's inputs, using transfer functions H_{c1}, H_{c2}, and H_{c3}. The particulars of these transfer functions are related to their counterparts H_{bt1}, H_{bt2}, and H_{bt3}, and can be derived via nodal analysis.

9. TONE, LEVEL, AND OUTPUT BUFFERING STAGE

The bass drum's tone and level controls, and output buffering stage are simply a passive low pass filter, cascaded into a voltage divider, cascaded into a high pass filter.

There are non-negligible loading effects, but they mostly change the position of very low (sub-audible) poles. Since these discrepancies are minimal, inaudible, and are not part of a feedback configuration or upstream of a nonlinearity (where they might have more far-reaching effects), this work breaks this stage into three blocks and makes clear the function of each, despite non-zero connection currents and loading.

Nodal analysis yields first-order continuous-time transfer functions (summarized in Table 1) for each stage of the form:

$$H\left(s\right) = \frac{\beta_1 s + \beta_0}{\alpha_1 s + \alpha_0}\,. \tag{10}$$

Table 1: Output Stage Continuous-Time Filter Coefficients.

Stage	β_1	β_0	α_1	α_0
Low pass	0	1	$R_{eq}C_{45}$	1
Level	$\mathrm{VR}_6\left(1-m\right)C_{47}$	0	$\mathrm{VR}_6 C_{47}$	1
High pass	$R_{177}C_{49}$	0	$R_{176}C_{49}$	1

In Table 1, R_{eq} is the equivalent resistance of the network formed by $R_{171} + (R_{172} \parallel \mathrm{VR}_5)$:

$$R_{eq} = R_{171} + \frac{R_{172}\mathrm{VR}_5 l}{R_{172} + \mathrm{VR}_5 l}\,. \tag{11}$$

$\mathrm{VR}_5 l$ and $\mathrm{VR}_6 m$ are the resistances of the tone and level controls with maximum resistances VR_5 and VR_6 and knob positions $l, m \in [0.0, 1.0]$.

10. MODELING

A digital model is implemented in Cycling 74's `Gen~`, a low-level DSP environment in `Max/MSP`. This model contains stock

controls for bass drum decay, tone, and level, as well as "bends" controlling tuning, attack tuning, extended decay, and input pulse width.

Memoryless nonlinearities are implemented directly, and the trigger logic can be implemented with simple algebraic manipulations on the accent level and timing data. All discrete-time filter coefficients are calculated with the bilinear transform. Although the bilinear transform has many nice properties (including order preservation, stability preservation, and avoiding aliasing), the fact that it maps from continuous-time frequency $\omega_a \in [-\infty, \infty]$ to discrete-time frequency $\omega_d \in (-\pi, \pi]$ means that it necessarily causes some frequency warping [15].[18] The bilinear transform constant c can be tuned to map one continuous-time frequency to precisely the right discrete-time frequency. This is not incorporated in this model, however, since salient filter features are at low frequencies with respect to the sampling rate. So, the standard untuned value of $c = 2/T$ is used throughout [15].[19]

The choice of filter topology is particularly important, since many of the filter blocks (especially the bridged-T network) will feature time-varying coefficients. In the model, all filter blocks are implemented with Transposed Direct Form-II (TDF-II) topologies. The TDF-II topology has good numerical properties and behaves well under changing coefficient values, which is of special importance for the bridged-T network, with its constantly-shifting effective resistances. TDF-II seems sufficient, but in the future other filter topologies such as the normalized ladder filter [15][20] and Max Matthews' phasor filter [24] can be explored for even better properties under time-varying coefficients.

The feedback arrangement of the bridged-T network and the feedback buffer creates a delay-free loop. In the model, this can be addressed by inserting a unit delay after the feedback buffer, which will have only negligible effects on the frequency response.[21] Alternatively, the delay-free loop could be avoided by combining the analog prototypes in feedback before digitization via the bilinear transform.

Some parts of the analysis feature deviations from the behavior of the real device. In particular, the behavior of Q_{43} and the interaction between the bridged-T network and the envelope generator, though physically-informed, are oversimplified.

Where connection currents between sub-circuits would not be negligible, results could be improved by methods such as [25].

11. RESULTS

Fig. 10 shows a time-domain plot of the first 13 ms of a single bass drum note, showing good agreement between the physical model and a SPICE simulation.

Fig. 11 shows the estimated instantaneous frequency of the first 300 ms of a single bass drum note,[22] showing good agreement between the physical model and a SPICE simulation. Note well the unique characteristics of the pitch sigh.

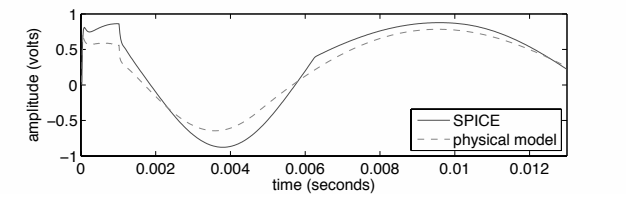


Figure 10: *Transient response.*

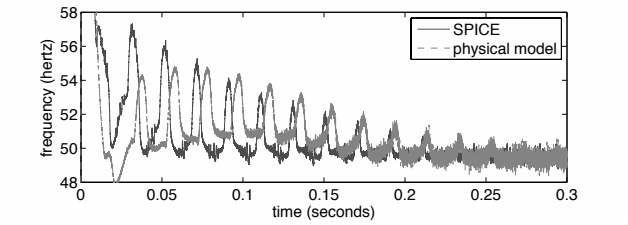


Figure 11: *Estimated instantaneous frequency.*

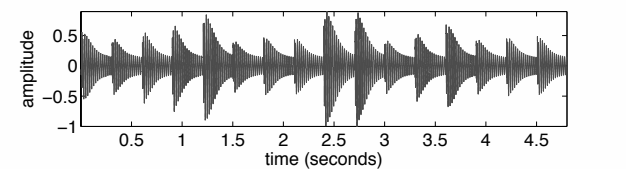


Figure 12: *Lack of "machine gun effect" in model output.*

Fig. 12 shows a time-domain plot of the bass drum physical model when it is retriggered before the previous note has gone silent. Each note is slightly different, as in a real 808, since the remaining filter states may interfere constructively or destructively with the response to a new trigger—the model avoids the "machine gun effect."

In light of this analysis, variations in sound between individual 808s are easily understood as a consequence of the tolerances of the circuit elements (the voice circuits featured $\pm 20\%$ capacitors and $\pm 5\%$ resistors). These variations have significant effects on the gain, center frequency, Q, decay time, &c. of filter sections, especially when they are in feedback configurations (the bridged-T network and feedback buffer).

Audio examples and other supplementary materials can be found online at this work's companion page.[23]

12. CONCLUSION

In addition to informing a real-time computer model of the TR-808 bass drum, these findings will have many secondary applications. Due to the original hardware's value (and the danger of working with mains-powered electronics), 808 modifications are often quite conservative, and it has been a rare subject for circuit benders. This work supports the design of further modifications and can inform the design of future drum machines.

This research forms a foundation for musicological inquiry into Roland's line of analog drum machines, which traces roots

[18] https://ccrma.stanford.edu/~jos/fp/Frequency_Warping.html

[19] https://ccrma.stanford.edu/~jos/pasp/Bilinear_Transformation.html

[20] https://ccrma.stanford.edu/~jos/pasp/Conventional_Ladder_Filters.html

[21] since all of the frequency response features are very low in frequency with respect to the sampling rate

[22] estimated by taking the Hilbert transform of the time-domain signal to obtain an analytic version, then estimating the derivative of its phase

[23] https://ccrma.stanford.edu/~kwerner/papers/dafx14.html

back to Ace Tone's 1964 R1 Rhythm Ace.

The modeling of additional 808 voices will open the door for experimentation with flexible signal routings and feedback between voices and sub-circuits.

We've partitioned the circuit into blocks, and made a detailed first-order model of each block. The performance of this model makes it clear that the architecture of the 808 bass drum and complex interactions between subcircuits are more important than subtle device nonlinearities.

Even without tuning, a modeling scheme derived from this analysis shows the same features as a SPICE model and recordings of a TR-808 [26], including basic tuning and timing, a lack of the "machine gun effect" on quickly repeated notes, accurate transient behavior on different accent levels, and proper handling of the complicated Q and center frequency trajectories of the bridged-T network under the entire range of decay settings and accents. This correspondance is consistent with the results of informal listening tests.

13. ACKNOWLEDGMENTS

Some analysis insights were developed with Kevin Tong as part of prof. Greg Kovacs's Analog Electronics course. Thanks to Melissa Kagen, Chet Cnegny, Christine Dakis, and Mayank Sanganeria for help with editing.

14. REFERENCES

[1] Roland Corporation, "TR-808 Service Notes, first edition," June 1981.

[2] Beastie Boys, "Hello Nasty," Compact Disc, July 1998.

[3] Micky Delp, "Anatomy of a Drum Machine," Available at http://mickeydelp.com/news/108-anatomy-of-a-drum-machine.html, July 31, 2012; accessed January 08, 2014.

[4] Propellorhead Software, "The Rebirth Museum," http://www.rebirthmuseum.com, September 2005.

[5] Eric Archer, "Blog archive » 808 clones," Available at http://ericarcher.net/devices/tr808-clone/, accessed January 08, 2014.

[6] Gordon Reid, "Synth Secrets: Practical Bass Drum Synthesis," *Sound on Sound*, July 2002; accessed January 8, 2014, Available at http://www.soundonsound.com/sos/Feb02/articles/synthsecrets0202.asp.

[7] John Pillans and James Ridgway, "ENGR3227 Analogue Electronics, Electronics Project: Bass Drum Synthesiser," Tech. Rep., Australian National University, 2006; accessed January 08, 2014, Available at http://users.cecs.anu.edu.au/~Salman.Durrani/_teaching/TB1.pdf.

[8] Mark Vail, *Vintage Synthesizers: Pioneering Designers, Groundbreaking Instruments, Collecting Tips, Mutants of Technology*, Backbeat Books, second edition, 2000.

[9] Derek Walmsley, "Monolake in full," *The Wire*, January 2010; accessed February 3, 2014, Available at http://thewire.co.uk/in-writing/interviews/monolake-in-full.

[10] Robin Whittle, "Modifications for the Roland TR-808," Available at http://www.firstpr.com.au/rwi/tr-808/, October 7, 2012; accessed January 08, 2014.

[11] Kurt James Werner and Mayank Sanganeria, "Bit Bending: an Introduction," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-16)*, Maynooth, Ireland, September 2–5, 2013.

[12] Julian Parker and Stephano D'Angelo, "A Digital Model of the Buchla Lowpass Gate," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-16)*, Maynooth, Ireland, September 2–5, 2013.

[13] David T. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*, Ph.D. thesis, Stanford University, 2009.

[14] David T. Yeh, John Nolting, and Julius O. Smith, "Physical and Behavioral Circuit Modeling of the SP-12 Sampler," in *Proceedings of the 33rd International Computer Music Conference (ICMC)*, Copenhagen, Denmark, August 30, 2007.

[15] Julius O. Smith, *Physical Audio Signal Processing*, http://ccrma.stanford.edu/~jos/pasp/, 2010; accessed January 2, 2104, online book.

[16] NEC Corporation, "μPC4558 data sheet," March 1993.

[17] David T. Yeh, Jonathan S. Abel, and Julius O. Smith, "Simulation of the Diode Limiter in Guitar Distortion Circuits by Numerical Solution of Ordinary Differential Equations," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-10)*, Bordeaux, France, September 10–15, 2007.

[18] David T. Yeh, Jonathan S. Abel, and Julius O. Smith, "Simplified, physically-informed models of distortion and overdrive guitar effect pedals," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-10)*, Bordeaux, France, September 10–15, 2007.

[19] Jaromir Macak and Jiri Schimmel, "Nonlinear Circuit Simulation Using Time-Variant Filter," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-12)*, York, UK, September 17–21, 2009.

[20] P. M. Honnel, "Bridged-T Measurement of High Resistances at Radio Frequencies," *Proceedings of the Institute of Radio Engineers*, vol. 28, no. 2, pp. 88–90, February 1940.

[21] Forrest M. Mims III, *Engineer's Notebook: A Handbook of Integrated Circuit Applications*, Radio Shack, first edition, 1979.

[22] Keio Electronic Laboratory Corporation, "Korg mini pops 120 Service Manual," 1976.

[23] T. Deliyannis, Yichuang Sun, and J.K. Fidler, *Continuous-Time Active Filter Design*, CRC Press, first edition, 1999.

[24] Dana Massie, "Coefficient Interpolation for the Max Mathews Phasor Filter," in *Proceedings of the 133rd Audio Engineering Society convention*, San Francisco, CA, October 26–29, 2012.

[25] Jaromir Macak, "Guitar Preamp Simulation Using Connection Currents," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-16)*, Maynooth, Ireland, September 2–5, 2013.

[26] Michael Fischer, "Roland TR-808 Rhythm Composer Sound Sample Set 1.0.0," September 1994.

THE MODULATION SCALE SPECTRUM AND ITS APPLICATION TO RHYTHM-CONTENT DESCRIPTION.

Ugo Marchand
STMS IRCAM-CNRS-UPMC
1 pl. Igor Stravinsky, 75004 Paris, France
ugo.marchand@ircam.fr

Geoffroy Peeters
STMS IRCAM-CNRS-UPMC
1 pl. Igor Stravinsky, 75004 Paris, France
geoffroy.peeters@ircam.fr

ABSTRACT

In this paper, we propose the Modulation Scale Spectrum as an extension of the Modulation Spectrum through the Scale domain. The Modulation Spectrum expresses the evolution over time of the amplitude content of various frequency bands by a second Fourier Transform. While its use has been proven for many applications, it is not scale-invariant. Because of this, we propose the use of the Scale Transform instead of the second Fourier Transform. The Scale Transform is a special case of the Mellin Transform. Among its properties is "scale-invariance". This implies that two time-stretched version of a same music track will have (almost) the same Scale Spectrum. Our proposed Modulation Scale Spectrum therefore inherits from this property while describing frequency content evolution over time. We then propose a specific implementation of the Modulation Scale Spectrum in order to represent rhythm content. This representation is therefore tempo-independent. We evaluate the ability of this representation to catch rhythm characteristics on a classification task. We demonstrate that for this task our proposed representation largely exceeds results obtained so far while being highly tempo-independent.

1. INTRODUCTION

Automatic description of audio content has become over the years an important research field. When the audio content is music, the related research field is named Music Information Retrieval. Its growth over the years is explained by the massive digitalization of music and its accessibility through online-services (on-line music sellers –as iTunes– or music streaming services –as Spotify or Deezer–). This necessitates the organization and tagging of the music data to enhance their accessibility. While manual annotation is still possible, it is time-consuming hence money-consuming. Because of this, tools are developed for automatically organizing, tagging and visualizing the music data. These tools rely on automatic audio content analysis. In the case of tagging, automatic content analysis tools rely on two parts: extracting the right information (named audio features) from the audio signal and matching this information to the required tags.

In this paper, we propose a new representation, named Modulation Scale Spectrum, that allows an efficient description of time/frequency content over time. We apply it to create an audio feature that allows an efficient description of the rhythm content of a music track. Rhythm, along harmony (melody) and timbre (orchestration) are the three perspectives that describe music content.

The direct applications of this feature are the search by rhythm pattern (for example looking for identical rhythm pattens without being affected by the tempo), the automatic classification into rhythm-classes. A better description of rhythm would be also beneficial to genre, mood classification or search by similarity systems. Applications of the Modulation Scale Spectrum outside the music field also concern generic audio recognition.

1.1. Related works

In this part, we only review works related to audio rhythm representation. We also provide an overview on the Modulation Spectrum since our proposal is based on it. For a good overview on other methods to represent time and frequency content see [1].

Works can generally be divided according to the use of temporal, spectral or spectro/temporal representations; according to the matching model (Dynamic Time Warping, or statistical models), according to the necessity to have a preliminary tempo detection.

In [2], Foote introduces the *beat spectrum*. The beat spectrum is computed by parametrizing audio (with STFT amplitude coefficients or with its MFCC). A similarity measure (cosine or euclidian distance) is then taken between each feature vector and embedded in a 2-dimensional representation named Self-Similarity-Matrix (SSM) S. Finally the beat-spectrum is found by looking for periodicities in S with diagonal sums or auto-correlation. In [3], Foote uses his beat spectrum to measure rhythmic similarity between songs. Bartsch extends the beat spectrum to create an audio thumbnail in [4]. Antonopoulos in [5] also uses the SSM approach. He extracts from a song (or its thumbnailed version) a chroma-based MFCC feature. He computes a rhythmic signature by using Dynamic Time Warping (DTW) to calculate the similarity between two feature vectors. He validates his method by measuring rhythmic similarity on a greek and an african music corpus.

Dixon [6] uses *rhythmic patterns* to classify rhythm on the Ballroom Dataset. It achieves 50% correctness, and up to 96% (85.7% without annotated tempo) by combining with other features derived from these patterns, from features of Gouyon [7], and from annotated tempo. However, a limitation of this method is that position of the first bars is needed to extract the rhythmic pattern. It has been estimated with BeatRoot algorithm [8] but have been "corrected manually". Paulus [9] firstly detects tatus, tatum and bar length to segment a song into patterns. Then he computes features on theses patterns (loudness, brightness for example) and measures the similarity of rhythmic patterns with DTW. Wright [10] creates clave pattern templates to analyse afro-cuban music. He uses matched-filtering to enhance claves, and a rotation-aware dynamic programming algorithm to find tempo, beat and downbeat positions. Jensen [11] uses a log-scale autocorrelation to create tempo-independent rhythmic patterns. His method works nicely on the rhythm classification task on the Ballroom Dataset if we ignore nearest neighbours with similar in tempo (50% accuracy

instead of 20% for state-of-the-art method).

Tzanetakis [12] proposes a *Beat histogram* computed by using an enhanced auto-correlation function and peak-picking algorithm. This beat histogram, along with timbre and pitch features is used for music genre classification. Gouyon [7] uses a set of 73 features based on the tempo, on the *periodicity histogram* and on the *inter-onset-interval histogram*. The system achieves a 78.9% accuracy on the Ballroom Dataset classification task, and up to 90.1% using the annotated tempo.

Peeters [13] compares various spectral and temporal periodicity representations to describe the rhythm of a song. He firstly extracts an onset function from the signal then computes three feature vectors (based on the amplitude of the Discrete Fourier Transform, the Auto-Correlation Function, and Hybrid-Axis-Autocorrelation-Fourier-Transform representation). He then makes these vectors tempo independent and compact by sampling them at multiple of the tempo frequency. Theses features are tested for rhythm classification on the ballroom dataset. They achieve 96.1% classification accuracy with the annotated tempo and 88% without.

Holzapfel [14] proposes *Dynamic Periodicity Warping* to classify rhythm. He uses a DTW distance on the periodicity spectrum between each song of the ballroom dataset as input of a k-nearest neighbours classifier. The process achieves 82.1% accuracy on the ballroom dataset, which is a bit less than state-of-the-art methods, but outperforms all other algorithms on a dataset made of Cretan dances (70% against 50%).

Directly related to our method are the following works.

Holzapfel [15] proposes a representation to describe rhythm: the *Scale Transform*. The main property of the scale transform is that the scale transforms of two identical songs played at different speed will be the same. He uses this scale transform in the creation of a tempo-independent descriptor. The performance of this descriptor on the ballroom dataset are good: 85.1% (state of the art is about 88%). In [16], Holzapfel tests his descriptor on two additional datasets (Turkish and Greek traditional music which have wider within-class tempo distribution). It achieves better classes recognition than other methods.

Rodet and Worms [17, 18] proposes the *Modulation Spectrum* (without naming it explicitly Modulation Spectrum) for a task of audio identification by fingerprint. It models the temporal evolution of the energy content in different frequency bands with a Fourier Transform. McKinney [19] computes the Modulation Spectrum on the output of 18 4^{th} order bandpass GammaTone filters, then sums the energy on four frequency bands to get 18x4 descriptors. He uses these features to classify five audio classes (classical, pop music, speech, noise, crowd) and 7 musical genres. Whitman [20] proposes the "Penny" features to create a system of automatic recording reviews that understands and labels songs based on their audio content. Atlas [21] proposes a joint acoustic/modulation frequency model to improve audio coding. This model coded at 32 kbit/s has been found better by listeners than MP3 coding at 56 kbit/s.

1.2. Paper overview and organization

The Modulation Spectrum allows a compact description of the time and frequency content. Its use has been proven for many applications (audio fingerprint [18], generic audio classes or genre recognition [19], auto-tagging [20], speaker separation [21]). However, this representation is not scale-invariant.

In this paper, we propose the Modulation Scale Spectrum as an extension of the Modulation Spectrum to the Scale domain. This makes it scale-invariant. This property allows then to derive audio features for music audio signals which are tempo invariant. As MFCC provides the spectral-envelope information complementary to the fundamental frequency, the Modulation Scale Spectrum provides the information complementary to the music tempo. While the Scale Transform by itself already allows this, the Scale Transform does not allow to represent the spectral-envelope information (only a single energy value is used to represent the whole spectrum). The Modulation Scale Spectrum does represent this spectral-envelope information, as the Modulation Spectrum does, but is tempo-invariant, while the Modulation Spectrum is not.

The paper is organized as follows. In part 2, we introduce the Modulation Scale Spectrum. We first describe the Scale Transform (part 2.1) and the Modulation Spectrum (part 2.2) it is based on and then present in part 2.3 our proposed Modulation Scale Spectrum. We provide a specific implementation of it for rhythm description in part 2.4 and successfully demonstrate its use for a task of rhythm class recognition in part 3.

2. MODULATION SCALE SPECTRUM

In this part we propose the Modulation Scale Spectrum. It is based on both the Scale Transform (ST) and the Modulation Spectrum (MS). We first briefly review those.

2.1. Scale Transform

The Scale Transform is a special case of the Mellin Transform. It has been introduced by Cohen in [22]. Scale, like frequency is a physical attribute of the signal. We can see the scale content of a signal by using the Scale Transform, just like we can see the frequency content by using the Fourier Transform. The scale transform is defined by :

$$D(c) = \frac{1}{\sqrt{2\pi}} \int x(t) t^{-jc-\frac{1}{2}} dt \tag{1}$$

where c is the scale variable, t the time, and x the signal.

It can be viewed as the Fourier transform of an exponentially re-sampled signal $x(e^t)$ weighted by an exponential window $e^{\frac{1}{2}t}$:

$$D(c) = \frac{1}{\sqrt{2\pi}} \int x(e^t) e^{\frac{1}{2}t} e^{-jct} dt \tag{2}$$

One of the most important property of the Scale Transform is *scale invariance*. Scale invariance is expressed as

$$\begin{aligned} x(t) &\rightleftharpoons D(c) \\ \sqrt{a}\, x(at) &\rightleftharpoons e^{jc \ln a} D(c) \end{aligned} \tag{3}$$

This implies that both $x(t)$ and $\sqrt{a}\, x(at)$ share the same modulus of the scale spectrum but differ in their phase.

It should be noted however that the Scale Transform is not shift invariant: $|ST(x(t))| \neq |ST(x(t+a))|$. For this reason, the Scale Transform is often applied to the auto-correlation of $x(t)$ instead of $x(t)$ itself.

If $x(t)$ represents the energy of the audio signal, two audio signals with the same rhythm pattern and starting at the same time but played at different speed will share the same modulus of the scale spectrum (this is the property used by Holzapfel [15, 16]).

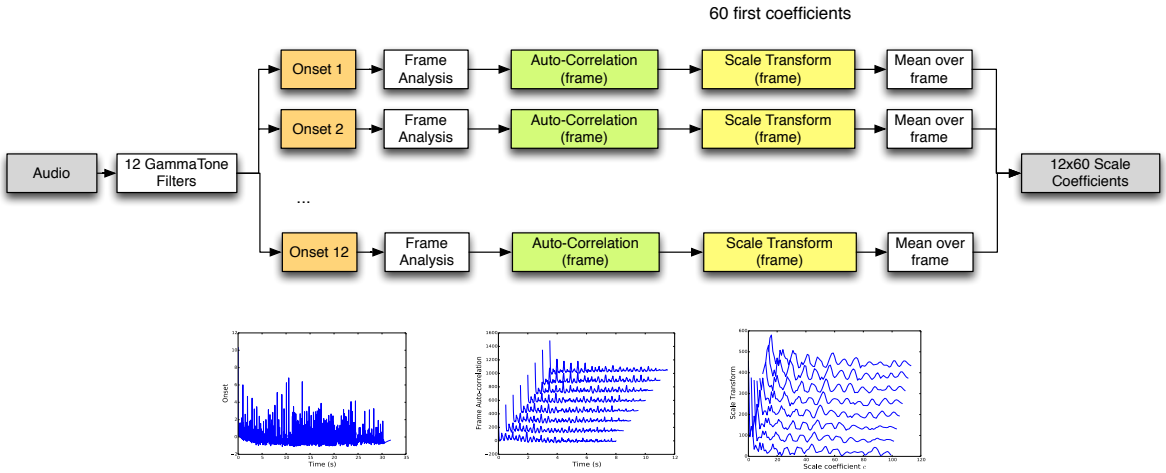


Figure 1: *Flowchart of our proposed Modulation Scale Spectrum for rhythm description.*

Computation: As seen, the ST can be computed in an efficient way from the Fourier Transform of $x(e^t)e^{\frac{1}{2}t}$ with $\omega = c$. In the case of the Discrete Fourier Transform $\omega_k = 2\pi\frac{k}{N}sr$ with $k \in \mathbb{N}$, N the size of the FFT and sr the sampling rate. In this case, c as ω_k is linearly-spaced. In the following we will use the implementation kindly provided by [23][1].

The exponential resampling step is detailled in [23]. We briefly review it here. We create an exponential axe between $t_0 = T_s$ and $t_e = nT_s$ where $T_s = \frac{1}{sr}$ is the sampling period and n the number of samples of the original signal. To fulfill the Nyquist-Shanon resampling condition, the maximum step between two adjacent exponential sample can not exceed T_s. In order not to have too many exponential samples, we choose as the distance between the two last samples the largest allowed distance: T_s. We then use a spline interpolation to create the resampled points. This leads to approximately $n\ln(n)$ exponential samples.

2.2. Modulation Spectrum

The modulation spectrum represents the evolution over time of the amplitude content of the various frequency bands ω of an STFT by a second Fourier Transform. It has been proposed by different authors under various names: dynamic features [18], modulation spectrum [21], auditory filterbank temporal envelopes [19], "penny" features [20]. It is also closely related to the scattering features proposed by [24]. The modulation spectrum $X(\omega, \Omega)$ can be expressed as (using [18] notation)

$$\begin{aligned} x(\omega,\tau) &= \frac{1}{\sqrt{2\pi}}\int_t x(t)h(\tau - t)e^{-j\omega t}dt \\ X(\omega,\Omega) &= \frac{1}{\sqrt{2\pi}}\int_\tau |x(\omega,\tau)|e^{-j\Omega\tau}d\tau \end{aligned} \quad (4)$$

In this ω denotes the frequencies of the STFT (ranging from 0 to half-Nyquist frequencies), t the time, τ the center-time of the STFT windows and Ω the frequencies of the second Fourier Transform (which upper frequency depends on the hop-size of the STFT).

[1] `http://profs.sci.univr.it/~desena/FMT/`

In [19], it is proposed to use 18 GammaTone filters to create $x(\omega, \tau)$ instead of the STFT. The bands of the GammaTone filters are centered on a log-space from 26 to 9795 Hz.

2.3. Modulation Scale Spectrum

We propose here the Modulation Scale Spectrum as an extension of the Scale Transform to allow its application to individual frequency bands. It therefore allows to take the benefits of the Modulation Spectrum while ensuring the *scale invariance* property.

It is expressed as

$$D(\omega, c) = \frac{1}{\sqrt{2\pi}}\int |x(\omega, e^\tau)|e^{\frac{1}{2}t}e^{-jc\tau}d\tau \quad (5)$$

where ω is defined as above and c is the scale (as in eq. (2)).

2.4. Modulation Scale Spectrum for rhythm description

We propose a specific implementation of the Modulation Scale Spectrum to describe the rhythm content of a music signal. In this, $x(\omega, \tau)$ does not represent the signal itself but the auto-correlation of an onset-energy function[2] derived from the signal in a specific frequency band ω.

The flowchart of its computation is indicated into Figure 1 and described below.

1. As in [19], we first separate the audio signal $x(t)$ using 4^{th} order bandpass GammaTone filters centered on a log-space from 26 to 9795 Hz. In our case, we use 12 filters instead of the 18 proposed by [19][3].
2. We then calculate an onset-energy function $o(\omega, \tau)$ on the output of each filter. For this, we used the function proposed by Ellis [26][4]. We parametrized it to have a sampling rate of 50 Hz.

[2] A onset-energy-function is a function taking high values when an onset is present and low values otherwise.

[3] For the GammaTone filters, we used the implementation kindly proposed by Ma [25] whose code is available at `http://staffwww.dcs.shef.ac.uk/people/N.Ma/resources/gammatone/`.

[4] `http://labrosa.ee.columbia.edu/projects/coversongs/`

3. We then perform a frame-analysis with an 8 s. length rectangular window and a 0.5 second step. We denote the frames by u.
4. As in [15], we then compute on each frame the autocorrelation of each $o(\omega, \tau)$. We denote the resulting functions by $R_{xx}(\omega, \tau, u)$.
5. Finally we compute the Modulation Scale Spectra on each autocorrelation functions $R_{xx}(\omega, \tau, u)$. We denote it by $D(\omega, c, u)$.
6. In order to obtain a single representation for the whole audio signal, we compute the average value of $D(\omega, c, u)$ over u. We denote it by $D(\omega, c)$. The resulting dimensionality is $[12, 2000]$.
7. In order to reduce this dimensionality, we could group the values of the various scale-bands c (as McKinney and Whitman did for the FFT frequencies). Another possibility starts from the consideration that most of the information of the Modulation Scale Spectrum is contained in the first scale coefficients c. Using this, Holzapfel in [15] only kept the first 40 coefficients of the Scale-Transform. In our case, we keep the first 60 scale coefficients ($c \in [1, 60]$) of the Modulation Scale Spectrum.

3. APPLICATION OF THE MODULATION SCALE SPECTRUM FOR RHYTHM RECOGNITION

In order to validate our Modulation Scale Spectrum we apply it to the task of rhythm recognition.

3.1. Test-set

For this, we use a test-set annotated into classes of rhythm: the Ballroom test-set. This one contains 698 titles divided into 8 music genres. Because in ballroom music, the genre ("ChaChaCha", "Rumba" ...) are closely related to the type of rhythm pattern, we consider this genre labels as classes of rhythm. This test-set was created for the ISMIR 2004 rhythm description contest [27]. It is extracted from the website `www.ballroomdancers.com`.

3.2. Experimental scenario

In the first experimental scenario, the task consists in recognizing the correct rhythm class.

Ballroom rhythm classes have a predominant tempo (i.e. ChaChaCha tracks are usually around 125bpm, QuickStep around 200bpm ...). Which means that it would be possible to recognize the rhythm classes simply by using the tempo of the track (if tempo==200 then class=QuickStep). This has been shown for example by Gouyon [7] who reported 82.3 % accuracy using only annotated tempo and a k-Nearest Neighbours (k-NN) classifier. Because of this, it is difficult to highlight the benefits of using a tempo-invariant-feature such as the Modulation Scale Spectrum. In order to show this benefit, we created a second experimental scenario. Considering that our classification algorithm is a KNN, in the second scenario we will ignore the tracks that have a tempo within 4 % of the tempo of the target[5].

[5]It should be noted that Viennese Waltz genre has been discarded in this experiment since all its tempi are within a 5 % range.

3.3. Classifier

As explained in part 2.4, each track is represented by its Modulation Scale Spectrum $D_i(c, \omega)$ as described. For the two scenario described above, we use a modified K-Nearest-Neighbor to perform the classification.

As showed in eq. (3), the Scale Transform is scale-invariant ignoring a global multiplication factor $\sqrt{\alpha}$. Because of this factor, we use the one-minus-cosine-distance[6] in the K-NN instead of the usual Euclidean distance.

If we denote by i the target point of the K-NN, and by $j \in [1, K]$ the K nearest-points, we assign to each j the following weight: $w_{i,j} = 1 - \frac{d_{i,j}}{d_{K+1}}$. We then attribute a global score to each class by summing the weights $w_{i,j}$ of the points j that belong to the class. The class with the largest score is then assigned to i. This modified K-NN algorithm has been found superior to the equally weighted K-NN by Holzapfel [14]. The best value of K and C (the number of scale coefficients) have been found by performing a grid-search.

The results presented in the following has been obtained using 10-fold cross-validation.

3.4. Results and discussions

Results are indicated into Table 1. In this table, we compare the performances obtained by our method to the ones obtained by Holzapfel [15] and Peeters [13]. The sign "-" denotes the fact that the result is not available for the given configuration. In order to be able to compare our algorithm to the one of Holzapfel in the second scenario (Exp. 2), we re-implemented Holzapfel method. Surprisingly, this re-implementation provides better results than the ones published by the author[7].

For the first scenario (Exp. 1), our proposed Modulation Scale Spectrum (93.12% accuracy) outperforms state-of-the-art methods by more than 5 %. Holzapfel method obtains 86.9 % and Peeters 87.96 %.

For the second scenario (Exp. 2), which remove tracks closely-related in tempo from the K-NN, our proposed Modulation Scale Spectrum (75.52% accuracy) outperforms state-of-the-art methods by more than 9 %. Jensen [11] reports 48.4% with his "tempo-insensive rhythmic pattern", our re-implementation of Holzapfel reaches 66.48%. This second experiment demonstrates that our Modulation Scale Spectrum is able to capture rhythmic content, without being dependent on the tempo.

On Fig 2, we represent the first 60 coefficients corresponding to the 9-th GammaTone filters[8]of the Modulation Scale Spectrum for the 698 tracks of the Ballroom test-set. Each row of the matrix represents the 60 coefficients of our Modulation Scale Spectrum for a given track. As can be seen, the various tracks are visually clustered according to the rhythm-classes (represented on the y-axis). Each rhythm class has a dominant pattern. Unsurprisingly, Waltz and Viennese Waltz have close patterns where most of their energy is concentrated on the first coefficient.

[6]$d_{i,j} = 1 - \frac{D_i . D_j}{|D_i||D_j|}$

[7]This may be due to a different split of the test-set into ten folds.

[8]It should be noted that we couldn't display all the Modulation Scale Spectrum for the 12 frequency bands, so we selected the 9^{th} GammaTone band as an example. All other bands show similar behavior.

Table 1: *Results and parameters (best values of C and K) for the recognition of the ballroom rhythm classes.*

Method	Exp. 1			Exp. 2		
	Accuracy	C	K	Accuracy.	C	K
Jensen	-	-	-	48.4 %		1
Holzapfel	86.9 %	40	5	-	-	-
Holzapfel (re-implemented)	87.82 %	40	11	66.48 %	20	5
Peeters	87.96 %	-	-	-	-	-
Modulation Scale Transform	**93.12 %**	**60**	**5**	**75.52** %	**20**	**5**

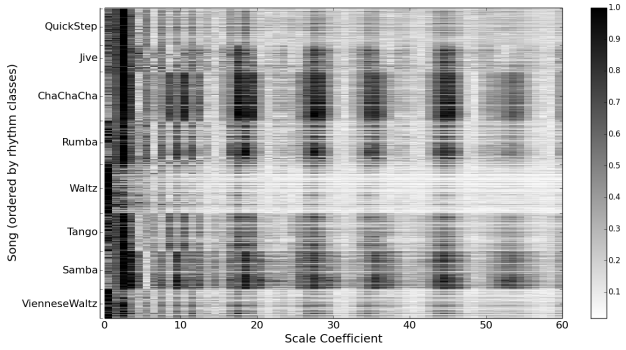


Figure 2: *First 60 coefficients (x-axis) of the Modulation Scale Spectrum for all the songs of the Ballroom test-set. The songs of the test-set are grouped by rhythm-classes on the y-axis. The coefficients are normalized : their amplitudes are represented in gray scale (black for a normalized amplitude of 1, white for 0)*

4. CONCLUSION

In this paper, we proposed the Modulation Scale Spectrum as an extension of both the Modulation Spectrum and the Scale Transform. It takes the benefits from the Modulation Spectrum while ensuring *scale invariance*. We used the Modulation Scale Spectrum to create a rhythm description feature. The whole process includes GammaTone filtering, onset-strength energy estimation, auto-correlation of those and finally Scale Transform. We tested this feature for a rhythm-class recognition task. We demonstrated that for this task our proposed feature largely exceeds results obtained so far. With a second experiment we showed that this feature is indeed tempo-independent. In future works, we will extend our use of the Modulation Scale Spectrum as audio representation for other MIR tasks such as genre classification.

Acknowledgements

This work was founded by the French government Programme Investissements d'Avenir (PIA) through the Bee Music Project.

5. REFERENCES

[1] Cyril Joder, Slim Essid, and Gaël Richard, "Temporal integration for audio classification with application to musical instrument classification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 17, no. 1, pp. 174–186, 2009.

[2] Jonathan Foote and Shingo Uchihashi, "The beat spectrum: A new approach to rhythm analysis.," in *ICME*, 2001.

[3] Jonathan Foote, Matthew L Cooper, and Unjung Nam, "Audio retrieval by rythmic similarity," in *ISMIR*, 2002.

[4] Mark A Bartsch and Gregory H Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *Multimedia, IEEE Transactions on*, vol. 7, no. 1, pp. 96–104, 2005.

[5] Iasonas Antonopoulos, Aggelos Pikrakis, Sergios Theodoridis, Olmo Cornelis, Dirk Moelants, and Marc Leman, "Music retrieval by rhythmic similarity applied on greek and african traditional music," in *ISMIR*, 2007.

[6] Simon Dixon, Fabien Gouyon, Gerhard Widmer, et al., "Towards characterisation of music via rhythmic patterns.," in *ISMIR*, 2004.

[7] Fabien Gouyon, Simon Dixon, Elias Pampalk, and Gerhard Widmer, "Evaluating rhythmic descriptors for musical genre classification," in *Proceedings of the AES 25th International Conference*. Citeseer, 2004, pp. 196–204.

[8] Simon Dixon, "n interactive beat tracking and visualisation system," in *Proceedings of the international computer music conference*, 2001, pp. 215–218.

[9] Jouni Paulus and Anssi Klapuri, "Measuring the similarity of rhythmic patterns.," in *ISMIR*, 2002.

[10] Matthew Wright, W Andrew Schloss, and George Tzanetakis, "Analyzing afro-cuban rhythms using rotation-aware clave template matching with dynamic programming.," in *ISMIR*, 2008, pp. 647–652.

[11] Jesper Højvang Jensen, Mads Græsbøll Christensen, and Søren Holdt Jensen, "A tempo-insensitive representation of rhythmic patterns," in *Proc. of the 17th European Signal Processing Conf.(EUSIPCO-09)*. Citeseer, 2009.

[12] George Tzanetakis and Perry Cook, "Musical genre classification of audio signals," *Speech and Audio Processing, IEEE transactions on*, vol. 10, no. 5, pp. 293–302, 2002.

[13] Geoffroy Peeters, "Spectral and temporal periodicity representation of rhythm for the automatic classification of music audio signal," *IEEE*, 2011.

[14] Andre Holzapfel and Yannis Stylianou, "Rhythmic similarity of music based on dynamic periodicity warping," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 2217–2220.

[15] Andre Holzapfel and Yannis Stylianou, "A scale transform based method for rhythmic similarity of music," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 317–320.

[16] André Holzapfel and Yannis Stylianou, "Scale transform in rhythmic similarity of music," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 1, pp. 176–185, 2011.

[17] L. Worms, *Reconnaissance d'extraits sonores dans une large base de donnees*, Practical lessons, Ircam, 1998.

[18] Xavier Rodet, Laurent Worms, and Geoffroy Peeters, "Method for characterinz a sound signal," July 11 2003, WO Patent 2,003,056,455.

[19] Martin F McKinney and Jeroen Breebaart, "Features for audio and music classification.," in *ISMIR*, 2003, vol. 3, pp. 151–158.

[20] Brian Whitman and Daniel PW Ellis, "Automatic record reviews," in *ISMIR 2004: 5th International Conference on Music Information Retrieval: Proceedings: Universitat Pompeu Fabra, October 10-14, 2004*. Audiovisual Institute, Pompeu Fabra University, 2004, pp. 86–93.

[21] Les Atlas and Shihab A Shamma, "Joint acoustic and modulation frequency," *EURASIP Journal on Applied Signal Processing*, vol. 2003, pp. 668–675, 2003.

[22] Leon Cohen, "The scale representation," *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3275–3292, 1993.

[23] Antonio De Sena and Davide Rocchesso, "A fast mellin and scale transform," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, 2007.

[24] Joakim Andén and Stéphane Mallat, "Multiscale scattering for audio classification.," in *ISMIR*, 2011, pp. 657–662.

[25] Ning Ma, Phil Green, Jon Barker, and André Coy, "Exploiting correlogram structure for robust speech recognition with multiple speech sources," *Speech Communication*, vol. 49, no. 12, pp. 874–891, 2007.

[26] Daniel PW Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[27] Beesuan Ong, Xavier Serra, Sebastian Streich, Nicolas Wack, Pedro Cano, Emilia Gómez, Fabien Gouyon, and Perfecto Herrera, "Ismir 2004 audio description contest," 2006.

QUAD-BASED AUDIO FINGERPRINTING ROBUST TO TIME AND FREQUENCY SCALING

Reinhard Sonnleitner
Department of Computational Perception,
Johannes Kepler University
Linz, Austria
reinhard.sonnleitner@jku.at

Gerhard Widmer
Department of Computational Perception,
Johannes Kepler University
Linz, Austria
gerhard.widmer@jku.at

ABSTRACT

We propose a new audio fingerprinting method that adapts findings from the field of blind astrometry to define simple, efficiently representable characteristic feature combinations called *quads*. Based on these, an audio identification algorithm is described that is robust to large amounts of noise and speed, tempo and pitch-shifting distortions. In addition to reliably identifying audio queries that are modified in this way, it also accurately estimates the scaling factors of the applied time/frequency distortions. We experimentally evaluate the performance of the method for a diverse set of noise, speed and tempo modifications, and identify a number of advantages of the new method over a recently published distortion-invariant audio copy detection algorithm.

1. INTRODUCTION

The typical use of an audio fingerprinting system is to precisely identify a piece of audio from a large collection, given a short query. This is typically done by extracting highly discriminative content features, a "fingerprint", from the collection of audio files as well as the query piece, and subsequently comparing these features. The focus in fingerprinting systems is on being able to discriminate between non-identical pieces of audio, even if they sound very similar to a listener, as in the case of different versions of the same song (e.g., "Album Version" vs. "Radio Version").

There are numerous application scenarios for audio fingerprinting. A well established one is to enable curious users to learn the name of the song they are currently listening to [1]. Other applications, mainly used by industry, include the large-scale tasks of media monitoring and audio copy/plagiarism detection [2].

Depending on the application, audio fingerprinting systems should be robust to different kinds of distortions of query audio material. The minimum requirement would seem to be robustness to various types of lossy audio compression and a certain amount of noise. Application domains such as DJ set track identification, media monitoring, audio copy detection and plagiarism detection, pose additional requirements. There, it is also necessary to recognize audio material that was modified in tempo and/or in pitch, and perhaps also tolerate nonlinear noise encountered when a DJ blends two songs in order to achieve a perceptually smooth transition. In the latter case, the time and frequency scale changes may not even be constant over the duration of a song. All these are significant challenges to automatic audio identification systems.

In this work we propose an efficient audio fingerprinting method that meets the above robustness requirements. It is not only robust to noise and audio quality degradation, but also to large amounts of speed, tempo or frequency scaling.[1] In addition, it can accurately estimate the scaling factors of applied time/frequency distortions. The key technique that makes this possible was found by researchers working on blind astrometry, who use a simple and fast geometric hashing approach to solve a generalization of the "lost in space" problem [3]. The specific task is to determine the pointing, scale and orientation of an astronomical image (a picture of a part of the sky), without any additional information beyond the pixel values of the image. We adapt this method to the needs of audio fingerprinting and based on this, develop an extremely efficient and robust audio identification algorithm. The central components in this are compact hash representations of audio that are invariant to translation and scaling, and thereby overcome the inherent robustness limitations of systems that depend on equal relative distances of reference and query features to find matches, such as the well-known Shazam algorithm [1]. More precisely, our algorithm uses a compact four-dimensional, continuous hash representation of quadruples of points, henceforth referred to as "quads". The quad descriptor [3] has also recently been adopted in the field of computer vision, for the task of accurate alignment of video streams [4, 5].

The system we propose can be used for DJ set monitoring and original track identification, audio copy detection, audio alignment, as well as other tasks that demand robustness to certain levels of noise and scale changes.

The paper is organized as follows. Section 2 discusses relevant related work and, in particular, focuses on a very recently published state-of-the art method [2] that will act as our main reference here. Section 3 gives a brief overview of the main points of our new method, in order to set the context for the precise method description, which comes in two parts: Section 4 describes the process of constructing audio fingerprints – the extraction of special features from audio, how to obtain hash representations from these features, and how to store these in special data structures for efficient retrieval. The actual identification method, i.e. the process of matching query audio with reference data by using the extracted features and their hash representations, is then explained in Section 5. Section 6 systematically evaluates the performance of our

[1] To clarify our terminology: if both scales are changed proportionally, we call this a change in *"speed"*: the song is played faster and at the same time at a higher pitch. This can be achieved by simply changing the rotational speed of the turntable, or by modifying the sampling rate of a digital media player – while keeping the sampling rate of the audio encoding unchanged. Changing the time scale only will be referred to as a *"tempo"* change: here, the audio is sped up or slowed down without observable changes in pitch. Vice versa, if only the frequency scale is modified, this will be called *pitch shifting*.

method in two experiments. The first experiment considers song identification on a database consisting of one thousand full length songs of different musical genres. For the second experiment we extend the database to 20,000 full length songs.

2. RELATED WORK

Numerous fingerprinting algorithms have been described in the literature (e.g., [1, 6, 7, 8, 9]), not all of which are applicable to the tasks described in the introduction. Some algorithms extract global fingerprints for entire songs, and are not suitable for our given tasks because identifying snippets or excerpts of songs in the reference database requires local fingerprints. In [9], a fingerprinting approach is proposed that computes the scale invariant feature transform (SIFT) [10] on the audio spectrograms with logarithmically scaled frequencies. While good results are shown, SIFT is not invariant to scale changes if only one of the two dimensions is scaled. Therefore, the algorithm proposed in [9] is not robust to what we call tempo changes.

A tempo- and speed-invariant audio fingerprinting algorithm has recently been proposed in [11]. By using pitch class histograms as fingerprint features, it tends to compute similar fingerprints for similar content, which is an interesting property on its own. The algorithm has a certain robustness to tempo and speed changes, but its performance degrades considerably for noisy queries. Also, the performance degrades when the query audio is shorter than the specific references, which is why the method is not well suited for the tasks we described above.

A very recent publication proposing a fingerprinting algorithm for audio copy detection, that also meets the demands of robustness against speedup, tempo changes and pitch-shifting of query audio is [2]. The work reports near perfect percentages of correct song association. The described method performs feature extraction on a two-dimensional time-chroma representation of the audio. First a set of candidate feature points is selected, which are then purified by extracting and comparing up to 30 two-dimensional image patches of different width, centered around the candidate feature point. A candidate point is selected as feature point if most of the (up to 30) extracted image patches fulfill a similarity criterion. This is determined via k-means clustering, which assigns the extracted patches to a number of c classes. Similarity is calculated by computing low frequency discrete cosine transformation (DCT) coefficients which represent the actual similarity metric. The proposed method performs feature point selection for an average of 20 candidate points per second of audio, of which approximately 40% pass the similarity constraints and are used for fingerprint computation. The actual fingerprints are generated from a number of low frequency DCT coefficients of the extracted image patches, and are scaled and translated to result in vectors of zero mean and unit variance. According to the explanation given in the work, such a fingerprint should result in a vector of 143 floating point values. Fingerprint matching is done by nearest neighbor lookups, with distance defined as the angle of two fingerprint vectors.

We will take this as our reference method in the present paper, because it is the latest publication on this topic, and it reports extremely high robustness and performance results for a large range of tempo and speed modifications (though based on experiments with a rather small reference database – see Section 6 below).

3. METHOD OVERVIEW

The basic idea of our proposed method is to extract spectral peaks from the two-dimensional time-frequency representation of reference audio material, then group quadruples of peaks into quads, and create a compact translation- and scale-invariant hash for each quad (a single hash is a point in a four-dimensional vector space). Quads and their corresponding hashes are stored in different data structures, i.e. quads are appended to a global index, and an inverse index is created to assign the corresponding audio file-id to its quad indices. The continuous hashes are stored, together with the index of their quad, in a spatial data structure, such that the index that is associated with the hash corresponds to the index of the quad that forms the hash.

For querying we extract quads and their hashes from the query audio excerpt. For each query hash we perform a range search in the spatial data structure and collect the indices of search results, which in turn give the indices of matching reference quads in the global index. The time and frequency scaling factor can be found by comparing a query quad to its matching reference quad. To predict the match file ID for a query snippet, we adapt the histogram method from [1].

4. FEATURE EXTRACTION

In this section we describe the extraction of audio features to be used for audio identification, how to obtain hashable representations from these features, and how to finally store these for efficient retrieval. The same feature extraction process is applied to the *reference recordings* that are used to build the fingerprint database, and the *query audio* that is to be identified in the recognition phase.

To begin with, all audio files are downmixed to one-channel monaural representations and processed with a sampling rate of 16 kHz. We compute the STFT magnitude spectrogram using a Hann-window of size 1024 samples (64 ms) and a hopsize of 128 samples (8 ms), discarding the phases.

4.1. Constructing Quads

The fingerprinting algorithm works on translation- and scale-invariai hashes of combinations of spectral peaks. Spectral peaks are local maxima in an STFT magnitude spectrogram, and identified by their coordinates in the spectrogram. Since the notion of a peak P as a point in 2D spectrogram space will be used extensively in the following, let us formally introduce the notation:

$$P = (P_x, P_y)$$

where P_x is the peak's time position (STFT frame index), and P_y is the peak's frequency (index of STFT frequency bin).

The extraction of spectral peaks is implemented via a pair of two-dimensional filters, a max filter and a min filter, where the neighbourhood size is given by the filter window size. We use a max filter to find the coordinates of spectral peak candidates in the spectrogram, and use a min filter with a smaller window size to reject peaks that were extracted from uniform regions in the spectrogram, e.g., silence. In the following we explain how quads are created from spectral peaks, and how compact hash values are computed from quads.

To create translation- and scale-invariant hashes from spectral peaks, we first have to group peaks into *quads* [3]. A quad consists of four spectral peaks A, B, C, D, where we define A to be the

root point of the quad, which is the peak with the smallest frame index (i.e. the first of the four peaks in time) and B is the most distant point in time from A (thus C, D lie somewhere between the STFT frames of A, B). The quad is *valid* if $B > A$ and C, D lie within the axis-parallel rectangle defined by A, B:

$$A_x < B_x \quad (1)$$

$$A_y < B_y \quad (2)$$

$$A_x < C_x, D_x \leq B_x \quad (3)$$

$$A_y < C_y, D_y \leq B_y \quad (4)$$

At the top level, the quad grouping process proceeds through an audio file from left to right, trying each spectral peak as a potential root point A of a set of quads, with the goal of creating up to a number of q quads for each peak.

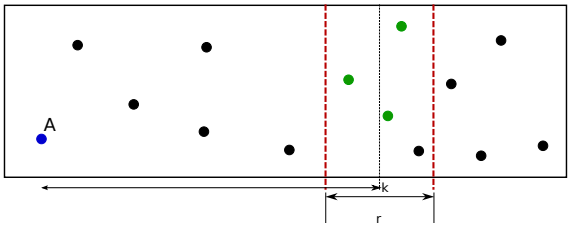
For a given root point A, the process of constructing up to q quads by selecting appropriate sets of B, C, D points is as follows.[2] We construct a region of width r, spanning r STFT frames, such that the region is centered k STFT-frames from A and A is outside of the region (earlier in time, i.e., the region is located to the right of A), as shown in Figure 1. We then sort the peaks that are contained in the region, by time. We let $t = 0$ and pick the first n peaks $p_t, p_{t+1}, \ldots p_{t+n-1}$ in the region and try all $\binom{n}{3}$ combinations of 3 peak indices in order to construct a valid quad with root point A and the points from the current combination. If a valid quad can be constructed we append it to a list of quads and proceed until q quads are created. If no valid quad could be constructed, we increase t by one and try again until there are no more peaks in the region.

The total number of resulting valid quads for a given root point A depends not only on the parameter values, but is fundamentally dependent on the specific layout of spectral peaks, and thus on the signal itself. As already mentioned, for creating the reference database we want to create a small number of quads. We therefore choose a small n and a region of small width r. For queries we create an extensive set of quads by choosing a larger n, $r_{\text{query}} \gg r_{\text{ref}}$, and $q_{\text{query}} \gg q_{\text{ref}}$. k is the same in both cases.

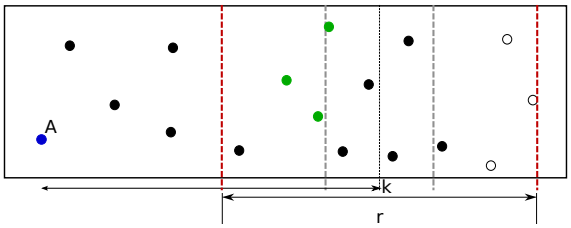
The reason for different parameterization for query quad construction is as follows: When the time scale of a query audio is modified, this affects not only the density of peaks in the given audio snippet, but also their relative positions. An example is given in Figure 1, which shows the grouping for a quad for a given root point A. In 1a a reference quad is created for a region of width r that is centered k frames from A. The analogous example for grouping a query quad for the same audio, but increased in tempo, or decreased in tempo, is given in 1b and 1c, respectively. We see that the green points, which are points B, C, D for the reference quad, may happen to move outside of the grouping region of width r if the time scale of the audio is modified. By choosing a larger region width r and a larger number q of quads that may be created for a root point A, we can ensure to obtain a quad that corresponds to the reference quad.

Note that when we consider audio queries of a fixed, limited duration d (e.g., $15sec$), there is an important difference between

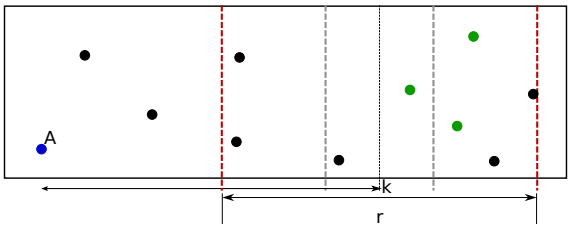
[2] We will parametrize this process differently, depending on whether we compute quads for the reference database, or for a piece of query audio. For reference database creation, we choose parameters in such a way that we only create a small number of reference quads to keep the resulting reference database as small as possible. For a query snippet, we will choose parameters to create a large amount of quads. The explanation for this will be given later in this section.



(a) Reference quad grouping



(b) Query quad grouping. Query audio was increased in tempo. New peaks are shown in white facecolor.



(c) Query quad grouping. Query audio was decreased in tempo.

Figure 1: *Reference quad grouping (1a) and query quad grouping with increased tempo (1b), and decreased tempo (1c).*

increased speed/tempo and decreased speed/tempo. Increasing the tempo of the query audio excerpt relative to the reference leads to a higher density of relevant audio content; all the content that was used during the phase of reference quad creation is also present when creating the quads for the query. However, decreasing the tempo of the query, i.e., stretching the time scale, may cause some of the relevant spectral peaks to fall out of the $15sec$ (i.e., not be part of the query any more), so some important quads do not emerge in the query. This problem arises when tempo or speed are decreased by large amounts. This difference in increasing vs. decreasing the time scale is actually reflected in the evaluation results (see Section 6). To summarize, if the same parameters are used for both reference and query quad grouping, and the time scale changes, it is very likely that no matching quads will be found in subsequent queries.

4.2. From Quads to Translation- and Scale-invariant Hashes

We now have created quads from spectral peaks in audio, but these quads are not the actual summarizing representation that we later use to find match candidates between a query audio and the reference database. That representation should be translation- and scale-invariant, and quickly retrievable. To achieve this, we compute *translation- and scale-invariant hashes* from the quads. For a given quad (A, B, C, D), the constellation of spectral peaks is translated to the origin and normalized to the unit square, resulting

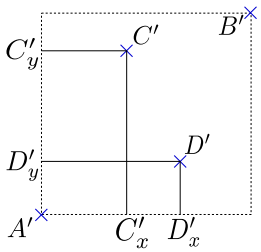


Figure 2: *Example of a quad hash computed from an arbitrary quad* A, B, C, D.

in the four points A', B', C', D' such that $A' = (0, 0)$ and $B' = (1, 1)$, as shown in Figure 2. The actual continuous hash of the quad is now given by C', D', and is stored as a four-dimensional point (C'_x, C'_y, D'_x, D'_y) in a spatial data structure. Essentially, C', D' are the relative distances of C, D to A, B in time and frequency, respectively. Thus, the hash C', D' is not only translation invariant (A' is always $(0, 0)$), but also scale invariant. A feature extraction example showing spectral peaks and resulting quads is shown in Figure 3

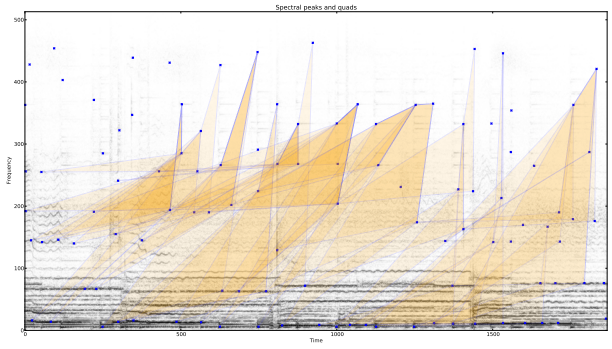


Figure 3: *Extracted spectral peaks and grouped quads on a* 15 *seconds excerpt of "Radiohead - Exit Music (For a Film)".*

4.3. Fingerprints: Storing Hashes for Efficient Recognition

Once peaks, quads and their hashes are computed, we store these in data structures that allow for efficient selection of match candidates and subsequent verification of match hypotheses from query audio. The reference data consist of four data structures:

- quadDB: A file that contains all reference quads (the original, *unnormalized* ones).
- fidindex: An index file that stores file-id, quad index range (into quadDB) and filename for each reference audio file.
- reftree: A spatial data structure containing all reference quad hashes.
- refpeaktrees: Two dimensional search trees for the spectral peaks that are extracted from reference audio files.

The quadDB is a binary file that stores the sequence of quads for all reference files, and the fidindex is an index file which maps each reference file to a unique file-id and also stores the index range (i.e. startindex, number of quads in quadDB) for the sequence of quads that was extracted from the reference files. For the spatial data structure (reftree) we use an R-Tree [12] with an R* split-heuristic that stores all quad hashes, together with their positional index in the quadDB. The R-Tree enables us to add songs the reference database without the need of rebuilding the tree in order to maintain high query performance. In addition, the R-Tree is well suited for large out-of-memory databases.

The refpeaktrees are used for the verification of match candidates, which will be explained later.

4.4. Chosen Parameter Values

The specific set of parameter values that we chose for our implementation and that are used in the evaluation in Section 6, is as follows: The extraction of spectral peaks is performed with a max-filter width of 91 STFT-frames, and a filter height of 65 frequency bins. The min-filter, used to reject maxima that resulted from uniform magnitude regions, has a width of 3 STFT-frames and a height of 3 frequency bins. For reference quad grouping we choose the center of the grouping window k to be four seconds from each root point A. The width r of this region window for reference quad extraction is two seconds. We group $q = 2$ quads for each root point A along with a group size of $n = 5$. This results in an average number of roughly 8.7 quads per second of audio. For query quad extraction we choose the same k of four seconds, and a large grouping window width r that spans 7.9 seconds. A number of up to $q = 500$ quads are extracted from a group size of $n = 8$.

5. RECOGNITION ALGORITHM

The method for identifying the correct reference recording, given a query audio excerpt, consists of several stages: the selection of match candidates, a filtering stage in which we try to discard false positive candidates, and a verification step adapted from the findings in [3]. After the verification stage we efficiently estimate a match sequence with the histogram binning approach that is used in algorithm [1]. In the following the selection of match candidates is explained.

5.1. Match Candidate Selection and Filtering

For each quad hash that was extracted from a query audio, an epsilon search in the reftree is performed. This lookup returns a set of raw match candidate indices: the indices of those quads in the quadDB whose quad-hashes are similar (identical up to epsilon: $B^q_x - \epsilon \leq B^r_x \leq B^q_x + \epsilon$ etc.) to the query quad-hashes. We call this the set of raw candidates, as it will most likely be a mixture of true positives and a (large) number of false positive matches. The raw candidates are used to obtain estimates of the time/frequency scale modification of the query audio, by looking at the different orientation of the original (non-normalized) quads corresponding to the query (q) and reference (r) hash, giving us the scaling factors for time and frequency:

$$s_{time} = (B^q_x - A^q_x)/(B^r_x - A^r_x) \quad (5)$$

$$s_{freq} = (B^q_y - A^q_y)/(B^r_y - A^r_y) \quad (6)$$

It makes sense to parametrize the system with scale tolerance bounds as, depending on the application, one might not be interested in trying to identify audio that is played at, e.g., half the speed or

double the tempo, or has undergone extreme pitch-shifting modifications. Such constrained tolerances considerably speed up the subsequent hypothesis testing by rejecting raw match candidates that lie outside the specified bounds.

Instead of directly starting with hypothesis tests on the raw candidate set we first apply filters in order to clean up the match candidates. This filtering process aims at discarding false positive matches while keeping a large number of true positives. In addition to the previously mentioned scale tolerances we perform a spectral coherence check, similar to the spatio-temporal coherence check described in [5]. Here we reject match candidate quads whose root point A is far away in the frequency domain compared to root point A of the query quad.

We now consult the fidindex to sort the remaining match candidates by file-id, and enter the verification step (Section 5.2) – those candidates that pass the following step are considered true matches and are passed to the match-sequence estimation.

5.2. Match Verification and Sequence Estimation

Match verification is performed once all match candidates for all query quads are collected and filtered as described above. Most likely, the remaining match candidates correspond to a large number of file-ids that are referenced in the database. Since our goal is to identify the correct file-id, we perform this stage of match candidate verification on a per-file-id basis. To do this we consult the fidindex file (cf. Section 4.3) and look up the file-ids for all match candidates, and sort the match candidates by file-id.

Verification is based on the insight that spectral peaks that are nearby a reference quad in the reference audio, should also be present in the query audio [3]. Naturally, depending on the audio compression, the amount of noise or other distortions, there might be a larger or smaller number of nearby peaks in the query. We define the nearby peaks as the set of N peaks closest to the match candidate's root point A (for some fixed N), and retrieve those by performing a k-nearest-neighbor search in the refpeak-trees (cf. Section 4.3) for the given file-id. We define a threshold t_{min}, the minimal number of nearby peaks that have to be present in the query in order to consider the candidate an actual match. Note that in order to find relevant nearby peaks in the query, we have to align the query- and reference-peaks by transforming the query peak locations according to the previously estimated time/frequency scale (cf. Section 5.1). The candidates that pass the verification step are considered true matches, and they are annotated with the number $v \leq N$ of correctly aligned spectral peaks, and the scale transformation estimates. This number v will be used for an optimization described below.

After match candidates for a given file-id are verified, we try to find a sequence of matches for this file-id by processing the matches with a histogram method similar to the one used in the Shazam algorithm [1], with the difference that the query time (the time value of root point A of each query quad in the sequence) is scaled according to the estimated time scale factor. Finally, the file-id for the largest histogram bin (the longest match sequence) is returned, together with the match position that is given by the minimal time value of the points in the histogram bin. We now know the reference file that identifies the query audio, the position of the query audio in the reference track, and the associated time/frequency scale modification estimates. Note that the reported scale transformation estimates are expected to be quite accurate, because with these estimates, for each "surviving" match candidate at least t_{min} nearby spectral query peaks could be correctly aligned to corresponding reference peaks during the verification phase. A lookup in the fidindex now gives us the filename of the reference audio as well as any kind of optional meta data.

To speed up the verification process, we define a threshold for the number of correctly aligned nearby peaks $t_{\text{v}} > t_{\text{min}}$. When the v value of a match reaches or exceeds this threshold, we allow a so-called "early exit" for this file-id. Once all match candidates of an early exit file-id are verified, we directly enter the match sequence estimation for this file-id, without subsequent verification of any other file-id.

5.3. Runtime and Data Size Considerations

Our system operates on a number of data structures (cf. Section 4.3) that together constitute what we call the *reference database*; the largest components are the reftree and the refpeak trees.

The quadDB linearly stores binary records of quads. A quad consists of four two-dimensional discrete points (coordinates in the STFT spectrogram) and can be represented and stored as $8 * 32$bit integers, which amounts to 32 byte per quad. It is not necessary to keep this file in-memory, as the proposed method is designed to operate on big out-of-memory reference data.

There exists exactly one quad hash per quad. A quad hash is a continuous four-dimensional point that is stored as an array of four float32 values by the reftree. The actual number of quads in the quadDB depends on the filter size parameters of the spectral peak extraction and the quad grouping parameters. For an example reference database consisting of 20,000 full length songs we choose the parameters such that we create an average of approximately 8.74 quads per second of audio, with a median of ≈ 8.68 and a standard deviation of $\sigma \approx 1.19$. The histogram of the number of quads per second is shown in Figure 4. This specific database consists of $\approx 4.29 * 10^7$ quads, or roughly 1.3GB. The reftree, a four-dimensional R-Tree, consumes approximately 4.8GB. To speed up the verification process we also store two dimensional trees of spectral peaks for each file-id, which consume roughly 3GB for $20,000$ songs. We currently store the fidindex file as text, along with some meta data. In this example the size of the fidindex amounts to 2.3MB.

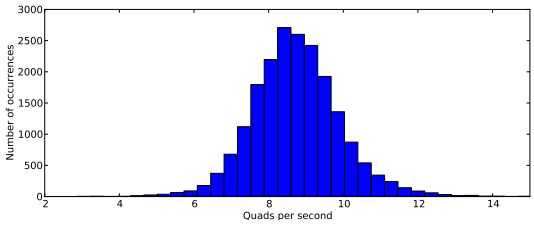


Figure 4: *Histogram of the average number of quads per second for all files in a reference database of 20,000 songs.*

Naturally, depending on application scenarios and hardware constraints, it is possible to trade runtime for storage space and vice versa. If minimal space consumption is of priority, one can pack the binary quad records of the quadDB to 16 bytes by exploiting the limited number of STFT frequency bins (i.e. 512), and storing the time values of points B, C, D as offsets from point A. This saves 50% per quad, reducing the size of the quadDB file to ≈ 650MB.

Regarding the runtime, using our unoptimized pure Python implementation of the method, feature extraction and quad creation for the database of $20,000$ songs took approximately $24h$ utilizing seven out of eight logical cores of an Intel Core i7 860 (2.8GHz) Processor.

The runtime for a query is made up of audio decoding, feature extraction, querying the database and filtering the results, match candidate verification and match sequence binning. For a query on a 15 seconds audio excerpt against a reference database of 1000 songs, this takes approximately 4 to 12 seconds. Here, half of the time is taken by the preprocessing (decoding, quad extraction).

Querying a larger database of $20,000$ songs takes considerably (though, of course, not proportionally) longer. The main reason is the higher number of match candidates that have to be processed. The same query excerpt as used above is processed in approximately 14 to 60 seconds. Here, at least half of the time is consumed by the reftree range queries. Again, this is based on an unoptimized, experimental Python implementation; there is ample room for improvement. Section 6.3 gives more detail.

6. EVALUATION

We systematically evaluate the performance of the system for different speed, tempo and noise modifications of 15 seconds query audio snippets. The reference database for the first experiment is constructed from $N = 1000$ full length songs in .mp3 format. To create test queries, we randomly choose 100 reference songs and subject these to different speed, tempo, and noise level modifications. We then randomly select a starting position for each selected song, and cut out 15 seconds from the audio, such that we end up with 100 $15sec$ audio queries. The evaluation considers speed and tempo ranges from 70% to 130% in steps of 5%. To evaluate the noise robustness of our system we mix each query snippet with white noise to create noisy audio in SNR level ranges from 0 dB to $+50$ dB in steps of 5 dB. Furthermore, we create all query audio snippets from .mp3 encoded data, and encode the modified snippets in the Ogg Vorbis format [13], using the default compression rate (cr $= 3$). We do this to show that the system is also robust to effects that results from a different lossy audio encoding. All modifications are realized with the free SoX audio toolkit [14]. The following terms are used in defining our performance measures: tp (*true positives*) is the number of cases in which the correct reference is identified from the query. fp (*false positives*) is the number of cases in which the system predicts the wrong reference. fn (*false negatives*) is the number of cases in which the system fails to return a reference id at all.[3]

We define two performance measures: *Recognition Accuracy* is the proportion of queries whose reference is correctly identified:

$$\text{Accuracy} = \frac{tp}{tp + fp + fn} = \frac{tp}{N} \tag{7}$$

Precision is the proportion of cases, out of all cases where the system claimed to have identified the reference, where its prediction is correct:

$$\text{Precision} = \frac{tp}{tp + fp} \tag{8}$$

Thus, high precision means low number of false positives.

[3]There are no *true negatives* (tn) in our scenario (i.e., cases where the system correctly abstains from identifying a reference because there *is* no correct reference) because for all queries, the matching reference is guaranteed to be in the DB.

	db 1000 songs				db 20,000 songs			
Speed	tot.	tree	feat.	match	tot.	tree	feat.	match
130%	12	1	6	5	60	31	7	22
110%	11	2	6	3	52	28	6	18
100%	9	1	6	2	35	20	6	9
90%	9	1	5	2	37	22	5	10
70%	4	0	3	1	14	7	3	1

Table 1: Query runtimes in seconds. "tot" is the total time, "tree" is the time taken by tree intersection, "feat." is the feature extraction time for spectral peaks and quad grouping and "match" is the matching and verification time.

6.1. Detailed Results on Small Reference-DB ($1,000$ Songs)

Each data point in the visualisation shows one of the aforementioned quality measures for 100 queries. The overall system performance for speed, tempo, and SNR changes is shown in Figure 5. For this experiment a total of 5900 queries of length 15 seconds were run against the database consisting of thousand songs.

Figure 6 shows the performance for the tested SNR levels for speed and tempo modifications of 95% and 105%.

Concerning the noise robustness of the proposed method, the results show that a stable performance of $> 95\%$ for the tested quality measures is achieved for SNR levels down to $+15$dB. According to these results the proposed quad-based hashes seem to be sufficiently robust for queries of various noise levels that may be encountered in real application scenarios.

6.2. Extending the Reference-DB to $20,000$ Songs

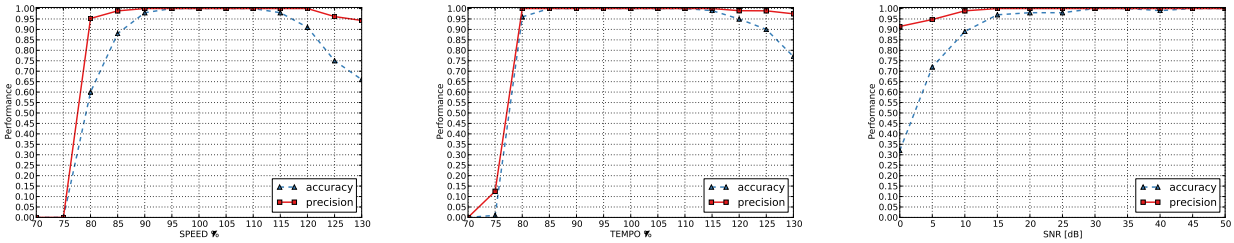
In the previous experiment on a database of thousand songs we reach a very high precision. To further investigate the precision of our proposed algorithm we extend the reference database to 20,000 songs, and query the same audio excerpts that we created for the previous experiment, with the same modifications, against this large database. Figure 7 shows that the performance of our approach does not degrade even if there are many songs in the reference. Note that we parametrized our system to discard any match candidates if their transformation estimates are outside the scale tolerance bounds of $\pm 32\%$ for either frequency and time scale. The performance is comparable to that of the first experiment, resulting in more false positives only for the larger speed modifications. For tempo modifications the system gives basically the same performance as in the first experiment.

6.3. Runtimes

In Table 1 we give information about the runtimes observed in the two above experiments. We randomly pick one of the generated audio query excerpts, and compare the query runtime for the small and the large databases for different scale modifications. The increased runtime for faster speed and tempo is a result of the higher number of spectral peaks in the audio excerpt, for which a larger number of tree-intersections and raw match candidates have to be processed.

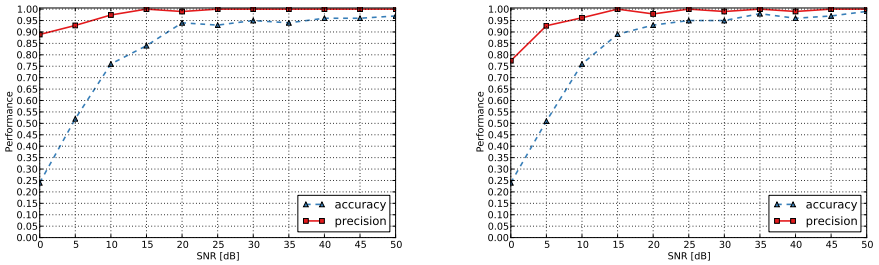
6.4. Comparison with Reference Method [2]

While it is not possible to directly compare our results to those of [2] (because we do not have access to their test data), from the published figures it seems fair to say that in terms of recognition



(a) Speed variations (b) Tempo variations (c) SNR variations

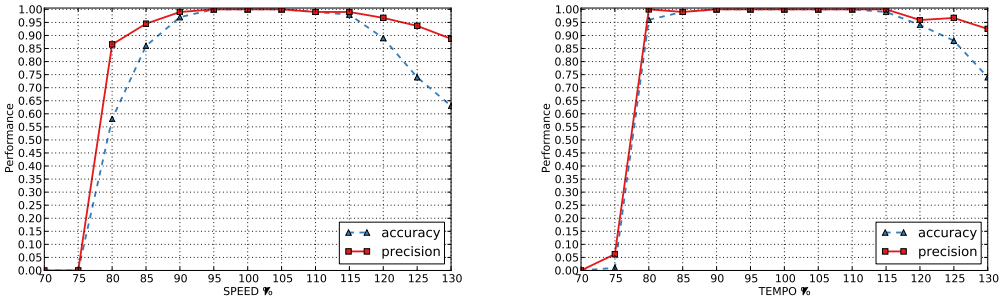
Figure 5: *Precision and accuracy for speed (5a), tempo (5b) and SNR (5c) modifications, on a database of 1000 songs.*



(a) SNR variations for speed 95% (b) SNR variations for speed 105%

(c) SNR variations for tempo 95% (d) SNR variations for tempo 105%

Figure 6: *SNR variations for* 95% *and* 105% *speed and tempo on a database of 1000 songs.*



(a) Speed variations (b) Tempo variations

Figure 7: *Precision and accuracy for speed (7a) and tempo modifications (7b) on a database of 20,000 songs.*

accuracy and robustness, both approaches seem comparable, and both are at the upper end of what can be expected of an audio identification algorithm – for tempo and pitch distortion ranges that are larger than what we expect to encounter in real applications. It should be noted that the results reported in [2] are based on a small reference collection of approximately 250 songs only and that, unfortunately, no information is given about either the size of the resulting database, or about runtimes.

The advantage of our proposed method is its efficiency in terms of data size and fingerprint computation. In contrast to [2], where log-spaced filter banks have to be applied to the spectrogram in order to compute the time-chroma representation, the selection of spectral peaks for quad grouping is done directly on the STFT magnitude spectrogram of the audio, and the quads can be grouped in linear time. Our proposed method chooses spectral peaks that are local maxima of the magnitudes in the spectrogram, in contrast to the method of [2], where 600 DCT computations per second of audio have to be performed (similarity computations for 30 rectangular image patches for each of approximately 20 feature candidates per second) in order to find stable feature points. The hash representation we propose is very compact and can be stored as four float32 values, while the algorithm of [2] uses fingerprints that are represented by 143-dimensional vectors.

Our match candidates are retrieved via an efficient range search in a spatial data structure, for which we use an R-Tree. The distance of hashes is the euclidean distance between four-dimensional points, while the distance measure used in [2] is the measure of the angles of the 143-dimensional fingerprint vectors.

7. CONCLUSIONS

We have presented a new audio identification method that is highly robust to noise, tempo and pitch distortions, and verified its ability to achieve overall high performance on a medium-large database consisting of 20,000 songs. While there is a lot of potential for false positive matches in a database of this size (roughly 43 million quads) in combination with the rather large tolerated scaling ranges, the method's filtering stage and the subsequent verification process enable the system to maintain high precision and accuracy. The results show a stable high performance for a large range of scale changes, with as few as ≈ 9 compact fingerprints per second of audio.

In preliminary experiments on DJ sets and media broadcast data we have not yet found any examples that exceeded 7% of speed or tempo scale modifications. We also assume that scale modification attacks against audio copy detection algorithms are usually done with very subtle scale changes, almost imperceptible to human ears. Our proposed algorithm achieves near perfect overall performance within scale modification ranges of 90% to 115% for speed, and 80% to 120% for tempo scale modifications.

8. ACKNOWLEDGMENTS

We would like to thank Rainer Kelz, Jan Schlüter and Harald Frostel for many intense and fruitful discussions. This research is supported by the Austrian Science Fund FWF under projects TRP307 and Z159.

9. REFERENCES

[1] A. Wang, "An industrial-strength audio search algorithm," in *Proc. Intl. Conf. on Music Information Retrieval*, 2003.

[2] M. Malekesmaeili and R. Ward, "A local fingerprinting approach for audio copy detection," *Signal Processing*, vol. 98, pp. 308–321, 2014.

[3] D. Lang, D. Hogg, K. Mierle, M. Blanton, and S. Roweis, "Astrometry.net: Blind astrometric calibration of arbitrary astronomical images," *The Astronomical Journal*, vol. 137, pp. 1782–2800, 2010.

[4] G. Evangelidis and C. Bauckhage, "Efficient and robust alignment of unsynchronized video sequences.," in *DAGM-Symposium*. 2011, vol. 6835 of *Lecture Notes in Computer Science*, pp. 286–295, Springer.

[5] G. Evangelidis and C. Bauckhage, "Efficient subframe video alignment using short descriptors.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 10, pp. 2371–2386, 2013.

[6] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. Intl. Conf. on Music Information Retrieval*, 2002.

[7] F. Kurth, T. Gehrmann, and M. Müller, "The cyclic beat spectrum: Tempo-related audio features for time-scale," in *Proc. Intl. Conf. on Music Information Retrieval*, Victoria (BC), Canada, October 8-12 2006.

[8] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recogn.*, vol. 41, no. 11, pp. 3467–3480, Nov. 2008.

[9] B. Zhu, W. Li, Z. Wang, and X. Xue, "A novel audio fingerprinting method robust to time scale modification and pitch shifting," in *Proc. Intl. Conf. on Multimedia*, New York, NY, USA, 2010, pp. 987–990, ACM.

[10] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[11] J. Six and O. Cornelis, "A robust audio fingerprinter based on pitch class histograms: applications for ethnic music archives," in *Proc. Intl. Workshop of Folk Music Analysis*. 2012, p. 7, Ghent University, Department of Art, music and theatre sciences.

[12] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *SIGMOD Rec.*, vol. 14, no. 2, pp. 47–57, June 1984.

[13] "Ogg Vorbis," Available at http://www.vorbis.com.

[14] "SoX - Sound eXchange," Available at http://sox.sourceforge.net/.

SCORE-INFORMED TRACKING AND CONTEXTUAL ANALYSIS OF FUNDAMENTAL FREQUENCY CONTOURS IN TRUMPET AND SAXOPHONE JAZZ SOLOS

Jakob Abeßer
Fraunhofer IDMT
Ilmenau, Germany
Liszt School of Music
Weimar, Germany
jakob.abesser@idmt.fhg.de

Martin Pfleiderer
Liszt School of Music
Weimar, Germany

Klaus Frieler
Liszt School of Music
Weimar, Germany

Wolf-Georg Zaddach
Liszt School of Music
Weimar, Germany

ABSTRACT

In this paper, we propose a novel algorithm for score-informed tracking of the fundamental frequency over the duration of single tones. The tracking algorithm is based on a peak-picking algorithm over spectral magnitudes and ensures time-continuous f_0-curves. From a set of 19 jazz solos from three saxophone and three trumpet players, we collected a set of 6785 f_0-contours in total. We report the results of two exploratory analyses. First, we compared typical contour feature values among different jazz musicians and different instruments. Second, we analyzed correlations between contour features and contextual parameters that describe the metrical position, the in-phrase position, and additional properties of each tone in a solo.

1. INTRODUCTION

The personal style of a musician or singer encompasses various features of her or his performances such as micro-timing, intonation (i.e., pitch accuracy according to a certain tone system), glidings at beginnings and endings or between successive tones and several features of sound, e.g. breathiness or roughness of tones or their overall timbre [1, 2]. However, for the task of Automatic Music Transcription (AMT), tones are commonly understood as acoustic events with a fixed pitch, onset, and offset time [3][1]. This symbolic music representation can be beneficial for a score-level analysis of musical properties such as interval distributions, chords, and scales. At the same time, further artist-specific aspects of a music performance such as pitch glides, intonation, or timbre are completely neglected. Some authors analyze pitch contours as part of automatic melody transcription systems. For instance, Salamon et al. extract different statistical features from pitch contours in polyphonic music, which are used as criteria to assign them to the main melody [4].

When observing jazz improvisation performances of trumpet and saxophone players, the fundamental frequency rarely remains constant over the full duration of a tone. Instead, frequency modulation techniques such as pitch bends, glissandi between tones, vibratos of varying speed and range, and other ornamentations are used to give individual expressiveness to the tones and melodic lines [5]. In African American music genres like jazz, especially thirds, fifths, and sevenths are played in a peculiar way—sometimes a bit too low, sometimes with a gliding movement of the pitch. This phenomenon is often referred to as blue notes or blue note areas by ethnomusicologists [6, 7]. Moreover, jazz musicians often play with vibrato and shape their vibrato in different ways, e.g. faster or slower, or with different amounts of pitch deviations. Depending on jazz style and artist, longer tones are played without vibrato at the beginning and then, often starting on a strong metrical position, are enriched by adding vibrato [8].

In this paper, we primarily focus on the intonation of tones in improvisation of jazz musicians, which could be a pivotal feature of their personal "sound" and playing style. Therefore, we neglect other perceptual aspects related to the instrumental timbre or micro-timing and solely focus on how the fundamental frequency evolves over the duration of a melody tone. In particular, we analyzed audio recordings from six well-known trumpet and saxophone players. We initially extracted tone-wise f_0-contours based on manual transcriptions of instrument solos and then performed two different analyses: Firstly, we investigated whether significant differences can be found for contour feature values among different artists and different instruments. Secondly, we analyzed whether and how contour features such as the deviation of the fundamental frequency from the annotated pitch depend on contextual properties such as the tone's pitch, duration, and metrical position.

This paper is structured as follows. The selection of commercial jazz recordings used for this publications will be described in Section 2.1 In section 2, the proposed algorithm for score-informed f_0-tracking as will be explained in detail. Section 2.7 will give a brief description of the features we extract for each f_0-contour. In Section 3, the differences between artist / instruments and the bias of contextual parameters will be explored. Finally, some conclusions for elaborated transcription strategies and jazz research will be drawn in Section 4.

[1] Throughout this publication, we use the terms *note* for annotated pitch values and *tone* for all sound events that were performed / played on a musical instrument.

2. NOVEL APPROACH

Figure 1 illustrates the proposed method for score-informed estimation of fundamental frequency contours from jazz solos.

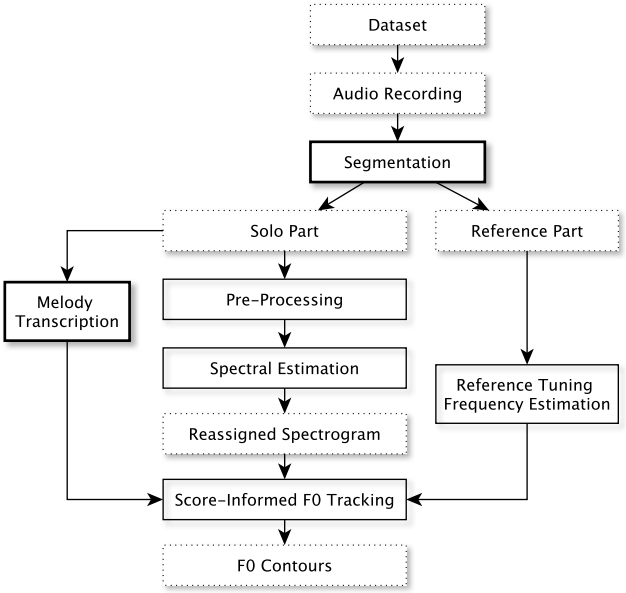


Figure 1: Flow-chart of the proposed algorithm for score-informed f_0-tracking. Steps illustrated with thicker outline are performed manually, all other processing steps are performed automatically.

2.1. Dataset & Melody Transcription

In this publications, we analyze 19 solos played by three saxophone and three trumpet players as listed in Table 1. These solos were taken from the *Weimar Jazz Database* (WJazzD) [9], which currently comprises 174 fully transcribed jazz solos. Based on the original audio recordings, the solos were manually transcribed and cross-checked by musicology and jazz students at the Liszt School of Music. The transcriptions include the MIDI pitch as well as the onset and offset time for each tone played by the soloist. Furthermore, each solo was segmented into melodic phrases, which often coincide with breathing cycles of the saxophone and trumpet players during their improvisation.

2.2. Segmentation

Given a jazz recording, we first manually extract two segments of interest. The *solo part* contains the transcribed instrumental solo and the *reference part* has only the accompanying rhythm section, i.e., piano, double bass, and drums, but no soloist playing. The last two columns of Table 1 show the durations of the solo parts and the reference parts for the used data set.

2.3. Reference Tuning Frequency Estimation

Jazz recordings often show tuning deviations, for instance due to speed variations of tape recorders in the recording process or the particular tuning of the piano used. Amongst others, we aim to analyze the deviation between the note intonation of the soloist and the note intonation of the accompanying rhythm section. Therefore, we first perfom a tuning estimation over the reference part discussed in the previous section to obtain a *reference tuning frequency* f_{ref}.

In particular, we follow the approach implemented in the Chroma Toolbox [10]: Based on a given tuning hypothesis (fundamental frequency of the pitch A4), a triangular filterbank is constructed in such way that its center frequencies are aligned to the semitone fundamental frequencies within the full pitch range of the piano. The STFT magnitude spectrogram is computed, averaged over the duration of the reference part, and filtered using the filterbank to get a measure-of-fit for the current tuning hypothesis. In contrast to the original implementation, we search for the tuning frequency around 440 Hz with a margin of $\pm\frac{1}{2}$ semitone (MIDI pitch range: 69 ± 0.5) and use a very small stepsize of 0.1 cents for the grid search.

2.4. Pre-processing

After the reference tuning frequency f_{ref} is estimated, the next step is to estimate the f_0-contours for each tone played in a given solo. In order to reduce the computation time, we apply a downsampling by factor 2 to a sampling rate of $f_{\text{s}} = 22.05\,\text{kHz}$, since all fudamental frequency values that can be played on the saxophone and the trumpet are below the Nyquist frequency of $f_{\text{s}}/2$. For each tone, the corresponding audio signal is extracted between the tone's onset time and offset time.

2.5. Spectral Estimation

In order to track the fundamental frequency contour over time, we compute a *reassigned magnitude spectrogram* M_{IF} based on the *instantaneous frequency* (IF) as follows. The instantaneous frequency $\hat{f}(k,n)$ for each time-frequency bin in the STFT spectrogram $X(k,n)$ is estimated using the method proposed by Abe in [11]. The approach uses the time derivative of the phase for a frequency correction. We use a zero-padding factor of 16, a blocksize of $b = 2048$, and a hopsize of $h = 64$. The magnitude spectrogram is computed as $M(k,n) = |X(k,n)|$.

We define a logarithmically-spaced frequency axis $f_{\text{log}}(k_{\text{log}})$ with a resolution of 50 bins per semitone. This axis is aligned to the reference tuning frequency f_{ref} and defined for each target tone with a pitch tolerance band of $\pm$ 2 semitones around its annotated MIDI pitch value P as

$$f_{\text{log}}(k_{\text{log}}) = f_{\text{ref}} \cdot 2^{\frac{P-69-2+k_{\text{log}}/50}{12}} \tag{1}$$

The MIDI pitch value of 69 refers to the pitch A4, which corresponds to f_{ref}. The frequency index is denoted as k_{log} with $0 \leq k_{\text{log}} \leq 201$.

The spectral magnitude reassignmend is performed as follows. In each time frame n, each magnitude value $M(k,n)$ (of the STFT magnitude spectrogram) is mapped to the frequency bin $\tilde{k}_{\text{log}}$ of the reassigned spectrogram M_{IF}, whose frequency $f_{\text{log}}(\tilde{k}_{\text{log}})$ is closest to the instantaneous frequency value $\hat{f}(k,n)$. Hence, the original magnitude values of $M(k,n)$ are accumulated in $M_{\text{IF}}(k_{\text{log}},n)$ as follows:

$$M_{\text{IF}}(\tilde{k}_{\text{log}},n) = \sum_k \sum_n \delta(k,n) \cdot M(k,n) \tag{2}$$

Table 1: Selection of saxophone and trumpet solos taken from *Weimar Jazz Database* (WJazzD) that is analyzed in this paper. The columns show the solo number, the artist name, the song title, the solo instrument, the number of notes per solo, the estimated tuning frequency f_{ref} from the reference part (see Section 2.3), the deviation of f_{ref} from the tuning frequency from 440 Hz in cent, as well as the duration of the reference part D_{ref} and the duration of the solo part D_{solo}. The total number of notes per artist is given in brackets after the artist name. The last row shows the mean (μ) and standard deviation (σ) values over all solos. Additional solo metadata can be found at [9].

Solo #	Artist	Title	Instrument	Notes	f_{ref} [Hz]	Δf_{ref} [cent]	D_{ref} [s]	D_{solo} [s]
1	Coleman Hawkins (1195)	Body And Soul	Saxophone	636	445.45	21.3	9.11	167.44
2		My Blue Heaven	Saxophone	213	447.28	28.4	10.58	58.1
3		Stompin' At The Savoy	Saxophone	346	438.71	-5.1	33.86	61.99
4	Michael Brecker (1271)	Midnight Voyage	Saxophone	589	441.94	7.6	33.27	153.86
5		Nothing Personal	Saxophone	682	440.84	3.3	23.82	118.59
6	Sonny Rollins (999)	Blue Seven - 1	Saxophone	354	442.5	9.8	21.51	109.71
7		Blue Seven - 2	Saxophone	138	442.5	9.8	21.51	38.41
8		Tenor Madness	Saxophone	507	438.73	-5	31.06	130.63
9	Clifford Brown (1085)	George's Dilemma	Trumpet	429	440.08	0.3	46.5	100.7
10		Joy Spring	Trumpet	455	441.35	5.3	43.38	94.73
11		Sandu	Trumpet	201	439.59	-1.6	50.68	44.79
12	Freddie Hubbard (1043)	245	Trumpet	481	435.78	-16.7	37.75	120.03
13		Down Under	Trumpet	114	440.28	1.1	25.37	38.56
14		Society Red	Trumpet	448	439.59	-1.6	36.25	149.83
15	Miles Davis (1192)	Blues By Five	Trumpet	371	443.24	12.7	39.49	130.62
16		Oleo - 1	Trumpet	223	442.06	8.1	30.92	56.36
17		Oleo - 2	Trumpet	224	442.06	8.1	30.92	55.26
18		So What	Trumpet	221	452.08	46.9	13.78	112.11
19		Vierd Blues	Trumpet	153	436.81	-12.6	28.62	100.17
μ (σ)				366.95 (213.91)	440.9 (3.24)	3.49 (12.63)	30.98 (9.57)	96.65 (42.38)

with

$$\delta(k,n) = \begin{cases} 1, & \text{if } \tilde{k}_{\log} = \arg\min_{k_{\log}} \left| f_{\log}(k_{\log}) - \hat{f}(k,n) \right| \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

2.6. Score-Informed f_0-tracking

After computing the reassigned spectrogram $M_{\text{IF}}(k_{\log}, n)$, the f_0-contour of the target tone is tracked over its complete duration. As an example, Figure 2 illustrates a tone taken from the solo "Stompin' At The Savoy" by the saxophonist Coleman Hawkins. The transcibed pitch value is $P = 65$. In the following sections, it will be detailed, how the starting location (indicated as blue circle) is derived and how the f_0-contour (indicated as red circles) is tracked.

2.6.1. Starting Location

Before the f_0-contour can be tracked, a suitable starting location $(k_{\log,\text{start}}, n_{\text{start}})$ must be identified. Therefore, we first retrieve the frequency bin positions $k_{\log,\max}(n)$ of the frame-wise magnitude maxima as:

$$k_{\log,\max}(n) = \arg\max_{k_{\log}} M_{\text{IF}}(k_{\log}, n) \tag{4}$$

Then, we aim to find the frame n_{start}, in which the magnitude peak is closest to the frequency bin $k_{\log} = 100$, which corresponds to the transcribed pitch of the given tone. Therefore, we compute the starting frame for the tracking as

$$n_{\text{start}} = \arg\min_{n} |k_{\log,\max}(n) - 100| \tag{5}$$

and set the starting frequency bin to $k_{\log,\text{start}} = k_{\log,\max}(n_{\text{start}})$. In case multiple frames show a minimum peak distance to the f_0 bin, we select the frame with the highest magnitude in M_{IF}.

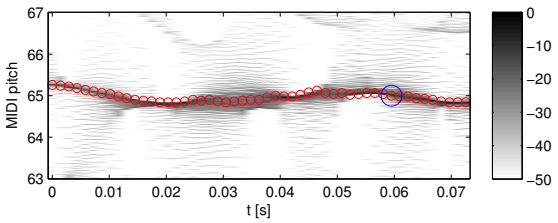


Figure 2: Example f_0-contour of a tone taken from the solo "Stompin' At The Savoy" by the saxophonist Coleman Hawkins with an annotated pitch of $P = 65$. The time axis is normalized such that $t = 0$ refers to the tone onset. The reassigned magnitude spectrogram M_{IF} is shown in dB in the background. The tracked f_0-contour is shown as red circles, the starting location for the forwards-backwards tracking is shown as the bigger blue circle at $t \approx 0.06$ s.

2.6.2. Contour Tracking

After finding the starting location $(k_{\log,\text{start}}, n_{\text{start}})$, the f_0-contour is tracked on a frame-wise basis forwards and backwards in time. We assume that the f_0-contours are continuous, hence we only allow a maximum absolute frequency deviation between the fundamental frequency values in adjacent frames of 10 bins, which corresponds to 20 cent for the given frequency axis. In each frame, we choose the f_0 frequency bin based on the maximum peak position in the search range around the previous f_0 estimate. For the backwards tracking, we obtain

$$k_{\log,0}(n) = \text{argmax}_{k_{\log}} M_{\text{IF}}(k_{\log}, n) \tag{6}$$
$$\text{for } k_{\log,0}(n+1) - 10 \leq k_{\log} \leq k_{\log,0}(n+1) + 10$$

The forward tracking is performed in a similar fashion. Hence, the estimated fundamental frequency is $\hat{f}_0(n) = f_{\log}\left(k_{\log,0}(n)\right)$.

2.7. Feature Extraction

This section details a set of *contour features*, which are computed to characterize each estimated f_0-contour. We measure the local deviation between the estimated fundamental frequency $\hat{f}_0(n)$ and the annotated fundamental frequency $f_0 = f_{\text{ref}} \cdot 2^{\frac{P-69}{12}}$ in cents as

$$\Delta f_0(n) = 1200 \log_2 \left(\hat{f}_0(n) / f_0 \right). \quad (7)$$

$\Delta f_0(n)$ provides a pitch-independent measure of frequency deviation, which is easy to interpret (100 cents correspond to one semitone). We extract the features

- **AvF0Dev**—median over the frequency deviation $\Delta f_0(n)$, which can indicate a sharp or flat intonation,
- **AvAbsF0Dev**—median over the absolute frequency deviation $|\Delta f_0(n)|$, which measures the total deviation from the reference pitch values,
- **LinF0Slope**—approximated (linear) slope of the f_0-contour over the duration of a tone (based on linear regression over $f_0(n)$),
- **F0Progression**—overall pitch progression in cent from the first to the last 5 % of the tone's total duration [12], and
- three features that can characterize a vibrato by measuring the modulation frequency (**ModFreq**), the total modulation range in cent (**ModRange**), as well as the number of modulation periods (**ModNumPeriod**) [12].

The most prominent modulation frequency detected as from the position of the highest peak of the FFT magnitude spectrogram over $f_0(n)$ in the range between 0.3 and 10 Hz. However, this approach will result in a modulation frequency value for all tones but doesn't necessarily imply a vibrato articulation. As will be discussed in Section 4, future work must adress an initial filtering of tones played with vibrato before the modulation frequency values are further interpreted.

2.8. Contextual Parameters

We want to closer investigate the hypothesis that the intonation of each tone depends on its position in the solo, its position in the current melodic phrase, as well as on its metrical position. Therefore, for each tone in a solo, we extract several *contextual parameters* based on the ground truth transcriptions (see Section 2.1).

While onset time (**Onset**) indicates the position of a tone in the solo in absolute time (seconds), we obtain two features of relative tone position from the melodic phrase annotations:

- **PhraseNum**—number of the corresponding melodic phrase and
- **RelPosInPhrase**—relative position of a tone within that phrase (the relative phrase position is a normalized value with 0 indicating the first tone and 1 indicating the last tone of a melodic phrase).

Besides duration (**Duration**) in seconds, we use two features to indicate the position according to the meter:

- **BeatNum**—corresponding beat number within a bar and
- **SubBeatNum**—corresponding sub-beat number (relates to the tatum, i.e., the metrical subdivision that coincides with most of the tone onsets).

Finally, pitch (**Pitch**) refers to the overall ambitus.

3. STATISTICAL ANALYSIS

In this section, we describe several exploratory analyses that we performed to reveal characteristic correlations and relationships within the data set.

3.1. Feature dependency of artist and instrument

In the first analysis, we investigated the distribution of feature values among solos played by different artists and solos played with different instruments. In particular, we focused on the features **AvF0Dev**, **ModRange**, and **F0Progression**. Figure 3 shows the boxplots over these four features for varying artists and instruments. Several observations can be made:

- Particularly Sonny Rollins and Clifford Brown show a tendency to a sharp intonation with median **AvF0Dev** values of 6.4 and 14.0 cent while Miles Davis tends to a flat intonation (-6.0 cent).
- No strong difference of the **AvF0Dev** feature values can be observed when averaged over all trumpet and saxophone players. This leads to the assumption that the tendency towards a sharp or flat intonation is not instrument-specific but rather artist-specific.
- It can be seen that the saxophone players, especially Coleman Hawkins (median **ModRange** value of 40.8 cent) show a higher modulation range than the trumpet players.
- Concerning the **F0Progression** feature, the results indicate that all of the investigated jazz musicians but Freddy Hubbard show a tendency towards upwards pitch glidings (positive **F0Progression** values). The difference between saxophone and trumpet solos is rather small (5.2 vs. 1.5)

3.2. Correlation between Contour Features and Contextual Parameters

In the second analysis, we investigated the correlations between contour features and contextual parameters. An initial Lilliefors test showed that none of the contour features nor the contextual parameters showed a normal distribution. Therefore, throughout the analyses discussed in this section, we used the Kendall τ rank correlation coefficient. The correlation results between pairs of features and contextual parameters are shown in Table 2 (moderate effect sizes of $|\tau| \geq 0.3$ are emphasized using bold print). The following observations can be made:

- While many highly significant correlations between features and contextual parameters exist (due to the large number of tones), most of them have only a small effect size.
- Neither the tuning deviation **AvF0Dev**, the absolute tuning deviation **AvAbsF0Dev**, the slope of the f_0-contour **LinF0Slope**, nor the **F0Progression** feature seem to depend on investigated contextual parameters.
- With increasing tone duration, the modulation frequency decreases ($\tau = -0.62$) while the modulation range and the number of periods increase ($\tau = 0.23$ and $\tau = 0.32$). Apparently, longer notes are played with slower but more extreme vibrato than shorter notes.
- Also, the modulation range in cent decreases with increasing pitch ($\tau = -0.28$), which is most likely caused by playing difficulties in higher pitch registers. As shown in

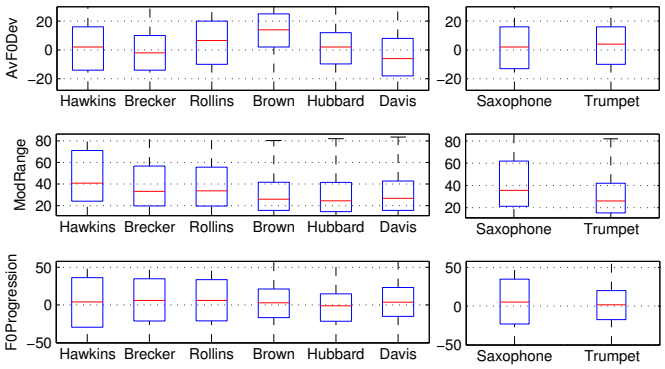


Figure 3: Boxplots over features **AvF0Dev**, **ModRange**, and **F0Progression** over different artists and different instruments.

Table 2: Kendall's τ between features and contextual parameters. Moderate correlation levels ($|\tau| \geq 0.3$) are indicated in bold print. Only significant correlations are shown ($p < .05$). The different significance levels based on the p-value are indicated as *** ($p < .001$), ** ($p < .01$), and * ($p < .05$).

Features	**Contextual Parameters**						
	Metrical Position		**In-phrase Position**		**Basic Tone Parameters**		
	BeatNum	**SubBeatNum**	**PhraseNum**	**RelPosInPhrase**	**Pitch**	**Onset**	**Duration**
AvF0Dev	0.02^{*}	0.02^{*}			-0.07^{***}		-0.04^{***}
AvAbsF0Dev	-0.02^{*}	0.04^{***}		-0.02^{**}	-0.11^{***}		-0.13^{***}
LinF0Slope		-0.03^{**}		-0.03^{***}			-0.02^{*}
ModFreq		0.15^{***}	-0.09^{***}	-0.05^{***}	-0.05^{***}	-0.06^{***}	$\mathbf{-0.62}^{***}$
ModRange		-0.05^{***}	0.05^{***}	0.06^{***}	-0.28^{***}	0.05^{***}	0.23^{***}
ModNumPeriod		-0.09^{***}		0.06^{***}			$\mathbf{0.32}^{***}$
F0Progression		-0.03^{***}		-0.02^{**}			

the boxplots in Figure 4 and 5, this phenomenon can be observed in a similar fashion for both trumpet and saxophone solos.

4. CONCLUSIONS

In this paper, we propose a novel method for score-informed tracking of fundamental frequency contours. Furthermore, we introduce a set of basic contour features that characterize various aspects such as modulation range, tuning deviation towards the equal-temperament scale, as well as the overall pitch progression. In the second part of our paper, we present several exploratory analyses to investigate, how feature values differ among different artists as well as different instruments and how the contour features and contextual parameters corelate with each other.

This leads to several observations which could be fruitful for future investigations of personal style in jazz improvisation as well as for music research in general. Obviously, different jazz musicians have different tendencies to glide towards or within pitches—with a general trend to glide upwards. Personal vibrato styles are characterized mainly by different modulation ranges as well as minor differences of vibrato frequency. Our method of score-informed tracking of fundamental frequency contours could help to characterize those idiosyncratic vibrato styles.

Preceding to a vibrato analysis, all tones in a solo that are played with vibrato must be identified first. As shown in [12], a supervised classification approach based on contour features such as discussed in Section 2.7 seems as a promising approach. Since vibrato is often put on longer tones (and only occasionally on shorter ones) by jazz musicians, vibrato analyses could be enhanced by filtering the data for longer tones only, which is easily done by the MeloSpySuite software that was developed in the Jazzomat Research Project [13]. Similarly, with MeloSpySuite we could simply filter the data for thirds, fifths, and seventhes according to the underlying chords in order to analyze selectively the pitch contours in those blue note areas and learn about the bias of different artists to play blue notes. In this manner, computer-based methods of transcription and analyses of audio recordings could be extended from the symbolic or structural level (pitch, onset, duration of tones) to the micro-level of musical sound which is, presumably, pivotal for the understanding of performance style in jazz and other music genres.

5. ACKNOWLEDGEMENTS

The JAZZOMAT project is supported by a grant DFG-PF 669/7-1 ("Melodisch-rhythmische Gestaltung von Jazzimprovisationen. Rechnerbasierte Musikanalyse einstimmiger Jazzsoli") by the Deutsche Forschungsgemeinschaft (DFG). The authors would like to thank all jazz and musicology students participating in the transcription and annotation process.

6. REFERENCES

[1] D. Gerhard, "Pitch track target deviation in natural singing," in *Proceedings of the 6th International Conference on Music*

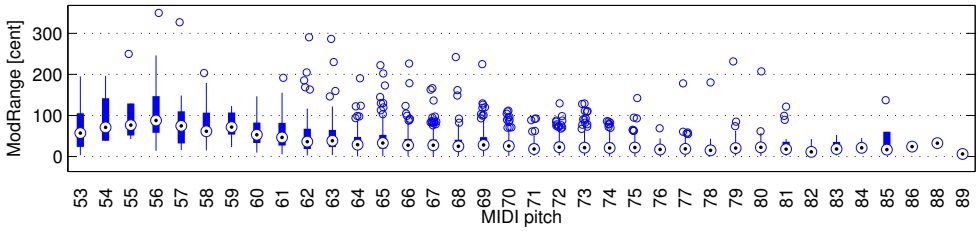


Figure 4: Boxplot over feature **ModRange** in cent over the pitch range for trumpet tones.

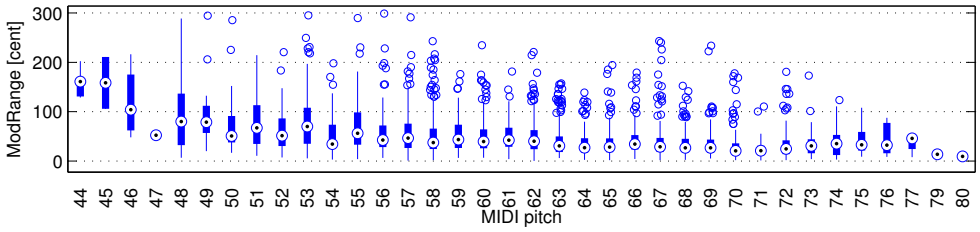


Figure 5: Boxplot over feature **ModRange** in cent over the pitch range for saxophone tones.

Information Retrieval (ISMIR), London, UK, 2005, pp. 514–519.

[2] S. S. Miryala, K. Bali, R. Bhagwan, and M. Choudhury, "Automatically identifying vocal expressions for music transcription," in *Proceedings of the 14th Conference of the International Society for Music Information Retrieval (ISMIR)*, Curitiba, Brazil, 2013, pp. 239–244.

[3] J. Salamon, E. Gómez, D. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *Signal Processing Magazine, IEEE*, vol. 31, no. 2, pp. 118–134, March 2014.

[4] J. Salamon, G. Peeters, and A. Röbel, "Statistical characterisation of melodic pitch contours and its application for melody extraction," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012, pp. 187–192.

[5] P. F. Berliner, *Thinking in Jazz: The Infinite Art of Improvisation*, Chicago Studies in Ethnomusicology. University of Chicago Press, 1994.

[6] D. Evans, *Big Road Blues: Tradition And Creativity In The Folk Blues*, Da Capo Press, 1987.

[7] J. T. Titon, *Early Downhome Blues. A Musical and Cultural Analysis*, The University of North Carolina Press, 2nd edition, 1995.

[8] M. Pfleiderer, *Five Perspectives on "Body and Soul" and Other Contributions to Music Performance Research*, chapter "Body and Soul" and the Mastery of the Jazz Tenor Saxophone, Chronos, 2009.

[9] "Weimar Jazz Database (WJazzD)," `http://jazzomat.hfm-weimar.de/`, Accessed: 2014-06-09.

[10] M. Müller and S. Ewert, "Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 215–220.

[11] T. Abe, T. Kobayashi, and S. Imai, "Harmonics tracking and pitch extraction based on instantaneous frequency," in *Proceedings of the Int. IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP-95)*, Detroid, USA, May 9 - 12, 1995, pp. 756–759.

[12] J. Abeßer, C. Dittmar, and G. Schuller, "Automatic recognition and parametrization of frequency modulation techniques in bass guitar recordings," in *Proceedings of the 42nd Audio Engineering Society (AES) Conference: Semantic Audio*, Ilmenau, Germany, 2011.

[13] J. Abeßer, K. Frieler, M. Pfleiderer, and W.-G. Zaddach, "Introducing the jazzomat project - jazz solo analysis using music information retrieval methods," in *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research (CMMR) Sound, Music and Motion*, Marseille, France, 2013, pp. 187–192.

REAL-TIME TRANSCRIPTION AND SEPARATION OF DRUM RECORDINGS BASED ON NMF DECOMPOSITION

*Christian Dittmar**
Fraunhofer IDMT,
Ilmenau, Germany
dmr@idmt.fraunhofer.de

Daniel Gärtner
Fraunhofer IDMT,
Ilmenau, Germany
gtr@idmt.fraunhofer.de

ABSTRACT

This paper proposes a real-time capable method for transcribing and separating occurrences of single drum instruments in polyphonic drum recordings. Both the detection and the decomposition are based on Non-Negative Matrix Factorization and can be implemented with very small systemic delay. We propose a simple modification to the update rules that allows to capture time-dynamic spectral characteristics of the involved drum sounds. The method can be applied in music production and music education software. Performance results with respect to drum transcription are presented and discussed. The evaluation data-set consisting of annotated drum recordings is published for use in further studies in the field.

Index Terms - drum transcription, source separation, non-negative matrix factorization, spectral processing, audio plug-in, music production, music education

1. INTRODUCTION

The rapid development of music technology in the past decades has inevitably changed the way people interact with music today. As one result, music production has shifted almost entirely to the digital domain. This evolution made music recording affordable for amateurs and semi-professionals. Furthermore, it enabled the rise of completely novel approaches to music practice and education. With these developments in mind, our work focuses on the real-time processing of drum set recordings. We aim at transcribing and isolating the single drum instruments that are played in a monaural, polyphonic drum set recording. Thus, our topic is at the intersection of automatic music transcription and source separation, two major fields in *Music Information Retrieval (MIR)* research [1, 2]. Strictly speaking, we are performing drum detection rather than transcription, since our approach is agnostic to the underlying metric structure (relations of beats and bars). However, we will use the term drum transcription for the sake of simplicity throughout the paper. We also use the term decomposition as synonym for source separation.

Our paper is structured as follows. First, the goals of our work are outlined in Sec. 2. After a review of the related work in Sec. 3, we explain the proposed transcription and separation algorithm in detail in Sec. 4. Finally, Sec. 5 describes the evaluation conducted and Sec. 6 summarizes this work.

* All correspondance should be adressed to this author.

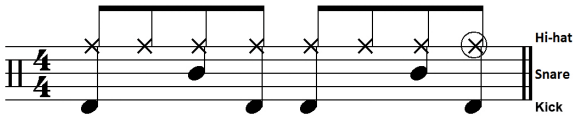


Figure 1: *A simple, one-bar drum rhythm in music notation. Taken from [3].*

2. GOALS

In professional music production, drum kits are usually recorded using several microphones that allow for separate processing of the different drum instrument signals via mixing desks. However, proper microphone setup is not trivial and even professional audio engineers often have to cope with heavy cross-talk between recording devices. In addition, amateur music producers might only have a single microphone available due to limited budget. Thus, our goal is to detect and separate occurrences of single drums within monaural polyphonic drum set recordings in real-time.

Our first application scenario is music production software, where post-processing of individual drum instruments in the mix plays an important role. In digital music production, so-called drum trigger plug-ins, such as Drumagog [1] or Steven Slate Trigger [2] are quite common. When applied to multi-channel drum set recordings, onsets can be detected in each drum channel and can be used to trigger additional digital audio samples. In a sense, these tools already perform monophonic drum transcription. One drawback of these plug-ins is the need for manual setting of trigger thresholds. Furthermore, they offer only conventional means (e.g., equalization, noise-gates) for attenuating cross-talk between drum channels. Of course, it would be desirable to better isolate the single drum-sounds automatically. As will be explained in Sec. 4, our approach requires to train the system with isolated drum sounds of the expected drum instruments. Having in mind that all drum instruments are played in succession during sound-checks, it is quire realistic to fulfill that requirement in practice.

The second application scenario are educational music games, such as Songs2See[3], BandFuse[4] and RockSmith[5]. Only a small number of music video games and music education software also offer the possibility to practice drums. In all cases, this functionality is enabled by using MIDI-fied drum sets. However, none of the existing applications allows users to practice on real-world acous-

[1] http://www.drumagog.com/
[2] http://www.stevenslatedrums.com/
[3] http://www.songs2see.com/
[4] http://bandfuse.com/
[5] http://rocksmith.ubi.com/

tic drum sets. We want to enable budding drummers to play along to a given rhythm pattern or song, while their performances, in terms of striking the correct drums to the correct points in time, are assessed in real-time. As a pre-requisite, it is necessary to recognize the different drum instruments in a monaural audio signal. Having beginners in mind, the system is constrained to detect onsets of three drum instruments as explained in Sec. 2.1. In educational music video games available on the market, it is pretty common to have a tuning stage before playing a song. In the same manner, we can require the use to play all drum instruments in succession for training the system.

2.1. The drum kit

A conventional drum kit usually consists of the drum instruments shown in Figure 2. They comprise the kick (1), snare (2), toms (3,4), hi-hat (5) and cymbals (6,7). The drums can be classified into membranophones (kick, snare, toms) and ideophones (hi-hat, cymbals). The sound is produced by striking them with drum sticks usually made of wood. In this work we are focusing on kick, snare and hi-hat. The kick is played via a foot pedal, generating a low, bass-heavy sound. The snare has snare wires stretched across the lower head. When striking the upper head with a drum stick, the lower head vibration excites the snares, generating a bright, snappy sound. The hi-hat can be opened and closed with another foot pedal. In closed mode, it produces a clicking, instantly decaying sound. In opened mode, it can sound similar to a cymbal with many turbulent high frequency modes. Real-world acoustic drums generate sound spectra that vary slightly with each successive stroke. Sample-based drum kits usually feature a limited number of pre-recorded drum sounds, while synthetic drum kits often provide just one particular sound (given the synthesis parameters are fixed).

Generally speaking, kick, snare and hi-hat can be ordered ascending by their spectral centroid. However, when polyphonic drum rhythms are played on a drum set, it is pretty common that different drums are struck simultaneously. In many common drum rhythms, the hi-hat plays all quarter or eighth notes and therefore coincides with kick and snare quite often. An example is shown in Figure 1. If such short rhythms of one to four bars are constantly repeated they are also called drum loop. In these cases, discerning the instruments by their spectral centroid is no longer possible, since only the mixed sound can be measured. In the worst case, a kick occurring simultaneously with a hi-hat could be mistaken for a snare drum. Besides the recognition of ghost-notes and other special playing techniques, the ambiguity in classifying polyphonic drum sounds poses the major challenge in automatic drum transcription.

3. STATE-OF-THE-ART

In this section, the most important directions of research in automatic drum transcription are presented. As described in [4], the existing approaches can be discerned into three different categories.

3.1. Source separation methods

The first category is also known as *separate and detect* because the signal is first decomposed into individual streams via source separation, before onset candidates are detected in each individual stream. The pre-requisite is typically a time-frequency transform (e.g., the *Short-term Fourier Transform (STFT)*). The generic signal model decomposes the resulting magnitude spectrogram X into a linear superposition of individual component spectrograms. The components are usually represented by fixed spectral basis functions B and corresponding time-varying amplitude (or gain) envelopes G. An intuitive interpretation is that the B describe how the constituent components sound, whereas the G describe when and how intense they sound. The approaches described in the literature mostly differ in the decomposition method as well as the constraints and initialization imposed on B and G.

Independent component analysis (ICA) computes a factorization

$$X = B \cdot G \tag{1}$$

such that the separated source spectra are maximally independent and non-Gaussian. *Independent subspace analysis (ISA)*, first described in [5], applies *Principal Component Analysis (PCA)* and ICA in succession for decomposing X. In order to classify the arbitrarily permuted and scaled components afterwards, feature extraction and classifiers such as *k-Nearest-Neighbor (kNN)* or *Support Vector Machines (SVM)* can be used [6]. An extension to ICA called *Non-Negative ICA (NICA)* has the constraint that the matrix B must be non-negative [7]. In [8], it is shown how to use NICA for transcription of kick, snare and hi-hat from polyphonic music.

Prior subspace analysis (PSA) was first proposed in [9] and utilizes a set of template spectrum basis functions in a matrix B_p. These consist of the averaged spectra drawn from a large collection of isolated drum sounds. A first approximation of G can be computed by

$$\hat{G} = B_p^+ \cdot X \tag{2}$$

where B_p^+ denotes the pseudo-inverse of B_p. The rows of matrix $\hat{G}$ contain the temporal activations of the template spectra in the spectrogram, but are not independent. In order to make them independent, ICA is applied afterwards. This results in an unmixing matrix W transforming $\hat{G}$ into independent amplitude gain functions according to

$$G = W \cdot \hat{G} \tag{3}$$

Subsequently, an improved estimate of the source spectra can be computed by

$$B = X \cdot G^+ \tag{4}$$

which now contains the source spectra adapted to the actual signal. Using this method, [10] reports an F-measure of 0.75 for the detection of kick and snare in polyphonic music.

An early work applying *Non-negative Matrix Factorization (NMF)* [11] for the separation of drums from polyphonic music is presented in [12]. It uses NMF minimizing the *Kullback-Leibler Divergence (KL)*, with random initialization of B and G. From the resulting components, spectral and temporal features are computed and classified with an SVM trained on the classes drums vs. harmonic. The reported results show that the NMF and SVM approach performed better than ISA and SVM. Another variant of NMF for drum transcription is described in [13]. The NMF is first applied to individual drum samples for kick, snare and hi-hat in order to derive B_p, which are later fixed during the NMF iterations. The method shows good performance on drum loops, yielding an average F-measure of 0.96 for kick, snare and hi-hat detection. In [14, 15], it is shown how source separation of instruments with time-varying spectral characteristics (such as drums) may benefit from an NMF extension called *Non-Negative Matrix Factor Deconvolution (NMFD)*. Recently, NMF-based methods have also

Figure 2: *A conventional drum kit with annotated drum instruments, taken from [3].*

been applied to real-time drum detection [16], where each drum onset is identified with Probabilistic Spectral Clustering based on the *Itakura-Saito Divergence (IS)*.

3.2. Template matching

The second category of drum transcription techniques follow a so-called *match and adapt* approach. It relies on temporal or spectral templates for the events that should be detected. In a first approximation, the occurrences of events that are similar to the template are detected. Afterwards, the templates are iteratively adapted to the given signal. The work presented in [17] uses *seed templates* for kick and snare, that are constructed from a collection of isolated drum sound spectrograms. First, onset detection determines possible candidates for drum sounds. At each onset candidate, a spectrogram snippet with the same size as the template is stored and compared with the templates. The reciprocal of the distance between the observed spectrogram and the template yields the reliability for that drum's occurrence. In the adapt stage, the seed templates are updated by taking the median power over all previously selected frames. This suppresses highly variable spectral peaks from pitched, harmonic instruments. The process of template adaption is applied iteratively, so that the output of this median filtering is used as the next seed template. The final stage determines whether the drum sound actually occurs at the onset candidate. Application of the template matching in conjunction with harmonic structure suppression, yielded an F-measure of 0.82 for kick and 0.58 for snare. A combination of template matching and sound separation is described in [18], where the candidates for template extraction are first detected using NMF. Instead of median filtering, a modified minimum filtering is applied. Another example of template matching is given in [19], where characteristic band pass filter parameters are learned. The training process is realized as an optimization of the characteristic filters with the *Differential Evolution (DE)* algorithm and fitness evaluation measures for determining each filter's ability to correctly detect the onset of the respective drum. The output of each filter represents the activations of the single drums and can be transcribed by means of envelope extraction and peak picking.

3.3. Supervised classification

The last category of transcription algorithms is referred to as *segment and classify*. It first employs temporal segmentation of the audio track into onset events. Usually, a fixed number of frames following each detected onsets is kept or a temporal grid of fixed periodicity is aligned to the audio track. Subsequently, each temporal event is identified by a classifier. Often, well-known machine learning methods such as SVM or GMM are used in conjunction with features extracted from each segment. The method in [20] uses a set of features comprising averaged MFCCs, various spectral shape parameters and the log-energy in six frequency bands corresponding to the spectral centroids of different drum instruments. The features are classified by a set of eight binary SVMs that have been trained on the classes kick, snare, hi-hat, clap, cymbal, rim shot, toms and percussion. Evaluated on a data-set of drum loops, the best configuration yielded a recognition rate of 83.9%. The method proposed in [21] uses a similar approach, but is applied for drum transcription in polyphonic music. The algorithm achieved an average F-measure of 0.61 for kick, snare and hi-hat. Finally, *Hidden Markov models (HMM)* are a machine learning method that can be used to model drum sequences. Although they are often counted as part of the *segment and classify* approach, they stand out as they are able to perform the segmentation and detection jointly. HMMs model temporal sequences by computing the probability that a given sequence of observed states were generated by hidden random variables, i.e., the activations of the drum classes. In [4], HMMs are used to model MFCCs and their temporal derivatives. The method achieves an F-measure of 0.81 for the recognition of kick, snare and hi-hat in drum loops and 0.74 in polyphonic music.

4. PROPOSED METHOD

In the preceeding section, we showed that good results have already been achieved in drum loop transcription. However, only a fraction of the methods is capable of real-time processing and only very few are suited for sound separation as well. As laid out in Section 2, our approach should cover both aspects. An overview about our proposed method is given in Figure 3. As in other works, we

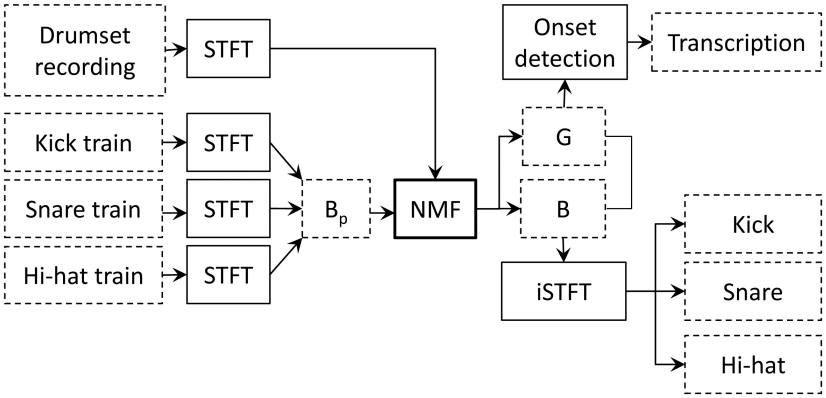


Figure 3: *Overview of the proposed method. Prior basis vectors B_p are derived from isolated drum sound spectrograms. Drum set recordings are split into spectral frames individually subjected to NMF. The resulting B and G are used for onset detection as well as inverse STFT in order to obtain isolated drum instrument recordings.*

also transform the drum recording to the time-frequency domain via STFT. As indicated in Sec. 2, we follow the approaches described in [13, 16]. We assume that an initial training phase can be conducted, where the individual drum sounds expected in the drum recordings are available in isolation. During training, we compute one prior basis vector B_p per drum instrument by just averaging along the time axis of each training spectrogram. The choice of just a single basis vector per drum is motivated by the findings in [22] as well as our general goal to spare computation time for real-time applicability. Of course, it is possible to use more than one component per drum and still reach real-time capability. In order to keep the number of samples required for processing as small as possible, the NMF decomposition is applied to each spectral frame of the drum recording individually, thus generating a succession of activations for kick, snare and hi-hat in G. In the following, three variants of the NMF decomposition are detailed.

4.1. NMF decomposition with adaptive bases

For decomposition, we use the KL Divergence resulting in the well known update rules [11] for both the spectral bases (5) as well as the amplitude envelopes (6):

$$B \leftarrow B \cdot \frac{\frac{X}{BG} G^T}{1B^T} \tag{5}$$

$$G \leftarrow G \cdot \frac{B^T \frac{X}{BG}}{B^T 1} \tag{6}$$

It should again be noted, that X represents an $N \times 1$ matrix corresponding to one individual spectral frame with N linearly spaced frequency bins. The matrix 1 consists of all ones in the appropriate dimensions. The spectral basis matrix B is initialized with B_p as proposed in other works [13, 23, 16].

4.2. NMF decomposition with fixed bases

As proposed by other authors [24], we optionally omit the update of B in Eq. 5 and just replace B with the fixed prior basis B_p. This way, it can be ensured that only the expected spectra will lead to activations in G. It can be assumed, that NMF with only one fixed basis vector per instrument will not be able to model time-dynamic spectral characteristics of drum sounds, which is in line with the findings of [16], where separate NMF templates for head and tail of a drum sound are used. Intuitively, this method is also likely to produce spurious activations in case the incoming signal consists of other components than the previously trained drum sounds. The NMF updates rules will try to model the currently observed spectra as good as possible given the fixed prior basis vectors, thus yielding activations of all drum components in the form of cross-talk. Consequences for the resulting approximation of X will be explained in 5.3.

4.3. NMF decomposition with semi-adaptive bases

In our novel approach, we introduce a modification imposing semi-adaptive behavior on B during the NMF iterations. In contrast to the procedure described in Sec. 4.1, we do not just initialize B with B_p and let them iterate freely afterwards. Instead, we allow the spectral content in B to deviate more from the initial value, the closer we are to the NMF iteration limit. This behavior is simply achieved by blending between the initial B_p and B from the current iteration as given in Equation 7. The blending parameter α depends on the ratio of current iteration count k to iteration limit K taken to the power of β as show in Equation 8.

$$B = \alpha \cdot B_p + (1 - \alpha) \cdot B \tag{7}$$

$$\alpha = (1 - \frac{k}{K})^{\beta} \tag{8}$$

Thus, the NMF components are first pushed towards the expected drum sounds. The adaption to subtle variations in the incoming spectra are allowed later. It should be noted, that the proposed procedure is not equal to *Online Non-Negative Matrix Factorization (ONMF)* algorithms (e.g., [25, 26]). Instead of learning the final NMF decomposition of an infinite stream of spectral frames over time by updating B with every incoming spectral frame, we revert to B_p prior to the NMF decomposition of every individual frame.

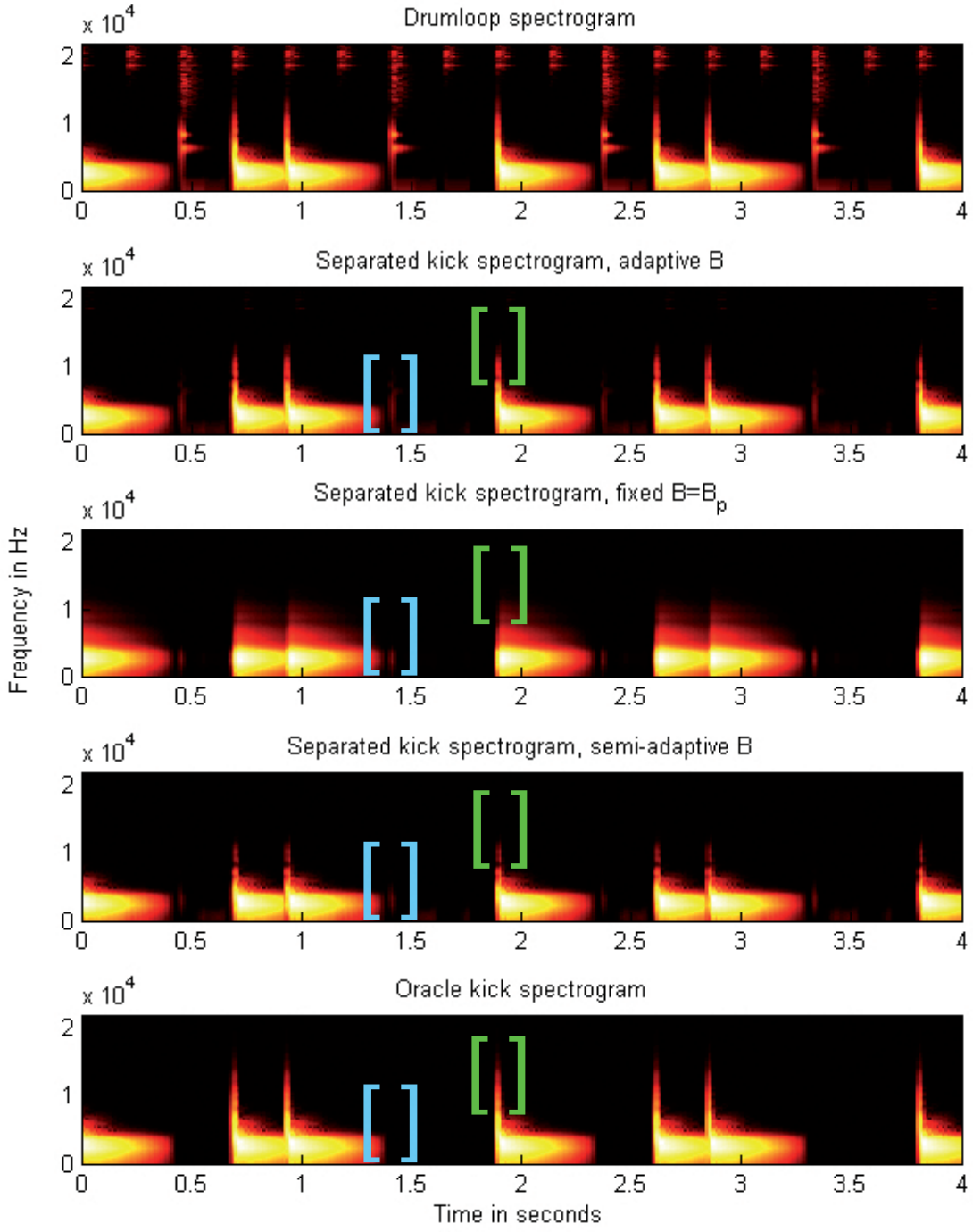


Figure 4: *Comparison of drum loop spectrograms obtained from the different decomposition methods. The top spectrogram is obtained from the input drum loop. The bottom spectrogram shows the idealized (oracle) target when separating the kick. The second, third and fourth spectrogram show the separation results obtained with adaptive B, fixed $B = B_p$ and semi-adaptive B, respectively. The kick separated using fixed B is clearly inferior compared to the oracle kick. This is evident by the smeared transient (light green brackets). It does exhibit less cross-talk from the snare (light blue brackets) yielding better transcription results than adaptive B (see Sec. 5.2). Thus, semi-adaptive B seems to be the optimal compromise between both.*

4.4. Onset detection

After decomposition, frame-wise matrix multiplication of the activations in G corresponding to a single drum with the corresponding columns in B yields well separated individual spectrograms for kick, snare and hi-hat. Based on these, onset detection is performed in a straightforward manner by means of peak-picking. While other authors used the amplitude envelopes in G directly, we encounter different spectra in every frame for the adaptive and semi-adaptive bases. Thus, we take the extra step of spectrogram reconstruction prior to onset detection. Following the approach proposed in [27], novelty curves D are extracted from the successive spectral frames for each drum by differentiating the logarithmic magnitude along time. Afterwards, all bins per frame are summed up and half-wave rectification is applied since only salient positive peaks corresponding to onsets are of interest. Inevitably, cross-talk artifacts that can occur due to imperfect decomposition may lead to erroneous spikes that can be mistaken as drum onsets. Thus, an adaptive threshold procedure is applied to the novelty curve. The threshold T is derived by element-wise nonlinear compression $D^{0.5}$, subsequent application of an exponential moving average filter and nonlinear expansion of the result $D^{2.0}$. A variable boost factor b can be used to adjust the additive offset of T manually. This is done by simply multiplying b with the arithmetic mean of T and adding the result to T. In real-time mode, the long-term arithmetic mean is derived by a frame-wise iterative update. If the novelty curve rises above T for several frames and fulfills additional plausibility criteria (see [16]), it is marked as an onset. Finally, the onset detection stage returns a list of onset times per drum instrument, yielding the final transcription result.

5. EVALUATION

In order to assess the drum transcription performance, experiments with manually transcribed drum set recordings were conducted. The well known Precision, Recall and F-measure were used as evaluation metrics with a tolerance of 50 ms between annotated and detected onsets.

5.1. Test data

A training set was created for initialization of single drums (kick, snare, hi-hat) in [3]. In order to capture the individual characteristics, the drums were hit separately with varying velocity. For recording, an overhead microphone at a fixed height of 1 m was used. The recordings were made with 10 different drum kits, consisting of different drum sizes and a broad range of materials. The size of the kick drum ranges from 18 inch to a 24 inch diameter, and depths of 16 inch up to 22 inch. Materials were birch, mahogany or maple. The snare drums all had the same size of 14 inch diameter and 6.5 inch in depth but different materials (such as metal, wood or acrylic). The sizes for hi-hat ranged from 13 inch to 15 inch. A second subset was generated using sample-based drum sets from the BFD[6] plug-in. The third part of the set featuring purely synthetic drum kits was generated using Steinberg's Groove Agent[7] plug-in. The onsets were transcribed manually by an experienced drummer using the software Sonic Visualiser [28].

In total, the test set consisted of 33 drum sequences which were fairly simple groove patterns of kick, snare and hi-hat. The tempo of the performed drum rhythms varies between 100 and 140 BPM. Overall, 10 minutes of audio were recorded (in 44.1 kHz, mono, 16 Bit) resulting in 3471 annotated onsets. The shortest annotated interval between consecutive onsets is 107 ms (16th note at 140 BPM). The combined data-set is available online as a public benchmark for drum transcription[8].

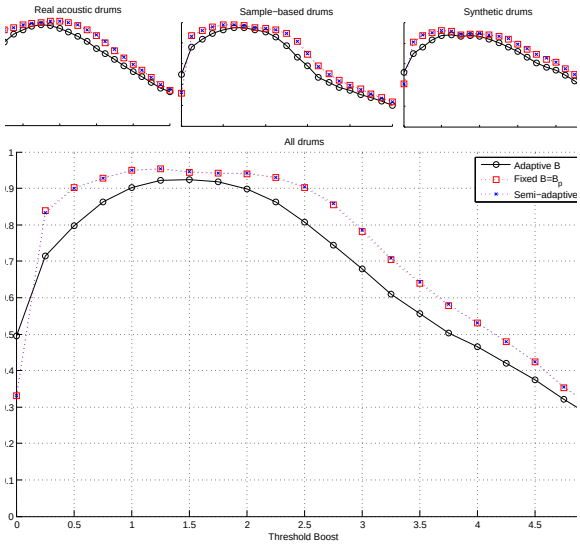


Figure 5: *Overview of transcription F-measure rates versus the threshold boost b on the different drum sets. The largest plot shows the combined results for all drum sets. The F-measures for fixed $B = B_p$ and semi-adaptive B are very similar, thus the corresponding curves are almost indistinguishable.*

5.2. Results

Using the described test data, an extensive grid search was performed in order to estimate the optimal set of parameters. We omit the details and just explain that the most influential parameters were the threshold boost and the number of NMF iterations used during decomposition. The best average F-measure of 0.95 across all drum kits was obtained with $H = 512$ samples hopsize, $N = 2048$ bins spectrum size, $b = 1.25$ threshold boost, $K = 25$ NMF iterations and $\beta = 4$ blending non-linearity in case of the semi-adaptive bases. Most surprisingly, the acoustic and sample-based drum kits lead to better F-measure scores than

[6] www.fxpansion.com/BFD

[7] http://www.steinberg.net/en/products/vst/groove_agent/groove_agent.html

[8] http://www.idmt.fraunhofer.de/en/business_units/smt/drums.html

the synthetic drum kits. This is somewhat counter-intuitive, since we expected the drum transcription performance to decrease when dealing with drum recordings under larger natural variation in the single drum sounds. We interpret this as a benefit of the semi-adaptive bases, which can be seen by the comparison between the different approaches in Figure 5. There, we show the influence of b on the F-measure scores across the different drum kits as well as the three different adaption degrees of the spectral bases. It can clearly be seen, that the adaptive B yield slightly worse transcription results, which we account to the more pronounced cross-talk artifacts. Differences between F-measure scores of fixed B and semi-adaptive B are extremely small. Nevertheless, the discussion in Sec. 5.3 shows that fixed basis vectors have their weaknesses when the drum sounds to be separated exhibit high spectral variability over time.

5.3. Influence of basis adaption

We present an illustrative example for the different degrees of adaptivity. The uppermost plot in Figure 4 shows the spectrogram of a synthetic drum loop consisting of kick, snare and hi-hat playing the rhythm given in Figure 1. It should be noted that the magnitude of the spectrograms has been converted to dB and has been resampled to a logarithmically spaced frequency axis for visualization purposes only. The bottom plot shows the oracle spectrogram of the kick playing in isolation. This kick, sampled from a Roland TR 808 drum computer, is obviously rather invariant across the repeated onsets but exhibits a very time-dynamic behavior per onset. One can clearly see a strong vertical head-transient caused by the sharp attack. Afterwards, a slightly decreasing center frequency can be observed in the tail. In the second plot of Figure 4 we see the kick spectrogram obtained from NMF decomposition with adaptive bases. The third plot shows the approximation of the kick spectrogram achieved with only one fixed spectral basis vector per drum instrument. The modeling of the attack transient is inferior, since it is smeared into the tail of the drum sound. The fourth plot shows the kick spectrogram resulting from decomposition with semi-adaptive spectral bases. When compared to the oracle spectrogram, one can clearly see that the attack transients are preserved very well. On closer inspection, all NMF variants exhibit cross talk from hi-hat and snare in the kick spectrogram (marked with light blue brackets). They are most pronounced for the fully adaptive B and can cause erroneous onset candidates during onset detection (see Sec. 4.4).

5.4. Real-time capability

The proposed algorithm has been implemented as VST plugin. A screen-shot of the user interface is shown in Figure 6. Three different drum sounds can be trained via live input or prepared audio files. Alternatively, artificial spectral basis templates can be used and refined in an iterative update. The plugin works in quasi-real-time, the systemic delay is only dependent on the input delay of the audio hardware and the used hop-size. For the optimal parameter settings given in Sec. 5.2, we could measure a delay of approximately 6 ms.

6. CONCLUSIONS

This paper presented a method for real-time transcription and separation of drum sounds from drum set recordings. It is based on

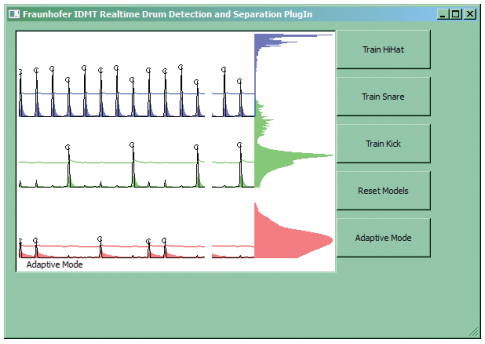


Figure 6: *Screen-shot of a VST plug-in encapsulating the proposed algorithm. The semi-transparent colored curves visualize the extracted G of the individual drums, the colored spectra on the right show the extracted B. Blue corresponds to hi-hat, green to snare and red to kick. The individual onset detection functions D are overlayed as black lines and the dynamic thresholds T as solid colored lines.*

NMF decomposition initialized with prior spectral basis templates for the expected drums. Under the assumption, that the isolated drum sounds are available for training, the transcription performance for polyphonic drum input featuring the specific instruments is on par with state-of-the-art results. The novel concept of semi-adaptive spectral bases does not yield improvements in transcription but seems promising for enhancing the perceptual quality of drum sound separation. Our collected data-set used for evaluation is contributed to the research community in order to foster reproducible research results. Future work will be directed to systematically evaluate alternative decomposition strategies, such as ONMF and NMFD. Furthermore, the applicability to a larger variety of different drum instruments (toms, cymbals, etc.) will be assessed allowing the inclusion of commonly used test corpora, such as the ENST drums data-set.

7. ACKNOWLEDGMENTS

Parts of this research have been developed during the SyncGlobal project. SyncGlobal was a 2-year collaborative research project between Piranha Womex AG, Bach Technology GmbH, 4FriendsOnly AG and the Fraunhofer IDMT in Ilmenau, Germany. The project was co-financed by the Germany Ministry of Education and Research in the framework of the SME innovation program (FKZ 01/S11007).

8. REFERENCES

[1] M. Plumbley, S. Abdallah, J. P. Bello, M. Davies, G. Monti, and M. Sandler, "Automatic music transcription and audio source separation," *Cybernetics & Systems*, vol. 33, no. 6, pp. 1–21, 2002.

[2] C. Dittmar, E. Cano, S. Grollmisch, J. Abeßer, A. Männchen, and C. Kehling, *Springer Handbook for Systematic Musicology*, chapter Music Technology and Music Education, Springer, 2014.

[3] F. Weber, "Development of a real-time algorithm for drum-

sound detection," Diploma thesis, Ilmenau University of Technology, 2013.

[4] J. Paulus, *Signal Processing Methods for Drum Transciption and Music Structure Analysis*, Ph.D. thesis, Tampere University of Technology, Tampere, Finland, 2009.

[5] M. Casey, "Separation of mixed audio sources by independent subspace analysis," in *Proceedings of the International Computer Music Conference*, 2000.

[6] C. Uhle, C. Dittmar, and T. Sporer, "Extraction of drum tracks from polyphonic music using independent subspace analysis," in *Proceedings of the 4th International Symposium on Independent Component Analysis and Blind Signal Separation*, 2003.

[7] M. Plumbley, "Algorithms for non-negative independent component analysis," *IEEE Transactions on Neural Networks*, vol. 14, pp. 30–37, 2003.

[8] C. Dittmar and C. Uhle, "Further steps towards drum transcription of polyphonic music," in *Proceedings of the AES 116th Convention*, 2004.

[9] D. FitzGerald, B. Lawlor, and E. Coyle, "Prior subspace analysis for drum transcription," in *Proceedings of the 114th AES Convention 114th Convention*, 2003.

[10] A. Spich, M. Zanoni, A. Sarti, and S. Tubaro, "Drum music transcription using prior subspace analysis and pattern recognition," in *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx)*, 2010.

[11] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, 2001.

[12] M. Helén and T. Virtanen, "Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine," in *Proceedings of the 13th European Signal Processing Conference (EUSIPCO)*, 2005.

[13] J. Paulus and T. Virtanen, "Drum transcription with non-negative spectrogram factorisation," in *Proceedings of the 13th European Signal Processing Conference (EUSIPCO)*, 2005.

[14] M.N. Schmidt and M. Mørup, "Nonnegative matrix factor 2-d deconvolution for blind single channel source separation," in *Independent Component Analysis and Blind Signal Separation*, pp. 700–707. Springer, 2006.

[15] P. Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *Independent Component Analysis and Blind Signal Separation*, pp. 494–499. Springer, 2004.

[16] E. Battenberg, V. Huang, and D. Wessel, "Live drum separation using probabilistic spectral clustering based on the itakura-saito divergence," in *Proceedings of the AES 45th Conference on Time-Frequency Processing in Audio*, Helsinki, Finland, 2012.

[17] K. Yoshii, M. Goto, and H. Okuno, "Automatic drum sound description for real-world music using template adaption and matching methods," in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, 2004.

[18] C. Dittmar, D. Wagner, and D. Gärtner, "Drumloop separation using adaptive spectrogram templates," in *Proceedings of the 36th Jahrestagung fuer Akustik (DAGA)*, 2010.

[19] A. Maximos, A. Floros, M. Vrahatis, and N. Kanellopoulos, "Real-time drums transcription with characteristic bandpass filtering," in *Proceedings of the 7th Audio Mostly Conference: A Conference on Interaction with Sound*, 2012.

[20] O. Gillet and G. Richard, "Automatic transcription of drum loops," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2004.

[21] K. Tanghe, S. Degroeve, and B. De Baets, "An algorithm for detecting and labeling drum events in polyphonic music," in *Proceedings of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX '05)*, 2005.

[22] D. Fitzgerald, *Automatic drum transcription and source separation*, Ph.D. thesis, Dublin Institute of Technology, Dublin, Ireland, 2004.

[23] S. Ewert and M. Müller, "Score-informed voice separation for piano recordings," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[24] M. N. Schmidt, *Single-channel source separation using non-negative matrix factorization*, Ph.D. thesis, Technical University of Denmark, Aalborg, Denmark, 2008.

[25] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal on Machine Learning Research*, 2010.

[26] B. Cao, D. Shen, J.T. Sun, X. Wang, Q. Yang, and Z. Chen, "Detect and track latent factors with online nonnegative matrix factorization," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, 2007, pp. 2689–2694.

[27] P. Grosche and M. Müller, "Extracting predominant local pulse information from music recordings," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1688–1701, 2011.

[28] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proceedings of the International Conference on Multimedia*, 2010, pp. 1467–1468.

[29] A. Zils, F. Pachet, O. Delerue, and F. Gouyon, "Automatic extraction of drum tracks from polyphonic music signals," in *Proceedings of the 2nd International Conference on Web Delivering of Music (WedelMusic2002)*, 2002, p. 5.

[30] J. Paulus and A. Klapuri, "Conventional and periodic n-grams in the transcription of drum sequences," in *Proceedings of the IEEE International Conference Multimedia and Expo*, 2003.

[31] J. Paulus and A. Klapuri, "Drum sound detection in polyphonic music with hidden markov models," *EURASIP Journal on Audio, Speech, and Music Processing*, 2009.

[32] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[33] J. Abeßer and O. Lartillot, "Modelling musical attributes to characterize two-track recordings with bass and drums," in *Proceedings of the International Society of Music Information Retrieval (ISMIR)*, 2011.

A PITCH SALIENCE FUNCTION DERIVED FROM HARMONIC FREQUENCY DEVIATIONS FOR POLYPHONIC MUSIC ANALYSIS

A. Degani, R. Leonardi, P. Migliorati
University of Brescia
DII, Signals and Communication Lab
38, Via Branze - 25123 Brescia (ITALY)
a.degani@unibs.it

*G. Peeters**
STMS - IRCAM - CNRS - UPMC
Sound Analysis and Synthesis
1,pl. Igor Stravinsky - 75004 Paris (FRANCE)
peeters@ircam.fr

ABSTRACT

In this paper, a novel approach for the computation of a pitch salience function is presented. The aim of a pitch (considered here as synonym for fundamental frequency) salience function is to estimate the relevance of the most salient musical pitches that are present in a certain audio excerpt. Such a function is used in numerous Music Information Retrieval (MIR) tasks such as pitch, multiple-pitch estimation, melody extraction and audio features computation (such as chroma or Pitch Class Profiles). In order to compute the salience of a pitch candidate f, the classical approach uses the weighted sum of the energy of the short time spectrum at its integer multiples frequencies hf. In the present work, we propose a different approach which does not rely on energy but only on frequency location. For this, we first estimate the peaks of the short time spectrum. From the frequency location of these peaks, we evaluate the likelihood that each peak is an harmonic of a given fundamental frequency. The specificity of our method is to use as likelihood the deviation of the harmonic frequency locations from the pitch locations of the equal tempered scale. This is used to create a theoretical sequence of deviations which is then compared to an observed one. The proposed method is then evaluated for a task of multiple-pitch estimation using the MAPS test-set.

1. INTRODUCTION

A salience function is a function that provides an estimation of the predominance of different frequencies in an audio signal at every time frame. It allows to obtain an improved spectral representation in which the fundamental frequencies have a greater relevance compared to the higher partials of a complex tone. The computation of a salience function is commonly used as a first step in melody, predominant-pitch (pitch is considered here as synonym to fundamental frequency or f_0) or multiple-pitch estimation systems [1, 2, 3, 4].

1.1. Classical approach

In the classical approach [5], the salience (or strength) of each f_0 candidate is calculated as a weighted sum of the amplitudes of the spectrum at its harmonic frequencies (integer multiples of f_0). In the discrete frequency case, this can be express as:

$$S[k] = \sum_{h=1}^{H} w_h |X[hk]| \quad (1)$$

where k is the spectral bin, H is the number of considered partials, w_h is a partials' weighting scheme and $|X[k]|$ is the amplitude spectrum. This process is repeated for each time frame m. In this approach, the choice of the number of considered harmonics H and the used weighting scheme w_h are important factors and directly affect the obtained results [5]. The weighting scheme w_h implicitly models the sound source. Since the classical approach is based on the amplitude/energy of the spectrum, it is sensitive to the timbre of the sources. In order to make the estimation more robust against timbre variations, spectral whitening or flattening processes have been proposed [2, 6, 7, 8].

Among other approaches, the one of [9] proposes to estimate the salient pitch of a complex tone mixture using a psychoacoustic motivated approach. It uses the notions of masking and virtual pitch (sub-harmonic coincidence) calculation.

1.2. Proposal

In this paper, we propose a novel salience function which does not rely on the amplitude/energy of the spectrum but only on the frequency location of the peaks of the spectrum. Doing this, our method is not sensitive to timbre variations hence does not necessitate whitening processes.

The specificity of our method is to use as likelihood the deviation of the harmonic frequency locations from the pitch locations of the equal tempered scale. This is illustrated in Figure 1, for the harmonic frequencies of the pitch $C4$ (MIDI Key Number $i = 60$), which 3-rd and 6-th harmonic frequencies are slightly above the pitches $i = 79$ and $i = 81$ respectively, while its 5-th and 7-th are below the pitches $i = 88$ and $i = 94$ respectively (in the equal tempered scale). This is used to create a theoretical sequence of deviations which is then compared to an observed one derived from the peaks detected in the spectrum.

Paper organization: In section 2 we present the motivation behind the concept of this novel salience computation approach (section 2.2) and the details of its computation (sections 2.3, 2.4, 2.5 and 2.6). In section 3 we propose a basic evaluation framework of salience function based on a multi-pitch estimation paradigm (section 3.1) and asses the performances of our proposed method (section 3.4). We finally conclude in section 4 and provide directions for future works.

* Thanks to the Quaero Program funded by Oseo French State agency for innovation.

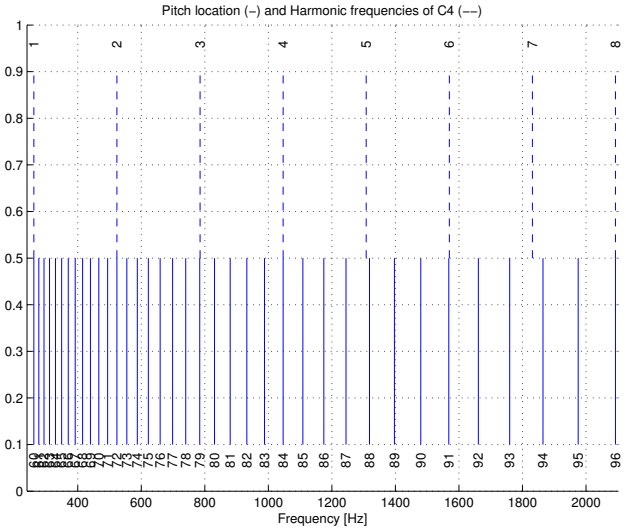


Figure 1: (Lower part) Frequency location of the pitches of the equal tempered scale for a tuning of 440 Hz. (Upper part) Frequencies of the harmonic series of the pitch C4 (261.6 Hz).

2. PROPOSED METHOD

2.1. Overview

The global flowchart of our method is represented in Fig. 2.

The content of the audio signal is first analyzed using Short Time Fourier Transform (STFT). At each time frame m, the peaks of the local Discrete Fourier Transform (DFT) are estimated using a peak-picking algorithm.
We denote by $\mathcal{P}_m = \{(f_1, a_1), \ldots, (f_P, a_P)\}$ the set of peaks detected at the frame m where f_p and a_p are the frequency and amplitude of the p-th peak. Since our salience function is based on an equal tempered cents grid, we then need to estimate the tuning frequency f_{ref} of the audio signal. We then compute at each frame m the salience value of each peak p by comparing its frequency to the ones of an equal tempered scale tuned on f_{ref}. This salience allows to discriminate peaks which are fundamental frequency from the ones that are harmonic partials.

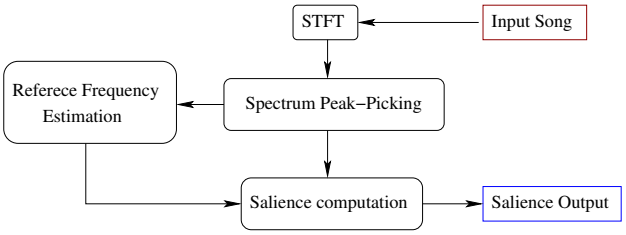


Figure 2: General scheme of the method.

2.2. Motivations for using frequency deviations for pitch salience computation

The computation of our salience function only relies on the frequency positions of the peaks of the spectrum (not on their energy). The basic idea we develop is the following: for a given note at fundamental frequency f_0 its h-th harmonic frequency exhibits a specific deviation from the equal tempered scale. For example, for a tuning at 440 Hz, the third ($h = 3$) harmonic of a A4 note ($f_0 = 440$ Hz) is at frequency 1320 Hz while the closest note of the equal tempered scale is at 1318.5 Hz. The specific deviation of the third harmonic is then 1.95 cents.

For a given frequency f_0, the frequency of its h-th harmonic is defined by

$$f_h^{f_0} = h \cdot f_0 \tag{2}$$

The deviation in cents of the harmonic $f_h^{f_0}$ from the equal tempered grid is defined as:

$$d_h^{f_0} = 100 \left[12 \log_2 \left(\frac{f_h^{f_0}}{f_{ref}} \right) - \left\lfloor 12 \log_2 \left(\frac{f_h^{f_0}}{f_{ref}} \right) \right\rceil \right] \tag{3}$$

where $\lfloor \cdot \rceil$ is the rounding operator and f_{ref} is the $A4$ tuning frequency estimated from the data[1]. We denote by $\{f_h^{f_0}\}$, the sequence of all the harmonic frequencies of f_0 and by $\{d_h^{f_0}\}$, the theoretical sequence of deviations.

This deviation is independent[2] of the actual f_0. We therefore simply denote it by $\{d_h\}$ in the following. In Fig. 3, we illustrate the deviation of the first 20 harmonics of a complex tone from the equal tempered note scale.

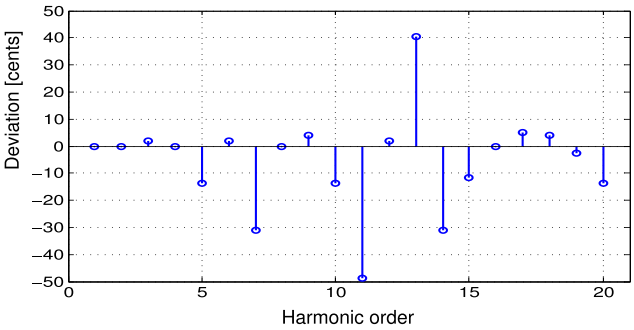


Figure 3: Deviation of the first 20 harmonic frequencies of a complex tone from the pitch of the equal tempered scale.

Salience computation: Since the sequence $\{d_h\}$ is independent of fundamental frequency, we can simply compute the salience of each f_0 candidate at frequency f_p, as the correlation between

[1] Or blindly chosen as 440 Hz.

[2] **Proof that $d_h^{f_0}$ is independent of f_0:** Under the hypothesis that the analysed musical excerpt is played on the equal temperament scale and using an accurately tuned instrument, we can calculate each equal-tempered note frequency f_i in the audio spectrum using an integer number i as follows:

$$f_i = f_{ref} \cdot 2^{(\frac{i}{12})} \tag{4}$$

The integer number i represents the note index in the MIDI notation without the offset of 69 (for the sake of simplicity, we assume that $A4$ correspond to $i = 0$ instead of $i = 69$). Now, it is easy to check that for all fundamental frequencies $f_0 = f_i$, (3) can be rewritten as:

$$\begin{aligned} d_h^{f_i} &= 100 \left[12 \log_2(h) + i - \lfloor 12 \log_2(h) + i \rceil \right] \\ &= 100 \left[12 \log_2(h) - \lfloor 12 \log_2(h) \rceil \right] \end{aligned} \tag{5}$$

Since $i \in \mathbb{Z}$, we can say that $\lfloor 12 \log_2(h) + i \rceil = i + \lfloor 12 \log_2(h) \rceil$ and it is clear that the sequence $\{d_h^{f_i}\}$ does not depend on the fundamental frequency f_i.

the theoretical sequence of deviation $\{d_h\}$ and the measured sequence of the deviation $\{\hat{d}_h^{f_p}\}$. The measured sequence of deviations is the one corresponding to the peak detected in the spectrum $\mathcal{P}_m$.

Extension to inharmonic signals: Inharmonicity is a phenomenon related to the physical characteristics of a non-ideal string. The frequencies of the modes of vibration of an ideal string are exact integer multiples of the fundamental, but the stiffness of the material of the real strings shifts the modes of vibration at non-integer multiples [10].
In mathematical terms, the relation between the h-th partial $f_h^{f_0}$ and the fundamental frequency f_0 can be modelled as

$$f_h^{f_0}(\beta) = h f_0 \sqrt{1 + \beta h^2} \tag{6}$$

where β is the *inharmonicity coefficient* which is related to the physical properties of a string. In order to take into account inharmonicity we use (6) instead of (2) into equation (3). It should be noted that whatever inharmonicity is taken into account or not, the theoretical sequence of deviations is always independent of f_0. However, the theoretical sequence of deviations now depends on the parameter β and it is denoted by $\{d_h(\beta)\}$.

In the next sections, we describe in details each block of our algorithm (see Fig. 2).

2.3. Short Time Fourier Transform

The N-terms STFT, at time frame m, of a discrete signal $x[n]$ is defined as

$$X_{m,k} = \sum_{n=0}^{N-1} x[n + \tau m] \cdot \bar{w}[n] \cdot e^{-j2\pi \frac{k}{N} n} \tag{7}$$

where $k \in [-N/2 + 1, \ldots, N/2]$, τ is the hop size (in samples) from two subsequent frames and $\bar{w}[n]$ is the windowing function. For our computation, we only use the amplitude of the STFT denoted by $|X_{m,k}|$. We use $N = 4096$ samples (which corresponds to 92.9 ms for a sampling rate of 44.1 KHz), $\tau = 2048$ samples (overlap of 50%) and $\bar{w}[n]$ is a Hanning windowing function.

2.4. Spectrum Peak Picking

In order to detect the local peaks of the spectrum, we use the algorithm proposed in the context of the Sinusoidal Modelling Synthesis framework (SMS) [11, 12]. In this context, a fixed number P of local maxima is detected in the amplitude spectrum $|X_{m,k}|$. For each local maximum, its frequency $\mathcal{M}_p$ is refined using a 3-point parabolic interpolation using $[\mathcal{M}_p - 1, \mathcal{M}_p, \mathcal{M}_p + 1]$. The obtained frequency is denoted by f_p in Hz. The result of the peak picking algorithm is the sequence $\mathcal{P}_m = \{(f_1, a_1), \ldots, (f_P, a_P)\}$ made of pairs of peaks frequency location f_p and amplitude a_p. The peak-picking is performed at each time frame $m \in [1 \ldots M]$. The concatenation of all peaks sequences, $\mathcal{P}_{tot} = \mathcal{P}_1 \| \mathcal{P}_2 \| \ldots \| \mathcal{P}_M$, is used as input for the reference tuning estimation algorithm.

2.5. Reference Frequency Estimation

Since our algorithm relies on the equal-tempered cent scale, the tuning f_{ref} (or reference frequency) of the audio signal being analyzed need to be estimated. For this, we use the method presented in [13]. This approach is entirely based on the observation that the deviation d is a periodic measure and not an absolute measure, since it is a "wrapped around" quantity that should be evaluated from the nearest 100 cents grid point. Each cent value is mapped onto a unit circle 100 cents-periodic, and represented as a vector as follows

$$\mathbf{u}_p = a_p \cdot e^{j\phi_p} \tag{8}$$

where

$$\phi_p = \frac{2\pi}{100} \cdot 1200 \cdot \log_2\left(\frac{f_p}{440}\right) \tag{9}$$

and a_p is the peak amplitude used to weight each vector to avoid high impact of small (noise) peaks. We take the mean vector $\hat{\mathbf{u}}$ of all circular quantities $\mathbf{u}_p$ as follows

$$\hat{\mathbf{u}} = \frac{\sum_{p=1}^{P_{tot}} \mathbf{u}_p}{\sum_{p=1}^{P_{tot}} a_p} \tag{10}$$

where P_{tot} is the element count of the concatenated sequence $\mathcal{P}_{tot}$. The overall deviation is then computed from the angle of the resulting vector $\hat{\mathbf{u}}$, that is

$$D = \frac{1}{2\pi} \angle(\hat{\mathbf{u}}) \tag{11}$$

The reference frequency of the entire music piece can be computed as

$$f_{ref} = 440 \cdot 2^{\frac{D}{12}} \tag{12}$$

2.6. Salience Function Computation

As previously said, the salience $S_p(\beta)$ of a given peak p can be calculated as the correlation C between the theoretical sequence of deviation $\{d_h(\beta)\}$ and the measured one $\{\hat{d}_h^{f_p}(\beta)\}$. From the abstract point of view, $S_p(\beta)$ is calculated using:

$$S_p(\beta) = C\left(\{d_h(\beta)\}, \{\hat{d}_h^{f_p}(\beta)\}\right) \tag{13}$$

where $C(\cdot, \cdot)$ is a generic correlation measure. The two deviation sequences can be seen as two vectors $\mathbf{d}(\beta) = [d_1(\beta), \ldots, d_H(\beta)]$ and $\hat{\mathbf{d}}^p(\beta) = [\hat{d}_1^{f_p}(\beta), \ldots, \hat{d}_H^{f_p}(\beta)]$, so that, a good correlation measure can be the inner product $< \cdot, \cdot >$. In practice, in order to reduce the influence of very small values (hence often noisy) in the computation of the salience, the correlation is weighted by the local amplitude a_p of the f_0 candidate f_p:

$$S_p(\beta) = a_p < \mathbf{d}(\beta), \hat{\mathbf{d}}^p(\beta) > = a_p \sum_{h=1}^{H} d_h(\beta) \cdot \hat{d}_h^{f_p}(\beta) \tag{14}$$

Computation of $\hat{d}_h^{f_p}(\beta)$: $\{f_h^{f_p}(\beta)\}$ is the sequence made of the harmonic frequencies of a detected peak p: $f_h^{f_p}(\beta) = h f_p \sqrt{1 + \beta h^2}$. $\{\hat{d}_h^{f_p}(\beta)\}$ is the vector of measured deviations corresponding to $\{f_h^{f_p}(\beta)\}$. $\{\hat{d}_h^{f_p}(\beta)\}$ is computed for all the detected peaks $p \in \mathcal{P}_m$ at frame m (i.e. we consider each detected peak as a potential pitch candidate).

To validate a given pitch candidate f_p, we look among the detected peaks the ones that are harmonics of this candidate. This is done by using a function G centered on the $h-$th harmonic of f_p and evaluated at the detected peaks $f_{p'}$.

More precisely, $G(f_{p'}; \mu_{h,p}(\beta), \sigma_{h,p}(\beta))$ is a Gaussian function evaluated at $f_{p'}$, with

- mean $\mu_{h,p}(\beta) = h f_p \sqrt{1 + \beta h^2}$ and

- standard deviation $\sigma_{h,p}(\beta) = \mu_{h,p}(\beta)\left(1 - 2^{\frac{\alpha}{1200}}\right)$

where the parameter $\alpha = 20$ cents is chosen experimentally in order to take into account the effect of the frequency location error of the peak picking step. We chose the α such that the number of the False Positive is reduced without losing Precision (see the section 3.2 for the explanation of the evaluation measures).
The Gaussian function we use, has a maximum value of one when $f_{p'} = \mu_{h,p}(\beta) = hf_p\sqrt{1+\beta h^2}$; in other words G will only take non-zero values for the $f_{p'}$ (the detected peaks) which are close to $hf_p\sqrt{1+\beta h^2}$.

To each detected peaks $f_{p'}$ is associated a deviation $\bar{d}_{p'}$ as defined in (3).

$$\bar{d}_{p'} = 100\left[12\log_2\left(\frac{f_{p'}}{f_{ref}}\right) - \left\lfloor 12\log_2\left(\frac{f_{p'}}{f_{ref}}\right)\right\rfloor\right] \quad (15)$$

The deviation of $f_h^{f_p}(\beta)$ is then computed as the following weighted sum:

$$\hat{d}_h^{f_p}(\beta) = \sum_{p'=1}^{P} G(f_{p'}; \mu_{h,p}(\beta), \sigma_{h,p}(\beta)) \cdot \bar{d}_{p'} \quad (16)$$

A single value of β is assigned to each pitch candidate f_p. The typical range of β for a piano string [10] is $\beta \in B = \{0\} \cup [10^{-5}, 10^{-3}]$. In order to estimate β we maximize

$$S_p = \max_{\beta \in B}[S_p(\beta)] \quad (17)$$

Notice that in the practical case, all the values of β in the search range must be tested exhaustively because $S_p(\beta)$ is an "unpredictable" function and no numerical optimized algorithm can be used in order to find the maximum of that function. The maximization of (17) provides simultaneously the value of S_p and the one of the inharmonicity coefficient β for each spectral peak p. Of course, only the values of β corresponding to true notes make sense.

The limits of the equal temperament: Using the equal-tempered grid of semitones is fundamental for the consideration made in Sec. 2.2. Moreover, it is reasonable to think that only exact tuned instruments[3] are needed in order to maintain the validity of equation (5). However, the gaussian weighting scheme used in (16) ensures that the slighted deviated fundamental frequencies are not much negatively affected. However, the spectral peaks that are detuned more than $\pm\alpha$ cents can be excessively penalized.

3. EVALUATION

There is no standard method to evaluate the performances of a salience function by itself. This is because such a function is usually a pre-processing step of a more complicated algorithm (as for example a pitch-estimation method [2, 14]). Therefore, in order to be able to test our salience, we chose to construct a very simple and straightforward multiple-pitch estimation algorithm from our salience function. In section 3.1, we explain the post-processing applied to the salience function in order to obtain a multi-pitch estimation.

[3]For example, the octave stretching in piano tuning can be a problem.

3.1. Multiple-pitch estimation: post-processing of the salience function

In order to test our salience function as a multi-pitch estimation algorithm, we chose to apply a basic post-processing process that transforms the salience function into a piano-roll representation. The piano-roll $\hat{\mathcal{R}}_{m,i}$ can be seen as a spectrogram-like binary representation where the rows are the time frames m and the columns are the MIDI Key Number i[4] . If a note i is marked as detected at the time frame m, the corresponding element $\hat{\mathcal{R}}_{m,i}$ is set to 1; otherwise it is set to 0.

At each time frame m, we have a sequence of P pairs of peak frequency and salience values $\mathcal{S}_m = \{(f_1, S_1), \ldots, (f_P, S_P)\}$. We normalize the values S_p in order to obtain maximum amplitude of one at each time frame. The negative values of salience are set to zero. Each peak frequency f_p is quantized to the nearest MIDI Key Number using

$$i_p = 69 + 12\log_2\left(\frac{f_p}{f_{ref}}\right) \quad (18)$$

where 69 correspond to the MIDI Key Number associated to the note $A4$ in the MIDI Tuning Standard (MTS). In order to remove holes (estimation errors) in the middle of notes (disruption in the salience value), we then apply a sliding median filter of size L frames along the time dimension m. Finally the binary piano-roll is obtained by applying a fixed threshold T to the values of $\hat{\mathcal{R}}_{m,i}$. We set to 1 all the values that are above T, and 0 the other ones. In Fig. 4, an example of piano-roll transcription is shown and different colors are used in order to highlight the True Positive, False Positive and False Negative. Notice that a considerable number of False Positive are just after a True Positive in the same MIDI Key Number. This is caused by the release time of the piano sound that is extended by the reverberation time simulated in the recordings.

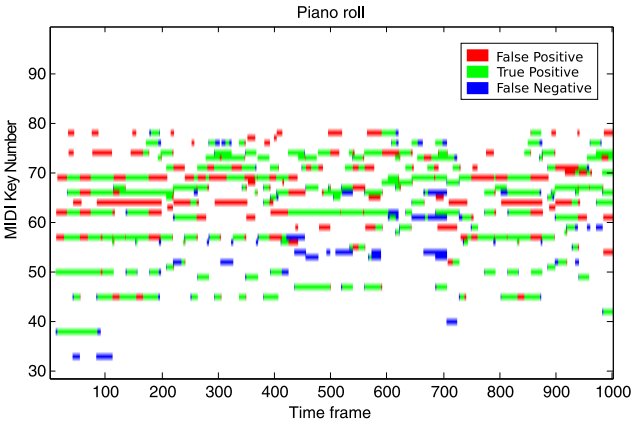


Figure 4: Piano roll representation $\hat{\mathcal{R}}_{m,i}$ obtained using our salience function. In this case $P = 0.65$, $R = 0.8$ and $F = 0.72$ (see explanation of the evaluation measures in section 3.2).

3.2. Evaluation measures

In order to evaluate our salience-based piano-roll, we have to compute a ground-truth piano-roll $\mathcal{R}_{m,i}$ for each song in the dataset.

[4]Ranging from 21 ($A0$) to 108 ($C8$).

$\mathcal{R}_{m,i}$ is obtained from the ground-truth text annotation that reports *onset* time, *offset* time and MIDI Key Number for each note played in a specific song. The note onset and offset time are quantized with the same hop size τ (converted in seconds) used by the algorithm.

We compare the ground-truth piano-roll $\mathcal{R}_{m,i}$ to the estimated piano-roll $\hat{\mathcal{R}}_{m,i}$ by comparing the values on each cell (m,i). We then compute the *Precision* (P), the *Recall* (R) and the *F-Measure* (F) defined as follows:

$$P = \frac{TP}{TP+FP}, \quad R = \frac{TP}{TP+FN}, \quad F = \frac{2PR}{P+R} \tag{19}$$

where TP (True Positive) is the total number of correctly identified notes, FN (False Negative) of missed notes and FP (False Positive) the number of false notes detected.

3.3. Test-Set

Experiments are performed on the MIDI Aligned Piano Sounds test-set [15]. MAPS provides CD quality piano recording (44.1 kHz, 16-bit). This test-set is available under Creative Commons license and consists of about 40GB (65 hours) of audio files recorded using both real and synthesized pianos. The aligned ground-truth is provided as MIDI or plain text files. The alignment and the reliability of the ground-truth is guaranteed by the fact that the sound files are generated form this MIDI files with high quality samples or a Disklavier (real piano with MIDI input). In order to have a generalized test-set, the pianos have been played in different conditions, such as various ambient with different reverberation characteristics (9 combinations in total). This collection is subdivided into four different subsets. The set ISOL contains monophonic excerpts, MUS contains polyphonic music, UCHO is a set of usual chords in western music, and RAND is a collection of chords with random notes.

3.4. Results

Setting the parameters: The parameters of our algorithm are:

- H: the total number of considered harmonics,
- L: the length of the median filter,
- T: the salience threshold.

In order to tune these parameters we used the *AkPnStgb* audio files of the test-set[5]. The values that maximize the *F-Measure* are $H = 8$, $L = 6$ and $T = 0.2$. The total number of peaks per frame P, is not itself a parameter of the salience algorithm. $P = 40$ is chosen experimentally.

Harmonic vs Inharmonic model: We first compare in Figure 5 the Pitch estimation obtained by our model in harmonic setting (the β parameters is forced to 0) to the inharmonic setting (β is estimated). This is done using the whole MAPS test-set.

As we expected, taking into account the inharmonicity brings an improvement on overall. The precision P increases by 12% (from 0.43 to 0.55) and the F-Measure increases by 5%. Since the Recall does not change significantly, while the Precision does, we can say that considering string inharmonicity allows reducing the number of False Positive. Because the results are better with our inharmonic model, we only consider this one in the following.

[5]This is one of the nine different piano and recording condition set-up in the MAPS test-set

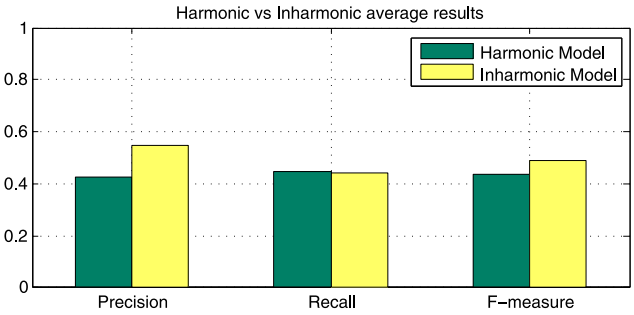


Figure 5: Pitch estimation results for our model in Harmonic (model forced to $\beta = 0$) vs Inharmonic setting (β is estimated).

Detailed Analysis: In Fig. 6, we provide the results in terms of Pitch estimation for each subset of MAPS using the inharmonic model. In Fig. 7, we provide the results in terms of Pitch-Class (i.e., without octave information). As we can see from the Figures, our approach is prone to octave errors. This is due to the fact that the deviation template itself does not exploit the octave information[6]. This octave ambiguity could only be solved with an ad hoc procedure. Figures 6 and 7 also show that on average, the precision P is greater than the recall R. For a fixed number of True Positive, this means that the number of False Negative (missed notes) is greater than the False Positive (added notes).

Influence of the T parameter: In Figure 8, we show the variation of the Recall and Precision in function of the choice of the parameter T (threshold on salience values). We see that the choice of T is a key parameter for the *Precision/Recall* trade-off, hence for the FP / FN trade-off. If our system is used as a front-end of a more complicated system which can filter-out the False Positives, we should use a value of T which maximizes Recall. It should be noted that Figure 8 is computed using only the MUS subset of MAPS. Because of this, the best value for T (in F-Measure sense) is $T = 0.1$ (which is different from the global optimum value for the entire MAPS test-set).

Comparison to state-of-the-art: In Table 1, we indicate the Pitch F-Measure results of our system in harmonic setting (P1, β is forced to 0) and in inharmonic setting (P2 , β is estimated). We compare our results to the ones obtained by Emiya et al. [15] and Benetos et al. [7] on the same test-set. Also, the results obtained by directly applying a threshold on the detected peaks are reported as a baseline results. As expected, the results obtained with our methods are not as good as the ones obtained with dedicated multi-pitch estimation algorithms.

The main reason is that our system is not a multi-pitch estimation method but only a pre-processing step to be used in a more complex system. Our straightforward post-processing procedure

[6]In a hypothetical scenario where the peak peaking algorithm detects the peaks at frequency f_p and in an infinite number of its harmonics with amplitude equal to a_p, the salience value for the peak p and for the peaks with frequency $\bar{h} f_p$ with $\bar{h} = 2^j$, $j \in \mathbb{N}^+$ will be the same. To put it in another way, the peaks with frequency that is j octaves above f_p will measure the same salience value as f_p.

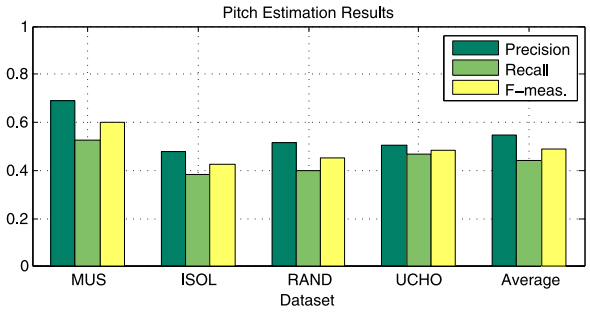


Figure 6: Pitch estimation result for each subset and the overall average (β is estimated).

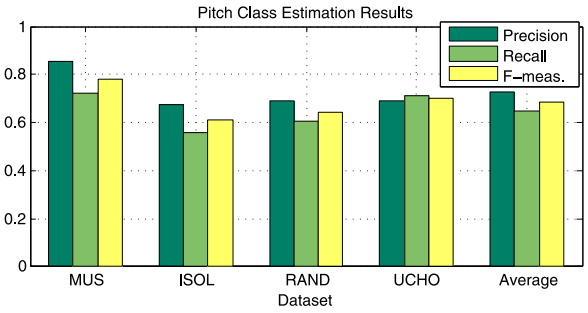


Figure 7: Pitch-Class estimation result for each subset and the overall average (β is estimated).

	Peak	P1	P2	Emiya et al.	Benetos et al.
F-Meas.	0.31	0.44	0.49	0.82	0.87

Table 1: Comparison of Pitch F-Measure results on MAPS test-set. *Peak* is a fixed threshold on detected peaks, *P1* is the proposed method without considering inharmonicity (β forced to 0) and *P2* is with the inharmonic model (β is estimated). *Emiya et al.* is presented in [15] and *Benetos et al.* in [7].

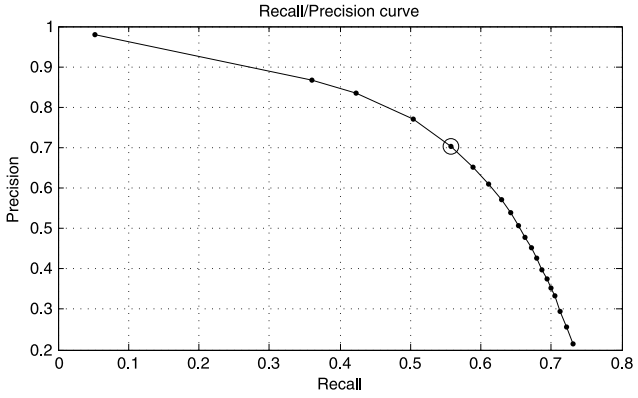


Figure 8: Pitch Recall/Precision curve for different values of T for the MUS subset. The best *F-Measure* (0.62) is obtained for $T = 0.1$ and is marked with the "O".

has been introduced only to asses the potential performances of our novel salience function design. In this context, our salience exhibit very promising results.

4. CONCLUSIONS

The performances obtained by our proposed salience function for the estimation of pitch-classes (Fig. 7) show that this kind of salience, even with simple post-processing procedure, is suitable for extracting audio features like Pitch Class Profile (PCP [16]) used in cover song detection or key/chord recognition tasks [17, 18]. Moreover, especially for a piano music test-set such as MAPS, considering the string inharmonicity is beneficial in terms of precision and F-Measure. Despite the fact that our salience function look promising, further development of an ad-hoc post-processing procedure is needed in order to be used for multi-pitch estimation. Moreover, as indicated in Section 3.4, the parameter T should be tuned depending on the application, in order to favour the F-Measure or the Recall. During our tests we have identified some weakness that are subjects for future research. The accuracy of the peak peaking algorithm is a key factor. A missing peak can negatively affect the overall accuracy performances. The octave ambiguity discussed in the previous section can be treated by developing specific procedure. Furthermore, the worst resolution in the low frequency spectrum can led to a large error when calculating the high order harmonic frequencies. Conversely, the note in the high portion of the audio spectrum does not have a sufficient number of partials to give a consistent value of salience because of the spectral roll-off near the Nyquist limit.

Acknowledgements

This work was partly founded by OSE through the Quaero project and by the French government Programme Investissements d'Avenir (PIA) through the Bee Music Project.

5. REFERENCES

[1] Karin Dressler, "Audio melody extraction for mirex 2009," *5th Music Information Retrieval Evaluation eXchange, (MIREX)*, 2009.

[2] Matti P. Ryynanen and Anssi P. Klapuri, "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, vol. 32, no. 3, 2008.

[3] Yipeng Li and DeLiang Wang, "Pitch detection in polyphonic music using instrument tone models," *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*, vol. 2, pp. 481–484, 2007.

[4] Tiago Fernandes Tavares, Jayme Garcia Arnal Barbedo, and Amauri Lopes, "Improving a multiple pitch estimation method with ar models," *Proc. of Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*, 2011.

[5] Justin Salamon, Emilia Gómez, and Jordi Bonada, "Sinusoid extraction and salience function design for predominant melody estimation," *Proc. of the 14th Int. Conference on Digital Audio Effects, (DAFx)*, 2011.

[6] Emmanouil Benetos and Simon Dixon, "Polyphonic music transcription using note onset and offset detection," *Proc.*

of IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP), pp. 37–40, 2011.

[7] Emmanouil Benetos and Simon Dixon, "Multiple-f0 estimation of piano sounds exploiting spectral structure and temporal evolution," *Proc. of ISCA Tutorial and Research Workshop on Statistical and Perceptual Audition*, pp. 13–18, 2010.

[8] Justin Salamon and Emilia Gómez, "A chroma-based salience function for melody and bass line estimation from music audio signals," *Proc. of Sound and Music Computing Conference (SMC)*, pp. 331–336, 2009.

[9] Ernst Terhardt, Gerhard Stoll, and Manfred Seewann, "Algorithm for extraction of pitch and pitch salience from complex tonal signals," *Journal of the Acoustical Society of America*, vol. 71, pp. 679–688, 1982.

[10] Harvey Fletcher, E. Donnell Blackham, and Richard Stratton, "Quality of piano tones," *Journal of Acoustical Society of America*, vol. 34, no. 6, pp. 749–761, 1962.

[11] Xavier Amatriain, Jordi Bonada, Alex Loscos, and Xavier Serra, *in Udo Zölzer DAFX-Digital Audio Effects*, chapter Spectral processing, pp. 373–438, John Wiley & Sons, 2002.

[12] Xavier Serra, "Musical sound modeling with sinusoids plus noise," in *Musical Signal Processing*, A. Picialli C. Roads, S. Pope and G. De Poli, Eds., chapter Musical Sound Modeling with Sinusoids plus Noise, pp. 91–122. Swets & Zeitlinger Publishers, 1997.

[13] Karin Dressler and Sebastian Streich, "Tuning frequency estimation using circular statistics," *Proc. of the 8th Int. Conf. on Music Information Retrieval, (ISMIR)*, pp. 357–360, 2007.

[14] Karin Dressler, "Pitch estimation by the pair-wise evaluation of spectral peaks," *Proc. of Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*, 2011.

[15] Valentin Emiya, Roland Badeau, and Bertrand David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[16] Takuya Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," *Proc. of the Int. Computer Music Conference, (ICMC)*, pp. 464–467, 1999.

[17] Joan Serrà, *Identification of Versions of the Same Musical Composition by Processing Audio Descriptions*, Ph.D. thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2011.

[18] Geoffroy Peeters, "Chroma-based estimation of musical key from audio-signal analysis," *Proc. of the 7th Int. Conf. on Music Information Retrieval, (ISMIR)*, 2006.

A COMPARISON OF EXTENDED SOURCE-FILTER MODELS FOR MUSICAL SIGNAL RECONSTRUCTION

Tian Cheng, Simon Dixon, Matthias Mauch†*

Centre for Digital Music
Queen Mary University of London
London, United Kingdom
{t.cheng, s.e.dixon, m.mauch}@qmul.ac.uk

ABSTRACT

Recently, we have witnessed an increasing use of the source-filter model in music analysis, which is achieved by integrating the source filter model into a non-negative matrix factorisation (NMF) framework or statistical models. The combination of the source-filter model and NMF framework reduces the number of free parameters needed and makes the model more flexible to extend. This paper compares four extended source-filter models: the source-filter-decay (SFD) model, the NMF with time-frequency activations (NMF-ARMA) model, the multi-excitation (ME) model and the source-filter model based on β-divergence (SFbeta model). The first two models represent the time-varying spectra by adding a loss filter and a time-varying filter, respectively. The latter two are extended by using multiple excitations and including a scale factor, respectively. The models are tested using sounds of 15 instruments from the RWC Music Database. Performance is evaluated based on the relative reconstruction error. The results show that the NMF-ARMA model outperforms other models, but uses the largest set of parameters.

1. INTRODUCTION

The source-filter model is a widely-used approximate physical model (considered as a physical model only when the coupling between the source and the filter is weak) for musical instrument modelling. The source (also called excitation) represents the vibrating object, and the filter models the frequency response of the instrument body. Introductions to physical modelling and source-filter models can be found in [1] and [2], respectively. Since Virtanen et al. used the source-filter model in audio analysis, and estimated the model parameters using the methods extended from NMF and non-negative matrix deconvolution (NMD) [3], we have observed some combinations of the source-filter model and NMF frameworks or statistical models for music analysis. In these combined models, the parameters are estimated in NMF framework [3, 4, 5, 6, 7, 8], with NMD [9], non-negative tensor factorisation [10], statistical models (Gaussian Scaled Mixture Model and extended Instantaneous Mixture Model) [11], or using EM [12].

By using the source-filter model, the spectral basis can be represented as a product of a source and filter, which reduces the number of free parameters and makes the estimation more reliable. On the other hand, the NMF or statistical model not only provides the baseline update rules for estimating parameters, but also makes the model more flexible to extend. For instance, the multi-excitation model extended the excitation as a weighted sum of multiple excitations with a harmonic constraint [8]; and Kirchhoff et al. extended the source-filter model with a scaling factor to compensate for gain differences [9]. A further extension is to include the time dimension in order to describe the time-varying spectral energy distribution. The source-filter-decay model proposed by Klapuri [13] extended the source-filter model with a loss filter to represent the time-varying spectral shape of pitched musical instruments. In [14], a model was proposed for representing the time-varying spectral characteristics of a quasi-harmonic instrument sound by assuming the spectral envelope to be determined by the partials' amplitude trajectories. A source-filter factorisation was proposed to model non-stationary audio events in [15]. In this model, the source works as the spectral basis of the NMF, and the filter is extended and works as the frequency-dependent temporal activations. The parameters of the source-filter model are estimated based on NMF. These models have shown their usefulness in several MIR applications, such as source separation [6, 10, 12, 13], melody extraction [5, 11] and music transcription [4, 7, 8, 9].

In this paper, four extended models are chosen for comparison: the source-filter-decay (SFD) model [13], the NMF with time-frequency activations (NMF-ARMA) model [15], the multi-excitation (ME) model [8] and the source-filter model based on β-divergence (SFbeta model) [9]. For the sake of completeness, a standard NMF is also included as a base line of comparison. The evaluation is based on the relative reconstruction error, and the complexities of the models are analysed in terms of parameter dimensionality. The results tested on the sounds from 15 instruments show that the average relative reconstruction error of the NMF-ARMA model is smallest, while its parameter dimensionality is largest. It approximates wind instruments well, while the other three models have better results on string instruments. All the models perform well on piano and guitar, while no model works well for all the instruments because of differences between the structures of the instruments. The poor performance for violin and vocals indicates a limitation of the models when encountering vibrato, which can be investigated in future work.

The rest of the paper is organised as follows: Section 2 gives a brief introduction to the models with modifications and parameters. The comparison results are illustrated in Section 3. Conclusions are drawn in Section 4.

* Tian Cheng is supported by a China Scholarship Council (CSC)/ Queen Mary Joint PhD scholarship.

† Matthias Mauch is funded by a Royal Academy of Engineering Research Fellowship.

2. MODELS

In this section, we present four different extended source-filter models. We mainly focus on the motivations and how the models are formulated rather than the detailed parameter learning equations which can be found in corresponding papers. Where applicable, we specify the modifications we made in order to run the methods and provide information on parameter settings.

2.1. Source-Filter-Decay Model

The source-filter-decay model [13] provides a way of representing the time-varying spectral energy distribution of pitched musical instruments. The changing spectral distribution of an instrument is modelled by extending the source-filter model by a loss filter, which models the frequency-dependent decay along the time axis. The model on a decibel scale is given as follows:

$$S^{(t)}_{\mathrm{dB}}(f_h) = \gamma_{\mathrm{dB}} + X_{\mathrm{dB}}(h) + B_{\mathrm{dB}}(f_h) + tL_{\mathrm{dB}}(f_h) + E^{(t)}_{\mathrm{dB}}(f_h) \tag{1}$$

where $f_h \approx hF$, is the frequency of h^{th} harmonic of the fundamental frequency F, $S^{(t)}_{\mathrm{dB}}(f_h)$ is the power spectrum (but only modelled at the positions of the harmonics), γ_{dB} denotes the overall gain of the sound, $X_{\mathrm{dB}}(h)$ is the initial level of the h^{th} harmonic, $B_{\mathrm{dB}}(f_h)$ represents the frequency response of the instrumental body, $L_{\mathrm{dB}}(f_h)$ is the frequency-dependent loss filter and $E^{(t)}_{\mathrm{dB}}(f_h)$ represents modelling error.

The 'source' X, 'filter' B and 'decay' L are further represented by the linear models:

$$\begin{aligned} X_{\mathrm{dB}}(h) &= \sum_{i=1}^{C_x} \xi_i x_i(h) \\ B_{\mathrm{dB}}(f) &= \sum_{j=1}^{C_b} \beta_j b_j(f) \text{ and } L_{\mathrm{dB}}(f) = \sum_{k=1}^{C_l} \lambda_k l_k(f) \end{aligned} \tag{2}$$

The basis functions $x_i(h)$ are found by performing PCA on the harmonics of sounds collected from 33 instruments, while $b_j(f)$ and $l_k(f)$ are defined in the same way with overlapped triangular bandpass responses on a critical-band frequency scale. After choosing the basis functions, $X_{\mathrm{dB}}(h)$, $B_{\mathrm{dB}}(f_h)$ and $L_{\mathrm{dB}}(f_h)$ are determined by the weights ξ_i, β_j and λ_k, respectively.

The parameters are estimated by minimizing the least-square (LS) error between the observed and modelled harmonic level using a weighted LS estimator. The influence of γ_{dB} is eliminated by performing subtraction between two observed harmonics or two consecutive frames.

2.1.1. Modifications

As the F0 estimation method used in the model [16] is unavailable to us, we use the pitches extracted by the SWIPE algorithm [17] with manual corrections (referred to as detected F0s, also used in Section 2.3.1 and 2.4.1). The model is built on the first two frames with stable pitches (frames after transient) of each note.

As the model only captures the harmonic levels of the sound, we convolve the result with the magnitude response of the window function to generate the reconstruction.

2.1.2. Parameters

The model is analysed in two scenarios: with and without decay filter, denoted by *SFD*(111) and *SFD*(110), respectively. However, as the decay rate modelled in two frames is not reliable to reconstruct the spectra of the whole note clip, we use only the model without decay filter for the reconstruction.

2.2. NMF-ARMA Model

Hennequin et al. extended the temporal activations of the standard NMF framework to be frequency-dependent, in order to model non-stationary notes [15]. The spectral basis and frequency-dependent activations in the NMF framework work as the sources and time-varying filters in the source-filter model. The time-varying filters are modelled using the Autoregressive Moving Average (ARMA) model and parameters are learned in the NMF framework, which is called source-filter factorisation. The spectrogram is modelled as follows:

$$V_{ft} \approx \hat{V}_{ft} = \sum_{r=1}^{R} \omega_{fr} h_{rt}(f) \tag{3}$$

where V_{ft} and $\hat{V}_{ft}$ are the original and reconstructed spectrograms, ω_{fr} are the spectral bases (the sources), $h_{rt}(f)$ are the frequency-dependent activations (the time-varying filters), which are parameterized following the general ARMA model:

$$h_{rt}(f) = \delta^2_{rt} \frac{|\sum_{q=0}^{Q} b^q_{rt} e^{-i2\pi v_f q}|^2}{|\sum_{p=0}^{P} a^p_{rt} e^{-i2\pi v_f p}|^2} \tag{4}$$

where δ^2_{rt} is the global gain of the filter, and b^q_{rt} and a^p_{rt} are the coefficients of the MA and AR parts of the filter, respectively. $v_f = (f-1)/(2(F-1))$, where f is frequency bin and F the total number of frequency bins.

This time-varying filter represents the spectral variations of the sound which are not modelled in standard NMF. The parameters are learned in an NMF framework using β-divergence.

2.2.1. Parameters

For each instrument, N sources are used, one for each note. Two sets of ARMA parameters are in use: $Q = 0, P = 2$ for the instruments with strongly varying spectral shapes and $Q = 1, P = 1$ for others. They are represented by *ARMA*(02) and *ARMA*(11), respectively.

2.3. Multi-Excitation Model

The multi-excitation model is motivated by the non-smooth structure of the spectral envelopes often observed in wind instruments [8]. To tackle this problem, note-varying excitations are represented by the weighted summation of excitation bases, which are modelled under a harmonic constraint as follows:

$$\begin{aligned} e_{n,j}(f) &= \sum_{m=1}^{M} a_{m,n,j} G(f - m f_0(n)) \\ a_{m,n,j} &= \sum_{i=1}^{I} w_{i,n,j} v_{i,m,j} \end{aligned} \tag{5}$$

where $e_{n,j}(f)$ is the excitation for pitch n and instrument j, $a_{m,n,j}$ is the amplitude of the m^{th} partial of the same note and $G(f - mf_0(n))$ is the harmonic component of pitch n. $v_{i,m,j}$ is the excitation basis vector belonging to partial m and instrument j, and $w_{i,n,j}$ is the weight of the i^{th} excitation basis for pitch n and instrument j.

The spectral basis function is modelled as the product of the excitation and the filter h in the usual way:

$$b_{n,j} = h_j(f)e_{n,j}(f) \tag{6}$$

and the reconstructed spectrogram is given as follows:

$$\hat{x}_t(f) = \sum_{n,j} g_{n,t,j} b_{n,j} \tag{7}$$

where $g_{n,t,j}$ are gains of instrument j.

The parameters are learnt in an NMF framework with KL divergence. For post-processing, temporal continuity is enforced over the gains by adding a cost term to penalize large changes in the gains between adjacent frames.

2.3.1. Modifications

The harmonic components are built based on detected F0s rather than the ideal pitches. We give up temporal continuity as no significant improvement is found (maybe because of an unsuitable parameter). Instead, we apply the sparsity constraint used in [18] for the post-processing as the test dataset only consists of isolated notes.

2.3.2. Parameters

The system is tested with 1,2 and 4 excitations (represented by *ME(I)*, where *I* is the number of excitations) to find out the relation between the number of excitations and the performance. In this paper, only the situation with one instrument at a time has been considered.

2.4. SFbeta Model

The source-filter model proposed by Kirchhoff et al. [9] is for estimating the missing templates for user-assisted music transcription. The model is built using a common excitation spectrum and a filter response on a log-frequency scale with a scaling factor. The proposed source-filter model represents the spectrum $\mathbf{w}_p$ of pitch ϕ_p as follows:

$$\mathbf{w}_p \approx \hat{\mathbf{w}}_p = \mathbf{s}_p \cdot \overset{\phi_p\downarrow}{\mathbf{e}} \otimes \mathbf{h} \tag{8}$$

where $\hat{\mathbf{w}}_p$ is the estimated spectrum, $\mathbf{s}_p$ is the scaling factor, $\mathbf{e}$ is the excitation, and $\mathbf{h}$ the filter response. The frequency is represented on a logarithmic scale. The $\otimes$ operator denotes element-wise multiplication of the vector, and the operator $\phi_p \downarrow$ shifts the excitation spectrum $\mathbf{e}$ along the frequency axis by ϕ_p frequency bins.

For all pitches ϕ_p ($p \in [1, \cdots, P]$), the scalars s_p are combined into a vector $\mathbf{s}$ of length P, and vectors $\mathbf{w}_p$ are combined into a matrix $\mathbf{W} \in R_+^{K,P}$, where K is the number of frequency bins. $\hat{\mathbf{W}}$ is a matrix with the same dimension as $\mathbf{W}$ combined from vectors of $\hat{\mathbf{w}}_p$.

The parameters $\mathbf{s}$, $\mathbf{e}$ and $\mathbf{h}$ are estimated by using gradient descent on each vector iteratively to gradually decrease the β-divergence between $\mathbf{W}$ and $\hat{\mathbf{W}}$. The vectors are randomly initialized and details of the derivation of the update equations can be found in [19].

Table 1: *Instrument categories*

Categories	Instrument
String	piano, harpsichord, guitar, violin
Wind	accordion, harmonica, pipe organ, horn, saxophone, oboe, bassoon, clarinet, flute
Vocal	alto (female), tenor (male)

2.4.1. Modifications

In the model [9], spectra are shifted down to get the relative spectra according to the note pitches. Here we shift the spectra down using detected F0s rather than the ideal pitches, as not all instruments in the dataset are tuned to the same reference frequency. Preliminary tests have shown that this is necessary in order to obtain reasonable results.

3. EVALUATION

To evaluate a physical model for music analysis, the criterion is mainly based on the difference between the model's output and the original sound. In this paper, we evaluate the models according to the relative reconstruction error between the modelled and observed spectra. In addition, the parameter dimension of each model is analysed.

3.1. Evaluation Metrics

The relative reconstruction error (*RRE*) is chosen for the evaluation, which is defined as below:

$$RRE = \|OS - RS\|_F / \|OS\|_F \tag{9}$$

where $\|\cdot\|_F$ is the Frobenius norm, *OS* is the observed spectrum and *RS* is the reconstructed spectrum, both are amplitude spectra. We use the relative reconstruction error instead of the reconstruction error as the time-frequency representations of the models are different and the lengths of the sounds vary with instruments.

The parameter dimensionality indicates the complexity of the model, which is analysed in association with the time-frequency representation, the note ranges of the instruments, harmonic number and so on.

3.2. Experimental Setup

3.2.1. Test Dataset

To evaluate the four models, we choose the sounds of 15 instruments from the Musical Instrument Sound Database in the RWC Music Database[20], including string, wind instruments, female and male vocals, as listed in Table 1. Two violin recordings are chosen: Violin and Violin2 referring to notes played with and without vibrato, respectively.

For each instrument, we use the first 1s of each recorded note or the duration of the note if the note lasts for less than 1s. The onsets are detected using SuperFlux [21] with manual corrections. The F0s of the notes of the instruments are extracted using the SWIPE algorithm [17] with manual corrections. The ground truth (onsets and pitches) for these files can be found on-line.[1]

[1] available at https://code.soundsoftware.ac.uk/projects/onsetpitch/files

3.2.2. Time-Frequency Representation

For the source-filter-decay model, the NMF-ARMA model and the multi-excitation model, the original spectra are computed using the Short-Time Fourier Transform (STFT). Frames are segmented by a 2048-sample Hamming window with a hop-size of 441. A Discrete Fourier Transform is performed on each frame with 2-fold zero-padding. The sampling rate is $f_s = 44100$ Hz. For the SFbeta model, the time-frequency representation is calculated using a constant-Q transform [22] with 48 frequency bins per octave. The frequency range of all models is from 25 to 12500 Hz covering about 9 octaves.

The reconstructed spectra of the time-varying models (SFD and NMF-ARMA) cover the whole duration of the sound clips. For the multi-excitation model and SFbeta model, we first generate the spectral dictionary for the instruments based on the model, then calculate the reconstructed spectra using a standard NMF framework (multiplicative update) with the dictionary. This is also done when not using the decay filter in the source-filter-decay model.

3.3. Results

For the sake of completeness, the reconstruction result for NMF with no constraint is also included as a bottom line for comparison. The models are analysed in terms of the relative reconstruction error and parameter dimensionality.

3.3.1. Relative Reconstruction Errors

The results of the relative reconstruction error of the models are listed in Table 2. Models are tested with different parameters. Detailed parameters can be found in Section 2.

The average *RRE* of the source-filter-decay model is largest among the models, up to 42.9%. The model works relatively well on guitar, bassoon and flute (*RRE* $< 30\%$) but performs badly on the harmonica and vocals. Although the model is simplified by using a small set of parameters based on data from only 2 frames, the performance of the model is then affected.

The NMF-ARMA model outperforms other models with an average *RRE* of 16.9%. The model was proposed to model sounds with a strongly varying spectral shape, and the results show that it works well on most instruments (except violin with vibrato and vocals). The missing results (denoted by '–') are caused by inverting a singular matrix, which shows that the spectra of these instruments are flat and are not suitable for a model designed to deal with strong spectral variations. On the other hand, the improvement brought by using the parameter set $(0, 2)$ indicates that the notes have time-varying spectral shapes. An advantage of this model is manifested in the performance on the wind instruments with an average *RRE* of only 12.1%. Best results are found in bassoon and horn with *RRE*s of 4.99% and 6.10%, respectively. However, performance dramatically drops on violin with vibrato and vocals with about 40% *RRE*s using the parameter set $(1, 1)$, while the errors decrease by 4% for violin and by about 0.6% for the vocals using the parameter set $(0, 2)$. The poor performance occurs on the vibrato sounds such as those shown in Figure 1 (b). This is mainly because the model uses one filter per note, which fails to model the fluctuating pitches.

The multi-excitation model is proposed to approximate the non-smooth spectral envelopes of wind instruments by using a combination of excitations. We observe that the performance of

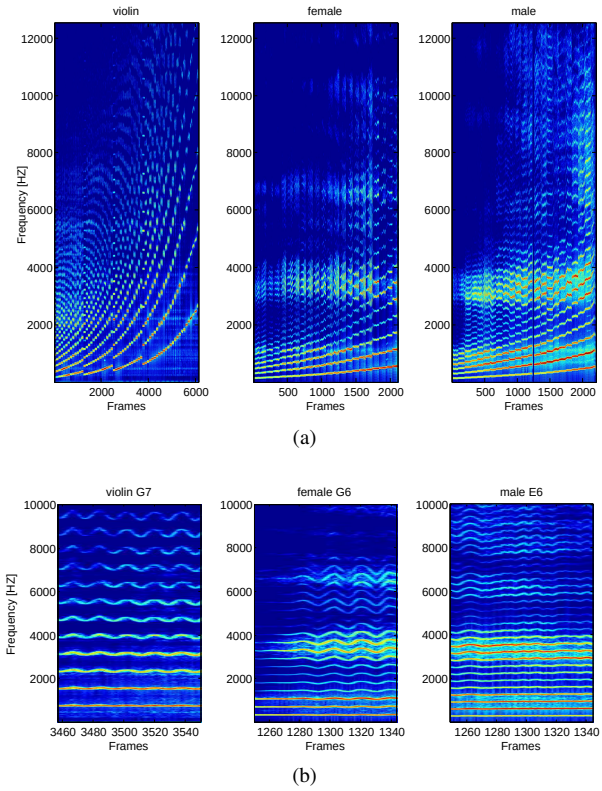


Figure 1: *Spectrograms of violin, female and male vocals: (a) all notes (64 notes for violin, 22 and 23 notes for female and male vocals, respectively), (b) individual note example.*

the model gradually improves with increasing number of excitations. The average *RRE* drops from 31.3% to 28.9% when using 2 excitations rather than 1; while a further 1.5% decrease is achieved by using 4 excitations. Notably, the errors fall by about 7% when employing 2 excitations on violin, bassoon and clarinet. The improvement by using more excitations indicates that the instrument has a non-smooth spectral envelope. The model works well on piano, pipe organ, guitar and flute even with one excitation. However, we also observe some noisy results when using more excitations in harpsichord, pipe organ, saxophone, flute and male voice.

The SFbeta model is the only model using a log-frequency scale among these models. The average *RRE* is 34.1% and the model is less sensitive to the choice of instrument. The model performs best on piano, guitar and tenor saxophone with *RRE*s of about 26%, while the worst results appear on clarinet, vocals, violin and bassoon. A notable phenomenon is that there are some inconsistencies of this model appearing on tenor saxophone, bassoon and flute, as we find that the other three models provide relatively poor results on tenor saxophone and perform well on flute. In addition, the best results of the source-filter-decay model and the NMF-ARMA model appear on bassoon, while the result on bassoon of the SFbeta model is one of the worst. By comparing the performance of the model on wind instruments, we find the model performs better on instruments with short and low frequency ranges.

Three out of four models, except the NMF-ARMA model, per-

Table 2: *Relative reconstruction errors (*RRE*), expressed as percentages. Results of piano and results better than that of piano are shown in bold. The symbol '–' means the result is not available, see text for details. The average error of the NMF-ARMA model is calculated using the better results of the two parameter sets.*

Instrument	MIDI range	*SFD*(110)	*ARMA*(11)	*ARMA*(02)	*ME*(1)	*ME*(2)	*ME*(4)	*SFbeta*	*NMF*
01 Piano	21-108	**31.7**	**7.43**	–	**11.4**	**10.7**	**10.7**	**26.6**	**5.34**
03 Harpsichord	28-88	52.1	16.4	–	19.7	19.0	22.4	31.2	7.69
06 Pipe Organ	36-91	42.9	13.1	–	15.1	15.5	15.4	36.8	9.19
07 Accordion	53-93	42.6	15.3	14.6	34.0	32.6	30.8	31.5	20.1
08 Harmonica	65-100	76.8	16.8	16.8	49.8	49.3	48.9	34.6	27.7
09 Guitar	40-76	**26.8**	11.7	–	12.7	**10.1**	**8.05**	**25.6**	**4.24**
15 Violin	55-101	37.2	40.4	36.4	38.0	31.1	29.9	38.5	15.4
15 Violin2	55-101	36.4	16.8	9.61	37.2	30.8	24.2	33.2	**4.92**
24 Horn	41-77	**31.5**	**6.10**	–	35.0	33.6	30.1	30.1	8.77
27 Tenor Sax	44-75	44.3	17.2	17.2	42.4	42.0	35.1	**25.7**	16.7
29 Oboe	58-91	44.6	8.23	–	22.8	20.2	19.1	35.2	14.3
30 Bassoon	34-72	**23.8**	**4.99**	–	32.3	25.8	23.7	38.5	9.13
31 Clarinet	50-89	48.5	15.6	–	51.7	44.9	42.2	43.7	20.9
33 Flute	60-96	**28.7**	12.2	–	14.6	14.6	15.3	37.2	9.15
46 Female	53-74	53.4	42.6	41.9	38.2	36.8	36.7	40.5	19.2
47 Male	53-74	64.8	38.4	37.9	45.8	45.2	45.7	37.2	22.1
Average		42.9	16.9		31.3	28.9	27.4	34.1	13.4
String Average		**36.8**	16.3		**23.8**	**20.3**	**19.1**	**31.0**	**7.52**
Wind Average		42.6	**12.1**		33.1	30.9	28.9	34.8	15.1
Vocal Average		59.1	39.9		42.0	41.0	41.2	38.9	20.7

form better on string instruments than on wind instruments, as shown in the average *RRE*s of string and wind instruments. We find that all models work well on piano and guitar, as a convincing evidence of the fitness of the source-filter model for these two instruments. On the other hand, all models perform badly on violin with vibrato and vocals. Two recordings of violin with vibrato and without vibrato are compared to find out whether the poor performance is caused by the vibrato. The *RRE*s of violin without vibrato (Violin2) are better than that of violin with vibrato (Violin) for all models. However, by checking the results of Violin2, a drop of 7.19% on *RRE* by using the parameter set $(0, 2)$ in the NMF-ARMA model shows that the violin sounds have a strong changing spectra distribution, while the improvement by using more excitations in the multi-excitation model indicates the non-smooth structure of the spectral envelope. So we could say that apart from the vibrato, the poor performance on violin is also caused by a changing spectral shape (as a result of, for instance, consistent changing pressure on the bow) and a non-smooth spectral envelope (4 strings). The bad results on vocals were not expected before the experiments, since the vocals have obvious filter responses as shown in Figure 1(a). And we found that the models capture the frequency response of the filter quite well in Figure 2. The frequency response corresponds to the vocal tract shape of the vowel /a:/ [23]. As no significant improvement is brought by using the parameter set $(0, 2)$ in the NMF-ARMA model and using more excitations in the ME model, the error is likely to stem from the reconstruction of the vibrato. The SFbeta model is least sensitive to vibrato. That is partly because for the constant-Q transform the frequency variations keep the same for all the partials, while the frequency variance gets larger at higher frequencies on the linear frequency scale. The SFbeta model performs worst on violin, bassoon and clarinet. They are exactly the same instruments which the multi-excitation model gets greatest improvement by using more excitations. This indicates the non-smooth spectral envelopes of these instruments and the utility of the multi-excitation model.

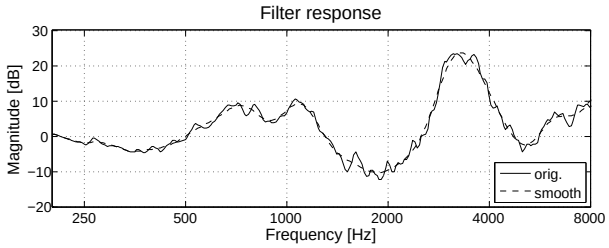


Figure 2: *Filter response of male vocal generated by the SFbeta model.*

3.3.2. Parameter Dimensionality

The parameter dimensions of the models are shown in Table 3. F and T are the numbers of frequency bins and time frames, respectively. The note range of each instrument is denoted by N. H is the number of the harmonics included in the model. I in the multi-excitation model indicates the number of excitations. To make it more intuitive, we list the dimensions for two instruments in Table 4, piano with 88 notes and harmonica with 20 notes.

The source-filter-decay model only has values at harmonic positions. The harmonic levels are represented by a weighted sum of C_H basis functions. The filter and decay are generated using a combination of C_B overlapped triangular bandpass filters. In this experiment, we use 15 basis functions and 20 bandpass filters, so only 55 parameters are used for each instrument in this model. When without the decay filter, the gains with NT parameters are also needed for the reconstruction.

The NMF-ARMA model builds each note using a source and

Table 4: *Parameter dimensions for piano and harmonica.*

Instrument	No.	SFD(111)	SFD(110)	ARMA	ME	SFbeta	NMF
Piano	88	55	7.4×10^5	2.3×10^6	7.4×10^5	5.9×10^6	9.2×10^5
Harmonica	20	55	3.8×10^4	1.4×10^5	4.0×10^4	3.1×10^5	7.9×10^4

Table 3: *Parameter dimensions.*

Algorithm	Dimension
SFD (111)	$C_H + 2C_B$
SFD (110)	$C_H + C_B + NT$
ARMA	$NF + NT(Q + P + 1)$
ME	$I(N + H) + F + NT$
SFbeta	$2F + NT$
NMF	$NF + NT$

a time-varying filter. The filter is represented by an ARMA model with $Q + P + 1$ parameters. So the number of parameters is $F + T(Q+P+1)$ per note. The parameter dimension increases linearly according to the note ranges of the instruments.

Both the multi-excitation model and the SFbeta model represent the spectra by multiplication of the dictionaries built by the models and the gains (NT). The source (excitation) of the multi-excitation model is a weighted sum of I excitation bases, and each basis is represented by H harmonics. The weights of each note is different, with NI weights in total. The filter is represented by F frequency bins. The whole model is represented by $I(N + H) + F + NT$ parameters.

For SFbeta model, the dictionary is generated by a source (F parameters) and filter (F parameters). The dimension of this model's parameters is $2F + NT$. The reason for the high figure of the model as shown in Table 4 is because the constant-Q transform has different numbers of frequency bins (F) and time frames (T). Apart from the influence of the TF representation, the parameter dimension of the SFbeta model is about the same as that of the multi-excitation model.

3.3.3. Comparison with NMF

With a large set of parameters, the average *RRE* of the NMF is smaller than that of all source-filter based models. Models with larger sets of parameters tend to have better results on the *RRE*. The NMF-ARMA model (with the largest set of parameters) outperforms the NMF on the average *RRE* of wind instruments. Besides reducing the number of free parameters, the source-filter models are employed because appropriate training data are not always available in real-world MIR applications and, as a result, pre-trained templates may not work [8].

4. CONCLUSIONS

In this paper, four extended source-filter models are evaluated according to the relative reconstruction error on sound clips from 15 instruments in the RWC Music Dataset. The results show that the source-filter-decay model captures the harmonic levels only with a small set of parameters, resulting in a large relative reconstruction error. The NMF-ARMA model obtains the smallest reconstruction result with the largest set of parameters. Performance is improved by using more excitations in the multi-excitation model, especially for violin, bassoon and clarinet, and the improvement indicates a non-smooth spectral envelope of the instrument. The results of the SFbeta model show low sensitivity to the choice of instrument. Overall, all the models perform well on piano and guitar, while no model works well for all the instruments because of differences between the structures of the instruments. The poor performance on vibrato indicates that a more flexible and shiftable structure is needed.

In future, we would like to develop a shiftable source-filter model for vibrato sounds using a constant-Q transform.

5. ACKNOWLEDGEMENTS

We thank Anssi Klapuri and Roland Badeau for generously sharing their code. We thank Holger Kirchhoff for making his code open-source.

6. REFERENCES

[1] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Reports on Progress in Physics*, vol. 69, no. 1, pp. 1–78, Jan. 2006.

[2] D. Arfib, F. Keiler, U. Zölzer, and V. Verfaille, "Source-Filter Processing," in *DAFX: Digital Audio Effects: Second Edition*, pp. 279–320. J. Wiley Sons, Chichester, 2011.

[3] T. Virtanen and A. Klapuri, "Analysis of polyphonic audio using source-filter model and non-negative matrix factorization," in *Advances in Models for Acoustic Processing, Neural Information Processing Systems Workshop*, Hawaii, USA, 2006.

[4] E. Vincent, N. Bertin, and R. Badeau, "Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, Nevada, USA, 2008, pp. 109–112.

[5] J. Durrieu, G. Richard, and B. David, "Singer melody extraction in polyphonic signals using source separation methods," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, Nevada, USA, 2008, pp. 169–172.

[6] T. Heittola, A. Klapuri, and T. Virtanen, "Musical instrument recognition in polyphonic audio using source-filter model for sound separation," in *10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 2009, pp. 327–332.

[7] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 3, pp. 528–537, 2010.

[8] J. Carabias-Orti, T. Virtanen, P. Vera-Candeas, N. Ruiz-Reyes, and F. Canadas-Quesada, "Musical instrument sound

multi-excitation model for non-negative spectrogram factorization," *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1144–1158, 2011.

[9] H. Kirchhoff, S. Dixon, and A. Klapuri, "Missing template estimation for user-assisted music transcription," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 26–30.

[10] D. Fitzgerald, M. Cranitch, and E. Coyle, "Extended nonnegative tensor factorisation models for musical sound source separation," *Computational Intelligence and Neuroscience*, 2008.

[11] J. Durrieu, G. Richard, B. David, and C. Févotte, "Source/filter model for unsupervised main melody extraction from polyphonic audio signals," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 3, pp. 564–575, 2010.

[12] A. Klapuri, T. Virtanen, and T. Heittola, "Sound source separation in monaural music signals using excitation-filter model and EM algorithm," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, Texas, USA, 2010, pp. 5510–5513.

[13] A. Klapuri, "Analysis of musical instrument sounds by source-filter-decay model," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, USA, 2007, pp. 53–56.

[14] H. Hahn, A. Röbel, J. Burred, and S. Weinzierl, "Source-filter model for quasi-harmonic instruments," in *International Conference on Digital Audio Effects (DAFx)*, Graz, Austria, 2010, pp. 1–6.

[15] R. Hennequin, R. Badeau, and B. David, "NMF with time-frequency activations to model nonstationary audio events," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 4, pp. 744–753, 2011.

[16] A. Klapuri, "Multiple fundamental frequency estimation by summing harmonic amplitudes," in *7th International Society for Music Information Retrieval Conference (ISMIR)*, Victoria, Canada, 2006, pp. 216–221.

[17] A. Camacho and J. Harris, "A sawtooth waveform inspired pitch estimator for speech and music.," *The Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008.

[18] E. Benetos and S. Dixon, "A shift-invariant latent variable model for automatic music transcription," *Computer Music Journal*, vol. 36, no. 4, pp. 81–94, 2012.

[19] H. Kirchhoff, S. Dixon, and A. Klapuri, "Derivation of update equations for a source-filter model based on beta-divergence," Tech. Rep., Queen Mary University of London, London, 2012.

[20] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Music genre database and musical instrument sound database," in *4th International Society for Music Information Retrieval Conference (ISMIR)*, Baltimore, Maryland, USA, 2003, pp. 229–230.

[21] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detection," in *International Conference on Digital Audio Effects (DAFx)*, Maynooth, Ireland, 2013, pp. 1–7.

[22] C. Schörkhuber and A. Klapuri, "Constant-Q transform toolbox for music processing," in *7th Sound and Music Computing Conference*, Barcelona, Spain, 2010.

[23] G. Bloothooft and R. Plomp, "Spectral analysis of sung vowels. III. Characteristics of singers and modes of singing," *The Journal of the Acoustical Society of America*, vol. 79, no. 3, pp. 852–864, Mar. 1986.

ONSET TIME ESTIMATION FOR THE EXPONENTIALLY DAMPED SINUSOIDS ANALYSIS OF PERCUSSIVE SOUNDS

Bertrand Scherrer, Philippe Depalle

SPCL/CIRMMT
McGill University
Montréal, QC, Canada
bertrand.scherrer@mail.mcgill.ca

ABSTRACT

Exponentially damped sinusoids (EDS) model-based analysis of sound signals often requires a precise estimation of initial amplitudes and phases of the components found in the sound, on top of a good estimation of their frequencies and damping. This can be of the utmost importance in many applications such as high-quality re-synthesis or identification of structural properties of sound generators (*e.g.* a physical coupling of vibrating devices). Therefore, in those specific applications, an accurate estimation of the onset time is required. In this paper we present a two-step onset time estimation procedure designed for that purpose. It consists of a "rough" estimation using an STFT-based method followed by a time-domain method to "refine" the previous results. Tests carried out on synthetic signals show that it is possible to estimate onset times with errors as small as 0.2ms. These tests also confirm that operating first in the frequency domain and then in the time domain allows to reach a better resolution *vs.* speed compromise than using only one frequency-based or one time-based onset detection method. Finally, experiments on real sounds (plucked strings and actual percussions) illustrate how well this method performs in more realistic situations.

1. INTRODUCTION

In this paper, the focus is set on percussive sounds that can be pitched (*e.g.* guitar, piano or glockenspiel sounds). Such sounds, and sometimes even non-pitched percussive sounds (see [1]), are well represented using a signal model of the form:

$$x[n] = \sum_{m=1}^{M} \left(\sum_{k=1}^{K_m} a_{m,k} e^{j\phi_{m,k}} z_{m,k}^{n} \right) u[n - n_m] + w[n] \quad (1)$$

where $x[n]$ is the real sound signal; M is the number of transient events in the sound; $u[n]$ is the unit step function; n_m is the sample marking the start of transient m; K_m is the order of the model for transient m; $z_{m,k} = e^{(\delta_{m,k} + j\omega_{m,k})}$ is its k^{th} pole with radian frequency $\omega_{m,k}$ and damping factor $\delta_{m,k}$; $a_{m,k}$ and $\phi_{m,k}$ are the initial amplitude and phase of $z_{m,k}$, respectively; and $w[n]$ represents the stochastic component of the signal.

The estimation of the parameters of this type of model has been extensively researched [1, 2, 3, 4]. In the scope of this study, however, it is relevant to note that it is necessary to have a good estimate of the time parameter n_m for those parametric methods to yield the best possible results. For example, when choosing on which segment to perform EDS modelling, it is important that the transient be close to the beginning of the segment to avoid pre-echo artifacts [1]. Also, one can desire a precise knowledge of the "initial" amplitudes and phases of the EDS of the model : for example, in [5], initial amplitudes and phases of the components forming a partial of a guitar's string sound are central to the estimation of the angle at which a guitar string is released.

In this paper we present an onset detection scheme designed to obtain transients with very fine time resolution in a reasonable amount of time. To borrow the terminology introduced in [6], this method makes use of two different detection functions one after the other.[1] More specifically, the detection functions used are based on frequency- and time-domain energy features rather than on probabilistic models [6] or a combination of the two [8].

The goal of this paper is to show that the sequential application of two simple detection functions leads to significant improvements over the results achievable using these two functions in isolation. Although percussive sounds might be seen as "easy" sounds to detect onsets on, and despite the fact that methods based on variations of the energy of the signal to segment audio have been used for a very long time (*e.g.* [9]), the valuable contribution of this paper lies in that the method proposed is of prime interest in the particular context of the analysis of percussive sounds using exponentially damped sinusoids. Indeed, it allows to obtain a finer time resolution than well known methods such as spectral flux [10] with acceptable computational demands.

The onset detection method is presented in Sec. 2. Experiments on synthetic and real percussive musical sounds are carried out in Sec. 3 and Sec. 4, respectively. The conclusions drawn from these experiments, as well as potential extensions are discussed in Sec. 5.

2. ONSET TIME ESTIMATION PROCEDURE

The onset time estimation procedure studied in this paper is comprised of two steps: a first onset determination over the whole duration of the signal based on its STFT with a "rough" time resolution; the second step involves another onset detection with finer time resolution around each "rough" onset.

The "rough" onset detection starts by computing the STFT of the signal $x[n]$ as follows:

$$X[l, b] = \sum_{n=0}^{N-1} w[n].x[n + lH].e^{j2\pi nb/N} \text{ with } b \in [0; N-1] \quad (2)$$

where $w[n]$ is a Hanning analysis window [11], l is the STFT frame index, b is the FFT bin index, N is the FFT size, and H

[1] A similar, though not identical, approach can be found in [7, p. 42].

is the hop size. The frequency-domain detection function $d_f[l]$ is given by:

$$d_f[l] = \sqrt{\sum_{b=0}^{N/2} \big(|X[l,b]| - |X[l-1,b]| \big)^2} \tag{3}$$

where $|X|$ is the modulus of the complex number X. In essence, $d_f[l]$ measures how different two consecutive STFT frames are from each other using an L_2-norm.[2] It is clear that the maximum time resolution is limited by H and depends on N.

As "rough" onsets are often late (see Sec. 3), the "refining" stage of onset detection is performed on smaller data segments starting a few hop sizes before each "rough" onset. Another detection function is put to use at this stage: a time-domain method, based on the variations of the energy of the signal [1]. More specifically, for a given sample index n, the power of the signal is computed over $\big[x[n-J]; x[n-1]\big]$ (a "backward" window) and over $\big[x[n+1]; x[n+J]\big]$ (a "forward" window), where J is an integer number of samples. The detection function $d_t[n]$ is then computed as follows:

$$d_t[n] = \frac{1}{J} \log \left(\frac{\sum_{m=n+1}^{n+J} x^2[m]}{\sum_{l=n-J}^{n-1} x^2[l] + \upsilon} \right) \cdot \sum_{k=n+1}^{n+J} x^2[k]. \tag{4}$$

The term in the $\log$ function is included in order to emphasize increases in energy. The variable υ is included in Eq. 4 as a regularization factor (*i.e.* to prevent divisions by zero).

The time offsets implied by the definitions of $X[l,b]$, $d_f[n]$ and $d_t[n]$[3] are compensated for in practice in order to be able to perform proper comparisons.

As suggested in [6], the detection function is first zero-meaned, normalized and finally smoothed using a normalized derivative filter:

$$\mathcal{H}(z) = \frac{1-\gamma}{1-\gamma z^{-1}}, \tag{5}$$

Peaks are identified on the smoothed detection function using parabolic interpolation and considering an extremum to be a peak if it is α dB above the neighbouring minima [12, p. 42]. Once peaks are detected, an adaptive thresholding scheme [6] is used: only the peaks with amplitude higher than τ_{ad} are considered to be onsets. The expression of τ_{ad} is as follows:

$$\tau_{\text{ad}} = \tau + \ell d_{\text{median,p}}, \tag{6}$$

where τ is an absolute threshold, $d_{\text{median,p}}$ is the normalized and smoothed detection function passed through a median filter of order p, and ℓ controls how much the absolute threshold is affected by $d_{\text{median,p}}$.

After both the "rough" and "refined" onset detection steps, a pruning mechanism is included to remove repeated onsets. That is, each onset is compared to neighbouring onsets within a given time interval (notated I in the rest of the paper). Then, in this interval, only the onset corresponding to the highest value of the detection function is kept.

[2]Note that only half of the bins are considered since $x[n] \in \mathbb{R}$.

[3]$N/2$ in Eq. 2, $(N-H)/2$ in Eq. 3 and $J/2$ in Eq. 4

Table 1: *Parameters of the onset estimation procedure used in Sec. 3.1. Their definition is found in Sec. 2. A sampling rate of 44.1kHz is used and the signals analyzed are such that* $|x[n]| < 1$.

Rough onsets	**Refined onsets**
N : 2048	J : 200
H : 1024	υ : 10^{-4}
γ : 0.3	γ : 0.1
τ : 0.1	τ : 0.5
p : 5	p : 5
ℓ : 0.5	ℓ : 0.5
α : 6dB	α : 6dB
I : 900	I : 900

3. EXPERIMENTS ON SYNTHETIC SOUNDS

In this section, the two-step onset detection procedure presented in Sec. 2 is evaluated in several experiments on synthetic pitched percussive sounds. These sounds reproduce the basic signal structure of sounds generated by a guitar or a piano. In other words, there are several modes, or poles $z_{m,k}$, inside a given string partial due to the coupling of strings through the bridge of the instrument [13, 5]. In this paper, synthetic signals are composed of EDS components grouped in pairs with very similar frequencies and quite different damping factors: 100 different sounds, lasting 1 s each (with $F_s = 44.1$ kHz), are synthesized with parameters randomly chosen within specific ranges as follows. Onset times are chosen within the first 0.5 s of sound segments. The number of partials, K, is such that $K \in [1;6]$. In order to approach the ideal structure of guitar sounds, partials are chosen to be strictly harmonic, with a fundamental frequency between 82 Hz and 900 Hz. Moreover, amplitudes of harmonics are weighted with a formula of the type $1/k^2$ to mimic the expected spectral slope for the displacement of an ideally plucked string with rigid terminations [14]. Each harmonic consists of two EDS, with slightly different frequencies, damping factors, amplitudes and phases. Only the real part of each signal is kept, so that the phasors in Eq. (1) are replaced by *cos* functions. Finally, some noise is added to the signal in such a way that the ratio of the maximum value of the signal squared to the power of the noise is 30dB. These sounds, as well as the real sounds used in Sec. 4 can be downloaded from this paper's companion webpage.[4]

3.1. Comparing "rough" and "refined" estimations

The two-step onset detection is applied to these sounds with the parameters found in Table 1. The choice of N was motivated by a desire to ensure that spectral components would be separate enough that a rapid temporal variation would clearly translate in energy spreading in more bins. The "refined" onset detection is performed on a portion of the sound starting 5 hop sizes before the onset detected at the "rough" stage and ending 1 hop size after.

Fig. 1 depicts the distribution of errors between the true onset, n_0, and its estimate, $\hat{n}_0$ after both "rough" and "refined" onset estimations, over the 100 sounds of the experiment. From these plots, it is clear that the refining step improves the performance of the onset detection: the median of the error (red line in Fig. 1)

[4]http://www.music.mcgill.ca/~scherrer/dafx14/

goes from 500 samples after the "rough" onset detection to 0 after the "refined" onset detection stage. Also, the spread of the errors for the "refined" estimation is dramatically reduced compared to the "rough" onset stage. In particular, Fig. 2a shows that 75 % of the error lies between 0 and 3 sample of the target (0-0.06ms at 44.1kHz) for the "refined" onset detection compared to the [250;700] sample range (6-16ms) for the "rough" estimation. It also appears that most of the onsets detected at the "rough" stage were late compared to the actual onset time ($e_{n_0} < 0$). This justifies the choice to look for "refined" onsets 5 hop sizes before the estimated onset and 1 hop size after during the refinement stage.

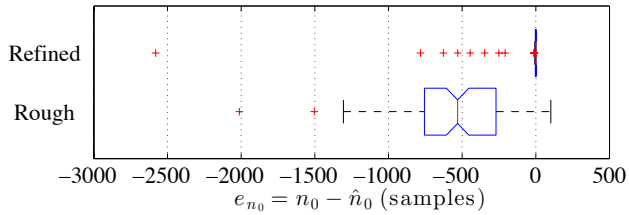


Figure 1: *Distribution of errors made on the estimation of the onset time (n_0) for synthetic signals with randomly chosen parameters. The plot labelled "Rough" corresponds to the distribution of errors made at the "rough" estimation stage, while the plot labelled "Refined" represents the error distribution after refinement.*

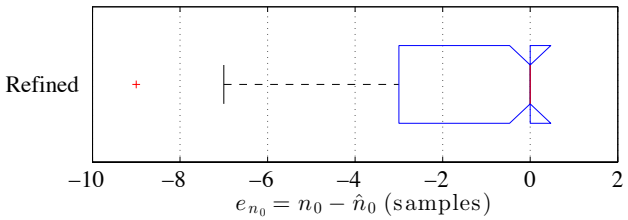
3.2. Testing the robustness to "soft" onsets

After closer inspection of Fig. 1, it appears the outliers (red crosses) in the "refined" stage correspond to signals where the waveform of the signal has a smoother start form 0 compared to the other sounds; signals with "soft" onsets. Thus, another experiment is carried out to better quantify the performance of the method on such sounds.

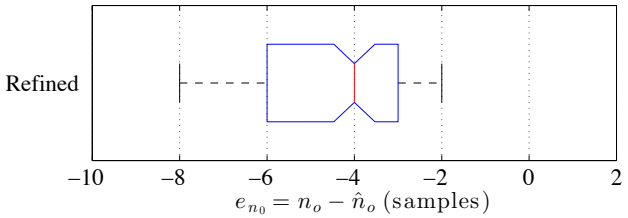
To that end, another set of synthetic sounds with the same general structure as those studied in Sec.3.1 is generated. The difference lies in the fact that the phases of the EDS's are now all set to $\pi/2$. The results of this experiment are presented in Fig. 3. The advantage of using the two-step method is still clear, judging from the drastic improvement of the median of the error between "rough" and "refined" steps. When comparing the "refined" onset detection in this experiment and in the previous experiment, as in Fig. 2b, one can note a very slight degradation of performances when all phases are set to $\pi/2$. For example, there are slightly more outliers at the "refined" stage in the case where all the phases are set to $\pi/2$ than when the phases are all random. Also, as shown in Fig. 2b, when all phases are set to $\pi/2$, there is a small increase in the error: a median of -4 instead of 0 for the random phases case, and now 75% of the error is within [-3;-5] samples ([0.07ms-0.11ms] at 44.1kHz). Despite this slight degradation in this adverse scenario, the performance of the method is still very satisfying in terms of time resolution.

3.3. Computational time *vs.* onset time error

This last experiment on synthetic sounds aims at characterizing how the two-pass onset estimation procedure compares to onset estimations using only $d_f[l]$ (*cf.* Eq.3), or only $d_t[n]$ (*cf.* Eq.4). The sounds analyzed are the same as those used in Sec. 3.1. The



(a) *Random phases*



(b) *Phases all set to $\pi/2$*

Figure 2: *Distributions of the errors on the detected onset time after refinement, a) for the case of partials with random phases, b) with phases set to $\pi/2$.*

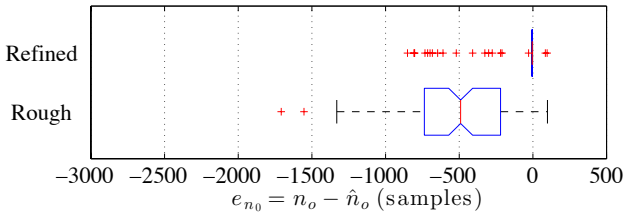


Figure 3: *Distribution of errors made on the estimation of the onset time, in the case where all partials have a phase of $\pi/2$. The labels "Rough" and "Refined" refer to the first and second pass of the onset detection.*

parameters for the STFT-based method are the same as those in Table 1 for the "rough" estimation, except that the hop size has been varied between 1024 and 256 samples (so between 23ms and 6ms at 44.1kHz). Hop sizes smaller than 256 samples yield computation speeds higher than that of the two-pass estimation procedure so they are not included in the plot. The parameters for the time-based method are identical to those in Table 1.

Fig.4 depicts the median value of the computational time[5], t_{comp}, versus the absolute value of the error on the onset,[6] $|e_{n_0}|$, for the different analysis scenarios. The two-pass onset estimation is represented by a white disk, while the time-based method is symbolized by a black triangle. Finally, the several instances of the STFT-based analysis are depicted using grey squares (one for each hop size used).

The ideal method would lie in the leftmost bottom corner of the plot as it would mean that this method is very quick and has no error. With this in mind, it is then clear that the two-pass onset estimation procedure studied here outperforms the time-based method in terms of speed, by a factor of 10. It also performs better

[5] in Matlab R2012b on a MacBook with a 2GHz processor, 4GB of RAM.

[6] As e_{n_0} can be negative.

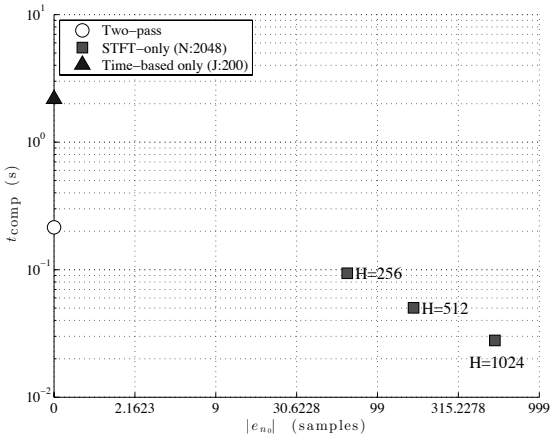
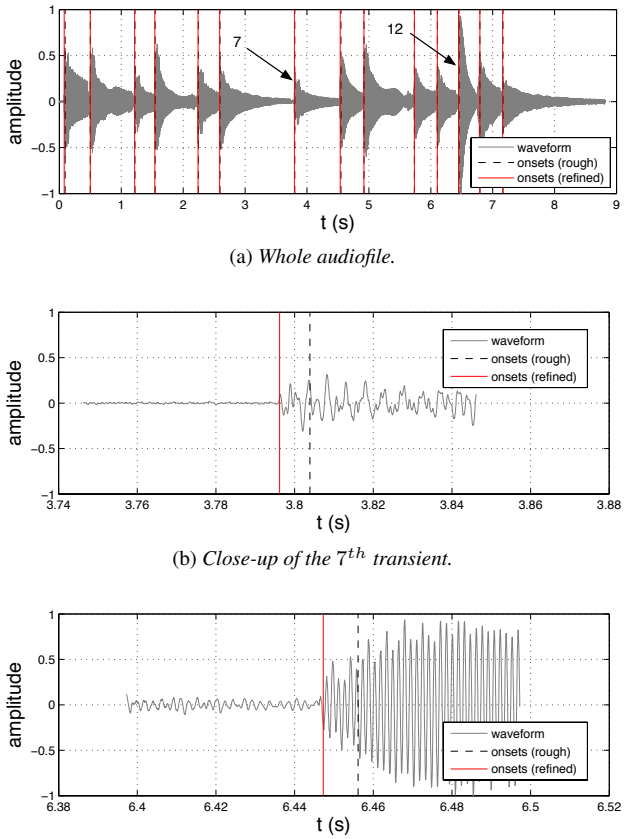


Figure 4: *Median value of t_{comp} the computation time vs. the median value of $|e_{n_0}| = |n_0 - \hat{n}_0|$ for different estimation scenarios: the two-pass procedure (white disk), the time-based method (black triangle), the STFT-based method with varying hop sizes, H (grey squares). Logarithmic scales are used on both axes.*



(a) *Whole audiofile.*

(b) *Close-up of the 7^{th} transient.*

(c) *Close-up of the 12^{th} transient.*

Figure 5: *Whole audiofile with the 7^{th} and 12^{th} transients singled out. The solid red lines are the "refined" onsets, while the dashed black lines are the onsets found at the "rough" onset detection step.* Close-ups on transients indicated by arrows in a) are found in b) and c).

than the STFT-based method only in terms of error on the onset, since its median is 0 compared to a minimum error of about 50 for $H = 256$. The presence of errors smaller than H is due to the parabolic interpolation performed after smoothing of the detection function $d_f[l]$.

This experiment also confirms the importance of choosing H properly as it seems to significantly reduce the error on the onset estimation: if $H = 256$, the error is about 7 times less than if $H = 1024$ while it only takes about 4 times more to compute. Even for $H = 256$, however, the error is still several orders of magnitude greater than using the two-pass onset estimation procedure.

4. EXPERIMENTS ON MUSICAL SOUNDS

In this section the two-pass onset detection scheme is used on more realistic signals, that do not exactly meet the model of Eq. 1. Sec. 4.1 presents a qualitative experiment that illustrates how the method performs on pitched percussive sounds (a monophonic recording of classical guitar) and on non-pitched percussive sounds (a monophonic recording of castanets). Sec. 4.2 discusses a quantitative evaluation of the method introduced in this paper on a small set of annotated guitar, piano and castanet sounds.

4.1. Qualitative experiment

An example of this two-pass onset detection method applied to a monophonic guitar recording is depicted in Fig. 5. In Fig. 5a, the waveform of the whole audio file is represented and includes both "rough" and "refined" onset times (dashed black lines and solid red lines, respectively). Fig. 5b and Fig. 5c feature close-ups of the 7th and 12th notes in Fig. 5a, respectively. Theses two examples were chosen to show the type of refinement on the "rough" onset detection that this method offers. One may note that, although on the whole signal the refinement may seem minor, the cases shown in Fig. 5b and Fig. 5c represent in fact a substantial improvement especially for analysis tasks requiring precise knowledge of the start of the transient. These two examples also indicate that the two-pass onset detection scheme is relevant for both isolated notes (Fig. 5b) and notes played closer together (Fig. 5c). The parameters used to obtain these results are listed in Table 2.

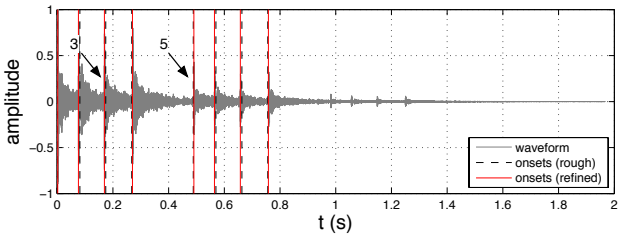
In order to demonstrate that the onset time estimation approach discussed here can also be applied to non-pitched percussive signals another test was conducted. It uses castanet sounds, often chosen as a test case in audio encoding experiments (*e.g.* [1]) as they are quite short with wide energy variations. Fig. 6 depicts both the rough and refined onsets for a castanet recording analyzed using the parameters in Table 3. Fig. 6 depicts two different examples of refinement where the rough onset was late (Fig. 6b) and where the rough onset was slightly too early (Fig. 6c). This shows that the method we propose is also well adapted to signals with very short and sharp transients and could be valuable in audio coding applications based on EDS modeling or other applications requiring very accurate estimations of transient times.

Table 2: *Values of the analysis for the guitar recording in Fig. 5a. The analysis parameters are those presented in Sec. 2 with a sampling rate of 44.1kHz.*

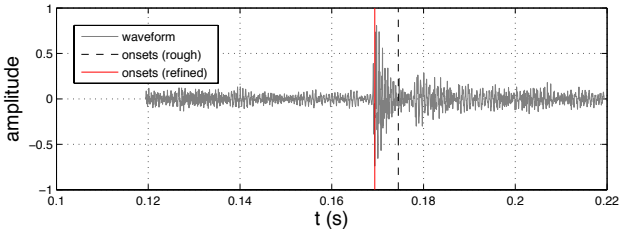
Rough onsets	**Refined onsets**
N : 1024	J : 400
H : 512	υ : 10^{-4}
γ : 0.3	γ : 0.1
τ : 0.15	τ : 0.5
p : 5	p : 5
ℓ : 0.5	ℓ : 0.5
α : 6dB	α : 6dB
I : 2205	I : 900

Table 3: *Values of the analysis for the castanet recording in Fig. 6. The analysis parameters are those presented in Sec. 2.*

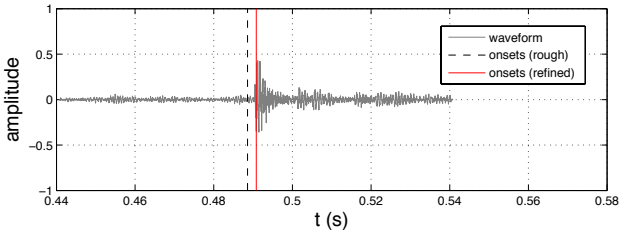
Rough onsets	**Refined onsets**
N : 1024	J : 400
H : 512	υ : 10^{-4}
γ : 0.3	γ : 0.1
τ : 0.1	τ : 0.5
p : 5	p : 5
ℓ : 0.5	ℓ : 0.5
α : 3dB	α : 6dB
I : 2205	I : 900



(a) *Whole audiofile with the 3^{rd} and 5^{th} transients indicated by arrows.*



(b) *Close-up of the 3^{rd} transient.*



(c) *Close-up of the 5^{th} transient.*

Figure 6: *Onset detection on a castanet signal. Solid red lines indicate "refined" onsets, while dashed black lines mark onsets found at the "rough" onset detection step. In a), the results over the whole file are depicted, while b) and c) illustrate how the initial onset time was refined in two particular instances.*

4.2. Quantitative experiment

A preliminary quantitative evaluation has also been carried out to complement the qualitative observations made on the two previous examples. The goal of this experiment is to evaluate the improvement resulting from adding a refining stage. The two-pass onset detection method is thus evaluated on a small set of annotated sounds. This set is comprised of the guitar and castanet sounds previously studied, as well as two non distorted guitar sounds and one piano sound from a small database used for the MIREX2005 Onset Detection task[7]; they are referred to as guitar2, guitar3 and piano1 in the rest of this paper. The annotation of the guitar and castanet sounds was done using the software accompanying [15].

The approach outlined in the instructions of the MIREX Audio Onset Detection task was implemented[8] to perform the evaluation. In particular, the F-measure [16] was used as the main evaluation metric:

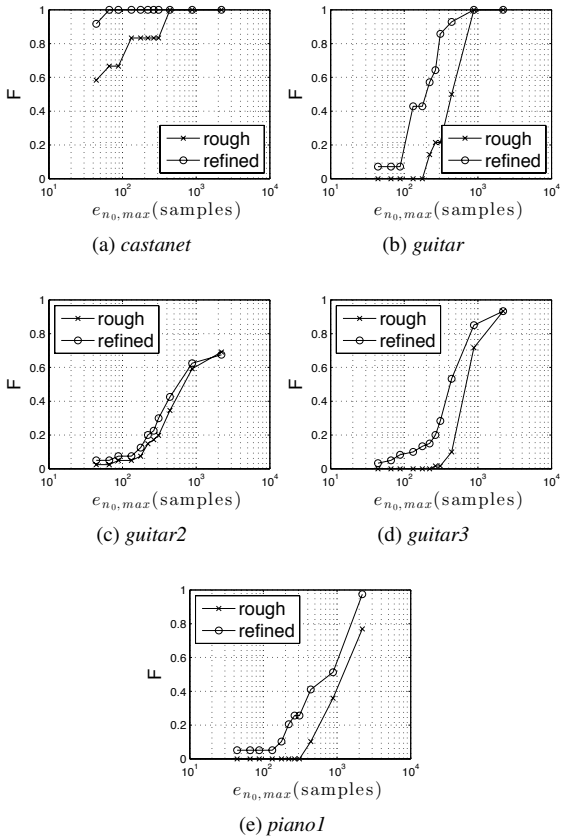
$$\begin{aligned} F &= 2\frac{P.R}{P+R} \\ \text{with } P &= \frac{n_{TP}}{n_{TP}+n_{FP}} \\ \text{and } R &= \frac{n_{TP}}{n_{TP}+n_{FN}} \end{aligned} \tag{7}$$

where P is the *precision*, R is the *recall* and n_{TP}, n_{FP}, n_{FN} are the numbers of *true positive*, *false positive* and *false negative* detections, respectively. In this paper, for each labelled onset, a *true positive* detection is counted if at least one detected onset is within a certain onset time tolerance, $e_{n_0,max}$, of that onset. When there are no *false positive* or *false negative* detections, $P = 1$ and $R = 1$, so $F = 1$ (see Eq. 7). Conversely, when there are no *true positive* detections, $F = 0$. Thus, for a given sound and a given set of detected onsets, the value of F will change depending on the chosen $e_{n_0,max}$. It is often taken to be equivalent to 50 ms [6] for musical tasks. Since the goal of the method presented in this paper is to provide a fine time resolution for the estimation of EDS parameters, $e_{n_0,max}$ values ranging between 44 samples ($\simeq$1 ms at 44.1 kHz) and 2205 samples (50 ms at 44.1 kHz) were used.

Fig. 7 depicts the evolution of the F-measure versus $e_{n_0,max}$, the onset time tolerance. Analysis parameters were found manually for all sounds so that they ensured most (if not all) onsets were found at the rough estimation stage. The tuning of analysis parameters for each sound is appropriate here as this experiment aims at quantifying the effect of the refinement of the onset detection, from the best possible rough onset estimation. The complete list of parameters used can be found on this paper's companion website. There are 5 subplots, one for each recording studied. In

[7] http://www.tsi.telecom-paristech.fr/aao/en/2011/07/13/onset_leveau-a-database-for-onset-detection/

[8] http://www.music-ir.org/mirex/wiki/2014:Audio_Onset_Detection



(a) *castanet* (b) *guitar*

(c) *guitar2* (d) *guitar3*

(e) *piano1*

Figure 7: *F-measure versus $e_{n_0,max}$ for various annotated sounds.*

each subplot, the F-measure obtained after rough (crosses) and refined estimation (circles) is represented as a function of $e_{n_0,max}$. As expected, the general trend for both rough and refined estimation is that F increases as the tolerance increases. In all plots of Fig. 7, it is clear that adding the refinement stage helps increase F as $e_{n_0,max}$ becomes smaller. The refinement stage does not seem to increase F significantly for a $e_{n_0,max}$ of 882 samples (20 ms) or 2205 samples (50 ms), except for the piano1 recording in Fig. 7e. For all recordings, however, it appears that one benefits most from the refinement stage for tolerances between 176 samples (4 ms) and 441 samples (10 ms).

The results obtained for the castanet recording (Fig. 7a) are much better for both stages of the estimation than all the other sounds. This is most likely due to the fact that this sound was comprised of well defined bursts of energy that do not really overlap in time whereas all the other sounds involve a fair amount of polyphony, where loud notes can overshadow softer new notes. In Fig. 7c, the refinement stage and the rough detections yield approximately similar F values, with a slight advantage to the refined detection. The reason for this is not entirely clear for now but after inspecting the results of the rough onset time estimations for all 5 sounds, it appears that the results obtained for that particular sound are the least satisfying of all five.

5. CONCLUSION

This paper presents and evaluates an onset time estimation procedure specifically designed for applications when a *very* precise onset time estimation is required (of the order of a few tenth of a millisecond). First, a frequency-domain onset estimation is performed. Then around each of those onsets a time-domain onset estimation is used in order to "refine" the onset time estimation.

Experiments on synthetic signals mimicking the structure of guitar sounds show that with this two-pass onset estimation procedure it is possible to obtain onset estimates with errors of at most 0.1ms, 75% of the time. It is also demonstrated that using this two-step method outperforms using either the STFT-based method or the time-based method in isolation. Indeed, Fig.4 shows that the two-pass procedure allows very small error with a small computation time.

Qualitative and quantitative tests on musical recordings help evaluate the performance of the method in more realistic conditions. It is indeed shown that adding the refining stage after the rough estimation helps improve the onset time estimation (increase the F-measure of that detection) when the time tolerance is between 4 ms and 10 ms. The difference with the synthetic case (where onsets were found within 0.1 ms) most likely comes from a combination of factors: real sounds do not conform exactly to the signal model of the synthetic sounds; onsets were manually annotated; the sounds used for testing were polyphonic.

As future work, we plan to investigate the nature of the outliers in the experiment of Sec. 3.2, itself designed to study the outliers of the study in Sec.3.1. Also, as hinted by the results in Fig.4, the choice of parameters of the STFT-based method can have important consequences on the error on the onset and on the computation time. A more systematic evaluation of the effect of N and H on the performances of the STFT-based method would help explain those observations. Another avenue for future work would be to try other methods for the "rough" detection that may be more robust to polyphony. Also, if one were to use this onset detection method on large datasets, and as with all onset time estimation methods, the automatic determination of parameters would be an interesting avenue for improvement.

6. REFERENCES

[1] R. Boyer and K. Abed-Meraim, "Audio modeling based on delayed sinusoids," *IEEE Trans. Speech Audio Process.*, vol. 12, no. 2, pp. 110–120, March 2004.

[2] K. Steiglitz and L. E. McBride, "A technique for the identification of linear systems," *IEEE Trans. Automatic Control*, vol. 10, pp. 461–4, 1965.

[3] R. Roy, A. Paulraj, and T. Kailath, "ESPRIT – a subspace rotation approach to estimation of parameters of cisoids in noise," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 5, pp. 1340–2, October 1986.

[4] R. Badeau, *Méthodes à Haute Résolution pour l'Estimation et le Suivi de Sinusoïdes Modulées. Application aux Signaux de Musique*, Ph.D. thesis, ENST, Paris, 2005.

[5] B. Scherrer and P. Depalle, "Extracting the angle of release from guitar tones: preliminary results," in *Proc. Acoustics 2012*, Nantes, France, 2012.

[6] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music

signals," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 5, pp. 1035–1047, September 2005.

[7] N. P. Donaldson, "Extending the phase vocoder with damped sinusoid atomic decomposition of transients," M.S. thesis, McGill University, June 2011.

[8] F. Eyben, S. Böck, and B. Schuller, "Universal onset detection with bidirectional long short-term memory neural networks," in *Proc. 11th ISMIR*, 2010.

[9] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice Hall, Upper Saddle River, NJ, 1978.

[10] S. Dixon, "Onset detection revisited," in *Proc. 9th DAFx*, 2006.

[11] Frederic J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proc. IEEE*, vol. 66, no. 1, pp. 51–84, January 1978.

[12] X. Serra, *A system for sound analysis / transformation / synthesis based on a deterministic plus stochastic decomposition*, Ph.D. thesis, Stanford University, October 1989.

[13] G. Weinreich, "Coupled piano strings," *J. Acoust. Soc. Am.*, vol. 62, no. 6, pp. 1474–84, 1977.

[14] N. H. Fletcher and T. D. Rossing, *The physics of musical instruments*, Springer, 2nd edition, 1998.

[15] Pierre Leveau, Laurent Daudet, and Gaël Richard, "Methodology and tools for the evaluation of automatic onset detection algorithms in music," in *Proc.4th ISMIR*, 2004.

[16] C. J. van Rijsbergen, *Information Retrieval*, Butterworth, London, 1979.

AUTOMATIC TABLATURE TRANSCRIPTION OF ELECTRIC GUITAR RECORDINGS BY ESTIMATION OF SCORE- AND INSTRUMENT-RELATED PARAMETERS

Christian Kehling
Fraunhofer IDMT,
Ilmenau, Germany

*Jakob Abeßer**
Fraunhofer IDMT,
Ilmenau, Germany
jakob.abesser@idmt.fraunhofer.de

Christian Dittmar
Fraunhofer IDMT,
Ilmenau, Germany

Gerald Schuller
Technical University of Ilmenau,
Ilmenau, Germany

ABSTRACT

In this paper we present a novel algorithm for automatic analysis, transcription, and parameter extraction from isolated polyphonic guitar recordings. In addition to general score-related information such as note onset, duration, and pitch, instrument-specific information such as the plucked string, the applied plucking and expression styles are retrieved automatically. For this purpose, we adapted several state-of-the-art approaches for onset and offset detection, multipitch estimation, string estimation, feature extraction, and multi-class classification. Furthermore we investigated a robust partial tracking algorithm with respect to inharmonicity, an extensive extraction of novel and known audio features as well as the exploitation of instrument-based knowledge in the form of plausability filtering to obtain more reliable prediction. Our system achieved very high accuracy values of 98 % for onset and offset detection as well as multipitch estimation. For the instrument-related parameters, the proposed algorithm also showed very good performance with accuracy values of 82 % for the string number, 93 % for the plucking style, and 83 % for the expression style.

Index Terms - playing techniques, plucking style, expression style, multiple fundamental frequency estimation, string classification, fretboard position, fingering, electric guitar, inharmonicity coefficient, tablature

1. INTRODUCTION

Audio recordings of plucked string instruments can be described as a sequence of acoustic note events having a characteristic harmonic structure, which strongly depends on the type of instrument and the playing techniques are being used. Scores and tablatures are common notation formats to store the most important parameters to describe each played note. In order to automatically generate such notations from a recorded audio signal, these parameters must be estimated beforehand.

As will be detailed in Section 3, various publications in the field of Music Information Retrieval (MIR) focused on the automatic extraction of either score-related parameters such as onset, offset, and pitch (a tasks that is commonly referred to as automatic music transcription), or instrument-related parameters such as the applied playing techniques and fretboard positions on string instruments. This work expands these approaches by fusing single parameter estimation algorithms to an overall transcription framework, which is tailored towards instrument-specific properties of the electric guitar.

The proposed automatic transcription algorithm extracts essential information about the recorded music piece that allows comparison with a ground truth notation. Hence possible application scenarios are music education software such as Songs2See[1] and BandFuse[2] as well as music games such as RockSmith[3]. Furthermore, the transcription algorithm can be applied for a detailed expressive performance analysis that provides information about artist-specific peculiarities related to micro-timing or to the preferred playing techniques. In combination with a sound synthesis algorithm, an efficient parametric audio coding model with very low bit rates can be realized due to the very compact symbolic representation of the instrument recording.

The paper is structured as follows. First, Section 2 provides important basics of the guitar sound generation. After a review of the related work in Section 3, we explain the proposed transcription algorithm in detail in Section 4. Finally, Section 5 describes all evaluation experiments and Section 6 summarizes this work.

2. GUITAR SPECIFIC BACKGROUND

The most influential parts of an electric guitar are the strings, the magnetic pick-up, and the passive electrical tone control. Body resonances only have a minor influence on the resulting tone and will not be taken into account here. The guitar strings determine the basic sound since when vibrating, they are the primary sound source. The sound is mainly affected by the string material, tension, and stiffness. These features manifest primarily in frequency shifts of partial vibrations also known as the effect of inharmonicity [1]. The standard arrangement and tuning of a 6-string guitar with corresponding fundamental frequencies and MIDI number specifications is given in Table 1. Electromagnetic pick-ups

* All correspondance should be adressed to this author.

[1] http://www.songs2see.com/
[2] http://bandfuse.com/
[3] http://rocksmith.ubi.com/

capture the existing vibrations depending on their position on the instrument neck and the corresponding possible displacement of partials. Their technical specifications determine the transfer function which is commonly approximated by a second order low pass filter with a cut-off frequency in the range from 2 to 5 kHz. The same applies to the subsequent tone control of the guitar, which can be represented by a first order low pass filter. Both can be combined to an overall transfer function.

Table 1: *Standard Tuning of Guitar Strings.*

String Number	Standard Tuning	Fundamental Frequency	MIDI Number
1	E2	82.4 Hz	40
2	A2	110.0 Hz	45
3	D3	146.8 Hz	50
4	G3	196.0 Hz	55
5	B3	246.9 Hz	59
6	E4	329.6 Hz	64

Another important means of tone manipulation is the playing technique applied by the musician. In this work we distinguish 3 different plucking styles—finger style, picked, and muted—as well as and 5 expression styles—bending, slide, vibrato, harmonics, and dead notes—executed with the fingering hand in addition to non-decorated, normal expression style 2. See [2] for a detailed description of the playing techniques.

Table 2: *Playing Techniques.*

Plucking Style	Expression Style
finger style (F)	bending (BE)
picked (P)	slide (SL)
muted (M)	vibrato (VI)
	harmonics (HA)
	dead notes (DN)

Besides common music notation, a widespread method of notating guitar music is the tablature. By indicating the fret and string numbers to be used, it provides an alternative and more intuitive view of the played score. Figure 1 shows an example of a score and corresponding tablature.

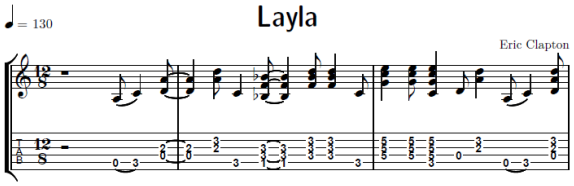


Figure 1: *Excerpt of the score and tablature representation of an interpretation from the Song Layla written by Eric Clapton [3].*

In tablature notation every drawn line symbolizes a string of the instrument, typically the lowest string corresponds to the bottom line. The numbers written on the single lines represent the used fret, where the fret number 0 corresponds to the open string.

3. PREVIOUS WORK

As will be discussed in Section 4, various Music Information Retrieval (MIR) tasks are relevant for our work. In the past, several authors focussed on *monophonic* guitar recordings, which contain isolated notes or simple melodies. The task of *onset detection*, i.e. the detection of note start times in audio recordings, was investigated in many publications. An overview over state-of-the-art methods can be found for instance in [4]. *Multipitch estimation*, i.e., the transcription of multiple simultaneously sounding notes, is up to this day a very challenging task to be performed in an automated manner [5]. In our paper, we build upon the method proposed by Fuentes et al. in [6]. For time-frequency-representation we use a spectral magnitude reassignment based on the instantaneous frequency as proposed in [7]. Fiss and Kwasinski proposed a multipitch estimation algorithm tailored towards the guitar in [8] by exploiting knowledge about the string tuning and pitch range of the instrument. Similarly, Yazawa et al. combine multipitch estimation with three constraints related to the guitar fretboard geometry to improve the transcription results [9]. In [3], an algorithm capable of real-time guitar string detection is presented, which is also the base for our work. Particularly for guitar chords, Barbancho et al. automatically classified between 330 different fingering configuration for three-voiced and four-voiced guitar chords by combining a multipitch estimation algorithm and a statistical modeling using a Hidden Markov Model (HMM) [10].

In addition to the score-based parametrization and the estimation of the fretboard position, we aim to estimate the *playing technique* that was used on the guitar to play each note. We showed in previous work, that the estimation of playing techniques [2] for electric bass guitar, which shares similar playing techniques with the electric guitar, can be performed from isolated note recordings with a high accuracy using a combination of audio features and machine learning techniques. Various publications analyzed guitar recordings with focus on playing techniques that modulate the fundamental frequency such as *vibrato* [11], *bending* [12], or *slides* [13, 12]. Other guitar playing techniques that were investigated in the literature are *slide*, *hammer-on*, and *pull-off* [13, 12]. A broader overview over state-of-the-art methods for the transcription and instrument-related parameters from string instrument recordings can be found in [14] and [15].

4. PROPOSED METHOD

4.1. Problem Formulation

The goal of this work is to develop an analysis algorithm, that extracts all essential parameters necessary for the automatic creation of guitar scores. Therefore, a robust event separation based on onset detection methods has to be implemented. Afterwards, the note duration and pitch must be extracted. In the next step, both the plucking and expression styles (see Table 2) as well as the string number must be estimated using feature extraction and subsequent classification methods. Finally, by using knowledge about the instrument string tuning, the fret position can be derived for each note.

The transcription parameters can be verified and corrected by exploiting knowledge about the instrument construction and physical limitations of the guitar player. Hence, a further goal is to develop adaptive algorithms that satisfy these conditions. The final model should be able to store the extracted parameters and

to generate a guitar tablature and score completely automatically based on a given polyphonic, monotimbral electric guitar recording. In this work exclusively clean guitar signals without any prior audio effect processing are considered. According to the diagram in Figure 2, the following sections will describe each step in detail.

4.2. Onset Detection

The purpose of this onset detection stage is the segmentation into musical note events. For the case of electric guitar recordings onsets corresponds to single plucks. The signal part between two plucks is interpreted as a note event. First, seven state-of-the-art onset detection functions (see appendix 8.1) were tested against a separate development set of guitar note recordings (see Section 5.1) using the same default blocksize and hopsize values. In general, these functions give an estimate of likelihood of a note onset to appear at each given time frame. Based on their superior performance, we selected the three best functions Spectral Flux, Pitchogram Novelty, and Rectified Complex Domain for the framework. Since all detection functions work in the frequency domain, we determined the optimal framesize for each function.

For the extraction of the onset positions, a peak picking algorithm proposed by Dixon [4] was used, which was optimized separately for each method. The results of each onset detection compared to the manually annotated ground truth are shown in Table 3. All detections are considered as true positives within an absolute tolerance area of 50 ms.

Table 3: *Optimal framesize and achieved F-measure for the best performing onset detection functions.*

Onset Detection Function	Optimal framesize	F-Measure
Spectral Flux	8 ms	0.93
Pitchogram Novelty	5 ms	0.87
Rectified Complex Domain	5 ms	0.95

The obtained onset positions of all detection functions are combined and filtered with an additional peak picking to avoid the detection of crackles, offsets, or duplicates that represent the same note onsets caused by this combination. Therefore, the mean square of energies $\bar{E}(n)$ in a variable interval τ before and after each onset candidate are analyzed and set into relation as

$$\bar{E}(n) = \frac{\sum_{i=-\tau}^{\tau} f(n+i)^2}{\tau}, \tag{1}$$

with $f(n)$ corresponding to the n^{th} frame of the summarized onset function f. With $\bar{E}_i$ denoting the mean squared energy of the ith interval ahead of the current onset, L corresponding to the length of the signal, f_s corresponding to the sampling frequency, and k_F and k_T being adjustment variables, the general conditions defining a detection as a valid onset are the following:

$$\min[\bar{E}_i(n-2\tau i)]_{i=1,2,3..I} < \bar{E}(n+\tau), \tag{2}$$

$$\frac{\sum_{i=1}^{L} f(i)^2}{L} < k_E \cdot \bar{E}(n+\tau) \tag{3}$$

and

$$n - n_{os(n-1)} > k_T \cdot f_s. \tag{4}$$

I is the maximum of intervals taken into account before the onset candidate, n is the sample index, and n_{os} is the sample number of the observed onset candidate.

In this work, the best results were achieved with $k_E = 100$, $k_T = 0.12$ ms, and $\tau = 331$ corresponding to 1.5 frames of 5 ms hopsize with a sample rate of 44100 Hz. The final method achieved an F-measure for onset detection of 98.5 %—all results are summarized in Section 5.3.

4.3. Multipitch Estimation

Next, the note segments of the audio signal are examined with respect to their spectral energy distribution. Large frame-sizes of $N = 4096$ and higher are necessary for the conventional Short-time Fourier Transform (STFT) to get a sufficient frequency resolution, which offers enough information for the pitch discrimination in the fundamental frequency register of the spectrum of a guitar. At the same time, large frame-sizes significantly reduce the achievable time resolution, which especially affects short notes. To avoid such complications, we compute a reassigned magnitude spectrogram based on the Instantaneous Frequency (IF) [7] representation in addition to the conventional time-frequency transform. By using the phase information for frequency correction, the IF supplies a high spectral accuracy while working with shorter frame sizes (here: $N = 1024$).

We use the IF magnitude spectrogram with a logarithmically-spaced frequency axis (84 bins per octave) as input for the subsequent Blind Harmonic Adaptive Decomposition (BHAD) algorithm proposed by Fuentes in [6]. It uses a frame overlap of 75 % and a downsampling by factor 4. The BHAD represents a multipitch estimation based on a framewise approach as previously used by Männchen et al. [3]. Several start frames (default: 5 frames) of each note event are left out to avoid the influence of noisy attack part transients. Furthermore, we aggregate over the following five frames in order to achieve more robust results. This way, note events with a minimum duration of 65 ms can be evaluated. For shorter events the amount of frames used for aggregation is reduced proportional.

We achieved an F-measure of 0.96 for pitch detection using this parameter setting. For the optimization of the algorithm concerning the number of aggregated frames and the parameters of the BHAD algorithm, we aimed at maximizing the Recall value (here: 0.98) in order to detect all possible fundamental frequency candidates. False positives are less critical since they can be eliminated by subsequent energy checks and checks of multiple pitch occurrences.

4.4. Partial Tracking

Based on the results of the pitch estimation, the fundamental frequency and the first 15 partials of each note event are tracked over time as follows. First, we apply a simple peak picking to the magnitude spectrum of each frame. The spectral peaks are assigned to harmonics of the different fundamental frequency candidates by minimizing the distance between the ideal harmonic frequency positions and the detected peak positions. We estimate the inharmonicity coefficient in each frame based on the detected partial peaks [3]. Results of the previous frames were used as initial inharmonicity values for the current frame and hence for more accurate partial estimation. The first frames were calculated with initial values based on [1].

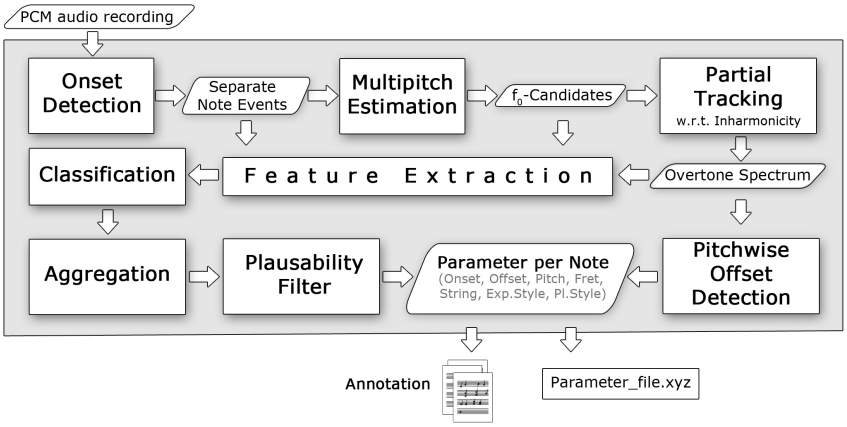


Figure 2: *Schematic model of the analysis framework.*

In addition, a window function is applied as a weighting function for variables of the tracking algorithm. These variables determine the frequency range around each partial where peaks are taken into account as well as the predicted frequency values of each partial for the following frame. A comparison of common window functions yields best performance for the use of a Kaiser function. The window length is fitted to the note duration and hence has the biggest impact in the middle of each note and almost no impact at the start and end position. Using this window, the considered search area for frequency peaks around each ideal harmonic frequency position is adjusted frame-wise. It adapts the extent of the range around each calculated partial which is taken into account for the performed peak picking to the relative note position.

Hence, at the temporal center of each note event this range is the largest and therefore more susceptible for frequency changes. Furthermore, the window function affects the amount of past frames taken into account when calculating the predicted harmonic frequencies of the current frame. At the center point of a note event less frames are considered emphasizing the affinity for frequency changes. Finally, the weight of magnitudes around each calculated harmonic frequency position is increased towards the middle of the note event. So, the comparison in the middle of note events yields lower dependency of the actual frequency distance but emphasizes high frequency magnitudes near the theoretical frequency. These three conditions are needed for an adaptive algorithm which reacts sensitive to frequency modulation techniques like bending, vibrato, and slide (see Section 2). A typical fundamental frequency envelope $f_0(t)$ for the frequency modulation technique *slide* is shown in Figure 3.

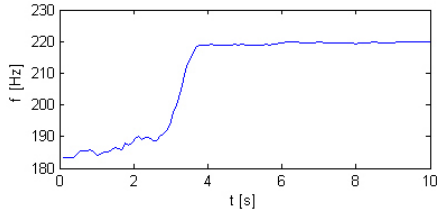


Figure 3: *Fundamental frequency envelope $f_0(t)$ of a slide tone.*

The obtained values allow for a correction of octave confusions, which can occur by the presence of weak sub-harmonics and their multiples, by summing up and comparing the odd and even harmonics of a note. Unlikely fundamentals are eliminated when the total energy of even partials falls below a quarter of the energy of odd partials.

4.5. Offset Detection

The detection of the note offset is performed based on the results of the partial tracking procedure as explained in the previous section. We obtain a temporal envelope for each time frame m by summing up the harmonic magnitude values $M_h(m)$ over all harmonics as

$$f_{\text{Env}}(m) = \sum_{h=1}^{H} M_h(m). \tag{5}$$

Figure 4 illustrates an example of a temporal magnitude envelope of a guitar note. The offset is obtained by detecting the first frame after the envelope peak with less than 5 % of the peak magnitude. Furthermore, an onset correction is performed by searching the lowest point of inflection before the peak. Therefore, the considered time area of the note excerpt is expanded in forward direction by 200 ms as safety margin. We initially smooth the envelope function by convolving it with a three-element-rectangle window to avoid the detection of random noise peaks.

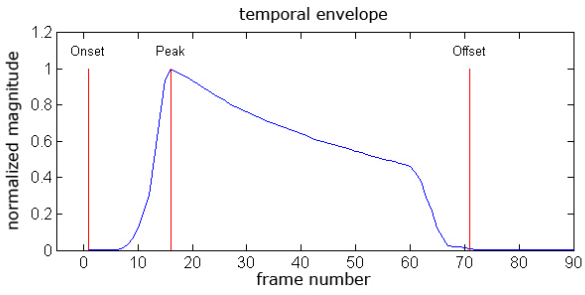


Figure 4: *Temporal magnitude envelope $f_{Env}(m)$ of the summed harmonics of a guitar note.*

4.6. Feature Extraction & Classification

Based on the extracted note parameters onset, offset, and pitch, various audio features can be extracted that allow to discriminate high-level parameters such as the played string or the applied playing techniques. We compute features on a frame-by-frame level and aggregate the features over the duration of each note event using different statistical measures such as minimum, maximum, mean, or median. A list of all 774 features can be found in appendix 8.2. A classification based on this amount of feature dimensions leads to high computational load and potential model overfitting. Therefore, prior to training the classification models, we first apply the feature selection algorithm *Inertia Ratio Maximization using Feature Space Projection* (IRMFSP) [16] for each classification task in order to reduce the dimensionality of the feature space. The amount of reduction depends on the performed classification task and is optimized separately.

For classification, we use a multi-class Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel. We perform three independent classification tasks—classification of the string number with 6 classes (see Table 1) as well as classification of the plucking style and expression style with three and six classes, respectively (see Table 2).

4.7. Plausability Filter

Depending on the classification task, the results can be aggregated over multiple notes using majority voting to obtain more robust classification. Furthermore, knowledge about typical guitar performance can be exploited in order to avoid impossible fingering positioning or improbable usage of playing techniques. In this section, two approaches to correct the classification results will be described. First, an *intra-note plausibility filter* deals with single notes and notes that are played simultaneously such as in chords. Second, an *inter-note plausibility filter* takes into account sequences of consecutive notes. Both filter aggregate classification results in dependence of the performed task for higher robustness. Expression styles are unified over all notes of a note event, plucking styles are aggregated over entire licks assuming that during one subsequently played lick no changes in plucking style occur.

4.7.1. Intra-Note Plausability Filter

The first corrections are applied to the estimated expression style. The most obvious restriction is a duration limit for the *dead note* class. Notes that are classified as dead notes and last longer than 0.6 seconds, are re-evaluated and the next probable expression style class is chosen. Second, we assume that all remaining expression styles (except the non-decorated *normal* expression style) are only plausible if less than three simultaneous pitches are detected by the multi-pitch estimation. Third, frequency modulated expression styles are compared against psychoacoustical thresholds so that detected bending, slide, or vibrato techniques with not noticeable frequency differences are set to the *normal* expression style. Especially for slide and bending, a check of the start and end pitch within the note is performed to detect and eliminate minor frequency changes below one semitone. Finally, the estimated pitches when marked as class *harmonics* are compared to possible harmonic pitches of the known guitar string tuning. Only certain pitches can occur at certain positions of the fretboard. Hence, *harmonics* detections with impossible pitch values are set to expression style *normal*.

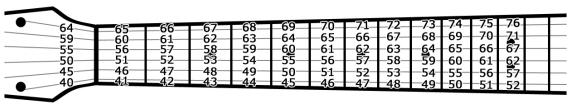


Figure 5: *Fretboard MIDI number references for standard tuning. The first column refers to empty string pitches.*

A second filter is applied to correct the string number. Each position on the fretboard is connected to a fixed pitch as shown in Figure 5, depending on the instrument tuning. Most pitch values within the pitch range of the guitar can be played on different fretboard position, hence, on different strings.

The first assumption of this filter connects to the expression style results by setting the probability of empty strings to zero if a decorating expression style has been classified. In addition, all probabilities from strings not allowing to play the observed pitch at any fretboard position are set to zero. Considering polyphonic fingerings, two or more pitches might collide by being assigned on the same string. To avoid this interception, our algorithm provides alternative fingering positions based on the string probabilities. Large spreads of the fingering are likely to occur as a result of alternative string assignment by using simple replacement to the most probable neighbour strings. Hence, spreads larger than four frets are eliminated. Instead, fingerings with smaller spreads are preferred by weighting their probabilities based on a computed fretboard centroid. Depending on the contribution of the fingering around the fretboard centroid the probability of each classification is lowered respectively to its relative fret and string distance to the centroid. Highest distances correspond to most intense lowering by half of the classification probability.

4.7.2. Inter-Note Plausability Filter

The inter-note plausability filter can correct classification results based on the parameters of preceding and subsequent notes. The first attempt of this filter is to find similar pitches played on the same string. Under the condition of comparable magnitudes and small gaps at the note borders notes are tied. As a consequence, detected expressions styles such as dead note become impossible for tied notes and are corrected. When comparing consecutive fingerings, fast and commonly applied position jumps (fingering changes with high local distances) are highly improbable if empty strings are not involved. Again, the fretboard centroid is used to weight and determine the most likely fingering if such jumps occur. This depends on the occurrence rate as well as the probability values of string estimation. The same corrections are performed for harmonic expression styles. Due to the characteristic of this playing technique, the fingering hand holds the string at an alternative fret position to obtain the perceived pitch. Here also the fretboard centroid is used to find the most probable position.

5. EVALUATION

For the evaluation of the proposed transcription algorithm, we use the common evaluation measures *Precision*, *Recall*, *F-Measure*, and *Accuracy*. In this section, a novel dataset of electric guitar recordings with extensive annotation of note parameters will be introduced. This dataset served as ground-truth in our experiments.

All results presented in Section 5.3 are based on 10-fold cross validation experiments.

5.1. Dataset

For the evaluation tasks, our novel dataset was recorded and manually annotated with all note parameters discussed in this paper. Six different guitars in standard tuning (see Table 1) were used with varying pick-up settings and different string measures to ensure a sufficient diversification in the field of electric guitars. The recording setup consisted of appropriate audio interfaces [4] which were directly connected to the guitar output. The recordings are provided in one channel RIFF WAVE format with 44100 Hz sample rate. The parameter annotations are stored in XML format.

The dataset consists of two sets. The first one created exclusively for this work contains all introduced playing techniques (see Table 2) and is provided with a bit depth of 24 Bit. It has been recorded using three different guitars and consists of about 4700 note events with monophonic and polyphonic structure. As a particularity the recorded files contain realistic guitar licks ranging from monophonic to polyphonic instrument tracks. In addition, a second set of data consisting of 400 monophonic and polyphonic note events with 3 different guitars is provided. No expression styles were applied here and each note event was recorded and stored in a separate file with a bit depth of 16 Bit [3]. The combined dataset will be made available as a public benchmark for guitar transcription research[5].

5.2. Experimental Procedure

For the onset detection, a detection within a tolerance of 50 ms to the annotated ground truth is considered as true positive. Since the offset detection is a harder task (due to smoothly decreasing note envelopes), a tolerance of 200 ms is used. Because of the time-frequency transform the duration of one additional frame (5 ms) has to be considered to obtain the effective tolerance of each temporal detection. The frequency tolerance adapts to the pitch and is scored as correct if both annotated and detected frequencies are rounded to the same MIDI pitch numbers. The three classification tasks discussed in Section 4.6 are measured using the mean normalized class accuracy.

5.3. Results

The performance of the final system for onset detection, offset detection, and pitch estimation are shown in Table 4. Because of the high specialization towards applications of guitar recordings the results clearly outperform existing approaches. Previous onset detection methods are on average placed around 90 % accuracy [4, 17], pitch estimation methods reached values up to 90 % [8, 5, 6].

The results of classification tasks are given in Table 5 - 7. In general, the typical decrease of accuracy for a higher number of classes can be observed. The string estimation still performed with good discrimination results of 82 % average accuracy including polyphonic estimation and the use of plausability filtering. The results differ from previous work [3, 10] where average accuracies around 90 % were reached due to different classification and evaluation methods. Plucking style estimation is performed with a very good score of 93 % average accuracy comparable to Abeßer et al. [2]. Here, a plausability filter was applied to combine the results of one note event. The classification of expression styles achieved good average accuracy of 83 %. State-of-the-art methods offer comparable results depending on the number of classes being distinguished. The plausability filter for expression styles introduced in Section 4.7 is used for correction and aggregation of the classification results.

[4] Tascam US 1641, M-Audio Fast Track Pro

[5] http://www.idmt.fraunhofer.de/en/business_units/smt/guitar.html

Table 4: *Precision, Recall and F-Measure results of onset detection, offset detection, and pitch estimation.*

Detection Function	Precision	Recall	F-Measure
Onset	0.98	0.99	0.99
Offset	0.98	0.98	0.98
Pitch Estimation	0.95	0.98	0.96

Table 5: *Accuracy results of the string estimation in percent displayed in a confusion matrix. Average accuracy = 82 %.*

string (correct) \ string (classified)	1	2	3	4	5	6
1	**81.3**	16.6	2.1	0.0	0.0	0.0
2	5.7	**86.0**	7.1	1.1	0.0	0.0
3	0.2	9.4	**78.8**	9.7	1.8	0.2
4	0.0	0.6	6.9	**81.8**	9.8	0.9
5	0.0	0.8	0.7	13.1	**76.7**	8.6
6	0.0	0.3	0.5	2.6	12.1	**84.5**

Table 6: *Accuracy results of the plucking style estimation in percent displayed in a confusion matrix. Average accuracy = 93 %. For abbreviations see Table 2.*

style (correct) \ style (classified)	F	P	M
F	**83.3**	16.7	0.0
P	2.5	**95.4**	2.0
M	1.9	1.9	**96.2**

With the automatically extracted transcription, guitar-specific tablature notation can be generated including information about the used playing techniques. A sample of the dataset is visualized in Figure 6. The tablature notation, which was automatically extracted from the audio recording, is compared against the reference notation taken from the dataset.

6. CONCLUSIONS

In this paper we introduced a novel algorithm for guitar transcription. The algorithm includes different estimation techniques for score-based and instrument-based parameters from isolated guitar recordings. By applying different optimization approaches, we received excellent detection results for onset, offset and pitch with an average accuracy of 96 % and higher. Estimations of more complex instrument-based parameters were performed with good results of 82 % and higher. Furthermore, a novel dataset was created and published to evaluate the proposed methods. We showed

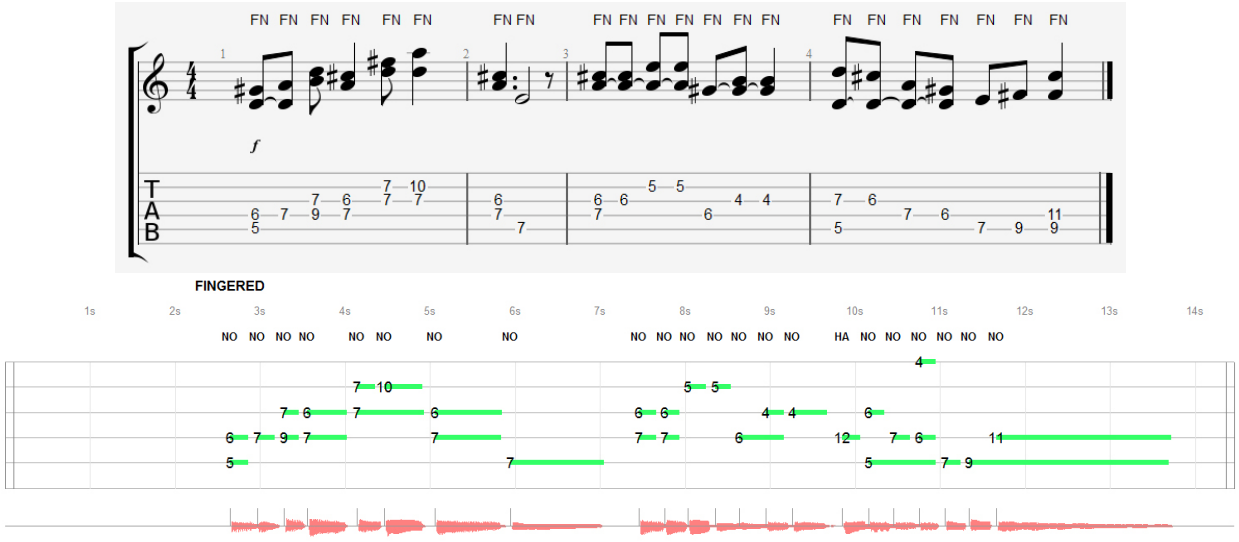


Figure 6: *Polyphonic guitar lick of the dataset.*
Top: *Manually notated tablature - Legend: FN above each note annotates the Plucking Style 'Finger Style' and Expression Style 'Normal'.*
Bottom: *Automatically notated tablature - Legend: Plucking Style is obtained for the entire lick. The letters above each note denote the Expression Style (NO - normal, HA - Harmonics).*

Table 7: *Accuracy results of the expression style estimation in percent displayed in a confusion matrix. Average accuracy = 83 %. For abbreviations see Table 2.*

stlye (correct) \ style (classified)	NO	BE	SL	VI	HA	DN
NO	**94.8**	0.7	0.5	0.9	1.5	1.6
BE	14.0	**71.3**	12.3	1.2	0.0	1.2
SL	20.7	11.2	**50.9**	8.6	4.3	4.3
VI	25.3	1.2	3.1	**66.7**	3.1	0.6
HA	10.5	0.0	0.0	2.0	**82.4**	5.2
DN	7.7	0.0	0.0	0.8	10.7	**80.8**

that an automatic transcription of guitar-based tablature is possible with a high accuracy.

7. REFERENCES

[1] Isabel Barbancho, Lorenzo J. Tardón, Simone Sammartino, and Ana M. Barbancho, "Inharmonicity-based method for the automatic generation of guitar tablature," in *Proceedings of the IEEE Transactions on Audio, Speech, and Language Processing*, August, 2012, pp. 1857–1868.

[2] Jakob Abeßer, Hanna Lukashevich, and Gerald Schuller, "Feature-based extraction of plucking and expression styles of the electric bass guitar," in *Proceedings of the Int. IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, USA, March 14-19, 2010, pp. 2290–2293.

[3] Christian Dittmar, Andreas Männchen, and Jakob Abeßer, "Real-time guitar string detection for music education software," *Proceedings of the 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pp. 1–4, 2013.

[4] Simon Dixon, "Onset detection revisited," in *Proceedings of the 9th Int. Conference on Digital Audio Effects (DAFx)*, Montreal, Canada, September 18 - 20, 2006, pp. 133–137.

[5] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri, "Automatic music transcription: challenges and future directions.," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[6] Benoit Fuentes, Roland Badeau, and Gaël Richard, "Blind harmonic adaptive decomposition applied to supervised source separation," in *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Bucharest, Romania, August 27 - 31, 2012, pp. 2654–2658.

[7] Toshihiko Abe, Takao Kobayashi, and Satoshi Imai, "Harmonics tracking and pitch extraction based on instantaneous frequency," in *Proceedings of the Int. IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Detroid, USA, May 9 - 12, 1995, pp. 756–759.

[8] Xander Fiss and Andres Kwasinski, "Automatic real-time electric guitar audio transcription," *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 373–376, 2011.

[9] Kazuki Yazawa, Daichi Sakaue, Kohei Nagira, Katsutoshi Itoyama, and Hiroshi G. Okuno, "Audio-based guitar tablature transcription using multipitch analysis and playability constraints," *Proceedings of the 38th IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 196–200, 2013.

[10] Ana M. Barbancho, Anssi Klapuri, Lorenzo J. Tardón, and Isabel Barbancho, "Automatic transcription of guitar chords

and fingering from audio," in *Proceedings of the IEEE Transactions on Speech and Language Processing 2012*, March, 2012, pp. 915–921.

[11] Cumhur Erkut, Matti Karjalainen, and Mikael Laurson, "Extraction of Physical and Expressive Parameters for Model-based Sound Synthesis of the Classical Guitar," in *Proceedings of the 108th Audio Engineering Society (AES) Convention*, 2000, pp. 19–22.

[12] Loïc Reboursière, Otso Lähdeoja, Thomas Drugman, Stéphane Dupont, Cécile Picard-Limpens, and Nicolas Riche, "Left and right-hand guitar playing techniques detection," in *Proceedings of the International Conference on New Interfaces for Muscial Expression (NIME)*, Ann Arbor, Michigan, USA, 2012, pp. 1–4.

[13] Tan Hakan Özaslan, Enric Guaus, Eric Palacios, and Josep Lluis Arcos, "Attack Based Articulation Analysis of Nylon String Guitar," in *Proceedings of the 7th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, Málaga, Spain, 2010, pp. 285–298.

[14] Jakob Abeßer, *Automatic Transcription of Bass Guitar Tracks applied for Music Genre Classification and Sound Synthesis*, Ph.D. thesis, Technische Universität Ilmenau, Germany, submitted: 2013.

[15] Christian Dittmar, Estefanía Cano, Sascha Grollmisch, Jakob Abeßer, Andreas Männchen, and Christian Kehling, "Music technology and music education," *Springer Handbook for Systematic Musicology*, 2014.

[16] Geoffroy Peeters and Xavier Rodet, "Hierarchical gaussian tree with inertia ratio maximization for the classification of large musical instrument databases," in *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx)*, London, UK, September 8-11, 2003, pp. 1–6.

[17] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler, "A tutorial on onset detection in music signals," in *Proceedings of the IEEE Transactions on Speech and Audio Processing*, September, 2005, pp. 1035–1047.

[18] Tan Hakan Özaslan and Josep Lluis Arcos, "Legato and Glissando Identification in Classical Guitar," in *Proceedings of Sound and Music Computing Conference (SMC), Barcelona, Spain*, 2010, pp. 457–463.

[19] Jakob Abeßer, "Automatic String Detection for Bass Guitar and Electric Guitar," *From Sounds to Music and Emotions - 9th International Symposium, CMMR 2012, London, UK, June 19-22, 2012, Revised Selected Papers*, pp. 333–352, 2013.

[20] Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Automatic recognition and parametrization of frequency modulation techniques in bass guitar recordings," in *Proceedings of the Audio Engeneering Society 42nd Int. Conference (AES)*, Ilmenau, Germany, July 22 - 24, 2011, pp. 1–8.

[21] Jakob Abeßer and Gerald Schuller, "Instrument-centered music transcription of bass guitar tracks," in *Proceedings of the Audio Engeneering Society 53rd Int. Conference (AES)*, London, UK, January 27 - 29, 2014, pp. 1–10.

[22] Loic Reboursière, Christian Frisson, Otso Lähdeoja, John Anderson Mills III, Cécile Picard, and Todor Todoroff, "Multimodal Guitar : A Toolbox For Augmented Guitar Performances," in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*, Syndney, Australia, 2010, pp. 415–418.

[23] David Wagner and Stefan Ziegler, "Erweiterung eines Systems zur Detektion von Onsets in Musiksignalen," in *Media Project of the Technical University of Ilmenau*, Ilmenau, Germany, 2008.

[24] Christian Kehling, "Entwicklung eines parametrischen Instrumentencoders basierend auf Analyse und Re-Synthese von Gitarrenaufnahmen," Diploma thesis, Technical University of Ilmenau, Germany, 2013.

8. APPENDIX

8.1. List of Onset Detection Functions

In this work, we compared the onset detection functions Spectral Flux [4], Rectified Complex Domain [4], Weighted Phase Deviation [4], High Frequency Content [23], Modified Kullback-Leibler Distance [23], Foote [23], and Pitchogram Novelty [21].

8.2. Audio Features

Table 8: *Feature list for classification. If features generate more than one return-value the amount is written in brackets after the feature name. Novel features are marked bold.*

· Spectral Centroid	· Odd To Even Harmonic Energy Ratio
· **Relative Spectral Centroid**	· **Harmonic Spectral Centroid**
· Spectral Roll Off	· Harmonic Magnitude Slope
· Spectral Slope	· Relative Harmonic Magnitude (14)
· Spectral Spread	· Normalized Harmonics Frequency Deviation (14)
· Spectral Decrease	
· Spectral Crest	· Frequency Statistics: Maximum, Minimum, Mean, Median, Variance (5)
· Spectral Flatness	
· Inharmonicity Factor	
· Tristimulus 1,2 und 3 (3)	· Frequency Statistics: Maximum, Minimum, Mean, Median, Variance (5)
· Spectral Irregularity	

Each frame-based audio feature listed so far is condensed to 14 statistic values per note. Maximum, Minimum, Mean, Variance, Median, Skewness and Kurtosis are computed for the attack and the decay part of each note. Both are known durations from Section 4.5 because of the performed temporal refinement. In Addition several novel note-based features are appended to the feature vector:

· High Frequency Pre Onset Arousal	· Envelope Part Length (3)
· Magnitude Range	· Temporal Slope (2)
· Envelope Sum	· Range Attack Time Deviation
· Temporal Centroid	· Mean Attack Time Deviation
· Envelope Fluctuation(2)	· Variance Attack Time Deviation
· Envelope Modulation Frequency and Range	· Subharmonic Attack Energy (21)
	· Subharmonic Decay Energy (21)

Concatenation of all features yields a feature vector of 774 elements. The detailed computation steps are explained in [24].

IMPROVING SINGING LANGUAGE IDENTIFICATION THROUGH I-VECTOR EXTRACTION

Anna M. Kruspe

Fraunhofer IDMT, Ilmenau, Germany
Center for Language and Speech Processing, Johns Hopkins University, Baltimore, USA
kpe@idmt.fhg.de

ABSTRACT

Automatic language identification for singing is a topic that has not received much attention in the past years. Possible application scenarios include searching for musical pieces in a certain language, improvement of similarity search algorithms for music, and improvement of regional music classification and genre classification. It could also serve to mitigate the "glass ceiling" effect. Most existing approaches employ PPRLM processing (Parallel Phone Recognition followed by Language Modeling).

We present a new approach for singing language identification. PLP, MFCC, and SDC features are extracted from audio files and then passed through an i-vector extractor. This algorithm reduces the training data for each sample to a single 450-dimensional feature vector. We then train Neural Networks and Support Vector Machines on these feature vectors. Due to the reduced data, the training process is very fast. The results are comparable to the state of the art, reaching accuracies of 83% on a large speech corpus and 78% on acapella singing. In contrast to PPRLM approaches, our algorithm does not require phoneme-wise annotations and is easier to implement.

1. INTRODUCTION

Language Identification (LID) describes the task of automatically detecting the language spoken in an audio document. In speech recognition, LID has been a topic of interest for more than 30 years and has been extensively researched. In the Music Information Retrieval field, a similar language identification can be performed for singing (Singing Language Identification, SLID). There have so far only been a handful of publications on SLID despite a number of interesting application scenarios, such as:

Direct search of music in a certain language SLID can be useful for private users who are, for example, looking for music for a holiday video, or for music to help them learn a language. Commercial users could use this for advertisement videos.

Improvement of similarity search Similarity dimensions could include the sung language.

Improvement of regional classification As mentioned in [1], human subjects tend to rely on the language to determine the region of origin of a musical piece. This is not taken into account by current regional classification systems.

Improvement of genre classification Similar to regional classification, certain musical genres are closely connected to a single singing language. Considering the "glass ceiling" of approximately 80% for most classification tasks[2], new hybrid approaches are necessary to improve them. SLID could serve this purpose, too.

Only a few SLID systems have been developed so far. They mostly use the principle of Parallel Phone Recognition followed by Language Modeling (PPRLM). In this paper, we present an approach that does not require the extensive annotations used in PPRLM. Our approach employs three commonly used audio features with Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) as backend classifiers. Using the i-vector extraction algorithm as a processing step in between, our approach is able to surpass the state of the art.

We will give a more detailed overview over the state of the art in section 2 before presenting the used datasets in section 3. We then describe our proposed system in section 4 and show experimental results in section 5. Finally, we draw conclusions in section 6 and make suggestions for further research in section 7.

2. STATE OF THE ART

2.1. Language identification for speech

Language identification has been extensively researched in the field of Automatic Speech Recognition since the 1980's. A number of successful algorithms have been developed over the years. An overview over the fundamental techniques is given by Zissman in [3].

Fundamentally, four properties of languages can be used to discriminate between them:

Phonetics The unique sounds that are used in a given language.

Phonotactics The probabilities of certain phonemes and phoneme sequences.

Prosody The "melody" of the spoken language.

Vocabulary The possible words made up by the phonemes and the probabilities of certain combinations of words.

Even modern system mostly focus on phonetics and phonotactics as the distinguishing factors between languages. Vocabulary is sometimes exploited in the shape of language models.

Zissman mentions Parallel Phone Recognition followed by Language Modeling (PPRLM) as one of the basic techniques. It requires audio data, language annotations, and phoneme annotations for each utterance. In order to make use of vocabulary characteristics, full sentence annotations and word-to-phoneme dictionaries are also necessary.

Using the audio and phoneme data, acoustic models are trained. They describe the probabilities of certain sound and sound sequences occurring. This is done separately for each considered language. Similarly, language models are generated using the sentence annotations and the dictionary. These models describe the probabilities of certain words and phrases. Again, this is done for each language.

New audio examples are then run through all pairs of acoustic and language models, and the likelihoods produced by each model are retained. The highest acoustic likelihood, the highest language likelihood, or the highest combined likelihood are then considered to determine the language. This approach achieves up to 79% accuracy for ten languages [4].
Another approach uses the idea to train Gaussian Mixture Models for each language. This technique can be considered a "bag of frames" approach, i.e. the single data frames are considered to be statistically independent of each other. The generated GMMs then describe probability densities for certain characteristics of each language. Using these, the language of new audio examples can be easily determined.
GMM approaches used to perform worse than their PPRLM counterparts, but the development of new features has made the difference negligible [5]. They are in general easier to implement since only audio examples and their language annotations are required. Allen et al. [6] report results of up to 76.4% accuracy for ten languages. Different backend classifiers, such as Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) [7] have also been used succesfully instead of GMMs.

2.2. Special challenges in singing

Singing presents a number of challenges for language identification when compared to pure speech. To mention a few examples:

Larger pitch fluctuations A singing voice varies its pitch to a much higher degree than a speaking voice. It often also has very different spectral properties.

Higher pronunciation variation Singers are often forced by the music to pronounce certain sounds and words differently than if they were speaking them.

Larger time variations In singing, sounds are often prolonged for a certain amount of time to fit them to the music. Conversely, they can also be shortened or left out completely.

Different vocabulary In musical lyrics, words and phrases often differ from normal conversation texts. Certain words and phrases have different probabilities (e.g. higher focus on emotional topics in singing).

Background music adds irrelevant data (for language identification) to the signal, which acts as an interfering factor to the algorithms. It therefore should be removed or suppressed prior to the language identification, e.g. by source separation algorithms.
In this paper, we only work with a-capella music to remove this difficulty.

So far, only a few approaches to perform language identification on singing have been proposed.
Schwenninger et al. [8] use MFCC features, but do not mention how they perform their actual model training. They test different pre-processing techniques, such as vocal/non-vocal segmentation, distortion reduction, and azimuth discrimination. None of these techniques seem to improve the over-all results. They achieve an accuracy of 68% on a-capella music for two languages (English and German).
The approach of Tsai and Wang [9] follows a traditional PPRLM flow. After vocal/non-vocal segmentation, they run their data through acoustic models using vector tokenization. One acoustic model for each language is used. The results are then processed by bigram language models, again for each language. The language model score is used for a maximum likelihood decision to determine the language. They achieve results of 70% accuracy for two languages (English and Mandarin).
Mehrabani and Hansen [10] also use a PPRLM system, with the difference that all combinations of acoustic and language models are tested. Their scores are combined by a classifier to determine the final language. This results in a score of 78% for three languages (English, Hindi, and Mandarin). Combining this technique with prosodic data improved the result even further.
Chandrasekhar et al.[11] try to determine the language for music videos using both audio and video features. They achieve accuracies of close to 50% for 25 languages. It is interesting to note that European languages seem to achieve much lower accuracies than Asian and Arabic ones. English, French, German, Spanish and Italian rank below 40%, while languages like Nepali, Arabic, and Pashto achieve accuracies above 60%.
Finally, we previously tested a different system based on Gaussian Mixture Models (GMMs) [12]. This approach does not require phoneme-wise annotations like the PPRLM approaches and is easier to implement. We achieved an accuracy of 68% on three languages (a-capella data).

3. DATASETS

In order to test our system on singing data, we used the data set previously presented in [12]. It consists of a-capella songs downloaded from *YouTube*[1]. The songs are performed by amateur singers in the languages English, German, and Spanish. We call it *YTAcap*.
For comparison, we also tested our algorithm on two well-known speech data sets: The *2003 NIST Language Recognition Evaluation (NIST2003LRE)* corpus [13] and the *OGI Multi-language Telephone Speech Corpus (OGIMultilang)*[14], using only the three previously mentioned languages.
An overview over the amount of data across the three corpora is given in table 1.

Table 1: *Amounts of data in the three used data sets: Sum duration on top, number of utterances in italics.*

hh:mm:ss ***#Utterances***	**NIST2003LRE**	**OGIMultilang**	**YTAcap**
English	00:59:08 *240*	05:13:17 *1912*	08:04:25 *1975*
German	00:59:35 *240*	02:52:27 *1059*	04:18:57 *1052*
Spanish	00:59:44 *240*	03:05:45 *1151*	07:21:55 *1810*

[1] `http://www.youtube.com/`

4. PROPOSED SYSTEM

Figure 1 shows a rough overview over our classification system. In the following, we will describe the selected features, the i-vector extraction algorithm, and the selected training backends in more detail.

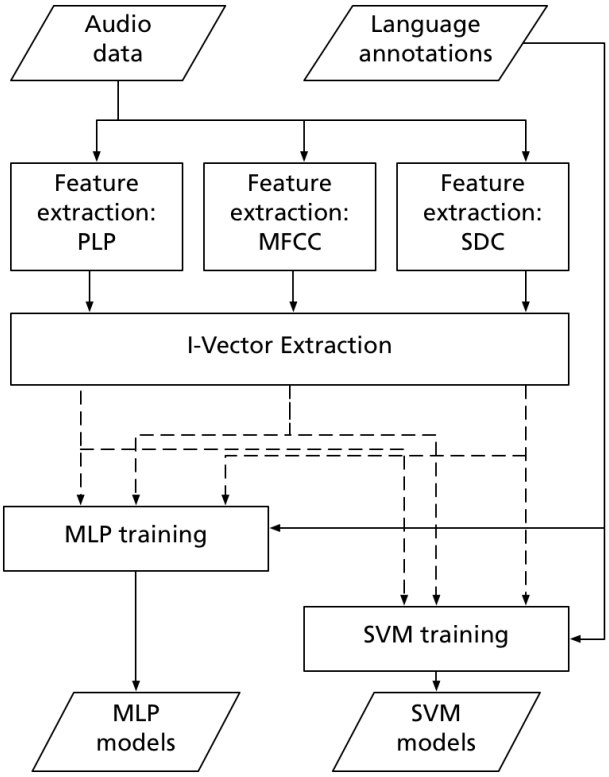


Figure 1: *Overview of the steps of our classification system.*

4.1. Features

We extracted a set of features for each audio file. Table 2 shows an overview over the various configurations used in training.

Perceptive Linear Predictive features (PLPs) PLP features, first introduced in [15], are among the most frequently used features in speech processing. They are based on the idea to use knowledge about human perception to emphasize important speech information in spectra while minimizing the differences between speakers. We use a model order of 13 in two experiments and one of 32 in another. Deltas and double deltas between frames are also calculated. We test PLPs with and without RASTA pre-processing [16].

Mel-Frequency Cepstral Coefficients (MFCCs) Just like PLPs, MFCCs are frequently used in all disciplines of automatic speech recognition [3]. We kept 20 cepstral coefficients for model training. Additionally, we calculated deltas and double deltas.

Shifted Delta Cepstrum (SDCs) Shifted Delta Cepstrum features were first described in [17] and have since been successfully used for speaker verification and language identification tasks on pure speech data [18] [7] [6]. They are calculated on MFCC vectors and take their temporal evolution into account. Their configuration is described by the four parameter $N-d-P-k$, where N is the number of cepstral coefficients for each frame, d is the time context (in frames) for the delta calculation, k is the number of delta blocks to use, and P is the shift between consecutive blocks. The delta cepstrals are then calculated as:

$$\Delta c(t) = c(t+iP+d) + c(t+iP-d), 0 <= i <= k \quad (1)$$

with $c \in [0, N-1]$ as the previously extracted cepstral coefficients. The resulting k delta cepstrals for each frame are concatenated to form a single SDC vector of the length kN. We used the common parameter combination $N = 7, d = 1, P = 3, k = 7$.

4.2. I-Vector extraction

I-Vector (identity vector) extraction was first introduced in [19] and has since become a state-of-the-art technique for various speech processing tasks, such as speaker verification, speaker recognition, and language identification [20]. To our knowledge, it has not been used for any Music Information Retrieval tasks before.
The main idea behind i-vectors is that all training utterances contain some common trends, which effectively add irrelevance to the data in respect to training. Using i-vector extraction, this irrelevance can be filtered out, while only the unique parts of the data relevant to the task at hand remain. The dimensionality of the training data is massively reduced, which also makes the training less computationally expensive. As a side effect, all feature matrices are transformed to i-vectors of equal length, eliminating problems that are caused by varying utterance lengths.
Mathematically, this assumption can be expressed as:

$$M(u) = m + Tw \quad (2)$$

In this equation, $M(u)$ is the GMM supervector for utterance u. The supervector approach was first presented in [21] and has since been successfully applied to a number of speech recognition problems. A music example can be found in [22]. m represents the language- and channel-independent component of u and is estimated using a Universal Background Model (UBM). T is a low-rank matrix modeling the relevant language- and channel-related variability, the so-called Total Variability Matrix. Finally, w is a normally distributed latent variable vector: The i-vector for utterance u.

Step 1: UBM training A Universal Background Model (UBM) is trained using Gaussian Mixture Models (GMMs) from all utterances. This UBM models the characteristics that are common to all of them.

Step 2: Statistics extraction 0th and 1st order Baum-Welch statistics are calculated for each of the utterances from the UBM according to:

$$N_c(u) = \sum_{t=1}^{L} P(c|y_t, \Omega) \quad (3)$$

$$\widetilde{F_c}(u) = \sum_{t=1}^{L} P(c|y_t, \Omega)(y_t - m_c) \quad (4)$$

Table 2: *Feature configurations used in training.*

Name	Description	Dimensions
PLP	PLP with RASTA processing, model order 13 with deltas and double deltas	39
PLP36	PLP with RASTA processing, model order 36 with deltas and double deltas	96
PLP_NORASTA	PLP without RASTA processing, model order 13 deltas and double deltas	39
MFCC	MFCC, 20 coefficients	20
MFCCDELTA	MFCC, 20 coefficients, deltas and double deltas	60
SDC	SDC with configuration $7-1-3-7$	91
MFCCDELTASDC	MFCCDELTA+SDC	117
COMB	PLP_NORASTA+MFCCDELTA	99

where $u = y_1, y_2, ..., y_L$ denotes an utterance with L frames, $c = 1, ..., C$ denotes the index of the Gaussian component, Ω denotes the UBM, m_c is the mean of the UBM mixture component c, and $P(c|y_t, \Omega)$ denotes the posterior probability that the frame y_t was generated by mixture component c. As the equation shows, the 1st order statistics are centered around the mean of each mixture component.

Step 3: T matrix training Using the Baum-Welch statistics for all utterances, the Total Variability Matrix T is now trained iteratively according to:

$$w = (I + T^t \Sigma^{-1} N(u) T)^{-1} T^t \Sigma^{-1} \widetilde{F}(u) \quad (5)$$

using Expectation Maximization.

Step 4: Actual i-vector extraction Finally, an i-vector w can be extracted for each utterance using equation 5 again. This can also be done for unseen utterances, using a previously trained T.

4.3. Classification backend

For classification, we tested Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs).
The MLPs were fixed at three layers, with the middle layer having a dimension of 256. They were implemented using Quicknet [23]. Additional layers did not seem to improve the result. A larger middle layer only improved it slightly.
The SVM parameters were determined using a grid-search. In the full-feature experiments, we additionally employed a previous feature selection using the "Inertia Ratio Maximization using Feature Space Processing" (IRMFSP) [24]. For both IRMFSP and SVMs, we used our own C++ implementation.

5. EXPERIMENTAL RESULTS

As described above, we performed experiments using both MLP and SVM classifiers on all three data sets (NIST, OGI, and YTA-cap). For each of those classifiers and data sets, we test all of the combinations of features listed in table 2 directly and with i-vector processing. All results were obtained using five-fold cross validation.

5.1. MLP results

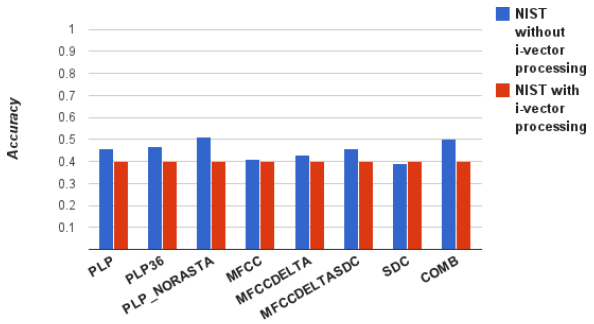


Figure 2: *Results for all feature combinations on the NIST2003LRE database, using MLP classifiers.*

NIST2003LRE As shown in figure 2, our MLP did not produce good results on the *NIST2003LRE* database for any of the feature combinations. *NIST2003LRE* is the smallest of the data sets by a large margin. Since we use a relatively high dimensional model, this is probably a case of overtraining. The i-vector processing step reduces the training data even further, thus aggravating the problem.

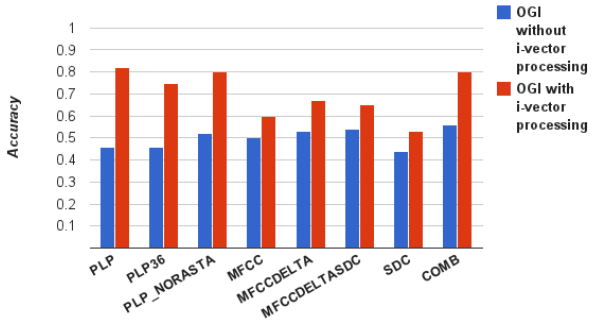


Figure 3: *Results for all feature combinations on the OGIMultilang database, using MLP classifiers.*

OGIMultilang The *OGIMultilang* data set contains roughly 4 times as much data as the *NIST2003LRE* set. With enough data, training an MLP classifier works a lot better. Without i-vector processing, we still only reach about 52% accuracy. I-Vector extraction improves the system massively. The best feature configurations are plain PLP (82%), PLP_NORASTA (80%) and COMB (80%).

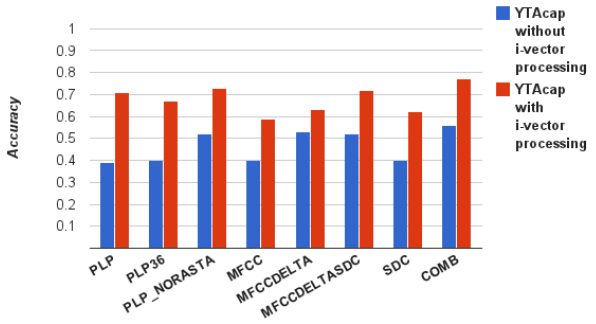


Figure 4: *Results for all feature combinations on the YTAcap database, using MLP classifiers.*

YTAcap In section 2.2, we described some factors that increase the difficulty for language identification on acapella data versus spoken data. As expected, the results on the *YTAcap* data set are somewhat worse than those on *OGIMultilang*, even though they contain a similar amount of data. The best result without i-vector extraction is still obtained using the COMB feature configuration at 56%. Similar to the *OGIMultilang* experiment, i-vector extraction yields a large improvement. COMB remains the best configuration, now at 77%.

5.2. SVM results

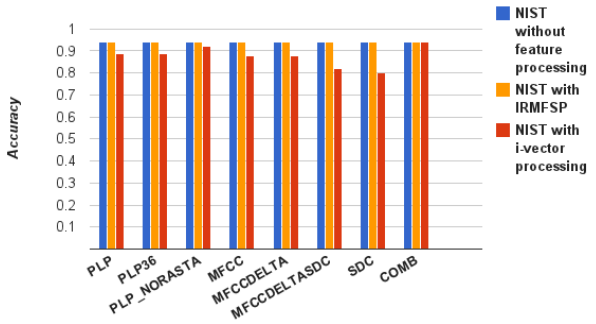


Figure 5: *Results for all feature combinations on the NIST2003LRE database, using SVM classifiers.*

NIST2003LRE In contrast to the MLP experiment, SVMs produced very good results on the *NIST2003LRE* data set for all of the features. They seem to be able to discriminate almost perfectly for this small, clean data set. We believe 94% might be an upper bound for the classification here, which might be caused by annotation errors or ambiguous data.

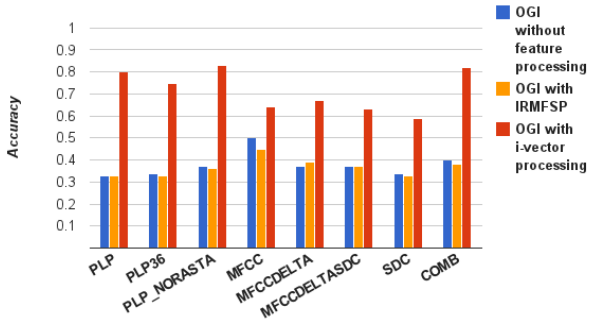


Figure 6: *Results for all feature combinations on the OGIMultilang database, using SVM classifiers.*

OGIMultilang The *OGIMultilang* corpus is bigger and more varied than the *NIST2003LRE* corpus, which makes it harder to classify. As shown, the high-dimensional pure features did not produce good results, with a maximum of 50% for MFCCs. Feature selection using IRMFSP did nothing to improve this result either. I-Vector extraction, however, improved the result by a large margin. Feature-wise, PLP without RASTA processing seems to work best at a result of 83%. MFCC and SDC features did not work quite as well, but did not hurt the result either when combined with PLPs (COMB result). It is interesting to see that the i-vector extraction provided the smallest improvement for MFCCs, the feature that worked best without it.

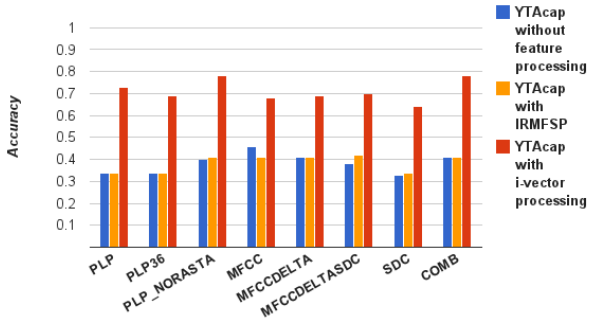


Figure 7: *Results for all feature combinations on the YTAcap database, using SVM classifiers.*

YTAcap Similar to the *OGIMultilang* corpus, the *YTAcap* corpus provides very complex and varied data. We see the same effects on the direct feature training here, too: MFCCs provide the best results, but the accuracy is not very high at just 46%. IRMFSP

still does not seem to be able to reduce the feature complexity in a salient way. I-Vector extraction, again, serves to improve the result by a large percentage. The highest result when using i-vector extraction is 78% when using PLP without RASTA processing or the COMB configuration.

6. CONCLUSION

In this paper, we presented an approach for automatic language identification on speech and acapella singing corpora. We used PLP, MFCC, and SDC features, and ran them through an i-vector extractor. We used the generated i-vectors as inputs for MLP and SVM training. To our best knowledge, the i-vector approach is new to music information retrieval. The basic idea behind it is the removal of speaker- and channel-dependent components of the signal. This effectively reduces irrelevance to the language identification tasks and also reduces the amount of training data massively.

Our smallest data set was the *NIST 2003 Language Recognition Evaluation (NIST2003LRE)* corpus. We did not achieve good results for any feature configuration when using the MLP backend. We believe that the small size of the corpus leads to overtraining. I-Vector processing only amplified this problem by reducing the amount of data even further. The SVM backend, however, produced good results of up to 94% for almost all features, with and without i-vector extraction.

The *OGIMultilang* corpus is a much bigger speech corpus. Training without i-vector extraction did not work well for any feature configuration. The best accuracy for this scenario was 52%. Feature selection using IRMFSP did not improve this result. I-Vector extraction, however, improved the results for all feature configurations immensely. We achieved results of up to 83% when i-vector processing was performed. There does not seem to be a large difference between SVM and MLP training, with SVM having just a slight advantage.

We expected language identification for singing to be a harder task than for speech due to the factors described in section 2.2. The results on the *YTAcap* corpus turned out to be somewhat worse than those for the *OGIMultilang* corpus, which is of similar size. We observed the same effect as on *OGIMultilang*: Fairly bad results on the raw features that improved by a large percentage through i-vector extraction. In this case, the accuracy jumped from 56% to 78%.

In general, MLPs seem to work a little better when raw features are used, while SVMs work better when i-vector processing is applied, but only by a small percentage.

Concerning the features, both PLPs and MFCCs seemed to be able to discriminate between languages. PLPs worked best when no RASTA pre-processing was used. We believe that this is because the recordings are all relatively high quality and not heavily spectrally distorted. A higher model order did not significantly increase the accuracy either.

MFCCs worked best when combined with deltas and double-deltas. SDCs by themselves did not work as well as PLP or MFCC features, but were able to increase the accuracy somewhat when combined with MFCCs and their deltas. This confirms our observation mentioned in [12].

The best accuracies were usually achieved when combining MFCCs, MFCC deltas, and PLP features, both features covering different relevant components.

When using an MLP backend, i-vector processing seems to increase the accuracy roughly equally for each feature configuration. Interestingly, this is not true for the SVM backend. In the SVM experiments, MFCCs usually produced the best results when used directly for training. I-Vector extraction provided the smallest improvement for this configuration, but improved the PLP configurations much more.

Overall, i-vector extraction reduces irrelevance in the training data and there leads to a more effective training. As additional benefits, the training process itself is much faster and less memory is used due to its data reduction properties. Using this system, we achieve results that are comparable to the system described in [10] and higher than other publications on the topic of singing language identification. Most of these approaches are based on PPRLM, which requires phoneme-wise annotations and a highly complex recognition system, using both acoustic and language models. In this respect, our system is easier to implement and merely requires language annotations.

7. FUTURE WORK

Since our algorithm produced good results on acapella data, we would now like to test it on polyphonic music. For this purpose, we will integrate additional pre-processing techniques, such as vocal activity detection and source separation.

We will then use the results produced by our language identification algorithm to improve other classification solutions. Genre classification and regional classification are of particular interest here.

In this context, we will expand the music material to different styles, such as opera music or especially non-western music.

We showed that the i-vector extraction algorithm improved our classification accuracy by a large percentage. To our best knowledge, it has not yet been applied to any other Music Information Retrieval problems, such as genre recognition or emotion detection. We are going to investigate these applications as well.

8. REFERENCES

[1] A. Kruspe, H. Lukashevich, J. Abesser, H. Grossmann, and C. Dittmar, "Automatic classification of musical pieces into global cultural areas," in *Proceedings of Audio Engineering Society 42nd Conference*, Ilmenau, Germany, 2011, pp. 44–53.

[2] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: How high is the sky?," in *Journal of Negative Results in Speech and Audio Sciences*, 2004, vol. 1.

[3] M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 31–44, Jan. 1996.

[4] Y. K. Muthusamy, E. Barnard, and R. A. Cole, "Reviewing automatic language identification," *IEEE Signal Procesing Magazine*, vol. 11, no. 4, pp. 33 – 41, Oct. 1994.

[5] E. Singer, P. A. Torres-Carrasquillo, T. P. Gleason, W. M. Campbell, and D. A. Reynolds, "Acoustic, phonetic, and discriminative approaches to automatic language identification," in *Proceedings of Eurospeech*, Geneva, Switzerland, 2003, pp. 1345–1348.

[6] F. Allen, E. Ambikairajah, and J. Epps, "Language identification using warping and the shifted delta cepstrum," in *2005 IEEE 7th Workshop on Multimedia Signal Processing*, Shanghai, China, 2006, pp. 1–4.

[7] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech and Language*, vol. 20, pp. 210–229, 2006.

[8] J. Schwenninger, R. Brueckner, D. Willett, and M. E. Hennecke, "Language identification in vocal music," in *7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006, pp. 377–379.

[9] W.-H. Tsai and H.-M. Wang, "Towards automatic identification of singing language in popular music recordings," in *5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004, pp. 568–576.

[10] M. Mehrabani and J. H. L. Hansen, "Language identification for singing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 4408–4411.

[11] V. Chandraskehar, M. E. Sargin, and D. A. Ross, "Automatic language identification in music videos with low level audio and visual features," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 5724–5727.

[12] A. Kruspe, J. Abesser, and C. Dittmar, "A GMM approach to singing language identification," in *AES 53*, London, UK, 2014.

[13] A. Martin and M. Pryzbocki, "2003 NIST Language Recognition Evaluation," Tech. Rep., Linguistic Data Consortium, Philadelphia, 2006.

[14] R. Cole and Y. Muthusamy, "OGI Multilanguage Corpus," Tech. Rep., Linguistic Data Consortium, Philadelphia, 1994.

[15] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am.*, vol. 57, no. 4, pp. 1738–52, Apr. 1990.

[16] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "RASTA-PLP speech analysis," Tech. Rep. TR-91-069, ICSI, 1991.

[17] B. Bielefeld, "Language identification using shifted delta cepstrum," in *Fourteenth annual speech research symposium*, Baltimore, MD, USA, 1994.

[18] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. Deller, Jr., "Approaches to language identification using gaussian mixture models and shifted delta cepstral features," in *International Conference on Spoken Language Processing (ICSLP)*, Denver, CO, USA, 2002, pp. 89–92.

[19] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, May 2011.

[20] D. Martinez, O. Plchot, and L. Burget, "Language Recognition in iVectors Space," in *Interspeech*, Florence, Italy, August 2011, pp. 861–864.

[21] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," in *Digital Signal Processing*, 2000, p. 2000.

[22] C. Charbuillet, D. Tardieu, and G. Peeters, "GMM Supervector for content based music similarity," 2011, number 1, pp. 1–4.

[23] D. Johnson, "ICSI quicknet software package," http://www.icsi.berkeley.edu/Speech/qn.html, 2004.

[24] G. Peeters and X. Rodet, "Hierarchical gaussian tree with inertia ratio maximization for the classification of large musical instrument databases," in *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFX-03)*, Sept. 2003, pp. 1–6.

UNISON SOURCE SEPARATION

Fabian-Robert Stöter, Stefan Bayer, Bernd Edler

International Audio Laboratories Erlangen,*
Erlangen, Germany
{fabian-robert.stoeter}@audiolabs-erlangen.de

ABSTRACT

In this work we present a new scenario of analyzing and separating linear mixtures of musical instrument signals. When instruments are playing in unison, traditional source separation methods are not performing well. Although the sources share the same pitch, they often still differ in their modulation frequency caused by vibrato and/or tremolo effects. In this paper we propose source separation schemes that exploit AM/FM characteristics to improve the separation quality of such mixtures. We show a method to process mixtures based on differences in their amplitude modulation frequency of the sources by using non-negative tensor factorization. Further, we propose an informed warped time domain approach for separating mixtures based on variations in the instantaneous frequencies of the sources.

1. INTRODUCTION

Audio source separation is a very active research field with a large number of contributions. Applications are dependent on the context of the scenario, ranging from enhancements of speech signals to musically motivated analysis tasks.

The separation of sound sources from a single channel mixture is considered as an under-determined case which does not have a single solution. Knowing the way in which source signals are mixed together is crucial to the quality of separation systems. In the context of speech separation even unsupervised methods can lead to good results. This is due to the fact that mixtures of speech signals (like in a cocktail party environment) show a high degree of statistical independence. Mixtures of musical instruments, however, are highly correlated which is a desired aim of musical performances in general.

The Signal Separation Evaluation Campaign (SiSEC) is a solid indicator of the progress in research within the field of source separation [1]. The results from 2013 [2] show that for professionally produced music it is still difficult to achieve a high quality separation. One reason is due to the fact that the wide use of non-linear post-processing techniques (e.g. dynamic compression or effects like reverb) break assumptions that often are required to enable good performance of source separation algorithms. Another reason is that non-stationary effects like vibrato introduce additional problems [3].

In most scenarios for source separation of instrument signals it is common to assume that the spectral harmonics do only partially overlap. This enables algorithms like non-negative matrix factorization (NMF) to approximate the mixture from a lower-rank decomposition in an unsupervised way. Such systems are described in [4] and [5]. Additionally the popularity of the class of NMF algorithms can be explained by the intuitive way in which they work on time-frequency representations of the mixture signal.

In the context of musical instrument source separation, many researchers have focused on including prior information about the sources in their algorithms [6]. The availability and detail of such a-priori information varies. Often systems learn spectral as well as temporal cues from training data or parts of the mixture where only one instrument is active. One example of such informed source separation systems is described by Ewert and Müller [7]. It incorporates the pitch and onset information encoded in a MIDI file to improve the separation result.

Even the number of sources is a simple but very important information for source separation algorithms. One of the main drawbacks of many source separation systems is that they rely on this information. In some scenarios, like popular western music, the sources to separate are grouped into Melody + Bass + Drums and a residual signal. Constraining the system to such a scenario allows the results to be evaluated even if the set of sources being separated is incomplete. Constrained systems like these are also sufficient for real-world applications such as the eminent karaoke scenario. Limiting the number of desired sources helps not only to improve the performance of the algorithms but is also related to the fact that the number of sources humans can perceive is limited, too. Although a threshold has not been systematically addressed so far, a variety of experiments have been carried out. David Huron found [8] that the number of voices humans can correctly identify is up to three. When Stöter and Schoeffler et. al. [9, 10] asked participants to identify the number of instruments in a piece of music, the participants were only able to identify up to three, similar to Huron's voice experiments. There is very little chance that listeners are able to detect the presence of more than three sources. However in trials with fewer than three instruments, listeners tended to be very sensitive: One of the stimuli in the [9, 10] experiments with 1168 participants consisted of a mixture of Violin and Flute played in unison. The results showed that 76% of the participants correctly identified two instruments. Only 18% of the participants underestimated by one instrument, 6% overestimated by one instrument.

Since humans are able to reliably detect even instruments played in unison, this is a good motivation to expect the same from an algorithm. In this paper we want to address this scenario which has not been brought up so far. We believe creating and evaluating new algorithms for separating sources playing in unison will improve source separation systems in general.

The remainder of this paper is organized as follows: Section 2 describes the challenges of a unison source separation scenario. We propose techniques based on the modulation characteristics of the signal to address the separation scenario in Section 3 and Section 4. In Section 5 we introduce a data set for the unison scenario.

* The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen (IIS).

Further we present and discuss the results from our study and a comparison between the algorithms in Section 5.3. Conclusions are then presented in Section 6.

2. UNISON SOURCE SEPARATION SCENARIO

Up to date there are very few proposed source separation methods which perform good on a variety of input signals without making general assumptions or constraints. Most of the current state-of-the-art algorithms address specific scenarios like voice or melody extraction, or harmonic percussive separation. Additionally assumptions about the mixture itself are important, too. In this paper we consider the linear single channel case:

$$x(n) = \sum_{s=1}^{N} x_s(n). \tag{1}$$

Describing a source separation scenario includes the number of sources N and the number of desired sources D which is normally smaller than N when the desired sources contain groups of sources like instrument classes (strings, woodwinds, etc.).

We propose a scenario where instruments play in unison. This means that they share the same fundamental frequency (regardless of the octave) so that the sources can overlap both in time and frequency. In fact unison[1] mixtures are meant to be as much overlapped as possible, hence they are very difficult to separate. However, due to masking effects, a relatively good subjective quality for the separated sources can be obtained, even if the other sources are not perfectly suppressed. As far as we know, there is no contribution to the source separation scene that focuses on mixtures of such unison sources.

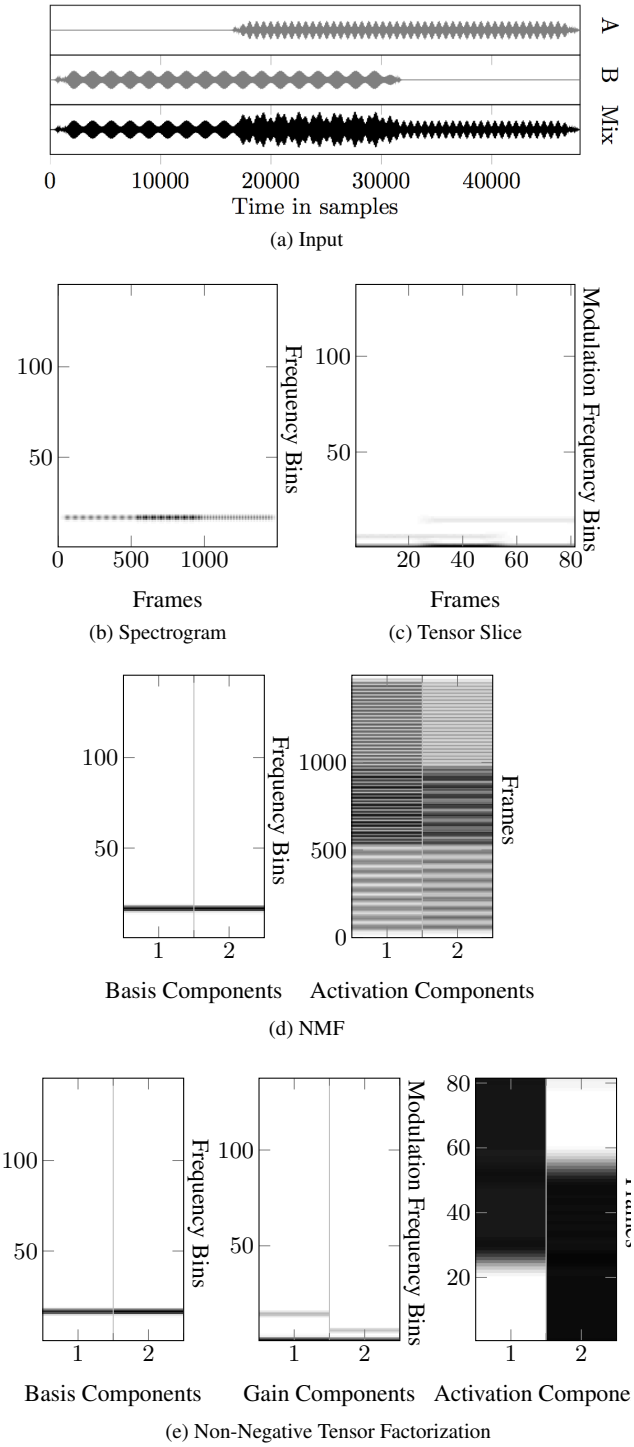
The decomposition of sources with overlapping partials are covered in several other publications like [3] and [11] which are based on non-negative matrix factorization. Lin et. al. [12] address the problem by defining invariant timbre based features. We propose to address the problem from a different perspective and focus on analyzing the non-stationarities of the source signals. For most musical instruments, the non-stationary features are intentionally created, for instance with vibrato or tremolo effects, which make them valuable to track. These non-stationarities can be modeled or learned from the signals themselves.

In this work we assume that we can separate overlapping partials of the sources based on differences in amplitude and/or frequency modulation, resulting in the following model for a signal with P commonly modulated partials

$$x(n) = \sum_{p=1}^{P} \Big[\big(1 + a(n)\big) \cdot \sin\Big(2\pi f_{p,0}\big(n + \frac{1}{f_{1,0}} \sum_{m=m_0}^{n} f(m)\big) + \phi_{p,0}\Big)\Big], \tag{2}$$

where effectively the amplitude modulation is $a(n)$ and the frequency modulation of the first partial is $f(n)$.

[1] greek: with *one voice*



(a) Input

(b) Spectrogram

(c) Tensor Slice

(d) NMF

(e) Non-Negative Tensor Factorization

Figure 1: Example of separating a mixture of two amplitude modulated signals by NMF and Modulation-NTF.
(a) Mixture of two sinusoids at 440 Hz with AM of 4.7 Hz and 12.6 Hz (fs=8 kHz), **(b)** STFT (FFT length = 256), **(c)** Slice of Modulation Tensor (FFT length = 256), **(d)** $\mathbf{W} \times \mathbf{H}$ Result of Non-Negative Matrix Factorization ($\beta = 1$) after 100 iterations, **(e)** $\mathbf{G} \times \mathbf{A} \times \mathbf{S}$ Result of Non-Negative Tensor Factorization ($\beta = 1$) after 100 iterations,

3. SEPARATING BY AMPLITUDE MODULATION

Amplitude modulation is normally not present, isolated in acoustical instruments. However electric pianos like Rhodes or Wurlitzers can generate a tremolo effect. Using the amplitude modulation to separate mixtures has already been done in [13] which makes use of the concept of *Common Amplitude Modulation*.

CAM is effectively the property of harmonics that share the same amplitude modulation across the bins. One way of analyzing it is a modulation spectrogram which is a frequency-frequency representation of a time domain input signal. There are also other ways to generate a modulation spectrum. A complete signal representation can be archived by a modulation tensor which holds the modulation spectrograms for each time frame. Barker and Virtanen [14] found a way to utilize the modulation tensor for single channel source separation. Standard NMF models the spectrogram by the sum of K components which are each factored into frequency/basis and time/activations components:

$$\mathbf{X}_{n,m} \approx \sum_{k=1}^{K} \mathbf{W}_{\mathbf{n},\mathbf{k}} \times \mathbf{H}_{\mathbf{k},\mathbf{m}}. \tag{3}$$

Non-negative Tensor factorization approximates a modulation tensor by a product of three matrices containing the frequency/basis, time/activation signals, and the modulation gain for each component. Compared to [14] we choose to generate the modulation tensor in way that is simpler and easier to invert. Barker and Virtanen use a Gammatone filter bank and rectification to model the characteristics of the human auditory system. We used a two-stage DFT filter bank where the modulation domain is based on magnitude spectrograms. Although this can give perceptually less optimal results, each step can be directly inverted by using the complex representation. Barker already showed that the NTF based approach gives better results on speech signals. We found that this approach can be used to separate two instrument mixtures by their amplitude modulation characteristics and is therefore ideal for the unison scenario.

In Figure 1 we show the factorization of a simple amplitude modulated input signal for comparison. The signal consists of two sinusoids which are linearly mixed. Both share the same carrier frequency but have different amplitude modulation frequencies. We choose a factorization into $K = 2$ components. From the activation components one can see that NMF is not able to separate the two signals sufficiently. NTF gives a smoother activation matrix and is able to generate the output with the separated amplitude modulations on each sinusoid. The modulation frequency gain matrix shows the two modulation frequency templates and the DC-component.

4. SEPARATING BY FREQUENCY MODULATION

Frequency modulation caused by vibrato is a very common playing style for string instruments but also for woodwind and brass instruments. Vibrato is an effect that is well studied especially in musicology. Performers tend to perform a vibrato in the same way when repeating a performance. This can be exploited in source separation scenarios. Typically, vibratos have modulation frequencies (rates) which vary between 4 and 8 Hz. Additionally vibrato rates vary across different instruments. In [15] the vibrato width (frequency deviation) was found to be significantly different between violinists and violists performers.

As with the amplitude modulated case NMF lacks the ability to model time varying frequencies since the $\mathbf{W}$ matrix is stationary. Several extensions for NMF have been proposed to improve the decomposition quality. [16] proposes frequency dependent activations matrices, [11] has developed a system which can be described as shift invariant NMF. Another approach is to model the spectral pattern changes by Markov chains [3]. All these approaches attempt to model the non-stationary effects within the decomposition model. In this paper we propose a method that increases the stationarity of the signal in preprocessing step and then use the standard NMF for the decomposition.

We make use of *time-warping* which refers to a mapping of the linear time scale t to a warped time scale τ via a mapping function $\tau = w(t)$. To ensure a unique mapping, the mapping function needs to be strictly increasing. For the discrete time case the mapping can be achieved by a time-varying re-sampling of the linear (i.e. regularly sampled) time signal under consideration. The instantaneous sampling frequency then corresponds to the first derivative of the mapping function. Although the mapping can be done from any time-span I on the linear time scale to any time span J on the warped time scale, in the discrete time case it is advantageous to have the same number of samples in the linear and warped time domain. This ensures that the average sampling frequency is the same in both domains. Such time-warping approaches have already been proposed for different purposes such as transform-based audio coding [17]. As in these applications, we derive the mapping function from the varying instantaneous fundamental frequency in such a manner that the variation of the frequency is reduced or removed. To be more precise the actual information needed is not the absolute instantaneous fundamental frequency but only its change over time. The discrete time warp map $w[n]$ is then simply the scaled sum of the relative frequencies $f[n]$:

$$w[n] = N \frac{\sum_{l=0}^{n} f[l]}{\sum_{k=0}^{N} f[k]} \qquad 0 \leq n < N, \tag{4}$$

where N being the number of samples of the signal under consideration. From the requirements for the mapping function it follows that the relative frequency $f[n]$ has to be positive at all instants and preferably should not exhibit large jumps. For the mapping from linear to warped time now the linear domain sample points $s[\nu]$ for the regularly spaced samples $x[\nu]$ in the warped domain are found by inverting $w[n]$. These sample points are then used to re-sample the linear time domain samples $x[n]$ to the warped time domain samples $x[\nu]$, in our case by employing an 128 times oversampled FIR low-pass filter. This processing leads to a sampling rate contour which is proportional to the pitch contour. Or in other words, a fixed number of samples are obtained in each period of the signal with the varying fundamental frequency. Mutatis mutandis the sample points $s[\nu]$ can be used for the re-sampling from warped time domain to linear time domain.

In this paper the time-warping was done globally over the full lengths of the signals under consideration. The globally time-warped sample sequence was then used in the further processing steps. In Figure 2 we show the results of the warping process in the time domain.

A similar approach using frequency modulation to separate a harmonic source from a mixture was proposed in [18]. Here the individual lines are demodulated to the base band using a com-

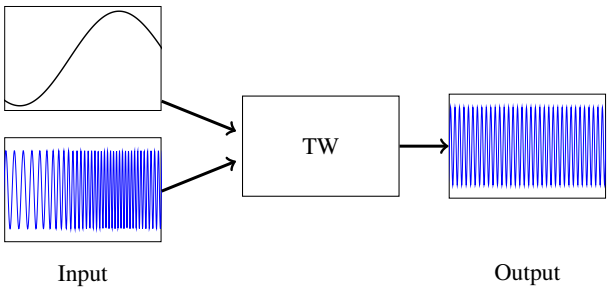


Figure 2: Example of applying warping to an input signal by using a frequency variation contour.

Instrument	Vibrato	General MIDI #
Violin	yes	40
Viola	yes	41
Violon Cello	yes	42
Trumpet	no	56
Trombone	no	57
Horn	no	60
Bariton Sax	yes	67
Oboe	no	68
Clarinet	no	71
Flute	yes	73

Table 1: Instrument item test set

bined frequency tracking/demodulation approach. The difference to our approach is that first the absolute instantaneous frequency for every harmonic line has to be known instead of a relative frequency that is common to all harmonic lines of a single source. This relative frequency might be obtained easier than its absolute value for a mixed signal. Secondly every harmonic line has to be individually frequency demodulated while in our approach the full signal is frequency demodulated in one algorithmic step.

4.1. Pitch Variation Informed Source Separation

With the ability to remove the frequency modulation from a signal we can then include this system in a source separation system to address the non-stationarity issues of NMF based approaches. Figure 4 shows how this system works on a harmonic FM signal mixture. Plots (a) and (b) show the two input signals which are linearly mixed (c). For each source the warp contour needs to be calculated. The mixture is then warped with pitch variation estimates of source 1 (d) and source 2 (e). The actual separation/filtering of the sources is then done by using NMF which is not shown here. To separate the components from the warped mixture we used NMF on a spectrogram computed with a very long DFT (about 0.5 s). NMF can work unsupervised by detecting the more tonal **W** component by using a spectral flatness measure. The separated signals (f) and (g) then need to be warped back into the original time domain resulting in (h) and (i).

It is important to clarify that this approach would not be able to separate two modulating instruments playing in unison without having prior knowledge about the individual modulation functions. Although a pitch variation estimate might be difficult to achieve in a mixture our approach shows that such a system can make sense.

5. EXPERIMENTS

We wanted to evaluate the methods proposed in Sections 3 and 4 so that they show the fundamental differences in their separation quality. Like in [14] we choose not to address the problem of clustering the components after the matrix factorization operation. Instead of processing mixtures in a $A-B-AB$ or $A-AB-B$ paradigm we went for a supervised learning phase where we had access to the original source individually. In this *oracle* supervised approach for each of the sources we then learned the spectral, temporal (for NMF), and modulation gain components (for MOD-NTF) and concatenated them. The learned coefficients were then used to initialize the final factorization process. This way we can achieve the maximum possible quality.

5.1. Test set

To build a test set we selected 10 instrumental items noted in Table 1. The items have each been generated by rendering C4 notes in a state of the art software sampler. All test have a duration of about three seconds. Items were equalized in loudness by using an iterative calculation of the loudness algorithm of the time varying Zwicker model. The implementation [19] was used. The 10 instrument items then generated 45 unique mixtures of two instruments each. The processing was done in 44.1 kHz / 16 bit.

5.2. Algorithms

The test set was processed by three algorithms: standard NMF, pitch variation informed NMF (PVI-NMF) (Section 4.1) and the non-negative tensor factorization based on modulation spectra (MOI NTF) as described in Section 3). All factorizations for NMF and NTF were computed by minimizing the $\beta = 1$ divergence (Kullback Leibler divergence). The Pitch Variation Informed-NMF (PVI-NMF) has been set up in the same way as the other algorithms. We choose to calculate results with $K = 2$ and $K = 4$. The pitch variation estimator is based on a method that was proposed by Bäckström in 2009 [20] with a subsequent post-processing to ensure the smoothness of the mapping.

Each of the algorithms did perform on the same filter bank and with the same sample rate. NMF approach did use a 2048 STFT with 512 samples hop size. For the MOD-NTF a second STFT based filter bank was used with 256 sample DFT size and 64 sample hop size. All methods use soft masking / wiener filtering for the actual synthesis.

5.3. Results

The results were evaluated by using commonly used evaluation measures provided by the PEASS Toolbox [21]. The evaluation measure are:

- Overall Perceptual Score (OPS)
- Target-related Perceptual Score (TPS)
- Interference-related Perceptual Score (IPS)
- Artifacts-related Perceptual Score (APS)
- Signal to Distortion Ratio (SDRi)
- Source to Interference Ratio (SIRi)

- Sources to Artifacts Ratio (SARi) [2]

The mean values of the PEASS evaluation are provided in Table 2. It can be seen that the SDR values give a different tendency than the OPS score, showing that the differences between both measures are substantial. Since unison mixtures are even very challening for humans to segregate we chose to focus on the psycho-acoustically weighted performance measures only. The results show a slightly better overall performance for the PVI-NMF. A more fine grained overview from the OPS results experiment is presented in Figure 3. It can be seen that results vary a lot between the mixtures. The modulation tensor factorization (MOD-NTF) performs good on mixtures like Clarinet-Viola (71-41) or Clarinet-Cello (71-42) where one source has vibrato and the other does not (see plots (e,f)). Although it performs well on average, MOD-NTF shows a high variance in the OPS results. The results have also been evaluated and confirmed subjectively by informal listening. Additionally we provide selected stimuli online on an acompanying webpage [3]. In general the PEASS scores give a good indication of quality. However the artifacts that are introduced by the standard NMF synthesis seem to be not well reflected. One possible reason is that PEASS toolbox has not been tested on artifacts from unison mixtures.
Future work could include a robust multi pitch variation estimator for musical instruments. Salamon and Gomez [22] describe the current state of the art of f0 estimation. Some approaches use source separation to estimate multiple f0 pitch tracks. Therefore our approach shows that a robust multi pitch f0 estimate can also help to improve source separation. In the future an iterative multi-step procedure could lead to better results in both problem domains.

6. CONCLUSIONS

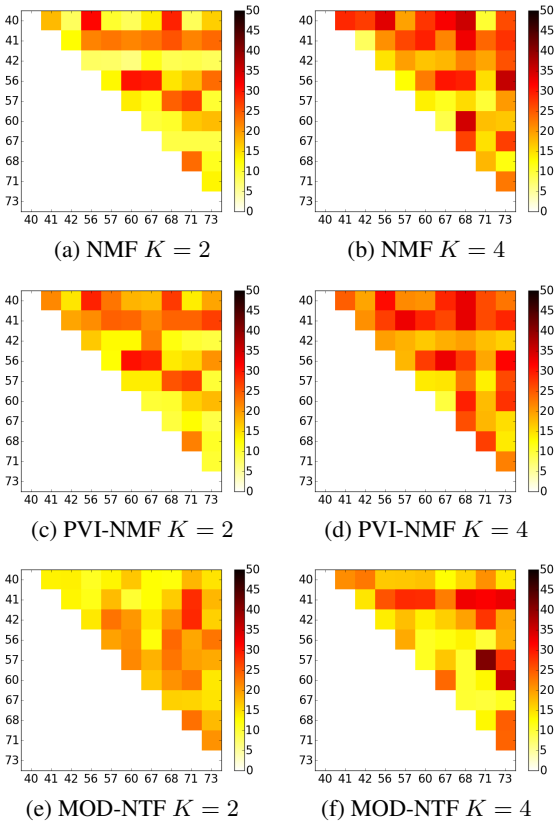
This paper proposes a new source separation scenario for instruments played in unison. It highlights the time-varying aspects of the signal sources like amplitude or frequency modulations. By addressing these aspects, the separation quality for non-unison mixtures can generally be improved, too. Furthermore we present two methods to decompose those mixtures based on differences in the amplitude or frequency modulation of the sources. One is using a method already published based on a modulation tensor factorization. The other is a novel method that uses an estimate of the pitch variation of the two input sources to warp the mixture. Within the warped domain the frequency modulation of the desired source is removed so that the sources can be separated more easily from the mixture. The results of 45 mixtures have been evaluated by using the PEASS toolbox. The scores indicate an improvement of about 2 OPS points in favor of the pitch variation informed NMF compared to the standard NMF.

[2] The i indicates that these scores have been calculated by decomposition with PEASS [21] instead of BSS EVAL.

[3] `http://www.audiolabs-erlangen.de/resources/2014-DAFx-Unison/`

Algorithm	NMF	PVI-NMF	MOD-NTF
OPS	15.76	**17.64**	17.35
TPS	30.17	32.80	**34.03**
IPS	26.07	**27.03**	22.73
APS	46.14	**54.74**	46.06
SDRi	**2.96**	2.54	2.20
SIRi	2.31	1.80	**3.13**
SARi	22.87	23.35	**26.09**

Table 2: Results from Evaluation with PEASS 2.0 Toolbox [21]. Best performing algorithm is marked bold.



(a) NMF $K = 2$ (b) NMF $K = 4$

(c) PVI-NMF $K = 2$ (d) PVI-NMF $K = 4$

(e) MOD-NTF $K = 2$ (f) MOD-NTF $K = 4$

Figure 3: Results of Overall Perceptual Score. Each matrix represents the mean OPS values for each individual mixture of two sources. The x and y axis represent the instrument IDs in General MIDI notation (See Table 1).

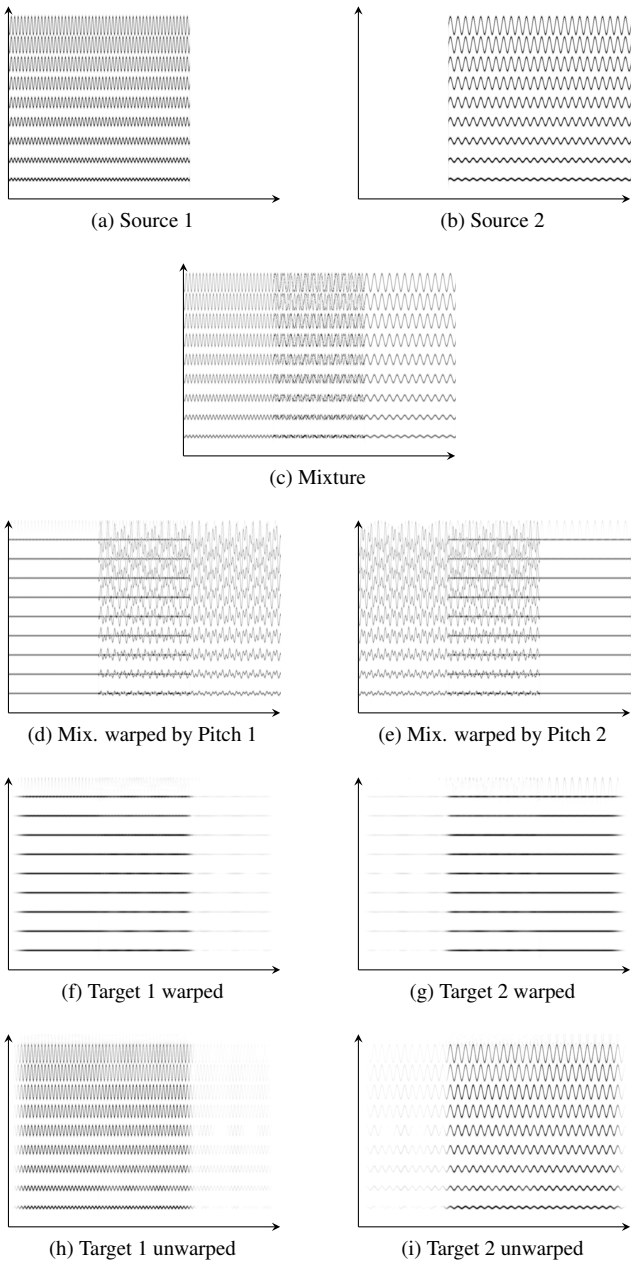


Figure 4: Example of pitch variation informed NMF in the warped domain. *Time* is shown on horizontal axes. *Frequency* is shown on vertical axes.

7. REFERENCES

[1] Emmanuel Vincent, Shoko Araki, Fabian Theis, Guido Nolte, Pau Bofill, Hiroshi Sawada, Alexey Ozerov, Vikrham Gowreesunker, Dominik Lutter, and Ngoc Q. K. Duong, "The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges," *Signal Processing*, vol. 92, no. 8, pp. 1928–1936, 2012.

[2] Nobutaka Ono, Zbynek Koldovsky, Shigeki Miyabe, and Nobutaka Ito, "The 2013 signal separation evaluation campaign," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2013, pp. 1–6.

[3] Masahiro Nakano, Jonathan Le Roux, Hirokazu Kameoka, Yu Kitano, Nobutaka Ono, and Shigeki Sagayama, "Nonnegative matrix factorization with markov-chained bases for modeling time-varying patterns in music spectrograms," in *Latent Variable Analysis and Signal Separation*, pp. 149–156. Springer, 2010.

[4] Paris Smaragdis and Judith C Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2003, pp. 177–180.

[5] Tuomas Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[6] Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1118–1133, 2012.

[7] Sebastian Ewert and Meinard Müller, "Using score-informed constraints for NMF-based source separation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012, pp. 129–132.

[8] D. Huron, "Voice denumerability in polyphonic music of homogeneous timbres," *Music Perception*, pp. 361–382, 1989.

[9] Fabian-Robert Stöter, Michael Schoeffler, Bernd Edler, and Jürgen Herre, "Human ability of counting the number of instruments in polyphonic music," in *Proceedings of Meetings on Acoustics*. Acoustical Society of America, 2013, vol. 19.

[10] Michael Schoeffler, Fabian-Robert Stöter, Harald Bayerlein, Bernd Edler, and Jürgen Herre, "An experiment about estimating the number of instruments in polyphonic music: a comparison between internet and laboratory results," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[11] Paris Smaragdis, Bhiksha Raj, and Madhusudana VS Shashanka, "Sparse and shift-invariant feature extraction from non-negative data.," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008, pp. 2069–2072.

[12] Yiju Lin, Wei-Chen Chang, Tien-Ming Wang, Alvin WY Su, and Wei-Hsiang Liao, "Timbre-constrained recursive time-varying analysis for musical note separation," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*, 2013, pp. 2–6.

[13] Yipeng Li, John Woodruff, and DeLiang Wang, "Monaural musical sound separation based on pitch and common amplitude modulation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1361–1371, 2009.

[14] Tom Barker and Tuomas Virtanen, "Non-negative tensor factorisation of modulation spectrograms for monaural sound source separation," in *Proceedings of INTERSPEECH*, 2013.

[15] Rebecca Bowman MacLeod, "Influences of dynamic level and pitch height on the vibrato rates and widths of violin and viola players," 2006.

[16] Romain Hennequin, Roland Badeau, and Bertrand David, "NMF with time–frequency activations to model nonstationary audio events," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 744–753, 2011.

[17] Bernd Edler, Sascha Disch, Stefan Bayer, Fuchs Guillaume, and Ralf Geiger, "A Time-Warped MDCT Approach to Speech Transform Coding," in *126th AES Convention*, Munich, Germany, May 2009, Preprint 7710.

[18] Avery Wang, "Instantaneous and frequency-warped techniques for source separation and signal parametrization," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (ASSP)*, 1995, pp. 47–50.

[19] "GENESIS S.A.: Loudness toolbox (version 1.2)," 2012.

[20] Tom Bäckström, Stefan Bayer, and Sascha Disch, "Pitch variation estimation," in *Proceedings of INTERSPEECH*, 2009, pp. 2595–2598.

[21] Valentin Emiya, Emmanuel Vincent, Niklas Harlander, and Volker Hohmann, "Subjective and objective quality assessment of audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2046–2057, 2011.

[22] Justin Salamon and Emilia Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759–1770, 2012.

[23] Romain Hennequin, Roland Badeau, and Bertrand David, "Time-dependent parametric and harmonic templates in non-negative matrix factorization," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2010, pp. 246–253.

[24] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[25] Estefanía Cano, Christian Dittmar, and Gerald Schuller, "Rethinking sound separation: Prior information and additivity constraint in separation algorithms," in *Proceedings of the 16th Int. Conference on Digital Audio Effects (DAFx)*, 2013.

[26] Alexey Ozerov, Ngoc Q. K. Duong, and Louis Chevallier, "Weighted nonnegative tensor factorization: on monotonicity of multiplicative update rules and application to user-guided audio source separation," Tech. Rep., 2013.

[27] Kazuyoshi Yoshii, Ryota Tomioka, Daichi Mochihashi, and Masataka Goto, "Beyond NMF: Time-domain audio source separation without phase reconstruction," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

A VERY LOW LATENCY PITCH TRACKER FOR AUDIO TO MIDI CONVERSION

Olivier Derrien,

Université de Toulon & CNRS LMA
Laboratoire de Mechanique et d'Acoustique
31 chemin Joseph-Aiguier, 13402 Marseille Cedex 20
derrien@lma.cnrs-mrs.fr

ABSTRACT

An algorithm for estimating the fundamental frequency of a single-pitch audio signal is described, for application to audio-to-MIDI conversion. In order to minimize latency, this method is based on the ESPRIT algorithm, together with a statistical model for partials frequencies. It is tested on real guitar recordings and compared to the YIN estimator. We show that, in this particular context, both methods exhibit a similar accuracy but the periodicity measure, used for note segmentation, is much more stable with the ESPRIT-based algorithm. This allows to significantly reduce ghost notes. This method is also able to get very close to the theoretical minimum latency, i.e. the fundamental period of the lowest observable pitch. Furthermore, it appears that fast implementations can reach a reasonable complexity and could be compatible with real-time, although this is not tested is this study.

1. INTRODUCTION

MIDI (Musical Interface for Digital Instruments) is the most widely used standard for connecting digital instruments. It specifies both the hardware interface and the data transmission protocol. It allows for instance to encode a melody as a collection of notes (note starting points, durations, pitches...) and to control a compatible synthesizer with an external interface, for instance a digital keyboard or a "wind controller" which mimics a wind instrument. However, for some instruments like guitars, designing an appropriate digital controller is difficult. Then, the original acoustic instrument can be used as a MIDI controller by adding an audio-to-MIDI converter. Such a device basically consists of a microphone that captures the acoustic signal produced by the instrument and a pitch-tracker which estimates the evolution of pitch during time. For guitars, one usually uses an under-saddle pickup for each string, connected to a series of monophonic pitch trackers, one for each string [1]. Such MIDI converters have been marketed for a few decades, but suffer from many flaws: latency, ghost notes, octave errors... The performance constantly improve, but latency still remains an issue: With an up-to-date Roland GR55 (built-in audio-to-MIDI converter and synthesizer) connected to a compatible acoustic guitar (Godin Multiac), we measured an average latency of 50 ms between the output of the under-saddle pickups and the audio output of the synthesizer (constant over the guitar frequency-range). Thus, playing a guitar synth is not easy and requires to develop specific skills.

In this study, we focus on the issue of latency for monophonic pitch tracking. We assume that pitch estimation is similar to fundamental frequency detection (noted f_0), and that the observed signal is harmonic. It appears that latency has several sources that add up. First, the "algorithmic delay", which is inherent to the pitch-detection algorithm. It corresponds to the length of the time-interval that is required for the algorithm to give an accurate estimation. This delay has a fundamental lower bound which related to the minimum f_0 value than can be detected. For a "Spanish" guitar[1], the minimum f_0 value is approximately 80 Hz, which corresponds to a minimum delay of 12.5 ms. Then, the "computational delay", which is the time required by the digital signal processor (DSP) to perform the pitch estimation. This delay can be reduced by increasing the speed of the DSP.

The issue of pitch detection is a classical problem and many algorithms have been proposed in the past decades. These methods can be roughly classified in two categories: time-domain and frequency-domain. Time-domain methods usually consist of finding a maximum of the auto-correlation function (or another similar function), while frequency-domain methods rely on a spectral analysis stage followed by a peak-picking stage. It was proved that time-domain methods are usually more efficient for real-time estimation of single-pitch [2]. Especially, the YIN algorithm, proposed by de Cheveigné et al. [3], can be considered as a reference f_0 estimator. It is based on the observation of a "cumulative mean normalized difference function", which is characterized by dips at the time-lags corresponding to the periodicity. This method is accurate, has a moderate complexity and a relatively low algorithmic delay. In [2], the delay of the full method was estimated around 30 ms for a "Spanish" guitar. However, this value is still approximately twice the theoretical minimum delay.

In this paper, we consider a new approach to reduce the algorithmic delay. Most f_0 estimators are non-parametric methods in the sense that they do not use *a priori* information about the signal. In contrast, parametric methods, which rely on a signal model, are known to be more precise when the observed signal correctly fits the model, but usually fail in the opposite case. For that reason, non-parametric methods are often considered more robust. However, audio signals coming from an under-saddle guitar pickup usually produce a quasi-harmonic sound with a very low noise, which justifies the use of a parametric method based on a sinusoidal signal model. In this study, we choose the Exponentially Damped Sinusoidal (EDS) model. The model parameters are estimated with a method derived from the ESPRIT algorithm [4]. This phase is similar to a spectral analysis and a peak-picking stage. To fulfill the pitch estimation, we use a spectral f_0 estimator inspired by the one proposed by Doval et al. [5]. Algorithms derived from ESPRIT are known for their good frequency resolution, but also have the reputation to require high computation time. However, fast algorithms have been proposed in the last decade [6, Chapter

[1] A "Spanish" guitar means a 6 string instrument tuned to the standard scale E-A-D-G-B-E. Thus, the lowest note is E2, corresponding to a fundamental frequency of 82.41 Hz.

V] which exhibit a complexity not much higher than a FFT. Thus, this method is theoretically suitable for real-time implementation, although this is not tested is this study.

This paper is organized as follows: In a first part, we consider more precisely the issue of algorithmic delay in a f_0 estimator. In a second part, we describe the proposed method. In a third part, we give results obtained from real guitar sounds both for our algorithm and for the YIN estimator, concerning pitch accuracy and delay. In the last part, we draw conclusions.

2. THE ISSUE OF ALGORITHMIC DELAY

Most f_0 estimators are frame-based methods. An input buffer of N samples is used, and the estimation of f_0 is made every a samples ($a \in \mathbb{N} \setminus \{0\}$). In other words, a sliding analysis window of N samples is used, with a hop-size a. The f_0 estimator should ideally be associated to the estimation of a "periodicity measure", i.e. whether the signal is pitched or not. A common periodicity measure is obtained by computing the energy of the periodic components in the signal, called "voiced" components in the case of a speech signal [7]. Such a periodicity measure influences the accuracy of the note segmentation process: A simple way to detect notes is to threshold the periodicity measure.

It is often believed that the algorithmic delay is equal to the window length, but this more complex. As exemplified on figure 1, the algorithmic delay corresponds to the time-interval between the beginning of a note and the last sample of the first window for which f_0 estimate is accurate (and eventually the periodicity measure is higher than the threshold). Sometimes, the algorithm returns the accurate f_0 even if the pitched signal does not "fill" the window (plotted case). Then, the delay can be shorter that N samples. Sometimes, the estimator takes some time to return the accurate f_0 and the delay can be longer that N samples.

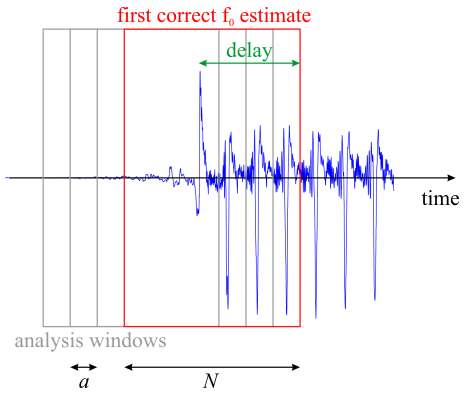


Figure 1: *Measurement of the algorithmic delay.*

As explained in [3], performing a correct f_0 estimation requires that the window length is no shorter than the largest expected period. But, according to the well known rule of thumb, a correct estimation requires enough signal to cover twice the largest period. Thus, the minimum delay is equal to $1/f_0^{\min}$, $f_0^{\min}$ being the lowest f_0 value than can be detected. As explained previously, this corresponds to approximately 12.5 ms for a "Spanish" guitar. As a consequence, the window length N must be higher that $f_s/f_0^{\min}$ where f_s is the sampling frequency. But practically, we expect a minimum delay of 25 ms.

3. THE PROPOSED METHOD

This method is divided in two stages: in the first one, the most significant sinusoidal components are extracted according to a signal model. Then, in the second stage, the most probable fundamental frequency is estimated using a statistical model.

3.1. Sinusoidal modeling

In this part, we describe the signal model and the estimation algorithm. Both have been extensively discussed in the literature. We choose to reproduce this description from a previous work [8] in order to render the paper self-contained.

In the EDS model, the signal to be analyzed is written:

$$x[n] = s[n] + w[n], \tag{1}$$

where the deterministic part $s[n]$ is a sum of K damped sinusoids:

$$s[n] = \sum_{k=0}^{K-1} \alpha_k z_k^n. \tag{2}$$

Complex amplitudes are defined as $\alpha_k = a_k\, e^{i\phi_k}$ (containing initial amplitude a_k and phase ϕ_k), and poles are defined as $z_k = e^{-d_k+2i\pi\nu_k}$ (containing damping d_k and normalized frequency ν_k). The stochastic part $w[n]$ is a gaussian white noise.

The estimation algorithm consists in finding the best values of K, α_k and z_k for a given signal in the least square sense. In this study, an estimation algorithm proposed by Badeau *et al.* [4] is used, which is derived from the ESPRIT algorithm. The principle consists of performing an SVD on an estimate of the signal correlation matrix. The eigenvectors corresponding to the K highest eigenvalues correspond to the so-called *signal space*, while the remaining vectors correspond to the so-called *noise space*. The shift invariance property of the signal space allows a simple solution for the optimal poles values z_k. Then, the amplitudes α_k can be recovered by solving a standard least square problem. The algorithm can be described as follows:

We define the signal vector:

$$\boldsymbol{x} = \begin{bmatrix} x[0] & x[1] & \dots & x[N-1] \end{bmatrix}^T, \tag{3}$$

where N is the length of the analysis window. We assume that N is even. The Hankel signal matrix is defined as:

$$\boldsymbol{X} = \begin{bmatrix} x[0] & x[1] & \dots & x[Q-1] \\ x[1] & x[2] & \dots & x[Q] \\ \vdots & \vdots & & \vdots \\ x[R-1] & x[R] & \dots & x[N-1], \end{bmatrix}, \tag{4}$$

where $Q, R > K$ and $Q + R - 1 = N$. $Q \approx R$ was proved to be an efficient solution, thus we choose $Q = N/2$ and $R = N/2+1$. We also define the amplitude vector:

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_0 & \alpha_1 & \dots & \alpha_{K-1} \end{bmatrix}^T, \tag{5}$$

and the Vandermonde matrix of the poles:

$$\boldsymbol{Z}^N = \begin{bmatrix} 1 & 1 & \dots & 1 \\ z_0 & z_1 & \dots & z_{K-1} \\ \vdots & \vdots & \vdots & \vdots \\ z_0^{N-1} & z_1^{N-1} & \dots & z_{K-1}^{N-1} \end{bmatrix}. \tag{6}$$

Performing a SVD on $\boldsymbol{X}$ leads to:

$$\boldsymbol{X} = [\boldsymbol{U_1 U_2}] \begin{bmatrix} \boldsymbol{\Sigma_1} & 0 \\ 0 & \boldsymbol{\Sigma_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{V_1} \\ \boldsymbol{V_2} \end{bmatrix}, \tag{7}$$

where $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are diagonal matrices containing respectively the K largest singular values and the remaining singular values. $[\boldsymbol{U_1 U_2}]$ and $[\boldsymbol{V_1 V_2}]$ are respectively the corresponding left and right singular vectors. The shift-invariance property of the signal space yields to:

$$\boldsymbol{U}_1^{\downarrow} \boldsymbol{\Phi}_1 = \boldsymbol{U}_1^{\uparrow}, \qquad \boldsymbol{V}_1^{\downarrow} \boldsymbol{\Phi}_2 = \boldsymbol{V}_1^{\uparrow}, \tag{8}$$

where the poles are eigenvalues of matrix $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$. $(.)^{\uparrow}$ and $(.)^{\downarrow}$ respectively stand for the operators that discard the first line and the last line of a matrix. Here, we estimate:

$$\boldsymbol{\Phi}_1 = (\boldsymbol{U}_1^{\downarrow})^{\dagger} \, \boldsymbol{U}_1^{\uparrow}, \tag{9}$$

where $(.)^{\dagger}$ denotes the pseudoinverse operator. The estimates of z_k are obtained by diagonalization of $\boldsymbol{\Phi}_1$. The associated Vandermonde matrix $\boldsymbol{Z}^N$ is computed. Finally, the estimates of amplitudes with respect to the least square criterion are obtained by:

$$\boldsymbol{\alpha} = (\boldsymbol{Z}^N)^{\dagger} \boldsymbol{x}. \tag{10}$$

Badeau *et al.* also proposed a criterion (ESTER) which measures the adequacy between the signal and the model [9]. It is based on the fact that equations (8) are strictly verified only when the signal exactly follows the EDS model defined by equation (2) without noise. In the general case, a distance between $\boldsymbol{U}_1^{\uparrow}$ and $\boldsymbol{U}_1^{\downarrow} \boldsymbol{\Phi}_1$ can be used to measure the model error. Is was observed that the original ESTER criterion naturally favors low values for the model order K. In order to minimize this effect, we propose a modified version of this criterion:

$$J = \frac{(K-1)^2}{\| \boldsymbol{U}_1^{\uparrow} - \boldsymbol{U}_1^{\downarrow} \boldsymbol{\Phi}_1 \|^2}. \tag{11}$$

The numerator simply performs a normalization of the denominator by the size of the matrix inside the norm. A high value for J means a good match between the signal and the model. This criterion can be used to automatically determine the best model order K, or in our case, to derive a periodicity measure.

3.2. Fundamental frequency estimation

It is assumed that each damped sinusoid in the EDS decomposition corresponds to a partial. Its frequency is related to the pole estimate by $f_k = f_s \nu_k = \frac{f_s}{2\pi} \arg(z_k)$. Doval *et al.* proposed a statistical method that allows estimating the most probable fundamental frequency of a harmonic signal given a set of partials [5]. The main idea is to compute a likelihood function of the fundamental frequency based on a probabilistic model of the observed partials. The best estimate for the fundamental frequency is the global maximum of this function. In the original method, the statistical model is elaborated and has many parameters. Estimating these parameters requires a learning database of recorded notes. Furthermore, the computation of the likelihood function can be time-consuming.

With our application, a low-complexity algorithm is desirable. We also wish that our method does not depend on a learning database. Thus, we modify the model in order to reduce the complexity and to minimize the number of parameters. In particular, the distribution of energy between partials is not modeled. This probably degrades the efficiency of the f_0 estimation compared to the original method, but it happens to be sufficient for this application.

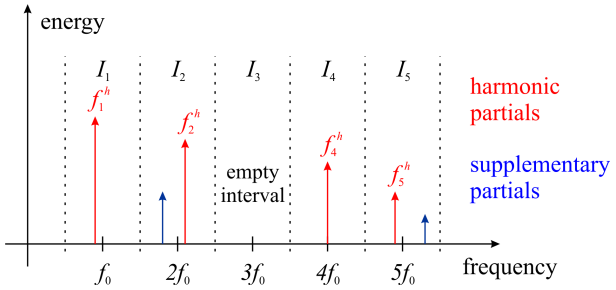


Figure 2: *Classification of partials for a given f_0.*

For a given value of f_0, we define a set of frequency intervals I_m centered on mf_0:

$$I_m = \left[\left(m - \frac{1}{2}\right) f_0, \left(m + \frac{1}{2}\right) f_0 \right], \quad m \in \mathbb{N} \setminus \{0\}, \tag{12}$$

which define a partition of the frequency scale. The partials are dispatched in these intervals according to their frequency f_k. Some intervals can contain several partials, and some others can be empty. In each non-empty interval, we define the most probable "harmonic partial" as the one which frequency is closer to mf_0, noted f_m^h. The others are called "supplementary partials" (see figure 2). The likelihood function is written as:

$$L(f_0) = \left[\prod_{m \in \mathcal{M}} g\left(\frac{f_m^h}{f_0} - m \right) \right] P_S(f_0) \, P_E(f_0), \tag{13}$$

where $\mathcal{M}$ is the set of indices m corresponding to non-empty intervals I_m. The first term is the *a posteriori* probability to observe the set of harmonic partials. The second and third terms, $P_S(f_0)$ and $P_E(f_0)$, are respectively the *a posteriori* probability to observe the set of supplementary partials and empty intervals. g is a probability function that models the frequencies of harmonic partials, which is assumed to be gaussian:

$$g\left(\frac{f_m^h}{f_0} - m \right) \propto e^{-\frac{1}{\sigma^2}\left(\frac{f_m^h}{f_0} - m\right)^2}. \tag{14}$$

σ^2 represents the variance of the reduced frequencies f_m^h / f_0 around the mean value m. $P_S(f_0)$ and $P_E(f_0)$ are estimated by:

$$P_s(f_0) = 1 - \left(\frac{N_S}{K} \right)^{\alpha_S}, \qquad P_E(f_0) = 1 - \left(\frac{N_E}{M} \right)^{\alpha_E}, \tag{15}$$

where N_S is the number of supplementary partials, N_E the number of empty intervals and M the total number of intervals. α_S and α_E are constants that allow adjusting the influence of N_S and N_E on the likelihood function.

Thus, when the frequencies of the harmonic-partials are close to mf_0, the likelihood increases. When the number of supplementary partials or empty intervals increases, the likelihood decreases. This method naturally avoids octave errors: A lower (resp. higher) octave generates supplementary partials (resp. empty intervals), which lowers the probability $P_S(f_0)$ (resp. $P_E(f_0)$) and finally lowers the likelihood. However, this requires a fine tuning on α_S and α_E.

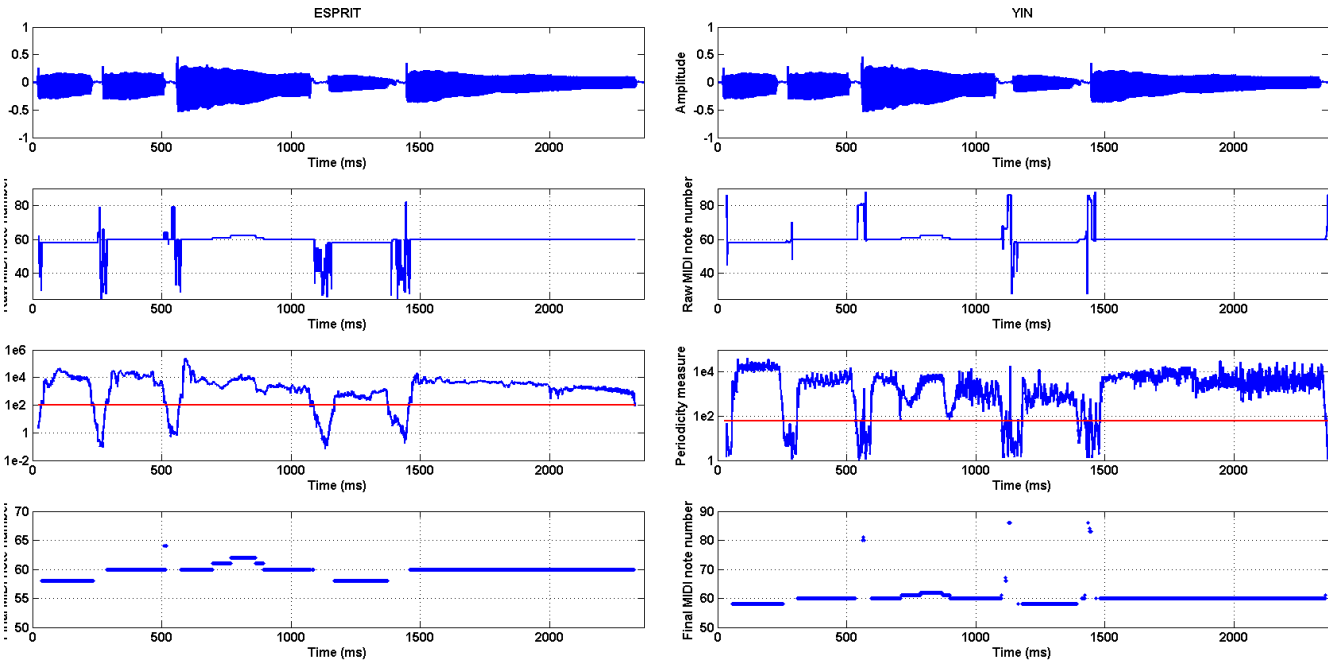


Figure 3: *Outputs of the estimation process for a short sequence of notes, ESPRIT (left column) and YIN (right column). First row: Signal waveform. Second row: MIDI note returned by the algorithm. Third row: Periodicity measure (blue) and threshold (red), log scale. Fourth row: Final MIDI note after thresholding the periodicity measure.*

3.3. Implementation details

The EDS estimation algorithm is applied on analysis segments without weighting function (also called "rectangular analysis window" in the literature). The model order K can be set to a constant, or one can define maximum and minimum values and use the ESTER criterion to select the optimal order. From our experiments on isolated guitar notes, it appears that $K = 6$ is good choice for a constant. Otherwise, K may vary between 4 and 12. Choosing a constant K saves execution time but this is sub-optimal. More precisely, bass notes usually have more harmonics than treble notes. Observing a larger set of harmonics on bass notes is desirable because these notes are more difficult to detect (there are fewer fundamental periods in the analysis window), and a larger set of harmonics gives a more robust estimation of f_0.

A periodicity measure ideally measures the energy of the periodic component in the signal. The criterion J defined in equation (11) is simply a ratio (without dimension) that measures the signal-to-model adequation. Thus, we derive our periodicity measure by multiplying J by the energy of the signal in the analysis window.

The estimation of f_0 implies computing the likelihood for all possible values of f_0. This can be accelerated by testing only discrete values, for instance on the tempered scale. If a finer estimation is required, a refinement stage can be added [5]. We set $\sigma = 1/8$, $\alpha_S = 8$ and $\alpha_E = 4$. This set of parameters appears to give a robust estimation over all the guitar frequency range. However, it is possible to choose a different set of parameters for each f_0 which could improve the detection accuracy.

Concerning complexity, ESPRIT is obviously the most critical part. A non-optimized version of ESPRIT has a complexity in $O(N^3)$ which is hardly suitable for real-time implementation. But a fast implementation of ESPRIT [6] has a complexity in $O(KN(K + \log(N)))$. When K is small, this reduces to $O(KN\log(N))$, which is not much more than a FFT. When the overlap between adjacent analysis windows is high, using adaptive algorithms allow to reduce again the complexity [6].

4. RESULTS AND DISCUSSION

In this section, we report test results for our algorithm and the YIN estimator on the same audio excerpts. The signal is the output of an under-saddle piezo-pickup on a solid-body acoustic guitar. The original signal is sampled at 44.1 kHz, downsampled at 11.025 kHz to reduce complexity. This appears to be sufficient for estimating the highest pitch on a Spanish guitar (between 930 and 1200 Hz). The implementation is in Matlab, and thus the estimation is an offline process. According to the results given in section 2, the minimum buffer length is $N = 138$ for $f_0^{\min} = 80$ Hz. We choose for both methods a hop-size of $a = 8$ samples, which corresponds to 0.72 ms.

For the YIN estimator, the author's implementation [10] is used. For the proposed method, the implementation of the ESPRIT-based estimator relies on the DESAM Toolbox [11], which is non-optimized. The minimum f_0 is set to 80 Hz for both methods. Buffer length was adjusted so that a correct estimation of f_0 is obtained for the whole guitar range. $N = 300$ is the minimal value for the YIN estimator, and $N = 260$ is the minimal value for the ESPRIT-based method. The YIN estimator gives a continuous frequency estimation that we round to the tempered scale. The new method is implemented only for discrete f_0 values corresponding to the tempered scale. Frequencies are then converted into MIDI note index for both methods. The periodicity measure in the case of YIN is the inverse of the so-called "aperiodicity mea-

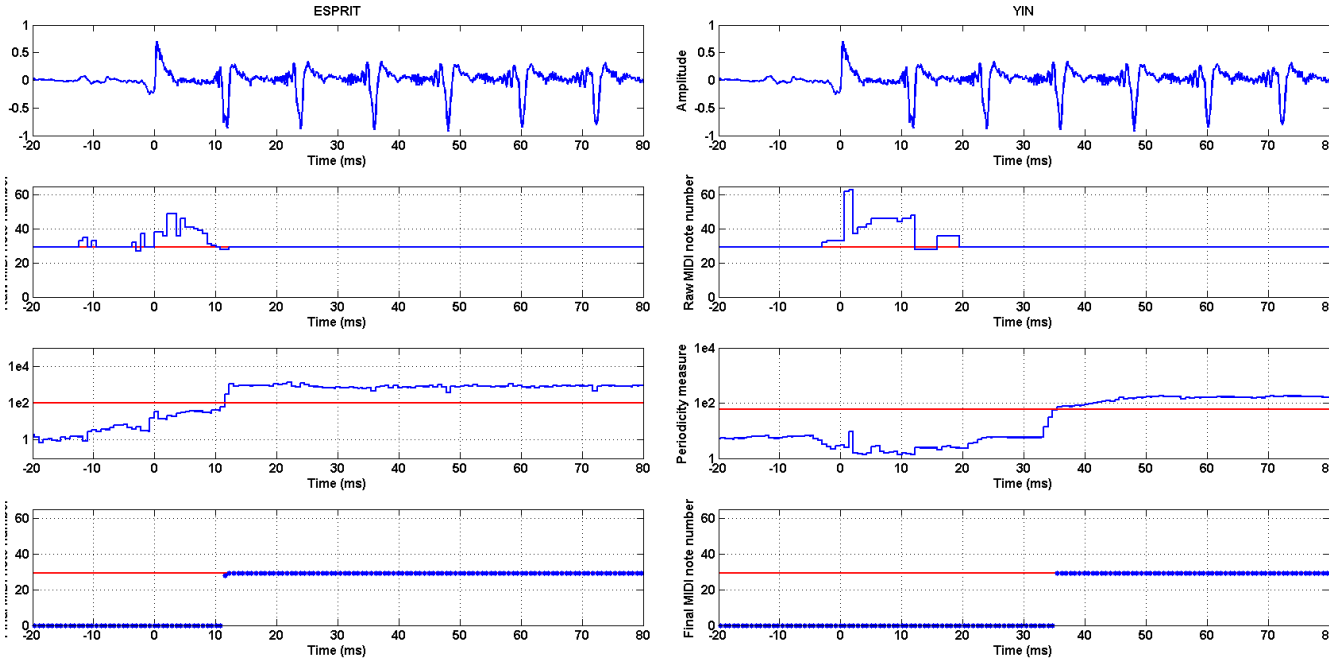


Figure 4: *Outputs of the estimation process for E2 note, ESPRIT (left column) and YIN (right column). First row: Signal waveform. Second row: MIDI note returned by the algorithm (blue) and theoretical value (red). Third row: Periodicity measure (blue) and threshold (red), log scale. Fourth row: Final MIDI note after thresholding the periodicity measure (blue) and reference value (red). Time origin is manually aligned with the onset.*

sure" returned by the algorithm [10]. For both methods, the note segmentation is obtained by thresholding the periodicity measure. The thresholds were adjusted empirically in order to get accurate segmentation on several test recordings. The threshold was set to 100 for ESPRIT and to 60 for YIN. Although, in a finely tuned application, a different threshold could be set for each string.

On figure 3, we plot the results for both algorithms on a sequence of high-pitched notes played on the same string, with a pitch-bend during the third note. The estimated MIDI note is accurate and stable for both methods when the signal is stationary. As expected, both return insignificant pitch values between the notes. The case of the periodicity measure is more contrasted: With the YIN algorithm, the periodicity evolves sharply in time, but is not very stable. One can not define a threshold on the periodicity measure that avoids ghost notes. With ESPRIT, the periodicity is more stable and an accurate thresholding can avoid most ghost notes, but it evolves more slowly in time.

On figure 4, we plot the results for a low-pitch single note (E2), which is the lowest note on a Spanish guitar, and zoom around the onset. With both methods, the estimated MIDI note is accurate and stable after a transition phase. As regards the algorithmic delay (we do not consider the computational delay in this section), the ESPRIT-based method returns the correct raw MIDI note after 12 ms, which is approximately one period of the signal (i.e. the theoretical minimum value), whereas YIN returns the correct raw MIDI note after 20 ms. However, one must take into account the periodicity measure to evaluate the actual delay. Both methods exhibit a raising front on the periodicity which allows a precise thresholding, approximately 12 ms after the onset for the ESPRIT-based method and 35 ms after the onset for the YIN algorithm. This is close to the value obtained by Knesebeck *et al.* in [2].

On figure 5, the results for a high-pitch single note (E4) are plotted. As expected, the results are globally similar to the previous case because the analysis parameters (especially the window size) did not change. However, one can see that the periodicity measure is less sharp with ESPRIT: there is a shelf between 12 and 24 ms that could extend the delay, or even generate ghost notes, depending on the threshold value. This can be explained by the fact that the signal exhibits a pseudo-periodicity before the onset that might come from the interaction between the string and the pick. The periodicity onset is sharper with YIN.

5. CONCLUSION

In this paper, an algorithm for estimating the fundamental frequency of single-pitch notes was described. The application to audio-to-MIDI conversion for guitar was especially considered. This application requires very-low algorithmic delay, which is still an issue with state-of-the-art pitch trackers. In order to minimize this delay, a new method was proposed. The first stage, equivalent to a spectral peak-picking algorithm, uses an algorithm from the literature derived form the ESPRIT method. The second stage is a fundamental frequency estimator inspired by the method proposed by Doval *et al.*, which consists in maximizing a likelihood function. The new method was tested on real guitar recordings and was compared to the YIN estimator proposed by de Cheveigné *et al.* which can be considered as a reference method. It was showed that, on this test material, both methods exhibit a similar accuracy, but it is important to notice that only the closest MIDI note was considered, and not the continuous fundamental frequency estimation. Concerning the periodicity measure which is used for

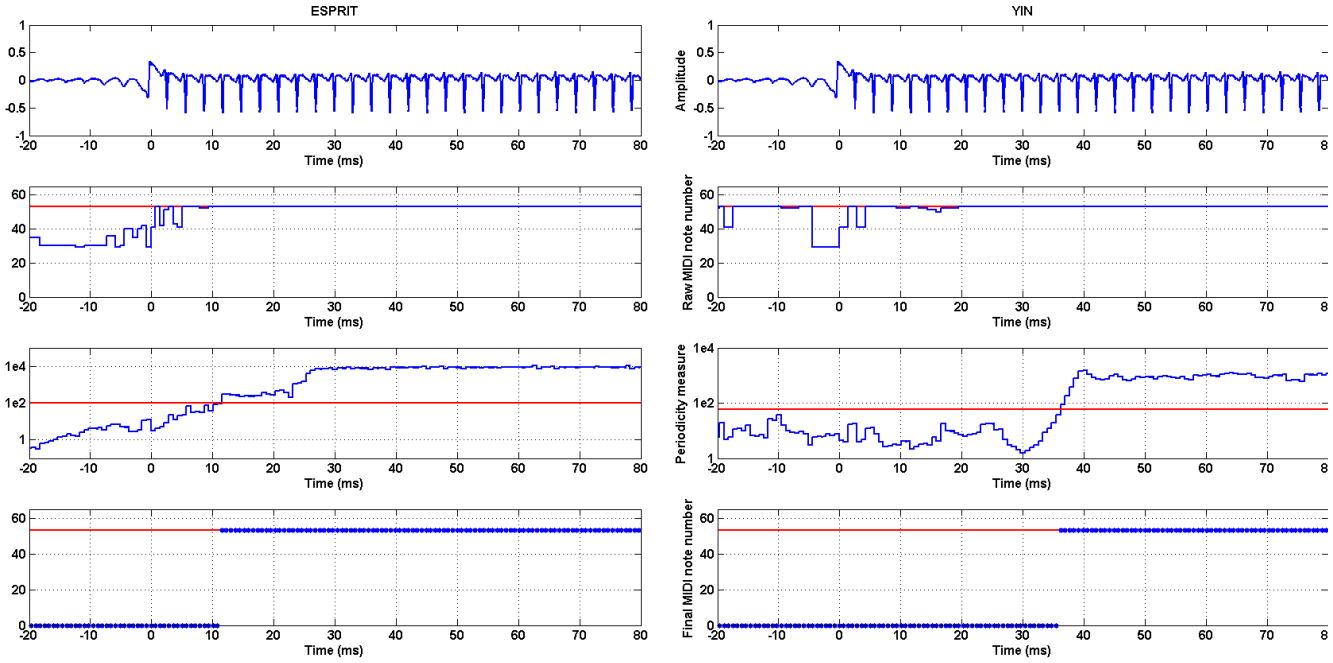


Figure 5: *Outputs of the estimation process for E4 note, ESPRIT (left column) and YIN (right column). First row: Signal waveform. Second row: MIDI note returned by the algorithm (blue) and theoretical value (red). Third row: Periodicity measure (blue) and threshold (red), log scale. Fourth row: Final MIDI note after thresholding the periodicity measure (blue) and reference value (red). Time origin is manually aligned with the onset.*

note segmentation, the new method was found more stable than the YIN estimator. This allows to significantly reduce ghost notes that are commonly observed in audio-to-MIDI conversion. It was also showed that the ESPRIT-based method is able to provide note tracking with an algorithmic delay that is very close to the theoretical limit, i.e. the fundamental period of the lowest observable pitch, which is not the case with the YIN method. For that reason, this new estimator may allow to significantly reduce the latency of audio-to-MIDI conversion. However, the issue of computational cost is crucial. The YIN estimator is a fast method, well suited for real-time implementation. In this preliminary study, our method was only tested off-line using a non-optimized implementation in Matlab. But theoretical studies have showed that fast implementations of the ESPRIT algorithm can reach a reasonable complexity in $O(KN\log(N))$ where K is the number of partials to be observed (typically 6) and N is the length of the analysis window (here less than 300 points). This is not much more than a FFT, which means that an optimized version would be theoretically compatible with real-time. This point will be investigated in the future.

6. REFERENCES

[1] U. Zölzer, Ed., *DAFX, Digital Audio Effects*, J. Wiley & Sons, New York, NY, USA, 2002.

[2] A. von dem Knesebeck and U. Zölzer, "Comparison of pitch trachers for real-time guitar effects," in *Proc. Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 2010.

[3] A. de Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 111, no. 4, pp. 1917–1930, Apr. 2002.

[4] R. Badeau, R. Boyer, and B. David, "EDS parametric modeling and tracking of audio signals," in *Proc. DAFx'02*, Hamburg, Germany, Sept. 2002.

[5] B. Doval and X. Rodet, "Estimation of fundamental frequency of musical sound signals," in *Proc. ICASSP'91*, Toronto, Ontario, Canada, May. 1991.

[6] Roland Badeau, *High resolution methods for estimating and tracking modulated sinusoids. Application to music signals.*, Ph.D. thesis, École Nationale Supérieure des Télécommunications, ENST2005E007, Paris, France, Apr. 2005, in French.

[7] G. Richard and C. d'Alessandro, "Analysis/synthesis and modification of the speech aperiodic component," *Speech Communication*, , no. 19, pp. 221–244, 1196.

[8] A. Sirdey, O. Derrien, R. Kronland-Martinet, and M. Aramaki, "Modal analysis of impact sounds with esprit in gabor frames," in *Proc. Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 2011.

[9] R. Badeau, B. David, and G. Richard, "Selecting the modeling order for the esprit high resolution method: an alternative approach," in *Proc. ICASSP'04*, Montreal, Quebec, Canada, May 2004.

[10] "The YIN algorithm documentation," Available at http://mroy.chez-alice.fr/yin/index.html.

[11] "The DESAM toolbox," Available at http://www.tsi.telecom-paristech.fr/aao/en/2010/03/29/desam-toolbox-2/.

TSM TOOLBOX: MATLAB IMPLEMENTATIONS OF TIME-SCALE MODIFICATION ALGORITHMS

Jonathan Driedger, Meinard Müller,

International Audio Laboratories Erlangen,*
Erlangen, Germany

{jonathan.driedger,meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Time-scale modification (TSM) algorithms have the purpose of stretching or compressing the time-scale of an input audio signal without altering its pitch. Such tools are frequently used in scenarios like music production or music remixing. There exists a large variety of different algorithmic approaches to TSM, all of them having their very own advantages and drawbacks. In this paper, we present the TSM toolbox, which contains MATLAB implementations of several conceptually different TSM algorithms. In particular, our toolbox provides the code for a recently proposed TSM approach, which integrates different classical TSM algorithms in combination with harmonic-percussive source separation (HPSS). Furthermore, our toolbox contains several demo applications and additional code examples. Providing MATLAB code on a well-documented website under a GNU-GPL license and including illustrative examples, our aim is to foster research and education in the field of audio processing.

1. INTRODUCTION

Time-scale modification (TSM) is the task of manipulating an audio signal such that it sounds as if its content was performed at a different tempo. TSM finds application for example in music remixing where it is used to adjust the playback speed of existing recordings such that they can be played simultaneously at the same tempo [1, 2]. Another field of application is the adjustment of the audio streams in video clips. For example, when generating a slow motion video, TSM can be used to synchronize the audio material with the visual content [3].

There exists a large variety of different TSM algorithms which all have their respective advantages and drawbacks. Some of the TSM procedures yield results of high perceptual quality only when applied to a certain class of audio signals. For example, 'classical' well-known TSM algorithms like WSOLA [4] or the phase vocoder [5, 6] are capable of preserving the perceptual quality of harmonic signals to a high degree, but introduce noticeable artifacts when modifying percussive signals. However, it has been shown that it is possible to substantially reduce artifacts by combining different TSM procedures. For example, in [7], a given audio signal is first decomposed into a harmonic and a percussive component. Afterwards, the two components are processed with different classical TSM algorithms, and final output signal is obtained by superimposing the two TSM results.

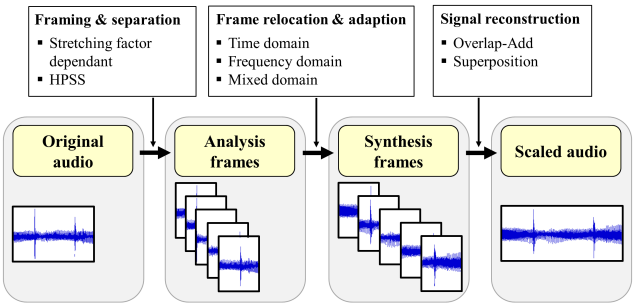


Figure 1: General processing pipeline of TSM procedures.

To foster research and to obtain a better understanding of TSM algorithms, we present in this paper the *TSM toolbox*. Published under a GNU-GPL license at [8], this self-contained toolbox serves various purposes. First, it delivers basic tools to work in the field of TSM. The toolbox includes well-documented reference implementations of the most important classical TSM algorithms within a unified framework. This not only allows users and researchers to get a better feeling for TSM results by experimenting with the algorithms, but also gives insights into implementation details and potential pitfalls. Second, to give an example of how those classical algorithms can be combined to improve TSM results, the toolbox also supplies the code of a recently proposed TSM approach based on *harmonic-percussive source separation* (HPSS), also including the code of the HPSS procedure itself. Third, the toolbox provides a MATLAB wrapper function for a commercial, proprietary, and widely used TSM algorithm. Because of its

* The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

'state-of-the-art' character, this is particularly interesting when conducting listening experiments which are the most common way of judging the perceptual quality of TSM results. Finally, the toolbox provides additional code for various example applications. Such applications include the automated generation of interfaces for comparing TSM results, the non-linear synchronization of audio recordings, and the pitch-shifting of audio signals. Although there already exist MATLAB implementations of individual TSM algorithms (for example [9, 10]), we believe that supplying an entire collection of different TSM approaches along with example applications within a unifying framework can be highly beneficial for both researchers as well as educators in the field of audio processing.

The remainder of this paper is structured as follows. In Section 2, we briefly review the basics of TSM in general as well as the TSM algorithms included in the TSM toolbox. Then, in Section 3, we describe the MATLAB functions contained in the toolbox. Some of the demo applications included in the toolbox are discussed in Section 4. Finally, in Section 5, we conclude this paper with some general remarks.

2. TIME-SCALE MODIFICATION

Most TSM procedures follow a common basic strategy which is sketched in Figure 1. Given an original audio signal x as an input, the first step of most TSM algorithms is to split up the waveform into short overlapping *analysis frames* which are spaced apart by an *analysis hopsize* H_a. In a second step, these frames are relocated on the time-axis to have a *synthesis hopsize* H_s and furthermore suitably adapted. While the relocation accounts for the actual modification of the time-scale of the audio signal, the objective of the adaption is to reduce possible artifacts introduced by the frame relocation. The modified frames, also known as *synthesis frames*, are then superimposed to form the output of the algorithm. The output signal is a time-scale modified version of the input signal x, altered in length by a constant *stretching factor* of $\alpha = H_s/H_a$.

The main differences between most procedures are therefore the strategies of how the analysis frames are chosen and how they are modified to form the synthesis frames. In the following, we review some of these strategies.

2.1. Overlap-Add (OLA)

One of the most basic TSM algorithms is known as *Overlap-Add* (OLA). In OLA, the synthesis frames are computed by just windowing the analysis frames with a window function w and not processing them any further. Although OLA is very efficient, adding up the unmodified synthesis frames usually introduces phase discontinuities into the output signal. Periodic, and therefore harmonic structures in the input signal are not preserved (see Figure 2). Perceptually, this manifests itself as strong harmonic artifacts in the output signal. However, especially when choosing the length of the analysis frames to be very short, OLA is particularly successful in preserving percussive sounds. This can be seen for example in Figure 3. Note that the sharp peak-like onsets which are visible in the original waveform (see Figure 3a) are preserved well by OLA (see Figure 3b).

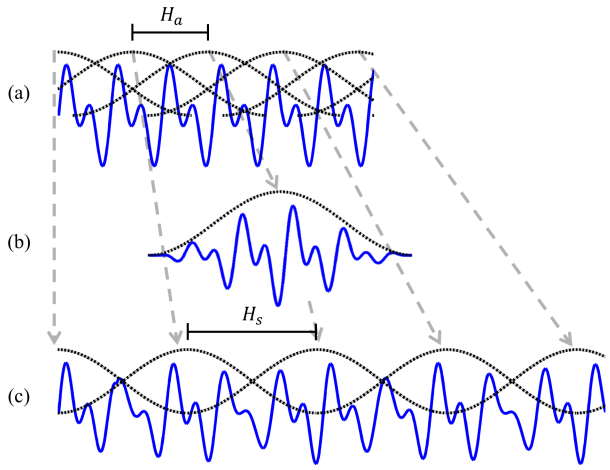


Figure 2: The principle of OLA TSM. **(a)**: Input signal x (solid line). The analysis frames are indicated by the window functions (dotted lines). **(b)**: One synthesis frame. **(c)**: Output signal as sum of all synthesis frames.

2.2. Waveform Similarity Overlap-Add (WSOLA)

One way of avoiding phase discontinuities as introduced by OLA is to choose the analysis frames such that successive synthesis frames better fit together when adding them up. The *Waveform Similarity Overlap-Add* algorithm (WSOLA) [4] achieves this by introducing an *analysis frame position tolerance* $\Delta_{\max}$. The position of each analysis frame in the input signal may be shifted on the time-axis by some $\Delta \in [-\Delta_{\max}:\Delta_{\max}]$ such that the waveforms of two overlapping synthesis frames are as similar as possible in the overlapping regions. Afterwards, the frames are windowed as in OLA and added up to form the output signal. Note that WSOLA reduces to OLA when using $\Delta_{\max} = 0$. The introduced tolerance for the analysis frames strongly reduces artifacts resulting from phase discontinuities. However, especially at transients in the input signal, the algorithm introduces noticeable *stuttering artifacts* in the output signal. These artifacts originate from shifted frame positions which tend to *cluster* around transients in the input signal. In the output signal, the transients

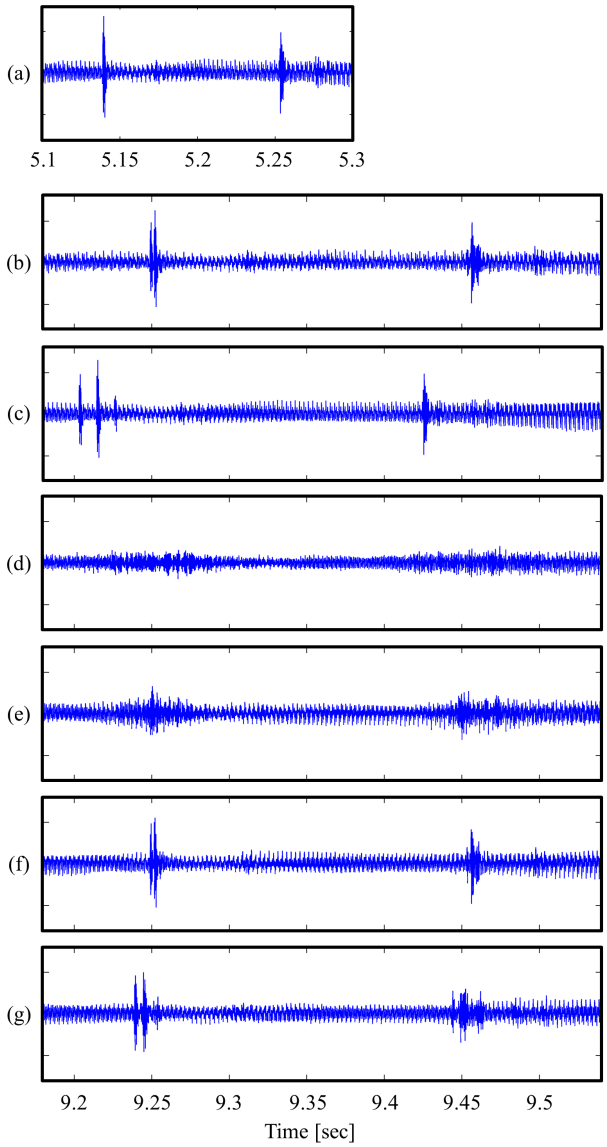


Figure 3: TSM results of different algorithms for an audio recording of a violin and castanets. **(a)**: Original waveform. **(b)**: OLA. **(c)**: WSOLA. **(d)**: Phase vocoder. **(e)**: Phase vocoder with identity phase locking. **(f)**: TSM based on HPSS. **(g)**: TSM based on the commercial *élastique* algorithm.

are therefore duplicated several times which results in the stuttering sound. For example, in Figure 3c, the first transient is repeated three times with different amplitudes.

2.3. Phase Vocoder

While WSOLA approaches the problem of phase discontinuities in the time-domain, the problem can also be targeted in the frequency-domain. The core idea of the *phase vocoder* [5, 6] is to see each analysis frame as a weighted sum of sinusoids with known frequency and phase. The synthesis frames are then computed by adapting the phases of these sinusoids such that no phase discontinuities are introduced when adding up the relocated synthesis frames.

In the first step of the procedure the Fourier transform is applied to every analysis frame resulting in a sequence of frequency spectra. Each frequency bin of a spectrum represents a sinusoid that contributes to the original signal. Afterwards, the *instantaneous frequencies* of the spectrum's frequency bins are computed from the phase differences of successive spectra, see [11]. Knowing the instantaneous frequencies and the synthesis hopsize H_s, the phases of the spectra can be adapted accordingly. Finally, all spectra are brought back to the time-domain by applying the inverse Fourier transform with the resulting waveforms constituting the synthesis frames. Note that the term "phase vocoder" generally describes the technique to estimate the instantaneous frequencies in an audio signal. However, the term is also frequently used to name the TSM algorithm.

By design, the phase vocoder guarantees phase continuity of all sinusoidals contributing to the output signal, which is also known as *horizontal phase coherence*. However, the *vertical phase coherence*, meaning the phase relationships of sinusoidals within one frame, is usually destroyed in the phase adaption process. Transients, which are highly dependent on preserving the vertical phase coherence of the signal, are therefore often *smeared* in phase vocoder TSM results, see Figure 3d for an example. The loss of vertical phase coherence also causes a very distinct sound coloration of phase vocoder TSM results known as *phasiness* [12].

2.4. Phase Vocoder with Identity Phase Locking

To reduce the loss of vertical phase coherence in the phase vocoder, Laroche and Dolson proposed a modification to the standard phase vocoder TSM algorithm [13]. Their core idea is to not adapt the phases of all frequency bins in the short-time Fourier spectra independently of each other. Instead, bins which contribute to the same partial of the audio signal are grouped. A peak in the magnitude spectrum is assumed to represent one partial of the audio signal, while the bins surrounding the peak are assumed to contribute to this partial as well. In the phase adaption process, only the frequency bins which contain spectral peaks are updated in the usual phase vocoder fashion. The phases of the remaining frequency bins are then *locked* to the phase of the closest spectral peak and the vertical phase coherence is therefore locally preserved. This technique, also known as *identity phase locking* leads to reduced phasiness artifacts and also to less transient smearing, see Figure 3e for an example.

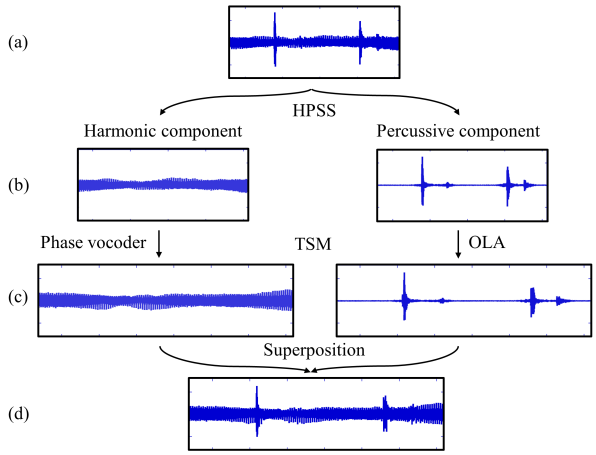


Figure 4: Overview of TSM based on HPSS. **(a)**: Input audio signal. **(b)**: Separation in harmonic component (left) and percussive component (right). **(c)**: TSM results for the harmonic component using the phase vocoder (left) and for the percussive component using OLA (right). **(d)**: Superposition of the TSM results from (c).

2.5. Combined TSM based on HPSS

TSM algorithms like the phase vocoder work particularly well for audio signals with harmonic content, while other algorithms like OLA are well suited for percussive signals. The authors of [7] therefore proposed a combined TSM approach using harmonic-percussive source separation (HPSS) techniques.

In HPSS, the goal is to decompose a given audio signal into a signal consisting of all harmonic sound components and a signal consisting of all percussive sounds. Fitzgerald [14] proposed a simple and effective HPSS procedure. This method exploits the fact that in a spectral representation of a signal, harmonic sounds form structures in time direction, while percussive sounds yield structures in frequency direction. By applying a median filter of length ℓ_h in time direction and a median filter of length ℓ_p in frequency direction to the magnitude spectrogram of the input signal, the respective structures are enhanced. Afterwards, by comparing the two filtered spectra element wise, each time-frequency instance of the signals spectrum can be assigned to either the harmonic or the percussive portion of the signal. This yields in the end the desired components.

After having decomposed the input signal using this HPSS method, the authors of [7] apply the phase vocoder with identity phase locking to the harmonic component and OLA to the percussive component. By treating the two components separately, both the characteristics of the harmonic sounds as well as the percussive sounds of the input signal can be preserved. The superimposed TSM results of both procedures finally form the output of the algorithm (see Figure 4). Note that there also exist other approaches to preserve both characteristics. For example, algorithms employing *transient preservation* aim for explicitly identifying the time positions of percussive events in the audio signal and giving them a special treatment in the TSM process [15, 16]. Such a strategy can also easily be integrated into the MATLAB code provided in the TSM toolbox.

2.6. TSM based on élastique

Besides these publically known TSM algorithms, there also exists a number of proprietary commercial products. One of these commercially available TSM algorithms, called *élastique*, has been developed by zPlane [17]. This algorithm, which is integrated in a wide range of music software like *Steinberg Cubase*[1] or *Ableton Live*[2], can be considered the state-of-the-art in the field of commercial TSM algorithms. An example of an audio signal stretched with *élastique* is shown in Figure 3g. In addition to the usual licensing model for their algorithm, the developers also offer a web-based interface called *sonicAPI*[3], which allows users to compute the TSM results for *élastique* over the internet. At least for the time being, this service is free of charge for personal usage. A MATLAB wrapper function for this webservice is included in the TSM toolbox.

3. TOOLBOX

The TSM algorithms as described in Section 2 form the core of our TSM toolbox, which is freely available at the website [8] under a GNU-GPL license. Table 1 gives an overview of the main MATLAB functions along with the most important parameters. Note that there are many more parameters and additional functions not discussed in this paper. However, for all parameters there are default settings such that none of the parameters need to be specified by the user.

To demonstrate how the TSM algorithms contained in our toolbox can be applied, we now discuss the code example shown in Table 2, which is also contained in the toolbox as script `demoTSMtoolbox.m`. Our example starts in lines 1-4 with specifying an audio signal as well as a time-stretch factor α. Furthermore, the audio signal is loaded from the hard disk using the MATLAB function `wavread` and stored in the variable `x` while its sampling rate is stored in `sr`.

The first TSM algorithm which is applied to the loaded signal is OLA in lines 6-10. Since OLA is a special case of WSOLA, this is done by calling the `wsolaTSM.m` function with a specialized set of parameters. In line 6, the analysis frame position tolerance $\Delta_{\max}$ of WSOLA is set to 0, turning WSOLA into OLA. Afterwards, the synthesis hopsize

[1] `http://www.steinberg.net`
[2] `https://www.ableton.com`
[3] `http://www.sonicapi.com/`

Filename	Main parameters	Optional parameters	Description
`wsolaTSM.m`	x, α	synHop$\hat{=}H_s$, win$\hat{=}w$, tolerance$\hat{=}\Delta_{\max}$	Application of OLA & WSOLA.
`pvTSM.m`	x, α	synHop$\hat{=}H_s$, win$\hat{=}w$, phaseLocking	Application of the phase vocoder (with or without identity phase locking).
`hpTSM.m`	x, α	hpsFilLenHarm$\hat{=}\ell_h$, hpsFilLenPerc$\hat{=}\ell_p$, pvSynHop, pvWin, olaSynHop, olaWin	Application of TSM based on HPSS.
`elastiqueTSM.m`	x, α	–	MATLAB wrapper for the *élastique* algorithm.
`win.m`	ℓ, β	–	Generates a $\sin^\beta$ window function of length ℓ.
`stft.m`	x	anaHop, win	Short-time Fourier transform of x.
`istft.m`	spec	synHop, win	Inversion of a short-time Fourier transform, see [18].
`hpSep.m`	x	filLenHarm$\hat{=}\ell_h$, filLenPerc$\hat{=}\ell_p$	Harmonic-percussive source separation.
`pitchShiftViaTSM.m`	x, n	algTSM	Pitch-shifting the signal x by n cents.
`visualizeWav.m`	x	fsAudio, timeRange	Visualization of TSM results.
`visualizeSpec.m`	spec	fAxis, tAxis, logComp	Visualization of a short-time Fourier transform.
`visualizeAP.m`	anchorpoints	fsAudio	Visualization of a set of anchorpoints.

Table 1: Overview of the main MATLAB functions contained in the TSM toolbox [8] and the most important parameters.

H_s is set to 128 samples in line 7. In lines 8 and 9, a $\sin^\beta$-window of length $\ell = 256$ samples and $\beta = 2$ is generated by calling `win.m`. The size of the generated window specifies at the same time the size of the analysis and synthesis frames. Together with the synthesis hopsize of 128 samples this means that in the output of the TSM algorithm the synthesis frames will have a half-overlap of 128 samples. Finally the actual TSM algorithm is applied to the input signal x with the stretching factor α and the specified set of parameters in line 10. The resulting waveform is stored in the variable `yOLA`.

Next, in lines 12-16, the WSOLA algorithm is applied. We first set the analysis frame position tolerance $\Delta_{\max}$ to 512 in line 12. Since WSOLA works optimally for medium sized frames which are half-overlapped, we set the synthesis hopsize H_s to 512 in line 13 and chose a $\sin^\beta$-window of length $\ell = 1024$ samples and $\beta = 2$ in lines 14 and 15. Finally, the function `wsolaTSM.m` is called in line 16.

In lines 18-22 the standard phase vocoder is applied by a call of `pvTSM.m`. To this end, we first specify that no phase locking should be applied (line 18). Being a frequency-domain TSM algorithm, the phase vocoder is dependent on a high frequency resolution of the used Fourier transform and therefore on a large frame size. Furthermore, also a large overlap of the synthesis frames is beneficial for the quality of the output signal as well as a sin-window function. We therefore set the synthesis hopsize H_s to 512 (line 19) and chose a $\sin^\beta$-window of length $\ell = 2048$ samples and $\beta = 1$ (lines 20 and 21), resulting in a 75% frame overlap. The actual function call is then executed in line 22. For the application of the phase vocoder with identity phase locking in lines 24-28, the only difference is the `phaseLocking` parameter set to one (line 24).

The TSM algorithm based on HPSS, which is applied in lines 30-38, is a combination of multiple techniques. First, we set the length of the median filters ℓ_h and ℓ_p used in the HPSS procedure both to 10 (lines 30 and 31). Then, the synthesis hopsizes and windows, which are used in the two TSM algorithms OLA and phase vocoder with identity phase locking, are set separately in lines 32-37. In line 38 the algorithm is then executed by a call of `hpTSM.m`.

The last TSM algorithm is the MATLAB wrapper for *élastique*. Since this function requires a *sonicAPI* access id as well as the additional tool `curl`, the function call in line 44 is commented out by default. However, when supplying the additional sources the algorithm can be applied by a call to `elastiqueTSM.m`. Since *élastique* is a proprietary procedure it is not possible to tweak the algorithm with additional parameters.

In lines 46 and 47, the visualization of the original input signal takes place. First, the segment of the input audio signal to be visualized is set to the section of the waveform between second 5.1 and 5.3 (line 46). Afterwards the visualization function `visualizeWav.m` is applied to `x` in line 47. To visualize the corresponding stretched audio segment, the segments boundaries are just multiplied with the stretching factor α in line 48. Afterwards, the visualization function is called again exemplarily for OLA's TSM result in line 49. Finally, in line 50, the TSM result of OLA is also written to the hard disk using the MATLAB function `wavwrite`.

4. APPLICATIONS

In this section, we discuss some additional functionalities of the TSM toolbox, including some demo applications.

4.1. Interface Generation

The most common way of comparing the quality of different TSM algorithms is by performing listening experiments. To this end, one usually generates time-stretched versions

```
filename = 'CastanetsViolin.wav';
alpha = 1.8;

[x,sr] = wavread(filename);

paramOLA.tolerance = 0;
paramOLA.synHop = 128;
len = 256; beta = 2;
paramOLA.win = win(len,beta);
yOLA = wsolaTSM(x,alpha,paramOLA);

paramWSOLA.tolerance = 512;
paramWSOLA.synHop = 512;
len = 1024; beta = 2;
paramWSOLA.win = win(len,beta);
yWSOLA = wsolaTSM(x,alpha,paramWSOLA);

paramPV.phaseLocking = 0;
paramPV.synHop = 512;
len = 2048; beta = 1;
paramPV.win = win(len,beta);
yPV = pvTSM(x,alpha,paramPV);

paramPVpl.phaseLocking = 1;
paramPVpl.synHop = 512;
len = 2048; beta = 1;
paramPVpl.win = win(len,beta);
yPVpl = pvTSM(x,alpha,paramPVpl);

paramHP.hpsFilLenHarm = 10;
paramHP.hpsFilLenPerc = 10;
paramHP.pvSynHop = 512;
len = 2048; beta = 1;
paramHP.pvWin = win(len,beta);
paramHP.olaSynHop = 128;
len = 128; beta = 2;
paramHP.olaWin = win(len,beta);
yHP = hpTSM(x,alpha,paramHP);

% To execute elastique, you will need
% an access id from http://www.sonicapi.com.
% Furthermore, you need to download 'curl'
% from http://curl.haxx.se/download.html.
% yELAST = elastiqueTSM(x,alpha);

paramVis.timeRange = [5.1 5.3];
visualizeWav(x,paramVis);
paramVis.timeRange = [5.1 5.3] * alpha;
visualizeWav(yOLA,paramVis);
wavwrite(yOLA,sr,'Output_OLA.wav'
```

Table 2: Code example for computing TSM results of various TSM algorithms, generating the visualizations, and writing the TSM results to the hard disk.

of several audio items using different TSM algorithms and stretching factors. This results in large amounts of audio data. To be able to compare the generated TSM results, interfaces which allow a user to order and access the audio signals in a convenient way are of great help. With the script `demoGenerateTSMwebsite.m`, which is contained in the TSM toolbox, we provide the code for generating such a HTML-based interface automatically (see Figure5). The toolbox also includes the set of audio items listed in Table 3, which has been already used for evaluation purposes in the context of TSM in [7, 19].

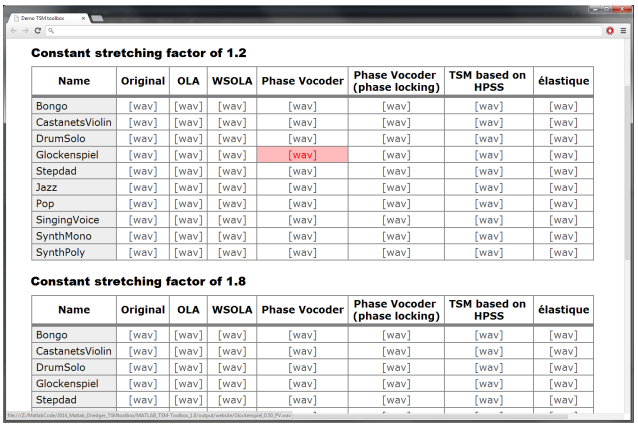


Figure 5: Screenshot of the interface generated using the function `demoGenerateTSMwebsite.m` of the TSM toolbox.

Item name	Description
Bongo	Regular beat played on bongos.
CastanetsViolin	Solo violin overlayed with castanets.
DrumSolo	A solo performed on a drum set.
Glockenspiel	Monophonic melody played on a glockenspiel.
Jazz	Synthetic polyphonic sound mixture of a trumpet, a piano, a bass and drums.
Pop	Synthetic polyphonic sound mixture of several synthesizers, a guitar and drums.
SingingVoice	Solo male singing voice.
Stepdad	Excerpt from *My Leather, My Fur, My Nails* by the band *Stepdad*.
SynthMono	Monophonic synthesizer with a very noisy and distorted sound.
SynthPoly	Sound mixture of several polyphonic synthesizers.

Table 3: List of audio items included in the TSM toolbox.

4.2. Non-linear Time-Scale Modification

In addition to stretching audio signals in a linear fashion by a constant stretching factor α, the implementations contained in the TSM toolbox (except for *élastique*) are also capable of stretching input signals in a non-linear way. To this end, one needs to define a *time-stretch function* which defines the mapping between time-positions in the input signal and the output signal of the TSM algorithm. A very convenient way of defining such a time-stretch function is by specifying a set of *anchorpoints*. An anchorpoint is a pair of time positions where the first entry specifies a time-position in the input signal and the second entry a time-position in the output signal. The actual time-stretch function is then obtained by a linear interpolation between the anchorpoints. In Figure 6, one can see an example of such a non-linear modification. In Figure 6b, we see the waveforms of two recorded performances of the first five measures of Beethoven's Symphony No. 5. The corresponding time-positions of the note onsets are indicated by red arrows. Ob-

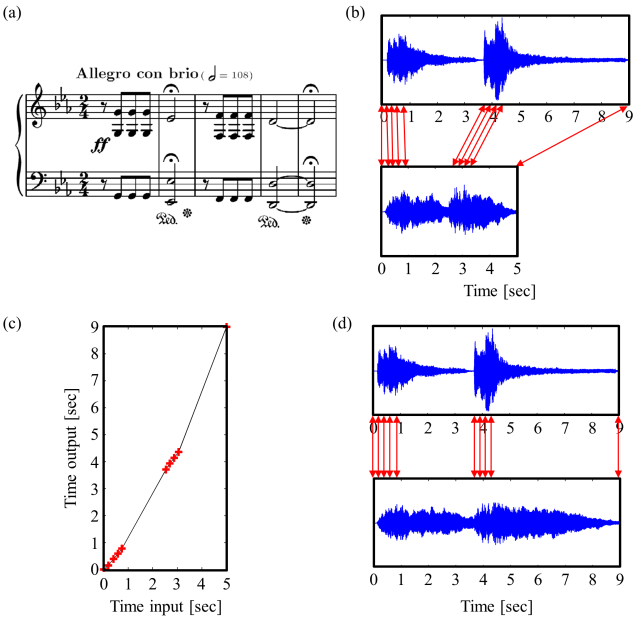


Figure 6: **(a)**: Score of the first five measures of Beethoven's Symphony No. 5. **(b)**: Waveforms of two performances. Corresponding onset positions are indicated by the red arrows. **(c)**: Set of anchorpoints. **(d)**: Onset-synchronized waveforms of the two performances, where the second performance was modified.

viously, the two performances differ strongly in their length. However, the tempo of the two performances does not differ by some constant factor. In fact, the tempo of the eighth notes in the first and third measure are played at almost the same tempo in both performances. Contrary, the durations of the half notes with fermata in measures two and five differ strongly in the two recordings. The mapping between the note onsets of the two performances is therefore non-linear. We define eight anchorpoints, which map the onset positions of the second performance to the onset positions of the first performance (plus two additional anchorpoints, which align the beginning and the end of the waveforms). Based on these anchorpoints, we then apply one of the TSM algorithms in the TSM toolbox to the second performance to obtain a version of the recording which is onset-synchronized with the first performance, see Figure 6d. The MATLAB code for this example, which also generates sonifications of the synchronization result, is also contained in the TSM toolbox in the file `demoNonlinearTSM.m`. In this example, the anchorpoints were chosen manually. However, one can also compute alignments between two recordings automatically and derive anchorpoints from them, see for example [20]. This functionality can, for example, be used in scenarios like *automated soundtrack generation* [21] or *automated DJing* [1, 2].

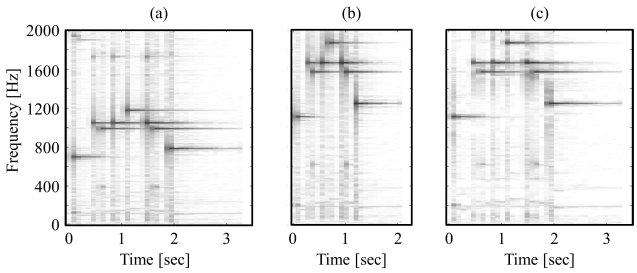


Figure 7: Pitch-shifting via resampling and TSM. **(a)**: Spectrogram of an input audio signal. **(b)**: Spectrogram of the resampled signal. **(c)**: Spectrogram after TSM application.

4.3. Pitch-Shifting

Pitch-shifting is the task of changing the pitch of an audio recording without altering its length. It can therefore be seen as the dual problem to TSM. While there exist specialized pitch-shifting algorithms [22, 23], it is also possible to approach the problem by combining TSM algorithms with resampling. Here, the core observation is, that stretching or compressing the whole waveform of an audio signal changes the length and the pitch of the signal at the same time. With vinyl records, this can for example be simulated by changing the rotation speed of the record player. In the world of digital audio signals, the same effect can be achieved by resampling a given signal. To this end, a given audio signal, sampled at a frequency of f_{in}, is resampled to have a new sampling frequency f_{out}. When playing back the resampled signal at the old sampling frequency f_{in}, this changes the pitch of the signal by $\log(f_{\text{in}}/f_{\text{out}})/\log(\sqrt[12]{2})$ semitones, as well as its length by a factor of $f_{\text{out}}/f_{\text{in}}$. To demonstrate this, we show an example in Figure 7. Here, the goal is to apply a pitch-shift of 8 semitones to the input audio signal. The original signal has a sampling frequency of f_{in}=44100 Hz (Figure 7a). To achieve a pitch-shift of 8 semitones, the signal is resampled to f_{out}=27781 Hz (Figure 7b). One can see, that the resampling changed the pitch of the signal as well as its length. While the change in pitch is desired, the change in length needs to be compensated. This can be done using a TSM algorithm at hand (Figure 7c). However, the quality of the pitch-shifting result crucially depends on the quality of the TSM algorithm. The MATLAB function `pitchShiftViaTSM.m`, which employs the above described strategy for pitch-shifting, is contained in the TSM toolbox. Furthermore, the script `demoPitchShift.m` gives an example of how this function can be applied.

5. CONCLUSIONS

In this paper, we have introduced the TSM toolbox, a unifying MATLAB framework which contains several TSM al-

gorithms, various code examples for demo applications, as well as audio material that has already been used for evaluating TSM algorithms. We hope that this toolbox not only provides a solid code basis to work in the field of TSM, but also helps to raise the awareness for potential problems of classical TSM algorithms, to foster the development of new TSM techniques, and to ease the design of listening experiments. Finally, we would like to encourage developers and researchers in the field of audio processing and music information retrieval to use the toolbox to realize their ideas of applications involving TSM of audio signals.

6. REFERENCES

[1] Dave Cliff, "Hang the DJ: Automatic sequencing and seamless mixing of dance-music tracks," Tech. Rep., HP Laboratories Bristol, 2000.

[2] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima, "Full-automatic DJ mixing system with optimal tempo adjustment based on measurement function of user discomfort," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 2009, pp. 135–140.

[3] Alexis Moinet, Thierry Dutoit, and Thierry Dutoit Alexis Moinet, "Audio time-scaling for slow motion sports videos," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*, Maynooth, Ireland, September 2013.

[4] Werner Verhelst and Marc Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Minneapolis, USA, 1993.

[5] James L. Flanagan and R. M. Golden, "Phase vocoder," *Bell System Technical Journal*, vol. 45, pp. 1493–1509, 1966.

[6] M. R. Portnoff, "Implementation of the digital phase vocoder using the fast fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 3, pp. 243–248, 1976.

[7] Jonathan Driedger, Meinard Müller, and Sebastian Ewert, "Improving time-scale modification of music signals using harmonic-percussive separation," *Signal Processing Letters, IEEE*, vol. 21, no. 1, pp. 105–109, 2014.

[8] Jonathan Driedger and Meinard Müller, "TSM toolbox," `http://www.audiolabs-erlangen.de/resources/MIR/TSMtoolbox/`.

[9] Amalia De Götzen, Nicola Bernardini, and Daniel Arfib, "Traditional (?) implementations of a phase vocoder: the tricks of the trade," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy, December 2000.

[10] Daniel P. W. Ellis, "A phase vocoder in Matlab," `http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/`, 2002, Web resource, last consulted in February 2014.

[11] Mark Dolson, "The phase vocoder: a tutorial," *Computer Musical Journal*, vol. 10, no. 4, pp. 14–27, 1986.

[12] Jean Laroche and Mark Dolson, "Phase-vocoder: about this phasiness business," in *IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics, 1997*, October 1997.

[13] Jean Laroche and Mark Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, 1999.

[14] Derry Fitzgerald, "Harmonic/percussive separation using median filtering," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Graz, Austria, 2010, pp. 246–253.

[15] Frederik Nagel and Andreas Walther, "A novel transient handling scheme for time stretching algorithms," in *127th Audio Engineering Society Convention 2009*, New York, NY, 2009, pp. 185–192.

[16] Shahaf Grofit and Yizhar Lavner, "Time-scale modification of audio signals using enhanced WSOLA with management of transients," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 16, no. 1, pp. 106–115, 2008.

[17] zplane development, "élastique time stretching & pitch shifting SDKs," `http://www.zplane.de/index.php?page=description-elastique`, Web resource, last consulted in August 2013.

[18] Daniel W. Griffin and Jae S. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[19] Jonathan Driedger, Meinard Müller, and Sebastian Ewert, "Accompanying website: Improving time-scale modification of music signals using harmonic-percussive separation," `http://www.audiolabs-erlangen.de/resources/2014-SPL-HPTSM/`, Web resource, last consulted in March 2014.

[20] Sebastian Ewert, Meinard Müller, and Peter Grosche, "High resolution audio synchronization using chroma onset features," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.

[21] Meinard Müller and Jonathan Driedger, "Data-driven sound track generation," in *Multimodal Music Processing*, Meinard Müller, Masataka Goto, and Markus Schedl, Eds., vol. 3 of *Dagstuhl Follow-Ups*, pp. 175–194. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[22] Azadeh Haghparast, Henri Penttinen, and Vesa Välimäki, "Real-time pitch-shifting of musical signals by a time-varying factor using normalized filtered correlation time-scale modification," Bordeaux, France, September 2007, pp. 7–14.

[23] Christian Schörkhuber, Anssi Klapuri, and Alois Sontacchi, "Audio pitch shifting using the constant-q transform," *Journal of the Audio Engineering Society*, vol. 61, no. 7/8, pp. 562–572, 2013.

FREEDSP: A LOW-BUDGET OPEN-SOURCE AUDIO-DSP MODULE

Sebastian Merchel and Ludwig Kormann,
Chair of Communication Acoustics,
TU Dresden
Dresden, Germany
sebastian.merchel@tu-dresden.de

ABSTRACT

In this paper, the development and application of a universal audio-DSP (digital signal processor) board will be described. It is called freeDSP. Our goal was to provide an affordable real-time signal processing solution for researchers and the do-it-yourself community. Easy assembling and simple programmability were the main focus. A solution based on Analog Devices' ADAU1701 DSP chip together with the free graphical development environment SigmaStudio is proposed. The applications range from active loudspeaker compensation over steerable microphone arrays to advanced audio effect processors. The freeDSP board is published under a creative commons license, which allows the unrestricted use and modification of the module.

1. INTRODUCTION

The development and application of DSP boards has been associated with high cost, difficult soldering and low-level programming in the past. The current project investigated if this is still the case. Therefore, the different steps from the selection of a suitable DSP chip to final application examples will be presented in the following. Before describing the design of the DSP module, basic demands and commercial alternatives will be discussed.

1.1. Requirements

It was decided that the board should have at least two audio inputs (stereo) and more than two outputs (e.g., to implement active cross-over networks for multi-way loudspeakers). A scalable number of input and output channels would be preferable. Additionally, the board shouldn't be too difficult to solder with as many through-hole components as possible. The overall cost should be as low as possible. An easy to learn programming language would be beneficial to allow quick experimentation and a short familiarization time. Finally, the overall processing power and functionality should allow for easy adaptation to different application scenarios.

1.2. Available DSP solutions

Few commercial solutions exist on the market. E.g., the company MiniDSP offers several easy to use DSP boards, mainly for audio filtering applications [1]. In order to configure the hardware, special software plugins have to be purchased separately - which increases the overall price. Another option is the use of evaluation boards from semiconductor companies. They are usually expensive and contain only the minimal support logic needed to learn programming the DSP or microcontroller. Several other companies provide costly audio amplifiers or loudspeakers with build-in DSPs. In those cases, configuration of the signal processing is quite limited. Many more DSP products exist which are specialized on one specific purpose like room compensation, audio effects or feedback suppression in a public address system. A low-cost open-source DSP solution provides much more flexibility. It enables the curious student to experiment with digital signal processing. Hobbyists can implement and share various projects using the same platform. Additionally, such a board could be easily adapted for professional prototyping or research projects.

2. DESIGN

First of all, a suitable DSP or microcontroller had to be selected.

2.1. Selection of the DSP

The requirements listed above are met by the SigmaDSP series from Analog Devices. E.g., the ADAU1701 has two integrated AD converters (inputs) with a sampling rate of 48 kHz and a bit depth of 24 bit. Four extra bits are added internally for better signal processing (28 bit total). Before outputting of the signal these extra bits are removed again. Four build in output DA converters can be used for various application scenarios. Additionally, it is possible to connect external AD/DA converters via the I^2S interface. Therefore, a maximum of 10 inputs and 12 outputs is realizable. A functional block diagram of the ADAU1701 is shown in Figure 1. Although the ADAU1701 comes in an SMD package (48-Lead LQFP), it can still be soldered by hand.

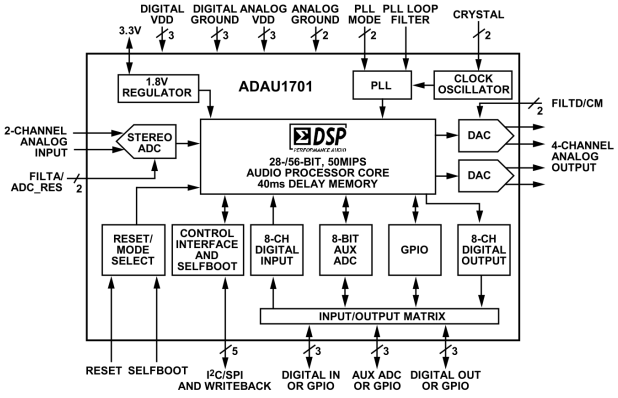


Figure 1: Functional block diagram of the ADAU1701 from Analog Devices [2].

2.2. Board Layout

The design fits on a two-layer board that has a width of 100 mm and a length of 80 mm. The components used on the board are mainly through-hole components, which are easier to solder. Only two SMD parts appear on the board, the ADAU1701 and AD8608, which is the amplifier used within the active output filter. Figure 2 shows the schematic of the audio-DSP board. The freeDSP supports a wide range of input voltages from 5 V up to 24 V. A voltage regulator supplies the 3.3V needed for all circuitry on the board. Additionally, the board might be powered over the USBi connector, which can be used for programming the processor and the EEPROM. Two integrated audio inputs and four audio outputs of the DSP can be accessed via RCA-connectors. The connectors are located on one side of the board for easier access. Figure 3 shows a photo of the assembled freeDSP in the current version. Jumpers allow fast and easy configuration of the input voltage range. One out of three user-defined configurations can be chosen individually for each input. This allows for easy adaptation to different signal levels. Active output filters are used to provide better audio performance. GPIO (general purpose input/output) pins, I²S and I²C interfaces are accessible via the GPIO header. Additional boards can be easily stacked on top of the freeDSP to add more functionality. This includes buttons, potentiometers, additional inputs and outputs or any other peripherals.

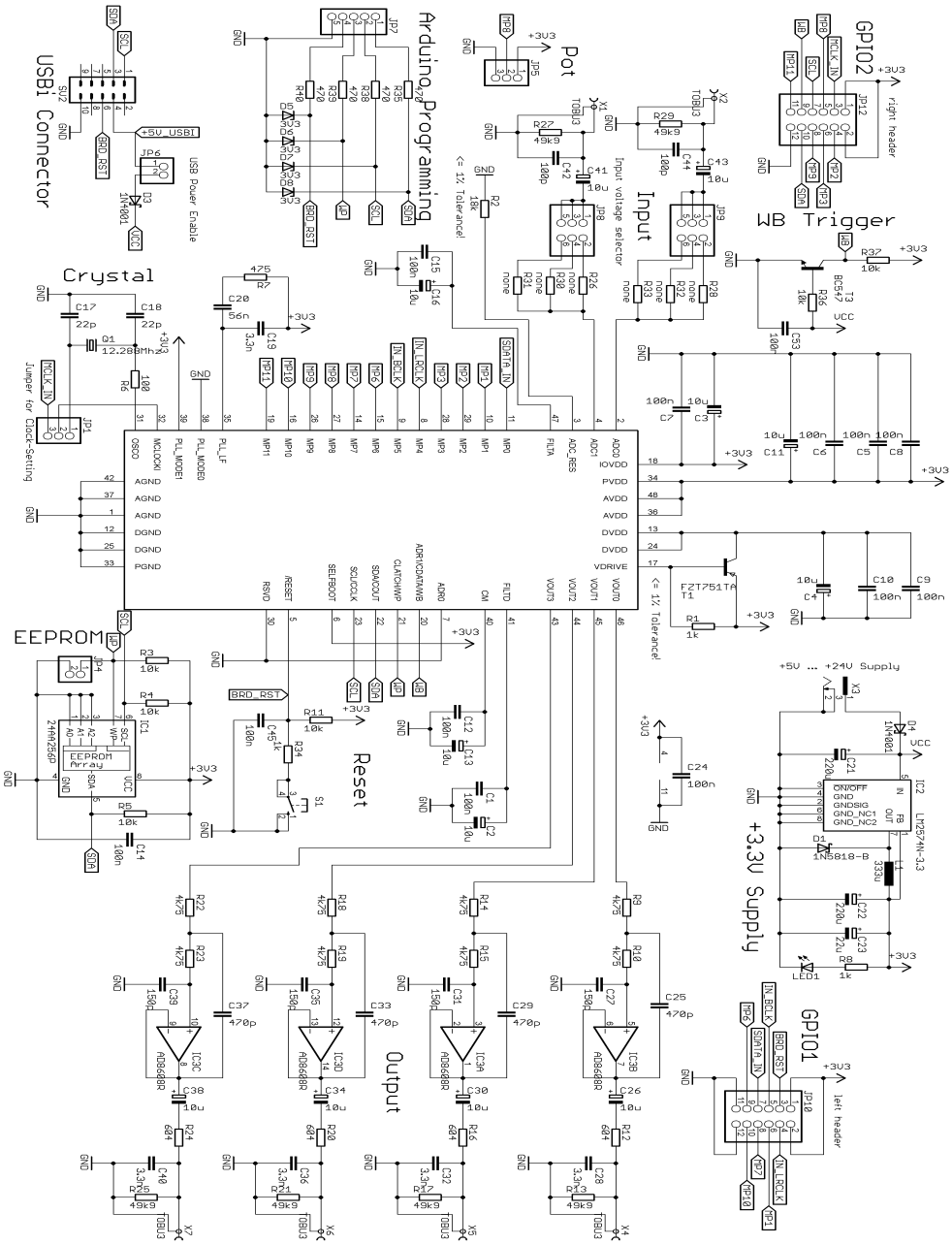


Figure 2: *Schematic of the freeDSP board version 0.3.*

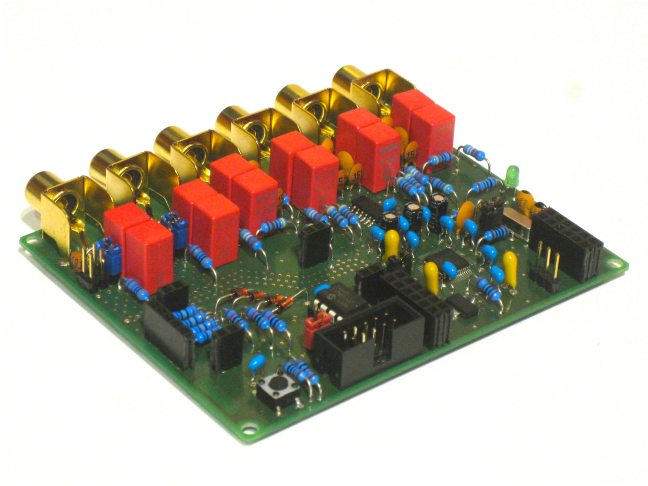


Figure 3: *Completely assembled freeDSP module (V 0.4).*

2.3. Programming

To get started with a SigmaDSP no advanced programming skills are necessary. A free development environment called SigmaStudio is available for Windows. It can be downloaded from the Analog Devices website [3] with no charge (account needed). The programming model is function-block based – comparable to other graphical programming languages like PureData [4] or Max/MSP [5]. Many prebuilt blocks (e.g., filters, compressors, effects, or logic) can be placed in the signal path via drag and drop. If the included libraries do not have the functions needed, low-level blocks, such as multipliers and delays, can be wired together to create custom algorithms. Figure 4 shows a simple example patch with two input channels. The incoming signals are filtered using an equalizer with adjustable frequencies and filter characteristics (band pass, low shelf, …). Finally, the summed up signal is played back via two outputs.

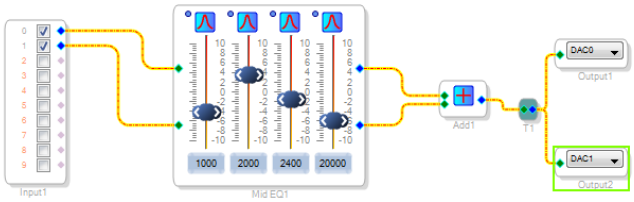


Figure 4: *A simple example patch in SigmaStudio.*

The final program needs to be compiled and transferred to the DSP. Therefore, an USBi programming adapter is available from Analog Devices. With this adapter the board can be controlled in real-time from the PC using SigmaStudio. Alternatively, the program can be written to the onboard EEPROM (electrically erasable programmable read-only memory) and booted in stand-alone mode. This offline programming can be done using a microcontroller, e.g. an Arduino or Raspberry Pi. Alternatively, a dedicated I^2C EEPROM programmer can be used, which supports the EEPROM mounted on the board.

2.4. Cost Calculation

The overall cost of the freeDSP depends on the number of boards build at the same time. Lets assume a total of five boards in the following. The exemplary calculation in Table 1 provides an overview of the expected costs per module.

Table 1: *Component overview with price examples.*

Components	Price / Module
Board	5 $
DSP ADAU1701	9 $
Resistors	3 $
Capacitors	4 $
Semiconductors	8 $
Miscellaneous	10 $
Total	**39 $**

Additionally, a suitable programmer is needed once. To offer real-time programmability the USBi adapter from Analog Devices is recommended, which is currently sold for approximately 80 $. If the application is intended to run the board only in self-boot mode, a cheaper programmer can be used.

3. APPLICATION EXAMPLES

The board was extensively tested and has been used successfully in several applications so far. Two exemplary projects will be described in the following.

3.1. Active Loudspeaker Concept

Active loudspeakers are speakers with build in amplifiers. Typical examples are subwoofers or studio monitors. For educational purposes, a desktop loudspeaker was designed and built in our audio lab. It was decided to implement a two-way system and to reduce the internal volume of the cabinet because of aesthetical reasons. A picture of the speaker is shown in Figure 5.

Figure 5: *Small active loudspeaker prototypes with digital crossover, loudspeaker protection and bass enhancement algorithms. An early version of the freeDSP can be seen in the foreground.*

An active crossover was built into the speakers using the freeDSP. This enabled fast and simple parameter modifications (like corner frequency or filter type) during the testing phase. A screenshot of one of the prebuild crossover blocks is shown in Figure 6.

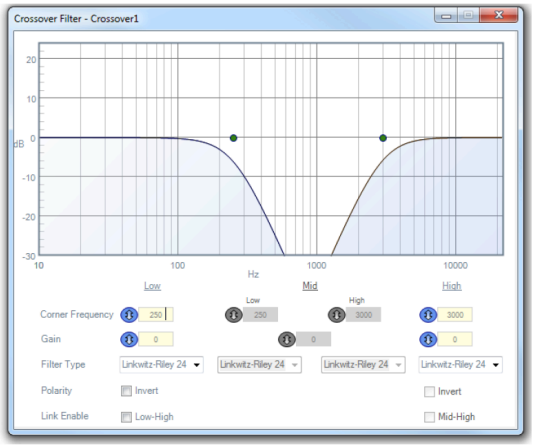


Figure 6: *Adjustable crossover settings in SigmaStudio.*

Further, the freeDSP was used to equalize the frequency response of the system. Therefore, impulse responses have been measured. Matlab was used to create inverse filters with different algorithms. The resulting IIR coefficients have been applied using a biquad filter bank in SigmaStudio. Alternatively, free software could be used for measurement and filter creation, e.g., the Room EQ Wizard (REW) by John Mulcahy [6]. However, there is also an easy to use auto-EQ filter designer available in SigmaStudio.
Additionally, the bass output was improved and the drivers were protected from overload. Experiments were carried out with dynamic bass boost, which provides variable gain for low frequencies dependent on input signal level.
If the general purpose input and output pins are taken into account, many more examples are imaginable. E.g., the GPIO pins of the module could be used to power down the loudspeaker amplifiers when no signal is present. Another useful option would be to connect configurable dip switches and a volume control potentiometer to the board.
The application of the freeDSP in this active loudspeaker example was intuitive and successful. However, also more complex projects can be realized.

3.2. Vibrotactile Feedback for an Electric Violin

When playing a violin, the musician communicates with his instrument not only through his ears but also his fingers, cheek, shoulder and eyes. While playing he uses multiple sensory channels, which are provided by different modalities, such as auditory, tactile, kinesthetic, and visual. In a research project, the influence of violin vibrations on the perceived quality of the instrument was investigated. Therefore, it was necessary to separately control the sound radiation and the vibrotactile feedback of the instrument. A Harley Benton electric violin was selected because of its non-resonant body. A vibration reproduction system was added using small electro-dynamic shakers mounted at the back of the instrument. Figure 7 shows the violin with one of the exciters mounted below the chin rest.

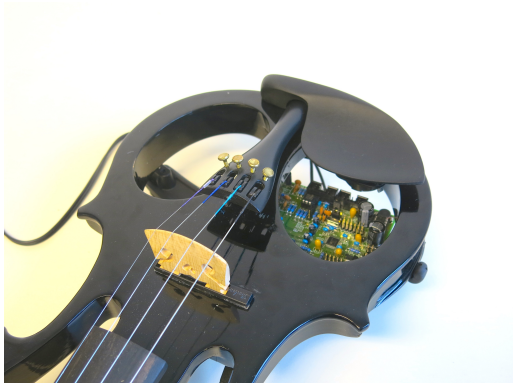


Figure 7: *Vibrotactile feedback for an electric violin using a small eletrodynamic on the backside of the chin rest. The freeDSP was integrated in the free space below.*

The freeDSP board was applied to generate audio-driven vibrations from the sound in real-time. First, the natural vibrations of a classical violin were simulated using the system. Additionally, different algorithms to modify the vibration signal were implemented and investigated, e.g., frequency shifting or dynamic compression. In Figure 8 an exemplary program patch can be seen. The results showed the importance of vibrations on the overall perception of the instrument and provide information on useful vibration features for the player-instrument interaction [7].

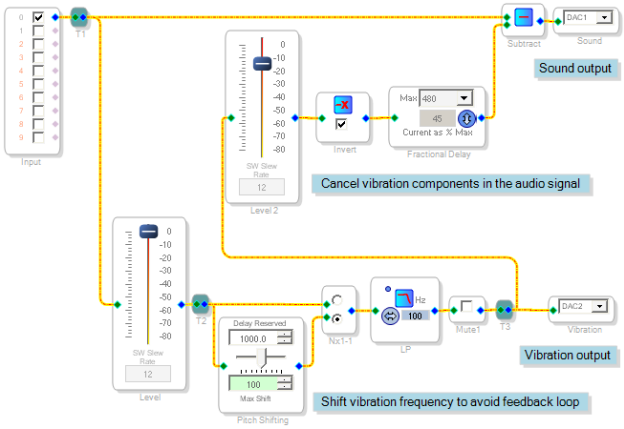


Figure 8: *Simplified program patch in SigmaStudio, showing one approach for vibration generation by frequency shifting the violin sound (bottom path). Additionally, the vibration crosstalk in the audio output is canceled actively using an inverter and a fractional delay (top path).*

In the area of vibrotactile feedback, many more innovative application scenarios for the freeDSP module exist. E.g., it can be used to process vibrations for a touch screen based audio mixer or a groove box [8]. It was found that different music instruments could be distinguished if vibrotactile feedback is rendered from the audio signal in an appropriate way. This helps to improve recognition of an audio source that is assigned e.g. to a specific mixing channel. Another example is the improvement of the concert experience in a music reproduction system. This is possible by generating seat vibrations in real-time from the audio signal using various signal processing algorithms [9].

4. SUMMARY

In this paper, the development and application of a universal low-budget audio-DSP board called freeDSP was described. The main features are summarized below:

- 2 integrated AD and 4 DA converters,
- Sampling rate 44.1 / 48 kHz (max. 96 kHz),
- Bit depth 24 bit (28 bit internal),
- Upgradeable up to 10 inputs and 12 outputs,
- Graphic oriented programming software available,
- Input voltage 5 to 24 V,
- 8 GPIO (general purpose input/output) pins,
- 4 AUX 8 bit ADC inputs (e.g. for potentiometers)
- Variable input range configurable via jumpers,
- Active filtered outputs.

The different application examples show that the freeDSP is a versatile tool that can be applied in various scenarios. The authors hope that it can be of use for others too. Therefore, the board layout is published under a creative commons license (CC BY-SA). Everyone is free to use the module as it is or to modify it as required. A complete list with all necessary components and the latest circuit board design will be published on our website [10]. A forum for discussions and the possibility for centralized buying are planned.

5. REFERENCES

[1] http://www.minidsp.com, Accessed February 27, 2014.

[2] Analog Devices, "Datasheet of the ADAU1701 Digital Signal Processor," Available at http://www.analog.com/static/imported-files/data_sheets/ADAU1701.pdf, Accessed January 05, 2014.

[3] http://www.analog.com/en/content/cu_over_sigmastudio_graphical_dev_tool_overview/fca.html, Accessed February 27, 2014.

[4] M., Puckette, "Pure Data: another integrated computer music environment." in *Proc. of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan, 1996.

[5] M., Puckette, "The patcher" in *Proc. of the Computer Music Conference*, San Francisco, USA, 1986.

[6] http://www.roomeqwizard.com, Accessed December 07, 2013.

[7] M.E. Altinsoy, S. Merchel, S. Tilsch, "Perceptual evaluation of violin vibrations and audio-tactile interaction," in *Proc. of Meetings on Acoustics (ICA 2013)*, Vol. 19, 015026, 2013.

[8] S. Merchel, M.E. Altinsoy, "Vibration in Music Perception," *in Proc. of the AES 134th Convention*, Rome, Italy, 2013.

[9] S. Merchel, M.E. Altinsoy, M. Stamm, "Touch the Sound: Audio-Driven Tactile Feedback for Audio Mixing Applications," in Journal of the Audio Engineering Society, 60(1/2), 2012.

[10] http://www.ias.et.tu-dresden.de/ias/kommunikationsakustik, Accessed March 13, 2014.

DECLARATIVELY PROGRAMMABLE ULTRA LOW-LATENCY AUDIO EFFECTS PROCESSING ON FPGA

Math Verstraelen, Jan Kuper, Gerard J.M. Smit

Computer Architecture for Embedded Systems
University of Twente
Zilverling 4078, P.O. Box 217, 7500 AE Enschede, the Netherlands
Email:{M.J.W.Verstraelen,G.J.M.Smit,J.Kuper}@utwente.nl

ABSTRACT

WaveCore is a coarse-grained reconfigurable processor architecture, based on data-flow principles. The processor architecture consists of a scalable and interconnected cluster of Processing Units (PU), where each PU embodies a small floating-point RISC processor. The processor has been designed in technology-independent VHDL and mapped on a commercially available FPGA development platform. The programming methodology is declarative, and optimized to the application domain of audio and acoustical modeling. A benchmark demonstrator algorithm (guitar-model, comprehensive effects-gear box, and distortion/cabinet model) has been developed and applied to the WaveCore development platform. The demonstrator algorithm proved that WaveCore is very well suited for efficient modeling of complex audio/acoustical algorithms with negligible latency and virtually zero jitter. An experimental Faust-to-WaveCore compiler has shown the feasibility of automated compilation of Faust code to the WaveCore processor target.

Keywords: ultra-low latency, zero-jitter, coarse-grained reconfigurable computing, declarative programming, automated many-core compilation, Faust-compatible, massively-parallel

1. INTRODUCTION

Modeling of physical/acoustical phenomena as a methodology to build electronic musical instruments has become increasingly popular since digital electronics became sufficiently powerful and cost-effective. Nowadays these models are often the mathematical core of products like synthesizers or sound effects gear (e.g. guitar-effects). Moreover, such physical models are increasingly being used during the development of acoustical music instruments [1], or used within hybrid musical instruments (like the hybrid piano). The General-Purpose Processor (GPP) in stand-alone computer systems, or in its embedded form in tablet computers or smartphones, has steadilly gained processing capacity during the past years. This has resulted in the fact that the GPP is often used as audio processing device. Consequently, audio/acoustical models are usually developed with a C-based approach. A few disadvantages of the application of the GPP are varying/unpredictable/long latency in the processing chain, the high power consumption and a limited processing capacity which limits the applicability of a GPP for complex physical modeling. Other processor technologies, like Field-Programmable Gate Arrays (FPGA), are an alternative for these types of complex modeling problems.

2. SCOPE OF THE WORK AND RELATED PROBLEMS

The scope of our work is to develop scalable (i.e. parallel computing) and low-latency processor technologies within the domain of physical modeling. State-of-the art GPP processor architectures are multi-core based. It is however a difficult task to exploit full parallelism, because it is not trivial to compile a conventional C-based program to a multi-core platform. Likewise, the usually shared and cached memory hierarchy adds another complexity level (data coherency among parallel processes, variable latencies and performance penalty when moving data between processors through a unified cached memory hierarchy). FPGAs are often used when low-latency and high-performance are crucial requirements. FPGAs offer flexibility and can be designed such that they offer the exactly required performance. However, creating an FPGA design is a specialistic task and differs from software design in many aspects. Moreover, FPGAs are not by definition flexible in the sense that the functionality can be changed easily/rapidly. As a result, an FPGA design is usually heavily optimized towards a specific task. In the field of physical modeling an example is a physical/acoustical model of a banjo instrument, based on Finite Difference Modeling [1]. Coarse-grained reconfigurable architectures (CGRA) aim to address the programmability problem of the bit-level configurable FPGA. CGRAs are usually based on regular matrices of configurable Processing Units (PUs). Usually a data-flow graph is mapped on a CGRA where the arithmetic functions map on the PUs and the communication on the interconnect network of such an architecture.

3. MAIN CONTRIBUTIONS

We have developed a CGRA, called WaveCore [2]. The WaveCore architecture is declaratively programmable through an explicit description of an algorithm in the form of a data-flow network. This data-flow network is automatically partitioned and mapped on the WaveCore architecture. The semantics of a WaveCore data-flow graph are conceptually close to properties of functional programming languages, like Faust [3]. Therefore, compilation of an algorithm which is described in a functional language towards a WaveCore data-flow graph is feasible. The WaveCore architecture is implemented as a softcore, which can be either mapped on FPGA or ASIC (Application Specific Integrated Circuit) technology. The scalability of the regular architecture, combined with the declarative and scalable programming methodology, results in the ability of automated partitioning and mapping of a data-flow graph on the architecture. A mapped algorithm behaves fully predictable, which means that stream buffering can be kept to a minimum. This

results in ultra-low processing latency. The WaveCore processor technology is optimized to characteristics which are dominant in physical modeling of audio systems (e.g. delay-lines). In this paper we will first focus on the WaveCore architecture, related to the programming methodology. Then we will explain how two example audio effects (flanger and auto-wah) are described and mapped on the WaveCore processor architecture, and how a reverberation effect which is described in Faust [3] can be automatically compiled to WaveCore. Next we will present a benchmark modeling algorithm, and highlight the efficiency. Finally we draw some conclusions and we give some directions to future work.

4. WAVECORE PROCESSOR TECHNOLOGY

Like we mentioned, WaveCore is a CGRA. This architecture consists of a cluster of interconnected Processing Units (PU). These PUs are small RISC (Reduced Instruction Set Computer) processors with a dedicated instruction-set which is specifically optimized towards audio/acoustical modeling. The WaveCore programming methodology is based on a native WaveCore programming language, and based on a declarative description of a hierarchical network of processing primitives. Despite the fact that the WaveCore language is native, the structure of the language matches closely with existing functional languages, like Faust [3]. The WaveCore processor architecture is implemented as a technology independent and configurable soft-core (VHDL), which as a prototype has been targeted to a commercially available FPGA development platform (Digilent Atlys, [4]). A WaveCore cluster and associated mapping tool can be automatically generated. In the following sections we will first outline the WaveCore programming methodology, followed by the associated WaveCore PU cluster architecture.

4.1. Programming model

The WaveCore programming model is based on explicit description of a data-flow-graph in a declarative manner. An example of such a graph is depicted in fig. 1. At top-level the graph consists of one or more 'actors' (i.e. processes) which are interconnected by means of 'edges'. An actor can have multiple inbound (i.e. input) and outbound (i.e. output) edges. Data is carried across the edges, where a data-packet is called a token. Each edge is associated with a predefined token-type. For example: an edge might represent a stereo audio channel, carrying tokens which consist of two floating-point numbers at a token rate of 48kHz. Each edge in the graph has a programmable token buffering capability. The example graph in fig. 1 represents an audio processing application. This application is controlled by an actor called "run-time control actor". This actor might run on a host processor and communicates with a WaveCore actor (which we call a WaveCore Process (WP)) through control edge E2. This edge E2 carries control tokens which represent for instance audio effects settings, like phasing depth. The "audio interface actor" within the example application graph represents an intermediate process between an audio codec (e.g. AC97 device) and the WP. This audio interface actor communicates with the WP through two edges: E1 which carries the input tokens(s) to the WP and E2 which carries the processed WP output tokens. The actual signal processing algorithm runs on two example WPs, called WP1 and WP2. WP1 on its turn consists of two WP-partitions WP1.a and WP1.b. Ultimately the WP, or WP-partition is composed of "primitive actors" (PA). As such, the

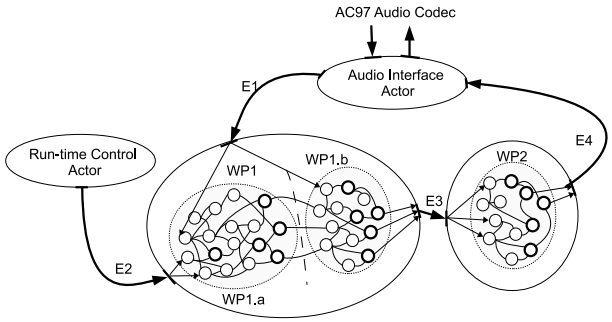


Figure 1: *Data-flow graph*

PA is the algorithmic primitive within the declarative WaveCore programming methodology. The PA is defined in the next subsection. Each actor in the graph is executed (i.e. fired) periodically, where each actor might be fired with a different (through coherently related) frequency. Within WaveCore, we chose to apply a centralized scheduler which orchestrates the firing of all the actors in the graph. The reason that we chose for this scheduling principle, rather than a purely data-flow driven schedule which is more common in data-flow architectures, is that a centralized scheduling principle yields a fully predictable and jitter-free execution of the overall graph. As a result, large closed-loop graphs with a relative large number of actors still yield a fully predictable overall execution when applying this centralized scheduler principle.

4.1.1. Primitive Actor

Like explained, the Primitive Actor (PA) is the basic processing element in the WaveCore programming methodology. The PA is depicted in fig. 2. The definition of the PA is based on fundamental discrete-time audio processing characteristics. Besides the common basic mathematical operations like addition, subtraction, multiplication, etc. a dominant property is the delay function. Delay-lines are dominantly present in many audio and acoustical modeling algorithms [5], like reverberation, string modeling etc. Furthermore, dynamic delay-line length variation is an additional basic property which is also present in many modeling phenomena (like Doppler shifting, or multi-path interference with time-modulated path-length such as "flanging"). The WaveCore PA is depicted in fig. 2. The PA has at most two inbound edges $x_1[n]$ and

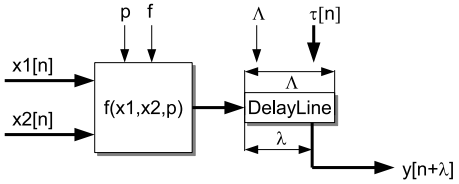


Figure 2: *WaveCore Primitive Actor (PA)*

$x_2[n]$ and one outbound edge $y[n + \lambda]$. There are different types of PAs which are indicated with the function identifier f (see table 1) A number of PAs (like the MUL-type PA) need a parameter p. Each PA is associated with an optional delay-line which is automatically inferred when the delay-line length Λ is greater than zero. The effective length λ of the delay-line can be run-time var-

ied with the inbound edge $\tau[n]$ according to the following relation:

$$1 < \lambda[n] < \Lambda \tag{1}$$

According to:

$$\lambda[n+1] = \lfloor \Lambda.(\tau[n] - 1) \rfloor \tag{2}$$

With:

$$1 < \tau[n] < 2 \tag{3}$$

The programmer should take care that $\tau[n]$ is bounded within the specified range. Hence, the behavior is not specified outside the specified range.

Table 1: *PA types*

PA type	Mnem	Function
Adder	ADD	$y[n+\lambda] = x_1[n] + x_2[n]$
Comparator	CMP	$y[n+\lambda] == (x_1[n] > x_2[n])$
Divider	DIV	$y[n+\lambda] = x_1[n]/x_2[n]$
Logic funct.	LGF	$y[n+\lambda] = LogFunc(x_1[n], x_2[n], p)$
Look-up	LUT	$y[n+\lambda] = Lookup[\lfloor Scale(x_1[n]) \rfloor]$
Multiplier	MUL	$y[n+\lambda] = x_1[n].x_2[n]$
Mul/Add	MAD	$y[n+\lambda] = p.x_1[n].x_2[n]$
Amplifier	AMP	$y[n+\lambda] = p.x_1[n]$
Noise Gen	RND	$y[n]$=$p.rnd < 0 : 1 >$

4.2. Processor architecture

The block diagram of an embedded WaveCore PU cluster instance is depicted in fig. 3. The cluster in fig. 3 consists of 5 interconnected PUs, and a shared local memory. Each PU represents a small IEEE754 compliant single-precision floating-point based RISC processor. The PU is optimized to sequentially fire all PAs in a WP-partition. If a WP is partitioned into two or more WP-partitions, then the data which is associated to the graph-cut(s) is streamed over the PU interconnect network which is called the GPN (Graph Partition Network). The memory hierarchy spans three levels and does not contain any caching. Level1 implies PU-proprietary tightly coupled memory within each PU instance. Level1 memory is used for in-place execution of WPs or WP-partitions (scratch memory and storage of state-variables) Level2 embodies a shared PU cluster memory. Level3 memory is located outside the PU cluster and is usually an off-chip bulk memory like DDR. WP token data buffer space is either allocated in Level2 (for spatially close WPs), or Level3 memory. Delay-line buffer space is also mapped onto Level2 or Level3 memory. The memory hierarchy has been designed in such a way that the locality of reference principle is maximally utilized. This implies that the data-traffic between the levels is kept to a minimum. The instruction set of the PU is optimized in such a way that it is enabled to fire a PA within a single instruction, including all the necessary memory references, pointer updates, arithmetic operations etc. Moreover, special measures have been taken to hide Level2/Level3 memory latency for the PU.

Like we mentioned in the introduction, all the actors in the data-flow graph are fired by means of a centralized scheduler. This means that all the PUs in the cluster are triggered by this centralized scheduler. The overall PU cluster architecture has been designed in such a way that the real-time constraints are always met. The cluster can be initialized through the "Host Processor Interface" (HPI), which provides access to instruction memory, data memory and registers for all the embedded PUs. Similarly, the HPI is used for run-time control of the WaveCore application. Run-time application control can either be performed by direct PU level1 memory write-access, or via control tokens through Level3 memory.

4.3. WaveCore FPGA development platform

We have generated an optimized WaveCore PU cluster for a commercially available Digilent Atlys FPGA platform. The heart of this platform is a Xilinx Spartan6 LX45 FPGA. The platform contains a rich variety of interfaces (USB, audio, HDMI, etc.) and memory. The WaveCore/Atlys architecture is depicted in fig. 3. This architecture uses the on-board AC97 codec, obviously the FPGA, the USB interface and DDR2 memory. The generated WaveCore PU cluster consists of 5 PUs and 16kByte Level2 memory.

The feasible clock frequency for the PUs is largely dependent on the target technology, in this case the Spartan6 LX45 FPGA. For this typical device we obtained a PU clock frequency of 86MHz (which equals 1792 times the programmed AC97 audio rate of 48kHz). Given the fact that a PU can execute a PA within a single instruction and it is a fully pipelined RISC processor architecture, this yields a processing capacity of 1792 PAs per audio sampling period per PU at 48kHz audio rate. Hence, the PU cluster on the Atlys board has a total processing capacity of 8960 PAs. This is equivalent to 860 MFLOPs sustained performance.

The PU cluster is embedded in a FPGA SoC (System on Chip) topology, as can be seen in fig. 3. The cluster has its own dedicated port to the embedded DDR2 controller, which provides access to the 128MByte DDR2 memory (level3 memory). The AC97 codec chip on the Atlys board streams directly into the DDR2 memory through the "Stream Actor" within the "Digital Audio Interface" on the FPGA. The system is controlled through the USB interface. Hence the host processor is supposed to be an external device like a notebook, smartphone (through BT), etc. The host processor is enabled to initialize/reconfigure the PU cluster and to control the application at run-time (e.g. virtual knobs).

Mapping of the example data-flow graph in fig. 1 onto the WaveCore PU-cluster in fig. 3 is straightforward. The "Run-time Control Actor" runs on the host computer (e.g. notebook). This control process can either pass control tokens into a dedicated buffer in DDR2 memory (represented by E2, see fig. 1), or directly write into PU level1 memories. The "Audio Interface Actor" runs on the FPGA as a hardwired process. This actor is basically a DMA controller which moves audio samples from the AC97 ADC (Analog to Digital Converter) to a dedicated token buffer in DDR2 memory, and audio samples from token buffer in DDR2 memory to the AC97 DAC (Digital to Analog Converter). The AC97 codec can be initialized by the host processor through USB. The actual processing is performed by the example WP1 and WP2 actors which are mapped on the WaveCore PU cluster. The number of occupied PUs within the cluster depends on the complexity of the WPs (number of PAs). It might be the case that WP1.a, WP1.b and WP2 run on three PUs, but these three WP(partitions) might also be mapped on a single PU if these are sufficiently small. The token

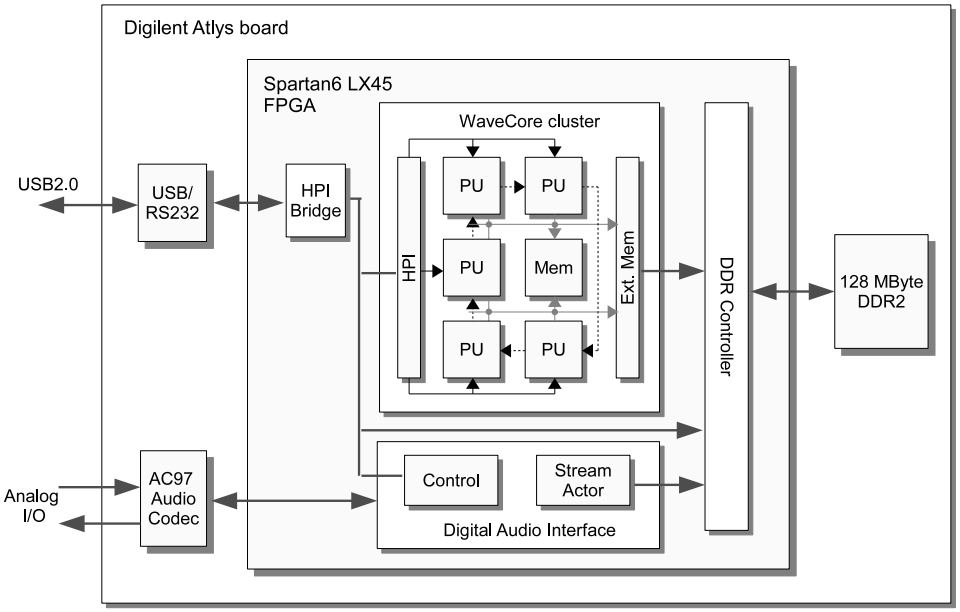


Figure 3: *WaveCore development platform architecture*

buffering for E3 is mapped on the locally embedded level2 memory within the cluster. Scheduling/firing of the actors is driven by the AC97 codec, which dictates the token frequency. The scheduler is embedded in the "Audio Interface Actor" and periodically fires the "Streaming Actor" and mapped WPs on the PU cluster through the HPI interface. Note that the "Run-time Control Actor" typically fires only incidentally, or at least at a rate which is far lower than the audio token rate of 48kHz.

4.4. FPGA mapping

The WaveCore SoC design (PU-cluster and infrastructure) in fig. 3 is mapped on the Xilinx LX45 FPGA on the Atlys board. The WaveCore PU cluster contains 5 PUs, where each PU is dimensioned to execute WP-partitions with up to 2048 PAs. The PU clock frequency is derived from the AC97 audio clock by an on-chip PLL (Phase Locked Loop). Each PU uses 120kByte level1 memory. The level2 memory is configured to 16kByte. This WaveCore configuration requires 21% of the slice registers, 68% of the slice LUTs, 75% of the block-RAMs, and 51% of the DSP48 units on the FPGA. The mapping floorplan is displayed in fig. 4. Each color in the floorplan represents a WaveCore PU.

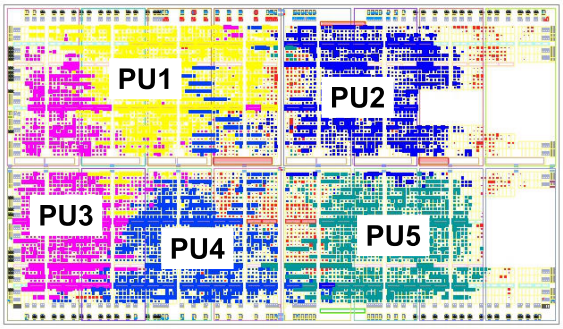


Figure 4: *Xilinx LX45 FPGA floorplan for WaveCore PU cluster*

4.5. Latency, jitter and buffering

Latency, jitter and sample buffering are closely related. Jitter is usually caused by unpredictable execution behavior. This unpredictability can be caused by several factors like cache-misses, interrupts, unpredictable round-trip delay times for shared memory read transactions or other process stalls due to shared resource conflicts. For an audio application it is unacceptable to stall the production of output samples, or to skip output samples in case of a stalled process. To prevent this, buffering is usually applied. The required buffering depth is directly related to the worst-case stall time of the processing device. Buffering however has the disadvantage that it introduces processing latency. This latency is critical in systems where part of the application runs on the processing system, and the other part communicates with the system in a closed-loop (e.g. the hybrid piano, or real-time guitar effects processing).

The processing latency within the WaveCore architecture can be as short as a few audio sampling periods (below 100 μsec @ 48kHz sampling rate). The actual processing latency depends on the token buffer sizes (number of tokens within an edge channel) and the nature of the processing algorithm which runs on the PU cluster. This latency can be that short because of the real-time guaranteed, fully predictable and jitter free execution of the overall WaveCore processing chain. Jitter-free execution is a direct result of the strict process-firing mechanism on the one hand, and guaranteed execution of WP (or WP-partitions) on the other hand. Moreover, the carefully designed memory hierarchy results in a modest load on the externally shared level2 and level3 memory resources. Finally, the static and fully predictable processing schedule prevents the need for extensive buffering.

5. AUDIO EFFECTS IN WAVECORE TECHNOLOGY

Like we explained, a WP (or WP-partition) consists of a network of interconnected PA instances. Furthermore, the available PA types enable both arithmetic as well as logical/control functionality. The abstraction level of a WaveCore WP matches closely with the level at which DSP functions at block diagram level are often specified. We will demonstrate this with two example audio effects algorithms, described in WaveCore: the 'Flanger' and 'Auto-Wah' sound effect algorithms. Additionally, we will show the results of an experimental compilation from the functional audio description language Faust to a WaveCore WP through an example algorithm: the 'Zita-Rev1' reverberator.

5.1. Flanger

The flanger audio effect is widely used within several musical instruments. The effect is based on the principle of varying multi-path acoustical wave interference. In this subsection we will focus on a WaveCore model of a flanger [5]. The core of the flanger algorithm is a delay-line which length is modulated by a Low-Frequency Oscillator (LFO). The acoustical input signal is split: one path goes through the modulated-length delay-line (path1) and the other travels without delay (path2). Subsequently, the path1 and path2 signals are added, yielding the output signal $y[n]$. The mentioned interference when varying delay-length(s) are applied yields the typical flanger effect.

The block diagram of the flanger is depicted in fig. 5. The

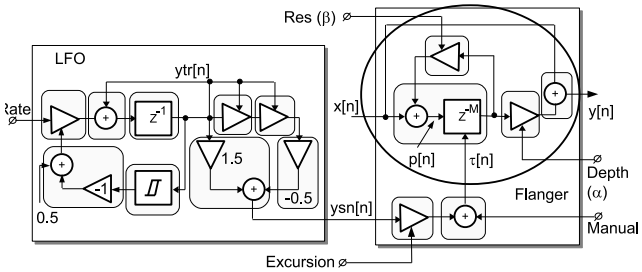


Figure 5: *Flanger WaveCore process*

flanger implementation consists of two WaveCore WP-partitions. The right WP-partition of the block diagram represents the core of the flanger. Those parts of the block diagram which are bounded by the round-edge boxes represent PAs (e.g. the adder and associated delay-line with input $x[n]$ is one PA). The primary input signal $x[n]$ is added to a delayed and scaled copy of the input signal $\alpha.x[n-\lambda]$, while a fraction β of the delay-line output is fed-back to the input of the delay-line. The actual multi-path interference takes place at the output adder where $x[n]$ is added to the delay-line output. The feedback-path, where a fraction of the delay-line output is fed back into the input of the delay-line, intensifies the effect.

In our next analysis, where we will show that the flanger core is a variant of a comb-filter, we assume λ to be constant. The difference equations for the upper-part of the flanger-core (within the ellipse) are given in equations 4 and 5 .

$$p[n] = x[n] + \beta.p[n-\lambda] \tag{4}$$

And

$$y[n] = x[n] + \alpha.p[n-\lambda] \tag{5}$$

With:

$$\lambda[n+1] = \lfloor M.(\tau[n]-1) \rfloor \tag{6}$$

After applying the z-transformation to the difference equations and merging the transformed equations, we find the transfer function in the z domain which is defined in equation 7

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1+(\alpha-\beta)z^{-\lambda}}{1-\beta.z^{-\lambda}} \tag{7}$$

In order to analyze the magnitude response of the flanger-core, we replace z by $e^{j\theta}$, and subsequently derive the magnitude response which is defined in equation 8.

$$|H(e^{j\theta})| = \sqrt{\frac{1+(\alpha-\beta)^2+2(\alpha-\beta)cos(\lambda\theta)}{1+\beta^2-2\beta cos(\lambda\theta)}} \tag{8}$$

The magnitude response of the comb filter reveals a number of equidistant peaks and notches. The location of the peaks in the magnitude response of the comb-filter follows from equation 8, and is as defined in equation 9.

$$\theta_k^{(p)} = k\frac{2\pi}{\lambda}, k = 0, 1, 2, \cdots, \lambda - 1 \tag{9}$$

So, the number of equidistant notches and peaks in the magnitude response of the comb-filter is equal to λ and is proportional to $\tau[n]$. The effect of the feedback path with the res amplifier is that the peaks in the magnitude response get smaller and intenser for high res values, which also follows from equation 8.

The delay-line length λ is modulated by the left-part of the block diagram in fig. 5: the LFO WP-partition. This LFO generates an approximated sine-wave with a sub-Hz frequency, which is determined by the $rate$ parameter. The core of the LFO is an integrator with a hysteresis-PA in its feedback path, as is depicted in fig. 5. The hysteresis-PA switches symmetrically between -1 and 1 (the output of the hysteresis-PA toggles between 0 and 1), which equals the amplitude of the generated triangular wave $ytr[n]$. The multipliers and adder with input $ytr[n]$ implement a third-order polynomial, as in equation 10

$$ysn[n] = 1.5ytr[n] - 0.5ytr[n]^3 \tag{10}$$

This polynomial shapes the triangular waveform into an approximated sine-wave $ysn[n]$. The amplitude of $ysn[n]$ is attenuated with the parameter $excursion$, and an offset tau is added. Finally, the resulting signal (which oscillates within range $1 < \tau < 2$) is used to modulate the delay-line length, according to equation 2. Note that we did not take fractional delay-line length interpolation into account (to suppress "zipper-noise" due to the discrete-length model of the delay-line).

The overall behavior of the flanger is represented by the spectrogram in fig. 6. This spectrogram is a representation of the WaveCore simulation of the flanger model, with an impulse-train as input signal.

The WaveCore implementation of the flanger is a netlist representation of the block diagram in fig. 5. The applied PAs in this figure are indicated by the round-edge rectangular boxes. The parameter

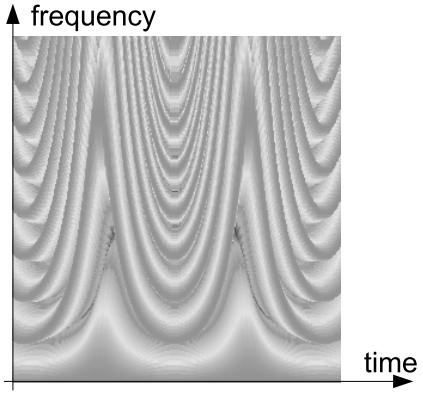


Figure 6: *Magnitude response spectrogram of flanger algorithm*

inputs ($rate, res, excursion, manual$ and $depth$) are run-time controllable, similar to the control knobs on a physical flanger device. As can be seen in fig. 5 the flanger implementation requires 16 WaveCore PAs, which implies less than 1% of the processing capacity of a single PU. The required minimum clock frequency for this application is 2.4 MHz. The PU memory requirement is 176 bytes. The processing latency of the flanger is zero, since the shortest path from $x[n]$ to $y[n]$ does not contain delay-elements. This implies that the overall flanger latency equals the latency in the AC97 codec, plus one additional sampling period for token buffering.

5.2. Auto-Wah

Similar to the flanger, the 'WahWah' audio effect is also widely used in several electronic musical instruments. The name reveals its sound effect. This sound effect is based on a bandpass filter with center frequency ω_0 and quality-factor Q, where the center frequency marks the middle of the pass-band and quality-factor determines the bandwidth of the pass-band. The center-frequency is dynamically varied within the WahWah effect. This can be done in various ways (e.g. expression pedal or controlled by input signal properties like envelope). In our example we chose an envelope controlled WahWah, which is sometimes referred to as 'Auto-Wah'. The continuous-time transfer function for a second order bandpass filter with ω_0 and Q parameters is defined in equation 11.

$$H(s) = \frac{(\frac{1}{Q.\omega_0}).s^2}{(\frac{1}{\omega_0^2}).s^2 + (\frac{1}{Q.\omega_0}).s + 1} \tag{11}$$

We need to translate the continuous-time transfer function to its discrete-time counterpart. For this purpose we use the bilinear transformation, which substitutes s with z, according to the substitution rule in equation 12.

$$s \leftarrow \frac{2}{T_s}.\frac{z-1}{z+1} \tag{12}$$

With T_s the sampling period. Application of the bilinear transformation yields the discrete-time transfer function $H(z)$ in equation 13, which embodies the required bandpass filter behavior with ω_0 and Q parameters.

$$H(z) = \frac{a(1-z^{-2})}{(a+b+1) + 2(1-b)z^{-1} + (1-a+b)z^{-2}} \tag{13}$$

With:

$$a = \frac{2}{Q.\omega_0.T_s} \text{ and } b = \frac{4}{T_s^2.\omega_0^2} \tag{14}$$

The final step is to correlate the coefficients in the derived transfer function $H(z)$ with the coefficients in the general second order discrete-time transfer function, which is defined in equation 15.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{15}$$

The relation between the system parameters in transfer function in equation 13 and the parameters in the generic second order transfer function in equation 15 is defined in equations 16,17 and 18.

$$b_0 = \frac{a}{d}\,,\, b_1 = 0\,,\, b_2 = \frac{-a}{d} \tag{16}$$

And:

$$a_1 = \frac{2(1-b)}{d}\,,\, a_2 = \frac{1-a+b}{d}, \tag{17}$$

With:

$$d = a + b + 1 \tag{18}$$

We map the generic transfer function in equation 15 on a direct-form II structure. This structure is depicted in the "DF2-IIR" WP-partition in fig. 7. The transfer function in equation 15, and associated parameters in equations 16, 17 and 18 apply to this DF2-IIR structure (note that the parameter b1 equals zero, and therefore is left out). The recipe for computing the coefficients b_0, b_2, a_1

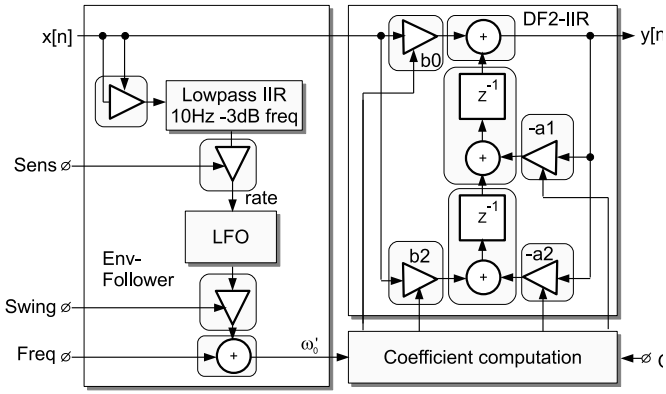


Figure 7: *AutoWah WaveCore process*

and a_2 from equations 16,17 and 18 is implemented in the WP-partition called "Coefficient Computation" (obviously with inputs ω_0 and Q). The "Env-Follower" WP-partition in fig. 7 detects the envelope of the input signal $x[n]$ and uses this envelope to compute the center frequency ω_0. Envelope detection is implemented by squaring the input signal, followed by lowpass filtering. The envelope signal is scaled and subsequently determines the frequency of the LFO (same LFO as used in the flanger in fig. 5). The sine wave shaped output of the LFO is scaled with the $swing$ parameter and added to the offset $freq$, and finally fed to the "Coefficient Computation" WP-partition.

The AutoWah WP requires 46 WaveCore PAs. This implies 2.2% of the processing capacity of a single PU. The AutoWah application requires a minimum clock frequency of 2.5 MHz. The PU memory requirement for the AutoWah is 368 bytes. The processing latency of the AutoWah algorithm is zero, since the shortest path from $x[n]$ to $y[n]$ does not contain delay-elements.

5.3. Automated compilation from Faust to WaveCore

In the third mapping example we focus on the WaveCore programming language compatibility with Faust [3]. Faust stands for Functional AUdio STream, and is a functional language designed for audio processing. The Faust compiler produces C++ code, specifically targeted to a C++ signal processor class. The Faust compiler can also produce a graph representation of the data-flow network which is extracted from the Faust language description. Typically an algorithm which is described in Faust consists of a data-flow process which is intended to run at real-time audio rate, and a control part which intends to implement the run-time control (e.g. sliders, knobs, etc.). We have developed an experimental compiler which maps the Faust graph representation onto a WaveCore process. This compiler writes a WP netlist description which subsequently can be compiled to WaveCore object code by the WaveCore mapping tool. We used the Faust implementation of a stereo variant of the Zita-Rev1 [6] reverberator to show the feasibility of WaveCore code generation from Faust. The Zita-Rev1 algorithm has been converted to WaveCore in a fully automated way.

The compiled Zita-Rev1 WP requires 284 WaveCore PAs, where 30 of these PAs infer delay-lines. This implies 12% of the processing capacity of a single PU. The Zita-Rev1 application requires a minimum clock frequency of 22 MHz. The PU memory requirement for Zita-Rev1 is 2608 bytes. The processing latency of the compiled Zita-rev1 Faust algorithm is zero since the shortest path from input $x[n]$ to output $y[n]$ does not contain delay-elements.

6. EVALUATION

We have developed a benchmark WaveCore algorithm set, which implements a combination of guitar effects processing and digital wave synthesis. This benchmark represents a guitar model (controlled by a console), a guitar-effects gear box and a model of distortion and speaker cabinet. The block diagram of this benchmark is depicted in fig. 8. The complete model is encoded in one composed WaveCore process which breaks down into four hierarchically built WP-partitions.

The first WP-partition in the chain instantiates 6 guitar string models. For a guitar string model we use a DWG (Digital Wave Guide) model, based on the Karplus-Strong algorithm [7]. Each string in the 6-string model can be tuned at run-time by a process which runs at the host computer. Moreover, plucking each individual string as well as adding damping characteristics (controlling the timbre of each individual string) can be controlled at run-time. Hence, a high level of "player" control is enabled by the model: like string bending, playing chords, artificial finger-picking etc.

The second WP-partition in the chain implements an acoustical model of a guitar body. The implementation of this model is based on a 1700-taps FIR filter. The six-string model, added with the acoustical guitar body model forms a digital model of a guitar: the "digitar".

The third WP-partition implements the effects gear-box. This is a model of a multi-effects rack. The digitized AC97 input signal is added to the "digitar" at the input of the gear model, and subsequently fed to the effects models in the rack. This enables to plug-in a real guitar into the analog input of the FPGA board. The effects in the rack are a 12-stage envelope Phaser, the "Zita-Rev1" reverberator [1] [6], the Auto-Wah, the Tremolo effect, and the flanger. The outputs of the individual effects are scaled and mixed with the unprocessed (i.e. "dry") signal which is fed through the "Bypass" unit. The effects in the gear-box, as well as the scaling/mixing is controlled at run-time by a process which runs on the host computer.

The fourth WP-partition implements a distortion model which is based an the BOSS DS1 distortion pedal [8], a speaker cabinet model and stereo rendering. The signal which is fed through the DS1 model is scaled and added with a scaled "dry" signal (through the "Bypass" unit). The output of the added signal is fed through a 1700-taps FIR filter which represents an acoustical model of a speaker cabinet. Finally, the output of the cabinet model is routed to the "left" channel output of the AC97 codec, and a delayed copy of this signal ("right") is routed to the "right" channel of the AC97 codec. The parameters for the DS1, the dry/wet mixing and the length of the stereo rendering delay can be varied at run-time by the host-computer. Table 2 shows the mapping results of the

Table 2: *Mapping results of Digitar benchmark algorithm set*

WP-partition	#PAs
6-String	184
GuitarBody	1712
GearBoxModel	551
Distortion/Cabinet-model	1774

PU	#Mapped PAs	Utilization	#Mapped DelayLines
1	738	41%	48
2	1714	95%	0
3	1775	99%	1
4	0	0%	0
5	0	0%	0

benchmark algorithm on the WaveCore/Atlys platform. The algorithm requires 365 MFLOPS and the overall processor utilization is 47% (2 idle PUs and the other ones not entirely loaded). Furthermore, the processing latency in the entire processing chain equals only two sampling periods (shortest path in the chain). The external memory bandwidth for this algorithm equals 9.6 MBytes/s, which is very modest. Hence, the real-time execution requirements of the overall algorithm are easily fulfilled with zero processing jitter.
Next to the described benchmark, we also applied another experiment to fully load the WaveCore cluster with interconnected biquad chains. We found that it is possible to compile 1710 biquad filter instances, divided over 30 WP-partitions within a single WP.

7. CONCLUSIONS AND FUTURE WORK

We have developed a scalable coarse-grained reconfigurable data-flow architecture, called WaveCore. WaveCore is optimized to the

[1] The Zita-Rev1 reverberator model has been generated from a Faust source by an experimental *Faust2WaveCore* compiler

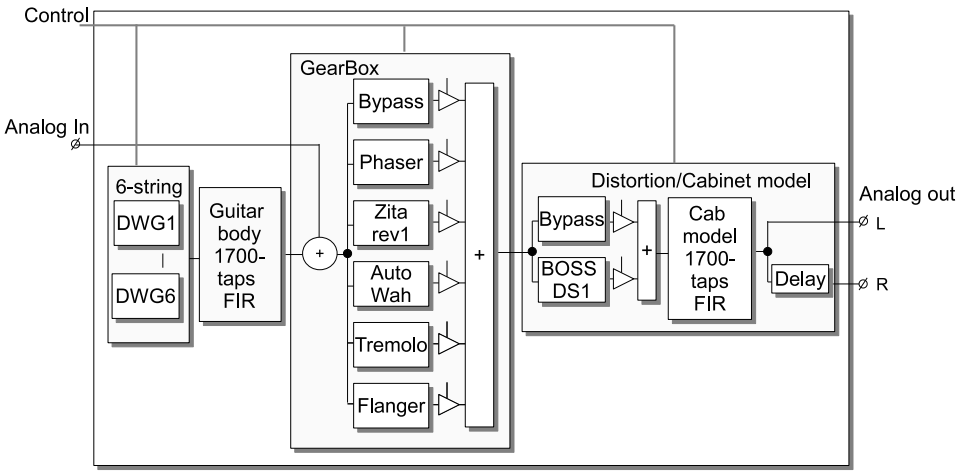


Figure 8: *Digitar benchmark setup.*

application domain of audio and acoustical processing. The programming model is declarative and based on explicit description of hierarchical data-flow networks. We found that the programming methodology is to a large extend compatible with Faust, which has been demonstrated with an experimentally compiled 'Zita-Rev1' reverberation algorithm. We have configured a WaveCore PU cluster for the Digilent Atlys, and applied this flashed WaveCore/Atlys board as a demonstrator platform.

We have developed a comprehensive digital effects processor for this platform which serves as a combination of guitar-effects processor and synthesizer with a DWG-based guitar model. The processing latency is negligible (few audio sampling periods) with zero processing jitter. The processing capacity of the demonstrator WaveCore/Atlys board proves that the architecture is very efficient.

Interesting target applications for the WaveCore technology range from cost effective audio effects solutions (e.g. multi-effects gear), to complex audio/acoustical modeling. The ultra low latency and real-time guaranteed execution makes it possible to apply the technology to hybrid instruments. The declarative programming methodology enables efficient interfacing of the technology to a class of functional languages with data-flow characteristics. Faust is an obvious example of this, but other existing data-flow programming methodologies are also interesting candidates. Functional language interfacing is an interesting topic for future work. Further benchmarking of the processor technology is also an interesting topic for future work.

8. ACKNOWLEDGMENTS

We thank Edgar Berdahl (CCRMA) for many constructive discussions on hybrid acoustical modeling, related to processor architectures. We thank Andreas Degert (Guitarix) for his contribution on the experimental *"Faust2WaveCore"* compiler which has resulted in a fully automatically compiled "Zita-Rev1" reverberation model. We also thank Jan Jacobs and Ramon Jongen from "Zuyd University of Applied Sciences" for their constructive discussions on architectures and signal processing applications.

9. REFERENCES

[1] Rolf Bader Florian Pfeifle, "Real-time finite difference physical models of musical instruments on a field programmable gate array (fpga)," in *Proc. of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, York, UK, Sept. 17-21 2012.

[2] Math Verstraelen, Jan Kuper, and Gerard J.M. Smit, "Wavecore: a reconfigurable mimd architecture," Submitted for publication, 2014.

[3] Stephane Letz Yann Orlarey, Dominique Fober, "FAUST (programming language)," Available at http://faust.grame.fr/, accessed Dec. 08, 2013.

[4] Digilent Inc., "Atlys Spartan-6 FPGA Development Board)," Available at http://www.digilentinc.com/Products/, accessed Jan. 21, 2014.

[5] Julius O. Smith, Ed., *Physical Audio Signal Processing for virtual musical instruments and digital audio effects*, W3K Publishing, USA, 2010.

[6] Julius O. Smith, "Zita-rev1," Available at https://ccrma.stanford.edu/ jos/pasp/Zita_Rev1.html, accessed Jan. 21, 2014.

[7] Kevin Karplus and Alex Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43–55, May 1983.

[8] Jonathan S. Abel David T. Yeh and Julius O. Smith, "Simplified, physically-informed models of distortion and overdrive guitar effects pedals," in *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10-15 2007.

[9] M. R. Schroeder and B. F. Logan, "Colorless artificial reverberation," *Journal of the Audio Engineering Society*, vol. 9, pp. 192–197, 1961.

[10] Udo Zolzer, *DAFX: Digital Audio Effects*, chapter Chapter 2: Filters, pp. 55–56, John Wiley & Sons, Ltd, 2002.

POLYPHONIC PITCH DETECTION BY ITERATIVE ANALYSIS OF THE AUTOCORRELATION FUNCTION

Sebastian Kraft, Udo Zölzer

Department of Signal Processing and Communications
Helmut-Schmidt-University
Hamburg, Germany
sebastian.kraft@hsu-hh.de

ABSTRACT

In this paper, a polyphonic pitch detection approach is presented, which is based on the iterative analysis of the autocorrelation function. The idea of a two-channel front-end with periodicity estimation by using the autocorrelation is inspired by an algorithm from Tolonen and Karjalainen. However, the analysis of the periodicity in the summary autocorrelation function is enhanced with a more advanced iterative peak picking and pruning procedure. The proposed algorithm is compared to other systems in an evaluation with common data sets and yields good results in the range of state of the art systems.

1. INTRODUCTION

Polyphonic and multipitch detection is still an unresolved problem in the field of music analysis. A lot of research has been conducted in this area in the last two or three decades and many quite different approaches were developed and published. While the best of these algorithms generally achieve detection accuracies above 60 % in objective evaluations on identical data sets, none of them ever reached values above 70 % [1]. Regarding the multitude of publications in this field it is difficult to give a complete overview. Therefore, the authors would like to point the interested reader to [1, 2, 3] for an extensive survey of state of the art algorithms and only mention the most important ones that served as a basis for this publication in the following paragraphs.

A subgroup of pitch detection algorithms utilises an auditory model as a front-end to mimic the human hearing system, where the unitary pitch perception model from Meddis and O'Mard [4] is the most prominent one. All these models usually include an input filter bank to imitate the frequency resolution capability of the human cochlea. The individual filter channel outputs are then half-wave rectified and lowpass filtered which corresponds to the mechanical to neural transduction of the inner hair cells. Periodicity information per channel is extracted (e.g. using the autocorrelation) and finally summarised or jointly evaluated over all channels.

The basic idea from Meddis' model was used by Tolonen and Karjalainen in their pitch detection algorithm [5], but they drastically reduced the amount of filters in the auditory filter bank and only chose two channels for a maximally efficient implementation. The redundancy in the resulting overall summary autocorrelation function (SACF) was then removed by simply stretching the SACF by integer factors and subtracting it from itself. The analysis procedure is computationally efficient and straight-forward to implement but the detection accuracy can not compete against recently developed methods.

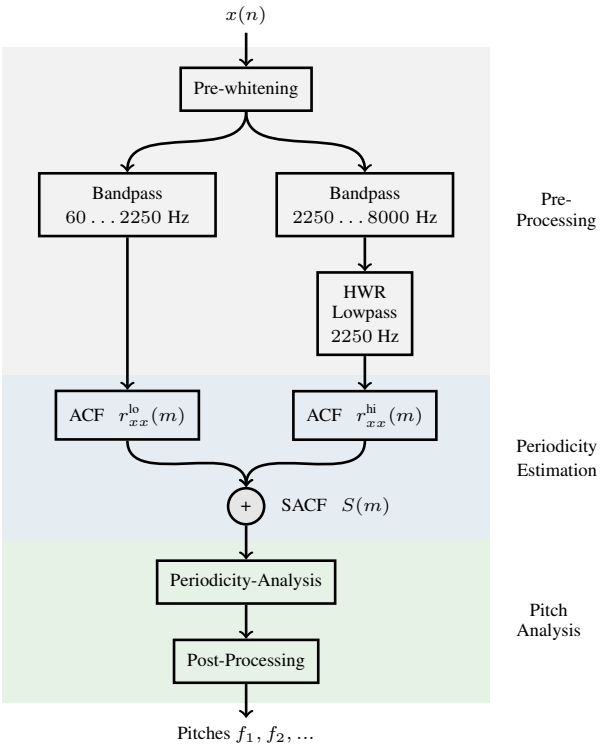


Figure 1: Block diagram of the presented pitch detection algorithm.

When it comes to the detection of multiple pitches with an auditory motivated front-end, one also has to consider the extensive research done by Klapuri [6, 7]. He uses an auditory model to split the input signal into several channels and periodicity information is retrieved from the sum of the individual channel spectra. The subsequent analysis process is looking for peaks with a strong corresponding harmonic series and iteratively removes the strongest series from the spectrum while selecting its base peak as a pitch candidate. The big filter bank (around 70 channels) and complex analysis induce high computational costs but the detection accuracies are good.

In this paper, a two channel auditory front-end like the one from Tolonen is used but the analysis of the periodicity information is replaced by a more advanced iterative peak picking and pruning procedure comparable to the one from Klapuri. Local

maxima in the SACF are detected and periodicity saliencies are calculated by summing the amplitudes at all integer multiples of a peak. High salience values will indicate a strong periodicity and the relating base period of the series can be assumed to be a good pitch candidate. A similar method has already been published by the same authors in [8] but the retrieved pitches were solely used as input for a chord detection and the whole algorithm was never optimised and evaluated in the context of multipitch analysis. Although it still shares the same basic idea, the implementation details and parameters changed a lot while the focus was shifted towards a pure polyphonic pitch detector.

In the following Section 2, the new algorithm will be described in detail followed by an evaluation with three well known data sets in Section 3, including a comparison with the state of the art approach from Benetos [9]. Section 4 will complete the paper with a summary and outlook to future developments.

2. PITCH DETECTION ALGORITHM

The block diagram of the presented pitch detector is depicted in Fig. 1 and in its underlying structure it is identical to the system of Tolonen [5]. Regarding the Pre-Processing and Periodicity Estimation stages, the main modification is a different parametrisation of the auditory front-end. However, the subsequent Pitch-Analysis block has been completely replaced by an iterative method. All signal processing is performed in overlapped blocks $x(n)$ of length N and the hop size between successive blocks N_h is set to $N/4$.

2.1. Pre-processing

The incoming signal block $x(n)$ is first of all processed by a pre-whitening filter. A signal model is estimated by linear prediction and inverse filtering with the model coefficients yields the pre-whitened input block with an equalised spectral envelope. To achieve a higher resolution in low frequency regions, the filter coefficients are determined by warped linear prediction (WLP) [10]. The WLP model was chosen to be of order 8 with a warping coefficient of 0.72 and the loss of signal energy by the filtering operation was compensated by comparing the overall power per block before and after the filter.

Afterwards the signal is split in two bands. The low channel bandpass filtering is realised by the sequential application of a lowpass and highpass at 2250 Hz and 60 Hz, respectively. The high channel bandpass is formed by a highpass at 2250 Hz followed by a lowpass at 8000 Hz. After half-wave rectification of the high channel signal, another lowpass at 2250 Hz is applied. All filters are second order IIR butterworth types [11] and the filtering is done per block in forward and backward directions to compensate for group delay but also to achieve steeper slopes. Finally, an individual periodicity estimation is performed in both channels.

2.2. Periodicity estimation

The autocorrelation function (ACF) is a common way to determine the periodicity of a signal and it has been frequently used to retrieve pitch information in the past. By using the Wiener-Khintchine theorem it can be efficiently calculated in the frequency domain as the inverse Fourier transform of the power spectrum. To avoid cyclic convolution from the DFT and to respect that the length of an autocorrelation sequence is $N_r = 2N - 1$, the input block has to be zero-padded to N_r before applying the DFT. In this case N_r is chosen to be $N_r = 2N$ (nearest power of two for an efficient FFT implementation). The input block $x(n)$ is first weighted by a Tukey (tapered cosine) window with a control parameter $\alpha = 0.4$ and after appending N zeros the resulting vector

$$\mathbf{x_p} = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \\ 0 \\ \vdots \\ 0 \end{bmatrix}^{N_r \times 1}, \qquad N_r = 2N \tag{1}$$

can be used to calculate the autocorrelation

$$\mathbf{r_{xx}} = \text{IDFT}\left(|\text{DFT}(\mathbf{x_p})|^2\right). \tag{2}$$

By replacing the square in (2) with a parameter γ

$$\mathbf{r_{xx}} = \text{IDFT}\left(|\text{DFT}(\mathbf{x_p})|^\gamma\right) \tag{3}$$

the ACF is non-linearly distorted and the amount of distortion can be easily adjusted. In the presented algorithm $\gamma = 0.6$ was used. The ACF is calculated individually in the high and low channel and the summary autocorrelation function (SACF)

$$S(m) = r_{xx}^{\text{lo}}(m) + r_{xx}^{\text{up}}(m), \quad m \in [0, \ldots, N_r], \tag{4}$$

with the time lag index m, is further analysed in the next step to extract the pitch information.

One interesting feature of the ACF in general, and also of the SACF as used in this paper, is the fact that its shape is approximately independent from the spectral envelope of the input signal. In Fig. 2 the SACFs of four harmonic signals with an identical fundamental frequency of 440 Hz but different spectral envelopes are shown. Although some of the signals have quite different partial amplitudes or even missing partials in the spectrum, the main period is clearly visible in all SACF plots and the corresponding peaks have an identical amplitude gradient. This is particularly beneficial for iterative detection approaches. Detected peaks have to be removed before the next iteration starts and the wrong estimation of peak amplitudes in the case of overlapping peaks is a common difficulty for algorithms that perform this kind of processing in the spectrum. In the SACF, the envelope is highly predictable and can be simply determined by fitting a smooth curve through the peak amplitudes.

2.3. Periodicity analysis

The SACF contains all the periodicity information from the input signal emphasised by the various pre-processing steps. The challenge is to analyse the SACF and to transfer this periodicity information to distinctive pitches. In [5] the SACF was iteratively stretched and subtracted from itself to remove redundant information. The remaining peaks above a final threshold eventually mark the most prominent fundamental periods in the signal. While being computationally efficient and easy to implement, the repeated reductions are not very specific as with increasing stretch factors the widening and subtraction of the SACF increasingly deforms the relevant peaks. Therefore, we propose to replace this analysis step with an iterative peak picking and pruning approach.

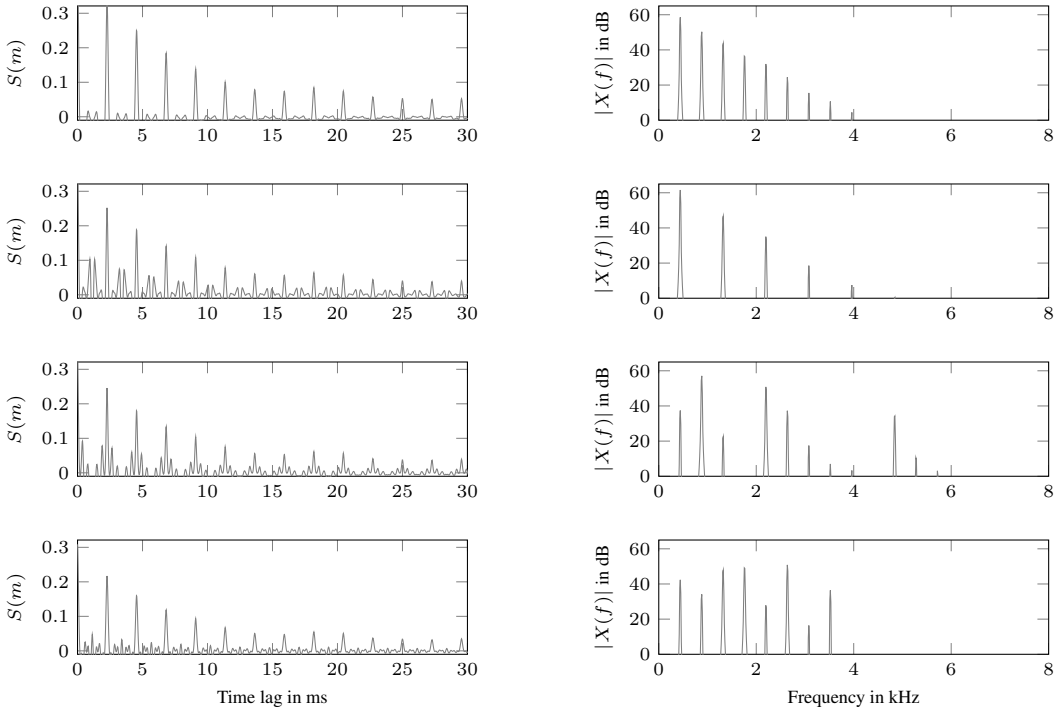


Figure 2: Outputs of the summary autocorrelation function (SACF) for input signals with different spectral envelopes. Fundamental frequency of all signals is 440 Hz which corresponds to a period of 2.27 ms.

2.3.1. Periodicity salience

Initially, a set of all local maxima (or peaks)

$$\mathbb{M} = [m_1, m_2, \ldots, m_i, \ldots, m_M], \quad m_{\text{lo}} < m_i < m_{\text{hi}} \tag{5}$$

above a threshold δ_1 in the SACF is identified, where m_{lo} and m_{hi} are the minimum and maximum lag values to take into account as fundamental frequencies and $i \in [1, \ldots, M]$ is the index of the maximum in the list. For every maximum a corresponding periodicity salience will be calculated by summing the SACF values at all integer multiples. A high salience will indicate that the investigated maximum is the base peak of a strong series in the SACF and hence, is a good candidate for a fundamental period.

The whole process is shown as pseudo code in Algorithm 1 and described in detail in the following paragraphs. The outer loop iterates over all detected maxima in $\mathbb{M}$. A tolerance value $\Delta_m = 4 + {}^{m_i}/_{25}$ is calculated for the maximum m_i and the corresponding salience s_i is initialised with the SACF amplitude $S(m_i)$ of the base peak. The peak counter k_i is set to one and the exact position of the first maximum $\hat{m}_{i,1}$ is initialised with m_i.

The inner loop iterates over all integer multiples k of the base peak, whereas k is bound to the nearest integer $[{}^{m_{\text{max}}}/_{m_i}]$ and m_{max} denotes the maximum lag that is considered being a multiple. The k-th multiple of m_i in the series is estimated to appear at

$$m_{i,k} = \hat{m}_{i,k-1} + m_i, \qquad k = 1, 2, 3, \ldots, [\frac{m_{\text{max}}}{m_i}] \tag{6}$$

and the exact location

$$\hat{m}_{i,k} = \underset{m_{i,k} \pm \Delta_m}{\text{argmax}} \; [S(m)] \tag{7}$$

is retrieved as the local maximum of $S(m)$ in a range of $\pm\, \Delta_m$ around the approximate position. If the periodicity error

$$\Delta_{\hat{m}_{i,k}} = |m_{i,k} - \hat{m}_{i,k}| \tag{8}$$

is smaller than the tolerance Δ_m, a valid peak in the current series is detected. Its amplitude $S(\hat{m}_{i,k})$ is added to the periodicity salience

$$s_i = s_i + S(\hat{m}_{i,k}) \tag{9}$$

and the counter of detected peaks in the current series

$$k_i = k_i + 1 \tag{10}$$

is incremented by one.

After the border m_{max} is reached and $m_{i,k} > m_{\text{max}}$ for the current k, a refined base peak position

$$\hat{m}_i = \frac{1}{k_i} \sum_{k \in \mathbb{K}} \frac{\hat{m}_{i,k}}{k} \tag{11}$$

can be calculated by taking the mean value of all peak positions in the series, where $\mathbb{K}$ is the set of all k where the maxima satisfy Eq. (8). This even allows sub-sample accuracy in the period measurement and therefore, an increased frequency resolution in particular for high frequencies. Otherwise, the precision would be limited by the sample time $T_s = {}^1/_{f_s}$. Furthermore, the saliencies

$$s_i = s_i \cdot \left(\frac{k_i}{\frac{m_{\text{max}}}{\hat{m}_i}} \right)^2 \tag{12}$$

// iterate over all maxima m_i in $\mathbb{M}$
for $i \leftarrow 1$ **to** M **do**
  $\Delta_m \leftarrow 4 + {}^{m_i}/_{25}$
  $s_i \leftarrow S(m_i)$
  $k_i \leftarrow 1$
  $\hat{m}_{i,1} \leftarrow m_i$

  // iterate over all multiples $m_{i,k}$
  for $k \leftarrow 2$ **to** $[{}^{m_{max}}/_{m_i}]$ **do**
    $m_{i,k} \leftarrow \hat{m}_{i,k-1} + m_i$
    $\hat{m}_{i,k} \leftarrow \text{argmax}_{m_{i,k} \pm \Delta_m} [S(m)]$
    $\Delta_{\hat{m}_{i,k}} \leftarrow |m_{i,k} - \hat{m}_{i,k}|$

    // if peak error is smaller than tolerance
    if $\Delta_{\hat{m}_{i,k}} < \Delta_m$ **then**
      $s_i \leftarrow s_i + \text{SACF}(\hat{m}_{i,k})$
      $k_i \leftarrow k_i + 1$
    end
  end
  $\hat{m}_i \leftarrow \frac{1}{k_i} \sum_{k \in \mathbb{K}} \frac{\hat{m}_{i,k}}{k}$
  $s_i \leftarrow s_i \cdot \left(\frac{k_i}{m_{\max}/\hat{m}_i} \right)^2$
end

Algorithm 1: Calculation of periodicity saliencies s_i for a set of detected maxima $\mathbb{M}$.

are weighted by the number of detected peaks over the number of potentially available peaks below $m_{\max}$. This factor can be interpreted as a measure of how complete a series is and it goes down to zero if only a few random or even no multiples were found.

The maximum $m_i^\star$ with the strongest salience s_i is then finally chosen as the first pitch candidate and the corresponding fundamental frequency

$$f_1 = \frac{f_s}{\hat{m}_i^\star} \tag{13}$$

is calculated with the help of the sampling frequency f_s.

2.3.2. Peak pruning

After selecting the strongest maximum, the corresponding peak series (base peak and multiples) has to be removed from the SACF before proceeding to the next iteration. The pruning procedure is shown in pseudo code in Algorithm 2. The detection of multiples in a series is identical to the one in Algorithm 1 and its detailed description is found in the previous section.

In Sec. 2.2 it was already mentioned that the envelope of a peak series is well predictable and in this case it is assumed to follow an exponential curve

$$\hat{S}(m) = a \cdot e^{b \cdot m}, \tag{14}$$

where the parameters a and b are estimated by a curve fitting algorithm. After erasing the base peak, all exact positions of the multiples are identified and removed. The removal of a peak with the removePeak() function in the pseudo code works as follows:

1. Find the inflection points left and right of $m_{i,k}^\star$ to determine the width of the peak.
2. Retrieve the estimated peak amplitude.

$\Delta_m \leftarrow 4 + {}^{m_i^\star}/_{25}$
$\hat{m}_{i,1}^\star \leftarrow m_i^\star$
// remove base peak $m_i^\star$
removePeak($m_i^\star$)

// remove all multiples of $m_i^\star$
for $k \leftarrow 2$ **to** $[{}^{m_{max}}/_{m_i^\star}]$ **do**
  $m_{i,k}^\star \leftarrow \hat{m}_{i,k-1}^\star + m_i^\star$
  $\hat{m}_{i,k}^\star \leftarrow \text{argmax}_{m_{i,k}^\star \pm \Delta_m} [S(m)]$
  $\Delta_{\hat{m}_{i,k}^\star} \leftarrow |m_{i,k}^\star - \hat{m}_{i,k}^\star|$

  // if peak error is smaller than tolerance
  if $\Delta_{\hat{m}_{i,k}^\star} < \Delta_m$ **then**
    removePeak($\hat{m}_{i,k}^\star$)
  end
end

Algorithm 2: Pruning of a periodic series from the SACF $S(m)$ starting with the most salient maximum at $m_i^\star$.

3. Create a tapered cosine window $w(m)$ (Tukey window) which spans the whole width of the peak (parameter $\alpha = 0.2$) and is zero elsewhere.
4. Remove the peak by multiplication with a properly scaled inverse window

$$w(m)' = \left(1 - \frac{\hat{S}(\hat{m}_{i,k}^\star)}{S(\hat{m}_{i,k}^\star)} \cdot w(m) \right) \tag{15}$$

$$S(m) = S(m) \cdot w'(m), \tag{16}$$

where $\hat{S}(\hat{m}_{i,k}^\star)$ is the expected peak amplitude determined by the curve fitting as in (14). In the case that $\hat{S}(\hat{m}_{i,k}^\star) > S(\hat{m}_{i,k}^\star)$, the quotient has to be bound to one to avoid a negative window amplitude.

After the removal of all peaks in the series the next iteration starts and the whole process is repeated until a certain break condition is met.

2.3.3. Break condition

There are two possible conditions to stop the iterations for the current frame and to proceed to the next one. First condition is to limit the average number of iterations to the expected count of simultaneous note events (polyphony). As this is usually unknown and may also change drastically throughout a musical piece, the polyphony alone is not a sufficient criterion. Therefore, iterations will also stop when the strongest salience does not any more excel a threshold δ_2, where usually $\delta_2 > \delta_1$.

2.3.4. Parameters

From the previous algorithmic description it could already be seen that there are a lot of free parameters. Most of them are quite empirical and can only be tweaked manually without any mathematical or physical relationship. This makes it difficult to give an optimal parameter set. However, the parameters in Table 1 turned out to yield good results with all data sets during the development process and also in the later evaluation. All parameters were determined for a sampling frequency of 44.1 kHz.

Description	Param.	Value
Block length	N	4096
Hop size	N_h	1024
Peak position tolerance	Δ_m	$4 + m_i/25$
Peak detection threshold	δ_1	0.025
Salience threshold	δ_2	$0.12 \approx 5\,\delta_1$
Max. number of iterations	-	6
Min. period of base peaks m_i	m_{lo}	30
Max. period of base peaks m_i	m_{hi}	735
Max. period of multiples $m_{i,k}$	m_{max}	2048

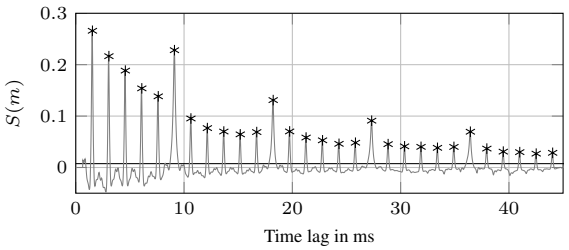
Table 1: Parameters of the periodicity analysis ($f_s = 44.1$ kHz).

2.3.5. Example

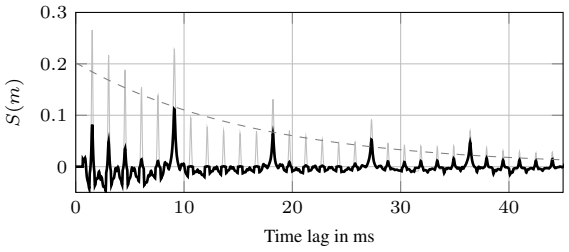
In Fig. 3 the peak picking and pruning procedure is depicted for a single iteration on a sample signal containing two harmonic tones with fundamental frequencies of 110 Hz and 659 Hz. The peaks of the strongest detected series in the first iteration are marked by an asterisk in Fig. 3a). This series is then removed in Fig. 3b) under the assumption of the estimated envelope which is drawn as a grey dashed line. Now, the residual thick black curve mainly contains periods of the lower fundamental frequency and the corresponding strongest series is chosen in Fig. 3c). Due to the smooth and well approximated envelope of the peak amplitudes it is possible to separate these tones even though the two series completely overlap.

2.4. Post-processing

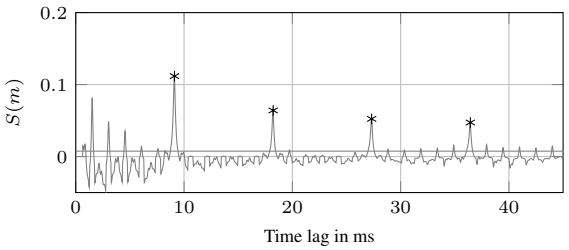
A simple post-processing filter was used to remove isolated and spurious detections with the length of a single frame. It is also intended to fill single frame gaps in otherwise stable detections over various frames. Despite its simplicity it turned out to be very effective. Applied to algorithms with many spurious false positives the post-processing has the ability to drastically raise the Precision with only negligible decrease of the Recall values.



(a) Selected peak series with the strongest salience



(b) After removal of the first series



(c) Selection of the next series

Figure 3: Peak picking and pruning in the SACF of a signal with fundamental frequencies of 110 Hz and 659 Hz. Subplot a) shows the selected peak series with the strongest salience in the first iteration which is then removed in b), where the dashed line shows the estimated envelope. The lower frequency series stays intact after the removal. In the residual c) the next series will be selected.

3. EVALUATION

3.1. Data sets

The pitch detection algorithm, described in the previous chapter, has been evaluated with three different data sets. All of them are established in the community and have been used to evaluate various other algorithms in the past:

- *Bach10 Data Set* [12] consists of ten excerpts from several J.S. Bach chorales played by violin, clarinet, saxophone and bassoon. Matlab data files with fundamental frequencies and onset/offset times are supplied as ground truth.
- *MIREX Multi-F0 Woodwind Development Data Set* [13, 14] is the recording of a woodwind quintet (flute, oboe, clarinet, horn and bassoon) with the respective pitch information as a MIDI file. The whole recording has a length of 9 minutes and is one of the pieces used in the evaluation of the annual *MIREX Multiple Fundamental Frequency Estimation and Tracking* task. Only a 30 second training snippet is publicly available and was used for this evaluation.
- *TRIOS Score-aligned Multitrack Recordings Data Set* [15] is a collection of 4 multitrack recordings of short extracts from classical trio pieces performed by piano, string and several wind instruments. It also includes an additional recording of the famous *Take Five* jazz piece played by piano, saxophone and drums.

Regarding the density and polyphony of the music, the Bach10 data set is the most simple one. Its pieces are played by a quartet of monophonic instruments and therefore, have a maximum polyphony of four. The same holds true for the MIREX piece, but as it is played by a quintet, its polyphony is limited to five. The most complex data set is TRIOS as it contains two monophonic instruments mixed with a difficult piano track which alone induces a high polyphony. All input signals are available at a sample rate of 44.1 kHz and were mixed down to mono if necessary. Additional normalisation to a mean sample power of one was applied to allow an almost data set independent parametrisation of the algorithms.

3.2. Metrics

For the calculation of the evaluation metrics, the amount of true positive, false positive and false negative detections were counted on a frame basis of 10 ms and accumulated over all songs in a data set. Based on these values the standard metrics Precision, Recall and F-measure were retrieved [13]. If the pitch detector output was given as a set of fundamental frequencies, they were converted and rounded to the closest integer MIDI value.

3.3. Algorithms and parameters

Besides the approach presented in this paper, three other algorithms were investigated. The algorithm from Tolonen [5] shares the same front-end as the presented approach. Hence, its purpose is to show if the new iterative analysis of the SACF yields any advantages. The algorithm from Klapuri [7] is also based on an auditory front-end but uses a far more complex filter bank as input stage. Its iterative analysis procedure is comparable to the presented one. Both algorithms were carefully implemented by the authors of this paper in Matlab.

Finally the publicly available Matlab implementation[1] of a recent algorithm presented by Benetos [9] is included. It is among the best algorithms that have participated in the MIREX campaign in the last years and well suited to compare the presented algorithm to a current state of the art system. Regarding its processing principle it is completely different to the other systems in this evaluation. The algorithm takes the log-frequency spectrogram matrix as input and tries to find a suitable factorisation into an activation matrix and accompanying spectral templates. In a training stage the spectral templates can be initialised with pre-trained spectra to guide the later factorisation process.

The three reference systems were parameterised as recommended in the respective papers. In particular:

- Benetos: sparsity for pitch activation $s_z = 1.05$, sparsity for source contribution $s_u = 1.5$, sparsity for pitch shifting $s_h = 1.1$. Time resolution of the resulting transcription matrix was 40 ms. Final threshold for the transcription matrix was set to $\delta_B = 45$.
- Klapuri: blocklength $N = 4096$, hop size $N_h = 2048$, all other parameters were chosen as proposed in [7].
- Tolonen: blocklength $N = 4096$, hop size $N_h = 1024$, all other parameters as in [5].

All parameters, and primarily the thresholds, were manually tweaked to yield a good balance between Precision and Recall throughout all data sets. Due to the huge amount of parameters it was not possible to iteratively optimize them automatically and it cannot be claimed that they are optimal under all conditions. However, the comparison with previously published evaluations in the next section will validate that the algorithms capabilities are well reflected in our results.

3.4. Results

The detailed results from the evaluation with all data sets are listed in Table 2. Every algorithm was evaluated in 4 different modes. The first block of results is from the pure pitch detector outputs. In the second block, the scores were calculated without taking the absolute octave into account and only the correct detection of the semitones was considered (chroma only). The post-processed results are achieved with the simple post-processing described in Sec. 2.4 and finally the post-processed results are also evaluated with chroma only metrics.

The auditory motivated iterative analysis of Klapuri yields generally better scores than the approach from Tolonen but it does not reach the results from recently developed algorithms. This matches the experience from various other evaluations [7, 16, 17] in the past. However, in absolute values our implementation of Klapuri's algorithm seems to be a few percent worse than reported in the above publications. In contrast the Tolonen algorithm performs a bit better than the implementation from the MIR Toolbox [18] used in [16, 17]. Comparing the post-processed Benetos results in Table 2 with the frame based F-measures in [9] (where a similar post processing was applied), one can see that the values are quite close for the MIREX and TRIOS data set (MIREX: 67.2 %, TRIOS: 66.5 % in [9]). The algorithm has also been evaluated in the context of the MIREX campaign [19] and detailed results are published on the corresponding website [14]. Again, the post-processed results from our evaluation of the Benetos implementation are in the same range. Small deviations of about 5 % may be caused by different parameter settings, thresholds, or in particular different training data. No data set specific training has been conducted during this evaluation and the pre-trained basis spectra from the available Matlab code have been used. However, in [19] it was mentioned that elaborate training with various instruments was performed for the MIREX contribution. After all, one can state that our results of the reference algorithms are plausible and they seem to be properly configured and evaluated.

The presented algorithm with an iterative analysis of the SACF clearly performs much better than the simple stretch and subtract procedure from Tolonen throughout all data sets and metrics. It also yields better results than our implementation of the Klapuri algorithm which uses a similar periodicity analysis but a much more complicated pre-processing. This is a good indication that it is not necessary to rely on a complex auditory model as a front-end. At least it seems possible to drastically reduce the amount of filters for a higher computational efficiency. The proposed system works best on the simple Bach10 data set, where the F-measure is 5.3 % better than Benetos when post-processing is applied. The results from all algorithms decrease with increasing complexity and polyphony of the music. Finally, on the most complex TRIOS data set, the presented approach and the one from Benetos reach a nearly identical F-measure of 62.9 % and 63.1 %, respectively. On all data sets, the Precision of the presented algorithm is constantly high and only the Recall degrades with increasing polyphony. This indicates a constantly low false positive rate and a slight penalty with highly polyphonic content.

The simple post processing turned out to be very effective and usually increases the Precision by 10-20 % on all algorithms with only minor impact on the Recall values. For future research it might be in particular interesting to see how it compares with more complex post-processing methods like note tracking, e.g. with a hidden Markov model (HMM) as in [20].

To summarize the evaluation, one can say that the presented algorithm with its iterative analysis of the SACF shows a clear advantage over the approach from Tolonen and is more accurate than the algorithm from Klapuri. In fact, the results indicate that the performance is in the range of current state of the art joint estimation approaches like the one from Benetos.

[1] `https://code.soundsoftware.ac.uk/projects/amt_mssiplca_fast`

	standard			chroma only			**post-proc.**			post-proc. + chroma only		
Algorithm	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	**F-meas.**	Prec.	Rec.	F-meas.	Prec.	Rec.
iterSACF	74.0 %	69.3 %	79.3 %	86.8 %	83.5 %	90.4 %	**85.0** %	90.2 %	80.3 %	94.4 %	100.0 %	89.3 %
Benetos[9]	68.4 %	61.6 %	76.8 %	86.4 %	81.7 %	91.7 %	**79.7** %	83.2 %	76.5 %	95.5 %	100.0 %	91.4 %
Klapuri[7]	61.9 %	60.0 %	64.0 %	72.1 %	67.5 %	77.3 %	**68.3** %	73.8 %	63.5 %	86.1 %	100.0 %	75.7 %
Tolonen[5]	61.4 %	61.5 %	61.2 %	72.9 %	70.7 %	75.3 %	**66.8** %	73.6 %	61.2 %	85.5 %	100.0 %	74.7 %

(a) Bach10 data set

	standard			chroma only			**post-proc.**			post-proc. + chroma only		
Algorithm	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	**F-meas.**	Prec.	Rec.	F-meas.	Prec.	Rec.
iterSACF	61.6 %	58.3 %	65.3 %	77.2 %	69.3 %	87.3 %	**73.2** %	83.7 %	64.9 %	90.7 %	100.0 %	83.0 %
Benetos[9]	63.9 %	62.0 %	65.9 %	78.0 %	71.5 %	85.9 %	**69.5** %	76.0 %	64.1 %	91.7 %	100.0 %	84.7 %
Klapuri[7]	51.0 %	50.5 %	51.5 %	68.2 %	60.9 %	77.6 %	**57.0** %	70.7 %	47.7 %	84.7 %	100.0 %	73.5 %
Tolonen[5]	41.4 %	40.5 %	42.3 %	62.9 %	54.2 %	74.9 %	**48.3** %	57.1 %	41.8 %	84.2 %	100.0 %	72.8 %

(b) MIREX data set

	standard			chroma only			**post-proc.**			post-proc. + chroma only		
Algorithm	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	**F-meas.**	Prec.	Rec.	F-meas.	Prec.	Rec.
iterSACF	54.5 %	58.8 %	50.8 %	73.3 %	71.8 %	74.8 %	**62.9** %	82.8 %	50.7 %	83.6 %	100.0 %	71.8 %
Benetos[9]	57.7 %	68.6 %	49.8 %	74.2 %	83.5 %	66.7 %	**63.1** %	86.6 %	49.6 %	79.4 %	100.0 %	65.9 %
Klapuri[7]	45.7 %	52.3 %	40.5 %	60.9 %	59.9 %	61.9 %	**50.5** %	70.7 %	39.2 %	73.6 %	100.0 %	58.2 %
Tolonen[5]	43.0 %	48.0 %	38.8 %	62.4 %	59.7 %	65.3 %	**47.4** %	61.7 %	38.5 %	77.9 %	100.0 %	63.8 %

(c) TRIOS data set

Table 2: Detailed evaluation results grouped by four different evaluation modes: standard rating from the pure pitch detector output, chroma only ratings, ratings with applied post-processing and finally with post-processing and chroma only ratings.

4. CONCLUSION

Starting from the two channel auditory front-end of Tolonen, a new method for the extraction of multiple fundamental frequencies from polyphonic signals was derived. It is based on a novel approach to iteratively extract pitch information from the autocorrelation function. The evaluation proves that the new algorithm is able to yield significantly higher scores than the basic system from Tolonen and also performs better compared to the similar iterative analysis from Klapuri. An average F-measure of 62.9 % was achieved with the TRIOS data set, 73.2 % with the MIREX piece and 85.0 % with the Bach10 data set. These are promising first results in the range of current state of the art algorithms. However, more extensive evaluations are necessary, e.g. in the context of the MIREX campaign, to give an absolute ranking.

One problem of the presented algorithm is its immense amount of parameters that can only be tweaked empirically. Detailed analysis of the parameters, thresholds and their influence on the metrics still has to be done but may be quite time consuming due to the high degree of freedom and existing parameter dependencies. Therefore, it may be interesting to keep the front-end and the advantages of the SACF as described here but apply a joint estimation analysis, as for example non-negative matrix factorisation (NMF) [21] or to make use of probabilistic methods like [9].

5. REFERENCES

[1] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri, "Automatic Music Transcription: Breaking the Glass Ceiling," in *Proc. 13th International Society for Music Information Retrieval Conference*, 2012.

[2] Anssi Klapuri, *Signal Processing Methods for the Automatic Transcription of Music*, Ph.D. thesis, 2004.

[3] Chunghsin Yeh, *Multiple Fundamental Frequency Estimation Of Polyphonic Recordings*, Ph.D. thesis, 2008.

[4] Ray Meddis and Lowel O'Mard, "A unitary model of pitch perception," *Journal of the Acoustical Society of America*, vol. 102, no. 3, pp. 1811–20, Sept. 1997.

[5] Tero Tolonen and Matti Karjalainen, "A computationally efficient multipitch analysis model," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 708–716, 2000.

[6] Anssi Klapuri, "A perceptually motivated multiple-f0 estimation method," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005.

[7] Anssi Klapuri, "Multipitch analysis of polyphonic music and speech signals using an auditory model," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 2, pp. 255–266, 2008.

[8] Adrian von dem Knesebeck, Sebastian Kraft, and Udo Zölzer, "Realtime System For Backing Vocal Harmonization," in *Proc. of the 14th Int. Conference on Digital Audio Effects*, 2011.

[9] Emmanouil Benetos, Srikanth Cherla, and Tillman Weyde, "An effcient shiftinvariant model for polyphonic music transcription," in *Proc. 6th International Workshop on Machine Learning and Music*, 2013.

[10] Unto K. Laine, Matti Karjalainen, and Toomas Altosaar, "Warped linear prediction (WLP) in speech and audio processing," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 1994.

[11] Udo Zölzer, *DAFX: Digital Audio Effects*, John Wiley & Sons, 2nd edition, 2011.

[12] Zhiyao Duan, Bryan Pardo, and Changshui Zhang, "Multiple Fundamental Frequency Estimation by Modeling Spectral Peaks and Non-Peak Regions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 8, pp. 2121–2133, Nov. 2010.

[13] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie, "Evaluation of multiple-f0 estimation and tracking systems," in *Proc. of the 10th International Society for Music Information Retrieval Conference*, 2009.

[14] MIREX, "Music Information Retrieval Evaluation eXchange," `http://music-ir.org/mirexwiki/`.

[15] Joachim Fritsch, *High Quality Musical Audio Source Separation*, Master, 2012.

[16] Emmanuel Vincent, Nancy Bertin, and Roland Badeau, "Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 528–537, Mar. 2010.

[17] Valentin Emiya, Roland Badeau, and Bertrand David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[18] Olivier Lartillot and Petri Toiviainen, "A matlab toolbox for musical feature extraction from audio," in *Proc. of the 10th Int. Conference on Digital Audio Effects*, 2007.

[19] Emmanouil Benetos and Tillman Weyde, "Multiple-f0 estimation and note tracking for mirex 2013 using an efficient latent variable model," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2013.

[20] Matti P. Ryynänen and Anssi Klapuri, "Polyphonic music transcription using note event modeling," in *Proc. IEEE-Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005.

[21] Paris Smaragdis and Judith C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003, number 3.

MUSIC-CONTENT-ADAPTIVE ROBUST PRINCIPAL COMPONENT ANALYSIS FOR A SEMANTICALLY CONSISTENT SEPARATION OF FOREGROUND AND BACKGROUND IN MUSIC AUDIO SIGNALS

Hélène Papadopoulos[*]
Laboratoire des Signaux et Systèmes
UMR 8506, CNRS-SUPELEC-Univ. Paris-Sud, France
helene.papadopoulos[at]lss.supelec.fr

Daniel P.W. Ellis
LabROSA
Columbia University
dpwe[at]ee.columbia.edu

ABSTRACT

Robust Principal Component Analysis (RPCA) is a technique to decompose signals into sparse and low rank components, and has recently drawn the attention of the MIR field for the problem of separating leading vocals from accompaniment, with appealing results obtained on small excerpts of music. However, the performance of the method drops when processing entire music tracks. We present an adaptive formulation of RPCA that incorporates music content information to guide the decomposition. Experiments on a set of complete music tracks of various genres show that the proposed algorithm is able to better process entire pieces of music that may exhibit large variations in the music content, and compares favorably with the state-of-the-art.

1. INTRODUCTION

In the general context of processing high-dimensional data, a recurrent problem consists in extracting specific information from a massive amount of related or unrelated information. Examples include recovering documents with specific topics from a collection of Web text documents [1] or detecting moving objects from camera recordings for video surveillance purpose [2]. Among numerous existing methods, the technique of Robust Principal Component Analysis (RPCA) [3, 4], has recently drawn a lot of attention. All the above-mentioned problems can be formulated as separating some foreground components (the keywords in Web data, the moving objects in video) from an underlying background (the background corpus topic in Web data, the stable environment in video), that can be respectively modeled as a sparse plus a low-rank contribution.

RPCA has been used extensively in the field of image processing (e.g. image segmentation [5], visual pattern correspondence [6], surveillance video processing [7], batch image alignment [8], etc.). However, its application in Music Information Retrieval (MIR) is much more recent. Existing applications in audio include audio classification, as in [9] where audio segments from video sound files are classified into classes (applause and laughter occurrences); [10] addresses the problem of refining available social tags obtained through social tagging websites to maximize their quality. The main application of the RPCA framework in music focuses on the task of separating a foreground component, usually the singing voice, from a background accompaniment in monaural polyphonic recordings, i.e., when only one channel of recording is available. This scenario is the primary focus of this paper.

The singing voice is a complex and important music signal attribute that has been much studied in MIR. Its separation is essential for many applications, such as singer identification [11], melody transcription [12], or query by humming [13]. We refer the reader to [14] for a recent review of singing voice separation methods. Recently, approaches that take advantage of repetition in the signal have emerged. These approaches assume that the background accompaniment has a repetitive musical structure, in contrast to the vocal signal whose repetitions, if any, occur only at a much larger timescale [15, 16, 17]. In [15] a simple method for separating music and voice is proposed based on the extraction of the underlying repeating musical structure using binary time-frequency masking (REPET algorithm). The methods assumes that there is no variations in the background and is thus limited to short excerpts. In [16], the method is generalized to permit the processing of complete musical tracks by relying on the assumption of local spectral-periodicity. Moreover, artifacts are reduced by using soft-masks. Inspired by these approaches, [17] proposes a model for singing voice separation based on repetition, but without using the hypothesis of local periodicity. The background musical accompaniment at a given frame is identified using the nearest neighbor frames in the whole mixture spectrogram.

Most recently, RPCA has emerged as a promising approach to singing voice separation based on the idea that the repetitive musical accompaniment may lie in a low-rank subspace, while the singing voice is relatively sparse in the time-frequency domain [18]. The voice and the accompaniment are separated by decomposing the Short-Time-Fourier Transform (STFT) magnitude (i.e., spectrogram) into sparse and low-rank components. When tested on short audio excerpts from the MIR-1K dataset[1] RPCA shows improvement over two state-of-the-art approaches [19, 15]. The decomposition is improved in [20] by adding a regularization term to incorporate a prior tendency towards harmonicity in the low-rank component, reflecting the fact that background voices can be described as a harmonic series of sinusoids at multiples of a fundamental frequency. A post-processing step is applied to the sparse component of the decomposition to eliminate the percussive sounds. [21] addresses the problem of jointly finding a sparse approximation of a varying component (e.g., the singing voice) and a repeating background (e.g., the musical accompaniment) in the same *redundant dictionary*. In parallel with the RPCA

[*] Part of this research was supported by a Marie Curie International Outgoing Fellowship within the 7th European Community Framework Program.

[1] The MIR-1K dataset [19] is a set of 1000 short excerpts (4 – 13s) extracted from 110 Chinese karaoke pop songs, where accompaniment and the singing voices are separately recorded. See https://sites.google.com/site/unvoicedsoundseparation/mir

idea of [3], the mixture is decomposed into a sum of two components: a *structured* sparse matrix and an *unstructured* sparse matrix. Structured sparsity is enforced using mixed norms, along with a greedy Matching Pursuit algorithm [22]. The model is evaluated on short popular music excerpts from the Beach Boys. [23] proposes a non-negative variant of RPCA, termed robust low-rank non-negative matrix factorization (RNMF). In this approach the low-rank model is represented as a non-negative linear combination of non-negative basis vectors. The proposed framework allows incorporating unsupervised, semi-, and fully-supervised learning, with supervised training drastically improving the results of the separation. Other related works including [24, 25] address singing voice separation based on low-rank representations alone but are beyond the scope of this article.

While RPCA performs well on the $\sim$10 sec clips of MIR-1K, the full-length Beach Boys examples of [14] give much less satisfying results. When dealing with whole recordings, the musical background may include significant changes in instrumentation and dynamics which may rival the variation in the foreground, and hence its rank in the spectrogram representation. Further, foreground may vary in its complexity (e.g., solo voice followed by a duet) and may be unevenly distributed throughout the piece (e.g., entire segments with background only). Thus, the best way to apply RPCA to separate *complete* music pieces remains an open question.

In this article, we explore an adaptive version of RPCA (A-RPCA) that is able to handle complex music signals by taking into account the intrinsic musical content. We aim to adjust the task through the incorporation of domain knowledge that guides the decomposition towards results that are physically and musically meaningful. Time-frequency representations of music audio may be structured in several ways according to their content. For instance, the frequency axis can be segmented into regions corresponding to the spectral range of each instrument of the mixture. In the singing separation scenario, coefficients that are not in the singing voice spectral band should not be selected in the sparse layer. In the time dimension, music audio signals can generally be organized into a hierarchy of segments at different scales, each with its own semantic function (bar, phrase, entire section etc.), and each having specific characteristics in terms of instrumentation, leading voice, etc. Importantly, as the segments become shorter, we expect the accompaniment to span less variation, and thus the rank of the background to reduce.

We will show a way for this music content information to be incorporated in the decomposition to allow an accurate processing of *entire* music tracks. More specifically, we incorporate voice activity information as a cue to separate the leading voice from the background. Music pieces can be segmented into vocal segments (where the leading voice is present) and background segments (that can be purely instrumental or may contain backing voices). Finding vocal segments (voicing detection [26]) is a subject that has received significant attention within MIR [26, 27, 28, 29]. The decomposition into sparse and low-rank components should be coherent with the semantic structure of the piece: the sparse (foreground) component should be denser in sections containing the leading voice while portions of the sparse matrix corresponding to non-singing segments should ideally be null. Thus, while the technique remains the same as [18] at the lowest level, we consider the problem of segmenting a longer track into suitable pieces, and how to locally adapt the parameters of the decomposition by incorporating prior information.

2. ROBUST PRINCIPAL COMPONENT ANALYSIS VIA PRINCIPAL COMPONENT PURSUIT

In [3] , Candès *et al.* show that, under very broad conditions, a data matrix $D \in \mathbb{R}^{m\times n}$ can be exactly and uniquely decomposed into a low-rank component A and a sparse component E via a convex program called *Principal Component Pursuit* (RPCA-PCP) given by:

$$\min_{A,E} \quad \|A\|_* + \lambda\|E\|_1 \quad \text{s.t.} \quad D = A + E \tag{1}$$

where $\lambda > 0$ is a regularization parameter that trades between the rank of A and the sparsity of E. The nuclear norm $\|\cdot\|_*$ – the sum of singular values – is used as surrogate for the rank of A [30], and the ℓ_1 norm $\|\cdot\|_1$ (sum of absolute values of the matrix entries) is an effective surrogate for the ℓ_0 pseudo-norm, the number of non-zero entries in the matrix [31, 32].

The Augmented Lagrange Multiplier Method (ALM) and its practical variant, the Alternating Direction Method of Multipliers (ADM), have been proposed as efficient optimization schemes to solve this problem [33, 34, 35]. ALM works by minimizing the augmented Lagrangian function of (1):

$$\mathcal{L}(A, E, Y, \mu) = \|A\|_* + \lambda\|E\|_1 + \langle Y, A+E-D\rangle + \frac{\mu}{2}\|A+E-D\|_F^2 \tag{2}$$

where $Y \in \mathbb{R}^{m\times n}$ is the Lagrange multiplier of the linear constraint that allows removing the equality constraint, $\mu > 0$ is a penalty parameter for the violation of the linear constraint, $\langle\cdot,\cdot\rangle$ denotes the trace inner product[2] and $\|\cdot\|_F$ is the Frobenius norm[3]. ALM [34] is an iterative scheme that works by repeatedly minimizing A and E simultaneously. In contrast, ADM splits the minimization of (2) into two smaller and easier subproblems, with A and E minimized sequentially:

$$A^{k+1} = \operatorname*{argmin}_A \quad \mathcal{L}(A, E^k, Y^k, \mu^k) \tag{3a}$$

$$E^{k+1} = \operatorname*{argmin}_E \quad \mathcal{L}(A^{k+1}, E, Y^k, \mu^k) \tag{3b}$$

Both subproblems (3a) and (3b) are shrinkage problems that have closed-form solutions that we briefly present here. We refer the reader to [34, 35] for more details. For convenience we introduce the scalar soft-thresholding (shrinkage) operator $\mathcal{S}_\epsilon[x]$:

$$\mathcal{S}_\epsilon[x] = \text{sgn}(x)\cdot\max(|x|-\epsilon, 0) = \begin{cases} x-\epsilon & \text{if} \quad x > \epsilon \\ x+\epsilon & \text{if} \quad x < -\epsilon \\ 0 & \text{otherwise} \end{cases}$$

where $x \in \mathbb{R}$ and $\epsilon > 0$. This operator can be extended to matrices by applying it element-wise.

Problem (3a) is equivalent to:

$$A^{k+1} = \min_A \left\{ \|A\|_* + \frac{\mu^k}{2}\|A - (D - E^k + \frac{1}{\mu^k}Y^k)\|_F^2 \right\} \tag{4}$$

that has, according to [36], a closed-from solution given by:

$$A^{k+1} = U\mathcal{S}_{\frac{1}{\mu^k}}[\Sigma]V^T$$

[2]The inner product between two matrices A and B is defined as $\langle A, B\rangle = trace(A^*B)$, * being the conjugate transpose.

[3]The Frobenius norm of matrix A is defined as $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$.

where $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ and $\Sigma \in \mathbb{R}^{r\times r}$ are obtained via the singular value decomposition $(U, \Sigma, V) = SVD(D - E^k + \frac{Y^k}{\mu^k})$.

Problem (3b) can be written as:

$$E^{k+1} = \min_E \left\{ \lambda\|E\|_1 + \frac{\mu^k}{2}\|E - (D - A^{k+1} + \frac{1}{\mu^k}Y^k)\|_F^2 \right\} \quad (5)$$

whose solution is given by the least-absolute shrinkage and selection operator (*Lasso*) [37], a method also known in the signal processing community as basis pursuit denoising [38]:

$$E^{k+1} = \mathcal{S}_{\frac{\lambda}{\mu_k}}[D - A^{k+1} + \frac{Y^k}{\mu_k}]$$

In other words, denoting $G^E = D - A^{k+1} + \frac{Y^k}{\mu^k}$:

$$\forall\, i \in [1, m], \forall\, j \in [1, n] \quad E_{ij}^{k+1} = \text{sgn}(G_{ij}^E)\cdot\max(|G_{ij}^E| - \frac{\lambda}{\mu^k}, 0)$$

3. ADAPTIVE RPCA (A-RPCA)

As discussed in Section 1, in a given song, the foreground vocals typically exhibit a clustered distribution in the time-frequency plane relating to the semantic structure of the piece that alternates between vocal and non-vocal (background) segments. This structure should be reflected in the decomposition: frames belonging to singing voice-inactive segments should result in zero-valued columns in E.

The balance between the sparse and low-rank contributions is set by the value of the regularization parameter λ. The voice separation quality with respect to the value of λ for the *Pink Noise Party* song *Their Shallow Singularity* is illustrated in Fig. 1. As we can observe, the best λ differs depending on whether we process the entire song, or restrict processing to just the singing voice-active parts. Because the separation for the background part is monotonically better as λ increases, the difference between the optimum λ indicates that the global separation quality is compromised between the singing voice and the background part.

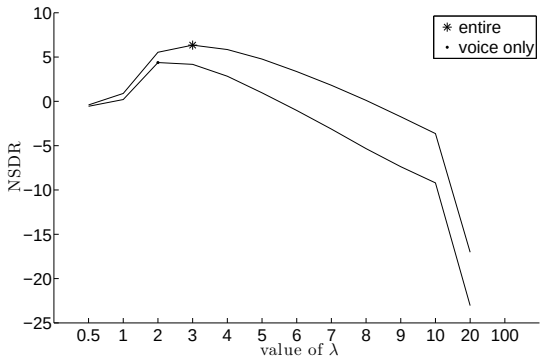


Figure 1: Variation of the estimated singing voice NSDR (see definition in Section 4) according to the value of λ under two situations. •: NSDR when only the singing voice-active parts of the separated signal are processed. ∗: NSDR when the entire signal is processed.

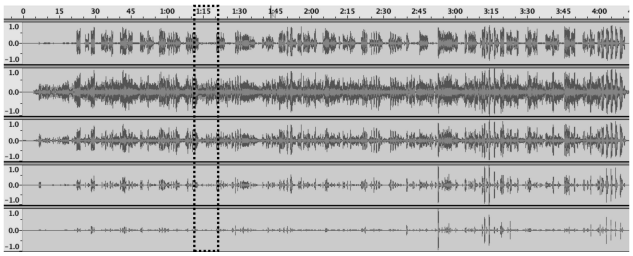


Figure 2: Waveform of the separated voice for various values of λ for the song *Is This Love* by Bob Marley. From top to bottom: clean voice, $\lambda = \lambda_1, 2*\lambda_1, 5*\lambda_1, 10*\lambda_1$.

In the theoretical formulation of RPCA-PCP [3], there is no single value of λ that works for separating sparse from low-rank components in all conditions. They recommend $\lambda = \max(m, n)^{-\frac{1}{2}}$ but also note that the decomposition can be improved by choosing λ in light of prior knowledge about the solution. In practice, we have found that the decomposition of music audio is very sensitive to the choice of λ with frequently no single value able to achieve a satisfying separation between voice and instrumental parts across a whole recording. This is illustrated in Fig. 2, which shows the waveforms of the resynthesized separated voice obtained with the RPCA-PCP formulation for various λ. For $\lambda = \lambda_1 = 1/\sqrt{\max(m, n)}$ and $\lambda_2 = 2{*}\lambda_1$, around $t = 1.15$ s (dashed rectangle) there is a non-zero contribution in the voice layer but no actual lead vocal. This is eliminated with larger values of λ, such as $\lambda = 5 * \lambda_1, 10 * \lambda_1$ but at the expense of a very poor quality voice estimate: the resulting signal consists of percussive sounds and higher harmonics of the instruments, and does not resemble the voice. Note that similar observations have been made in the context of video surveillance [39].

To address the problem of variations in λ, we propose an adaptive variant of the RPCA consisting of a weighted decomposition that incorporates prior information about the music content. Specifically, voice activity information is used as a cue to adjust the regularization parameter through the entire analyzed piece in the (3b) step, and therefore better match the balance between sparse and low-rank contributions to suit to the actual music content. This idea is related to previous theoretical work [40, 41, 42], but to our knowledge, its application in the framework of RPCA is new.

We consider a time segmentation of the magnitude spectrogram into N_{block} consecutive (non-overlapping) blocks of vocal / non-vocal (background accompaniment) segments. We can represent the magnitude spectrogram as a concatenation of column-blocks $D = [D_1 D_2 \cdots D_{N_{\text{block}}}]$, the sparse layer as $E = [E_1 \cdots E_\Lambda$ and $G^E = [G_1^E \cdots G_{N_{\text{block}}}^E]$.

We can minimize the objective function with respect to each column-block separately. To guide the separation, we aim at setting a different value of $\lambda_l, l \in [1, N_{blocks}]$ for each block according to the voice activity side information. For each block, the problem is equivalent to Eq. (5) and accordingly, the solution to the resulting problem:

$$E_l^{k+1} = \min_{E_l} \left\{ \lambda_l\|E_l\|_1 + \frac{\mu^k}{2}\|E_l - G_l^E\|_F^2 \right\}$$

is given by:

$$E_l^{k+1} = \mathcal{S}_{\frac{\lambda_l}{\mu_k}}[G_l^E] \quad (6)$$

Algorithm 1 Adaptive RPCA (A-RPCA)

Input: spectrogram D, blocks, $\lambda, \lambda_1, \ldots, \lambda_{N_{blocks}}$
Output: E, A
Initialization: $Y^0 = D/J(D)$ where $J(D) = max(\|D\|_2, \lambda^{-1}\|D\|_\infty)$; $E^0 = 0$; $\mu_0 > 0$; $\rho > 1$; $k = 0$
while not converged **do**
  update A:
  $(U, \Sigma, V) = SVD(D - E^k + \frac{Y^k}{\mu^k})$; $A^{k+1} = U\mathcal{S}_{\frac{1}{\mu^k}}[\Sigma]V^T$
  update E:
  for each block l **do**
    $\lambda = \lambda_l$;
    $E_l^{t+1} = \mathcal{S}_{\frac{\lambda_l}{\mu_k}}[D_l - A_l^{k+1} + \frac{Y_l^k}{\mu^k}]$
  end for
  $E^{t+1} = [E_1^{t+1} E_2^{t+1} \cdots E_{N_{\text{block}}}^{t+1}]$
  update Y, μ:
  $Y^{k+1} = Y^k - \mu^k(A^{k+1} + E^{k+1} - D)$
  $\mu^{k+1} = \rho \cdot \mu^k$
  $k = k + 1$
end while

Denote λ_{v} the constant value of the regularization parameter λ used in the basic formulation of RPCA for voice separation [18]. To guide the separation, in the A-RPCA formulation we assign to each block a value λ_l in accordance with the considered prior music structure information. Using a large λ_l in blocks without leading voice will favor retaining non-zero coefficients in the accompaniment layer. Denoting by Ω_V the set of time frames that contain voice, the values of λ_l are set as:

$$\forall\, l \in [1, N_{\text{block}}] \begin{cases} \lambda_l = \lambda_{\text{v}} \text{ if } E_l \subset \Omega_V \\ \lambda_l = \lambda_{\text{nv}} \text{ otherwise} \end{cases} \quad (7)$$

with $\lambda_{\text{nv}} > \lambda_{\text{v}}$ to enhance sparsity of E when no vocal activity is detected. Note that instead of two distinct values of λ_l, further improvements could be obtained by tuning λ_l more precisely to suit the segment characteristics. For instance, vibrato information could be used to quantify the amount of voice in the mixture within each block and to set a specific regularization parameter accordingly. The update rules of the A-RPCA algorithm are detailed in Algorithm 1.

In Section 4, we investigate the results of adaptive-RPCA with both exact (ground-truth) and estimated vocal activity information. For estimating vocal activity information, we use the voicing detection step of the melody extraction algorithm implemented in the MELODIA Melody Extraction vamp plug-in[4], as it is freely available for people to download and use. We refer the reader to [26] and references therein for other voicing detection algorithms. The algorithm for the automatic extraction of the main melody from polyphonic music recordings implemented in MELODIA is a salience-based model that is described in [43]. It is based on the creation and characterization of pitch contours grouped using auditory streaming cues, and includes a voice detection step that indicates when the melody is present; we use this melody location as an indicator of leading voice activity. Note that while melody can sometimes be carried by other instruments, in the evaluation dataset of Section 4 it is mainly singing.

[4] `http://mtg.upf.edu/technologies/melodia`

4. EVALUATION

In this section, we present the results of our approach evaluated on a database of complete music tracks of various genres. We compare the proposed adaptive method with the baseline method [18] as well as another state-of-the-art method [16]. Sound examples discussed in the article can be found at:
`http://papadopoulosellisdafx14.blogspot.fr`.

4.1. Parameters, Dataset and Evaluation Criteria

To evaluate the proposed approach, we have constructed a database of 12 complete music tracks of various genres, with separated vocal and accompaniment files, as well as mixture versions formed as the sum of the vocal and accompaniment files. The tracks, listed in Tab. 1, were created from multitracks mixed in Audacity[5], then exported with or without the vocal or accompaniment lines.

Following previous work [18, 44, 15], the separations are evaluated with metrics from the BSS-EVAL toolbox [45], which provides a framework for the evaluation of source separation algorithms when the original sources are available for comparison. Three ratios are considered for both sources: Source-to-Distortion (SDR), Sources-to-Interference (SIR), and Sources-to-Artifacts (SA In addition, we measure the improvement in SDR between the mixture d and the estimated resynthesized singing voice $\hat{e}$ by the Normalized SDR (NSDR, also known as *SDR improvement*, SDRI), defined for the voice as $\text{NSDR}(\hat{e}, e, d) = \text{SDR}(\hat{e}, e) - \text{SDR}(d, e)$, where e is the original clean singing voice. The same measure is used for the evaluation of the background. Each measure is computed globally on the whole track, but also locally according to the segmentation into vocal/non-vocal segments. Higher values of the metrics indicate better separation.

We compare the results of the A-RPCA with musically-informe adaptive λ and the baseline RPCA method [18] with fixed λ, using the same parameter settings in the analysis stage: the STFT of each mixture is computed using a Hanning window of 1024 samples length with 75% overlap at a sampling rate of 11.5KHz. No post-processing (such as masking) is added. After spectrogram decomposition, the signals are reconstructed using the inverse STFT and the phase of the original signal.

The parameter λ is set to $1/\sqrt{\max(m,n)}$ in the baseline metho Two different versions of the proposed A-RPCA algorithm are evaluated. First, A-RPCA with exact voice activity information, using manually annotated ground-truth (A-RPCA_GT), and $\lambda_l = \lambda$ for singing voice regions and $\lambda_l = 5 * \lambda$ for background only regions. In the other configuration, estimated voice activity location is used (A-RPCA_est), with same settings for the λ_l.

We also compare our approach with the REPET state-of-the-art algorithm based on repeating pattern discovery and binary time-frequency masking [16]. Note that we use for comparison the version of REPET that is designed for processing complete musical tracks (as opposed to the original one introduced in [15]). This method includes a simple low pass filtering post-processing step [46] that consists in removing all frequencies below 100Hz from the vocal signal and adding these components back into the background layer. We further apply this post-processing step to our model before comparison with the REPET algorithm.

Paired sample t-tests at the 5% significance level are performed to determine whether there is statistical significance in the results between various configurations.

[5] http://audacity.sourceforge.net

Table 1: Sound excerpts used for the evaluation; *back.* proportion of background (no leading voice) segments (in % of the whole excerpt duration); Recall *Rec.* and False Alarm *F.A.* voicing detection rate.

Name	% *back.*	*Rec.*	*F.A.*	Name	% *back.*	*Rec.*	*F.A*
1- *Beatles* Sgt Pepper's Lonely Hearts Club Band	49.3	74.74	45.56	8 - *Bob Marley* Is This Love	37.2	66.22	36.8
2 - *Beatles* With A Little Help From My Friends	13.5	70.10	14.71	9 - *Doobie Brothers* Long Train Running	65.6	84.12	58.5
3 - *Beatles* She's Leaving Home	24.6	77.52	30.17	10 - *Marvin Gaye* Heard it Through The Grapevine	30.2	79.22	17.9
4 - *Beatles* A Day in The Life	35.6	61.30	63.96	11 - *The Eagles* Take it Easy	35.5	78.68	30.2
5,6 - *Puccini* piece for soprano and piano	24.7	47.90	27.04	12 - *The Police* Message in aBottle	24.9	73.90	20.4
7 - *Pink Noise Party* Their Shallow Singularity	42.1	64.15	61.83				

4.2. Results and Discussion

Results of the separation for the sparse (singing voice) and low-rank (background accompaniment) layers are presented in Tables 2, 3, 4 and 5. To have a better insight of the results we present measures computed both on the entire song and on the singing voice-active part only, that is obtained by concatenating all segments labeled as vocal segments in the ground truth.

• **Global separation results.** As we can see from Tables 2 and 3, using a musically-informed adaptive regularization parameter allows improving the results of the separation both for the background and the leading voice components. Note that the larger the proportion of purely-instrumental segments in a piece (see Tab. 1), the larger the results improvement (see in particular pieces 1, 7, 8 and 9), which is consistent with the goal of the proposed method. Statistical tests show that the improvement in the results is significant.

As discussed in Section 3, the quality of the separation with the baseline method [18] depends on the value of the regularization parameter. Moreover, the value that leads to the best separation quality differs from one music excerpt to another. Thus, when processing automatically a collection of music tracks, the choice of this value results from a trade-off. We report here results obtained with the typical choice $\lambda_{\text{v}} = 1/\sqrt{\max(m,n)}$ in Eq. (7). Note that for a given value of λ_{v} in the baseline method, the separation can always be further improved by the A-RPCA algorithm using a regularization parameter that is adapted to the music content based on prior music structure information: in all experiments, for a given constant value λ_{v} in the baseline method, setting $\lambda_{\text{nv}} > \lambda_{\text{v}}$ in Eq. (7) improves the results.

For the singing voice layer, improved SDR (better overall separation performance) and SIR (better capability of removing music interferences from the singing voice) with A-RPCA are obtained at the price of introducing more artifacts in the estimated voice (lower SAR_{voice}). Listening tests reveal that in some segments processed by A-RPCA, as for instance segment $[1'00'' - 1'15'']$ in Fig. 3, one can hear some high frequency isolated coefficients superimposed to the separated voice. This drawback could be reduced by including harmonicity priors in the sparse component of RPCA, as proposed in [20]. This performance trade-off is commonly encountered in music/voice separation [14, 47]. However, we can notice that all three measures are significantly improved with A-RPCA for the background layer.

• **Ground truth versus estimated voice activity location.** Imperfect voice activity location information still allows an improvement, although to a lesser extent than with ground-truth voice activity information. In table 1, we report the accuracy results of the voicing detection step. Similarly to the measures used for melody detection in [48, 12], we consider the *Voicing Recall Rate*, defined as the proportion of frames labeled voiced in the ground truth that are estimated as voiced frames by the algorithm, and the *Voicing False Alarm Rate*, defined as the proportion of frames labeled as unvoiced in the ground truth that are mistakenly estimated to be voiced by the algorithm. The decrease in the results mainly comes from background segments classified as vocal segments. However, statistical tests show that the improvement in the results between RPCA and A-RPCA_est is still significant.

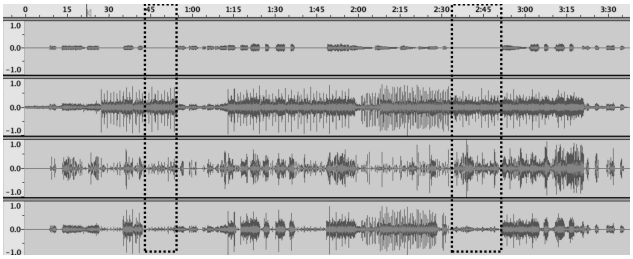


Figure 3: Separated voice for various values of λ for the *Pink Noise Party* song *Their Shallow Singularity*. From top to bottom: clean voice, constant $\lambda_1 = 1/\sqrt{\max(m,n)}$, constant $\lambda = 5 * \lambda_1$, adaptive $\lambda = (\lambda_1, 5 * \lambda_1)$.

• **Local separation results.** It is interesting to note that using an adaptive regularization parameter in a unified analysis of the whole piece is different from separately analyzing the successive vocal/non-vocal segments with different but constant values of λ (see for instance the dashed rectangles areas in Fig. 3).

• **Analysis of the results on vocal segments:** We expect the separation on background-only parts of the song to be improved with the A-RPCA algorithm. Indeed the side information directly indicates these regions where the foreground (sparse) components should be avoided; this can be clearly seen in Fig. 3. However, the improvements under the proposed model are not limited to non-vocal regions only. Results measured on the vocal segments alone indicate that by using the adaptive algorithm, the voice is also better estimated, as shown in Table 3. The improvement over RPCA is statistically significant, both when using ground truth and estimated voice activity location information. This indicates that side information helps not only to better determine the background only segments, but also enables improved recovery of the singing voice, presumably because the low-rank background model is a better match to the actual background.

Side information could have been added as a pre- or post-processing step to the RPCA algorithm. The adaptive-RPCA algorithm presents advantages over such approaches. To analyze this,

Table 2: SDR, SIR and SAR (in dB) and NSDR results for the voice (*Voice*) and background layer (*Back.*), computed across the whole song, for all models, averaged across all the songs. RPCA is the baseline system, A-RPCA_GT is the adaptive version using ground truth voice activity information, and A-RPCA_est uses estimated voice activity.

Entire song				
		RPCA	A-RPCA_GT	A-RPCA_est
Voice	SDR (dB)	-4.66	-2.16	-3.18
	SIR (dB)	-3.86	0.74	-0.46
	SAR (dB)	8.99	4.81	3.94
	NSDR	1.70	4.20	3.18
Back.	SDR (dB)	4.14	6.52	6.08
	SIR (dB)	11.48	13.30	12.07
	SAR (dB)	5.51	8.03	7.83
	NSDR	-2.35	0.03	-0.41

Table 3: SDR, SIR and SAR (in dB) and NSDR results for the v (*Voice*) and background layer (*Back.*), computed across the vocal s ments only, for all models, averaged across all the songs. RP is the baseline system, A-RPCA_GT is the adaptive version us ground truth voice activity information, and A-RPCA_est uses e mated voice activity.

Vocal segments				
		RPCA	A-RPCA_GT	A-RPCA_est
Voice	SDR (dB)	-3.19	-2.00	-1.96
	SIR (dB)	-2.33	-0.39	0.74
	SAR (dB)	9.44	7.27	4.64
	NSDR	1.67	2.85	2.90
Back.	SDR (dB)	3.63	5.18	5.28
	SIR (dB)	9.95	10.64	10.41
	SAR (dB)	5.39	7.32	7.54
	NSDR	-1.37	0.18	0.29

Table 4: SDR, SIR and SAR (in dB) and NSDR results for the voice (*Voice*) and background layer (*Back.*), computed across the whole song, for all models, averaged across all the songs. RPCA is the baseline system, A-RPCA_GT is the adaptive version using ground truth voice activity information, and A-RPCA_est uses estimated voice activity. Low-pass filtering post-processing is applied. REPET is the comparison algorithm [16].

Entire song					
		RPCA	A-RPCA_GT	A-RPCA_est	REPET
Voice	SDR (dB)	-2.76	-0.72	-2.11	-2.20
	SIR (dB)	-0.17	4.03	2.22	1.34
	SAR (dB)	4.33	3.33	2.32	3.19
	NSDR	3.60	5.64	4.25	4.16
Back.	SDR (dB)	5.16	7.61	6.81	5.01
	SIR (dB)	14.53	14.49	12.99	16.83
	SAR (dB)	5.96	9.02	8.44	5.47
	NSDR	-1.32	1.12	0.33	-1.48

Table 5: SDR, SIR and SAR (in dB) and NSDR results for the v (*Voice*) and background layer (*Back.*), computed across the vocal s ments only, for all models, averaged across all the songs. RP is the baseline system, A-RPCA_GT is the adaptive version us ground truth voice activity information, and A-RPCA_est uses e mated voice activity. Low-pass filtering post-processing is appl REPET is the comparison algorithm [16].

Vocal segments only					
		RPCA	A-RPCA_GT	A-RPCA_est	REPET
Voice	SDR (dB)	-1.25	-0.53	-0.83	-0.70
	SIR (dB)	1.49	3.04	3.62	3.02
	SAR (dB)	5.02	4.46	3.12	4.02
	NSDR	3.60	4.32	4.02	4.15
Back.	SDR (dB)	4.85	6.03	6.11	4.80
	SIR (dB)	13.07	12.38	11.41	15.33
	SAR (dB)	5.91	7.69	8.20	5.41
	NSDR	-0.14	1.03	1.11	-0.20

we compare the A-RPCA algorithm with two variants of RPCA incorporating side information either as a pre- or a post-processing step:

- RPCA_OV_{pre}: Only the concatenation of segments classified as vocal is processed by RPCA (the singing voice estimate being set to zero in the remaining non-vocal segments).
- RPCA_OV_{post}: The whole song is processed by RPCA and non-zeros coefficients estimated as belonging to the voice layer in non-vocal segments are transferred to the background layer.

Results of the decomposition computed across the vocal segments only are presented in Table 6. Note that the RPCA_OV_{post} results reduce to the RPCA results in Table 3 since they are computed on vocal segments only. There is no statistical difference between the estimated voice obtained by processing with RPCA the whole song and the vocal segments only. Results are significantly better using the A-RPCA algorithm than using RPCA_OV_{pre} and RPCA_OV_{post}. This is illustrated in Figure 4, which shows an example of the decomposition on an excerpt of the *Doobie Brothers* song *Long Train Running* composed of a non-vocal followed by a vocal segment. We can see that there are misclassified partials in the voice spectrogram obtained with the baseline RPCA that are removed with A-RPCA. Moreover, the gap in the singing voice around frame 50 (breathing) is cleaner in the case of A-RPCA than in the case of RPCA. Listening tests confirm that the background is better attenuated in the voice layer when using A-RPCA.

Table 6: SDR, SIR and SAR (in dB) and NSDR results for the voice (*Voice*) and background layer (*Back.*), computed across the vocal segments only, averaged across all the songs. RPCA_OV_{post} is when using the baseline system and set the voice estimate to zero in background-only segments, RPCA_OV_{pre} is when processing only the voice segments with the baseline model, A-RPCA_GT is the adaptive version using ground truth voice activity information, and A-RPCA_est uses estimated voice activity.

		RPCA_OV_{post}	RPCA_OV_{pre}	A-RPCA_GT	A-RPCA_est
Voice	SDR	-3.19	-3.28	-2.00	-1.96
	SIR	-2.33	-2.31	3.62	0.74
	SAR	9.44	8.97	7.27	4.64
	NSDR	1.67	1.57	2.85	2.90
Back.	SDR	3.63	3.72	5.18	5.28
	SIR	9.95	9.22	10.64	10.41
	SAR	5.39	5.85	7.32	7.54
	NSDR	-1.37	-1.28	0.18	0.29

• **Comparison with the state-of-the-art.** As we can see from Table 4, the results obtained with the RPCA baseline method are not better than those obtained with the REPET algorithm. On the contrary, the REPET algorithm is significantly outperformed by the A-RPCA algorithm when using ground truth voice activity information, both for the sparse and low-rank layers. However, note

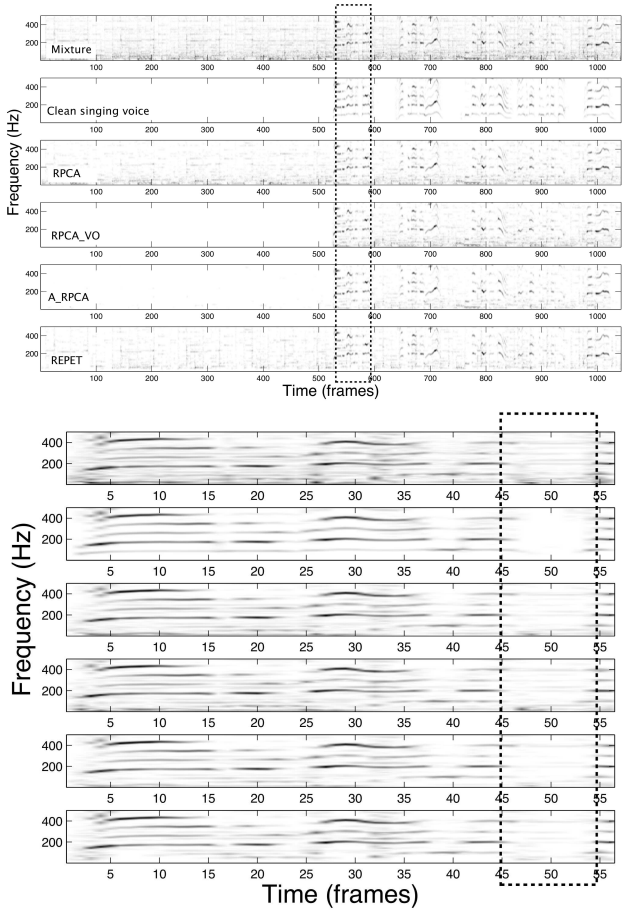


Figure 4: [Top Figure] Example decomposition on an excerpt of the *Doobie Brothers* song *Long Train Running* and [Bottom Figure] zoom between frames [525-580] (dashed rectangle in the Top Figure). For each figure, the top pane shows the part between 0 and 500Hz of the spectrogram of the original signal. The clean signing voice appears in the second pane. The separated signing voice obtained with baseline model (RPCA), with the baseline model when restricting the analysis to singing voice-active segments only (RPCA_OV_{pre}), and with the proposed A-RPCA model are represented in panes 3 to 5. For comparison, the sixth pane shows the results obtained with REPET [16].

that when using estimated voice activity information, the difference in the results between REPET and A-RPCA is not statistically significant for the sparse layer. If we look closer at the results, it is interesting to note that the voice estimation improvement by A-RPCA_GT over REPET mainly comes from the non-vocal parts where the voice estimated is favored to be null. Indeed, Table 5 indicate that the voice estimates on vocal segments obtained with A-RPCA_GT and REPET are similar. This is illustrated by the two last panes in the [bottom] Figure 4, which show similar spectrograms of the voice estimates obtained with the A-RPCA and REPET algorithms on the vocal part of the excerpt.

5. CONCLUSION

We have explored an adaptive version of the RPCA technique that allows the processing of entire pieces of music including local variations in the music structure. Music content information is incorporated in the decomposition to guide the selection of coefficients in the sparse and low-rank layers according to the semantic structure of the piece. This motivates the choice of using a regularization parameter that is informed by musical cues. Results indicate that with the proposed algorithm, not only the background segments are better discriminated, but also that the singing voice is better estimated in vocal segments, presumably because the low-rank background model is a better match to the actual background. The method could be extended with other criteria (singer identification, vibrato saliency. etc.). It could also be improved by incorporating additional information to set differently the regularization parameters for *each* track to better accommodate the varying contrast of foreground and background. The idea of an adaptive decomposition could also be improved with a more complex formulation of RPCA that incorporates additional constraints [20] or a learned dictionary [49].

6. REFERENCES

[1] K. Min, Z. Zhang, J. Wright, and Y. Ma, "Decomposing background topics from keywords by principal component pursuit," in *CIKM*, 2010.

[2] S. Brutzer, B. Hoferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *CCVPR*, 2011, pp. 1937–1944.

[3] E.J. Candès, X. Li, and J. Ma, Y. andb Wright, "Robust principal component analysis?," *Journal of the ACM*, vol. 58, no. 3, Article 11, 2011.

[4] V. Chandrasekaran, S. Sanghavi, P. Parrilo, and A. Willsky, "Sparse and low-rank matrix decompositions," in *Sysid*, 2009.

[5] B. Cheng, G. Liu, J. Wang, Z. Huang, and S. Yan, "Multi-task low-rank affinity pursuit for image segmentation," in *ICCV*, 2011, pp. 2439–2446.

[6] Z. Zeng, T.H. Chan, K. Jia, and D. Xu, "Finding correspondence from multiple images via sparse and low-rank decomposition," in *ECCV*, 2012, pp. 325–339.

[7] F. Yang, H. Jiang, Z. Shen, W. Deng, and D.N. Metaxas, "Adaptive low rank and sparse decomposition of video using compressive sensing," *CoRR*, vol. abs/1302.1610, 2013.

[8] Y. Peng, A. Ganesh, J. Wright, and Y. Xu, W. andMa, "Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2233–2246, 2012.

[9] Z. Shi, J. Han, T. Zheng, and S. Deng, "Online learning for classification of low-rank representation features and its applications in audio segment classification," *CoRR*, vol. abs/1112.4243, 2011.

[10] Y.H. Yang, D. Bogdanov, P. Herrera, and M. Sordo, "Music retagging using label propagation and robust principal component analysis," in *WWW*, New York, NY, USA, 2012, pp. 869–876.

[11] W. Cai, Q. Li, and X. Guan, "Automatic singer identification based on auditory features," 2011.

[12] J. Salamon, E. Gómez, D.P.W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications and challenges," *IEEE Signal Process. Mag.*, 2013.

[13] R.B. Dannenberg, W.P. Birmingham, B. Pardo, N. Hu, C. Meek, and G. Tzanetakis, "A comparative evaluation of search techniques for query-by-humming using the musart testbed," *J. Am. Soc. Inf. Sci. Technol.*, vol. 58, no. 5, pp. 687–701, 2007.

[14] B. Zhu, W. Li, R. Li, and X. Xue, "Multi-stage non-negative matrix factorization for monaural singing voice separation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 10, pp. 2096–2107, 2013.

[15] Z. Rafii and B. Pardo, "A simple music/voice separation method based on the extraction of the repeating musical structure," in *ICASSP*, 2011.

[16] A. Liutkus, Z. Rafii, R. Badeau, B. Pardo, and G. Richard, "Adaptive filtering for music/voice separation exploiting the repeating musical structure," in *ICASSP*, 2012.

[17] D. FitzGerald, "Vocal separation using nearest neighbours and median filtering," in *ISSC*, 2012.

[18] P.S. Huang, S.D. Chen, P. Smaragdis, and M. Hasegawa-Johnson, "Singing voice separation from monaural recordings using robust principal component analysis," in *ICASSP*, 2012.

[19] C.L. Hsu and J.S.R. Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 2, pp. 310–319, 2010.

[20] Y.H. Yang, "On sparse and low-rank matrix decomposition for singing voice separation," in *MM*, 2012, pp. 757–760.

[21] M. Moussallam, G. Richard, and L. Daudet, "Audio source separation informed by redundancy with greedy multiscale decompositions," in *EUSIPCO*, 2012, pp. 2644–2648.

[22] S.G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Audio, Speech, Language Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[23] P. Sprechmann, A. Bronstein, and G. Sapiro, "Real-time online singing voice separation from monaural recordings using robust low rank modeling," in *ISMIR*, 2012.

[24] A. Lefévre, F. Glineur, and P.A. Absil, "A nuclear-norm based convex formulation for informed source separation," in *ESANN*, 2013.

[25] Y.H. Yang, "Low-rank representation of both singing voice and music accompaniment via learned dictionaries," in *ISMIR*, 2013.

[26] J. Salamon, *Melody Extraction from Polyphonic Music Signals*, Ph.D. thesis, Department of Information and Communication Technologies Universitat Pompeu Fabra, Barcelona, Spain, 2013.

[27] A.L. Berenzweig and D.P.W. Ellis, "Locating singing voice segments within music signals," in *WASPAA*, 2001, pp. 119–122.

[28] T.L. Nwe and Y. Wang, "Automatic detection of vocal segments in popular songs," in *Proc. ISMIR*, 2004, pp. 138–145.

[29] L. Feng, A.B. Nielsen, and L.K. Hansen, "Vocal segment classification in popular music.," in *ISMIR*, 2008, pp. 121–126.

[30] M. Fazel, *Matrix Rank Minimization with Applications*, Ph.D. thesis, Dept of Elec. Eng., Stanford Univ., 2002.

[31] B. Recht, M. Fazel, and P.A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.

[32] E.J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, pp. 717–772, 2009.

[33] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," Tech. Rep. UILU-ENG-09-2214, UIUC Tech. Rep., 2009.

[34] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," Tech. Rep. UILU-ENG-09-2215, UIUC, 2009.

[35] Xiaoming Yuan and Junfeng Yang, "Sparse and low-rank matrix decomposition via alternating direction methods," *Preprint*, pp. 1–11, 2009.

[36] J.F. Cai, E.J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[37] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[38] S. Chen, L. David, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.

[39] Z. Gao, L.F. Cheong, and M. Shan, *Block-Sparse RPCA for Consistent Foreground Detection*, vol. 7576 of *Lecture Notes in Computer Science*, pp. 690–703, Springer Berlin Heidelberg, 2012.

[40] Y. Grandvalet, "Least absolute shrinkage is equivalent to quadratic penalization," in *ICANN 98*, L. Niklasson, M. Boden, and T. Ziemke, Eds., Perspectives in Neural Computing, pp. 201–206. Springer London, 1998.

[41] H. Zou, "The adaptive lasso and its oracle properties," *J. Am. Statist. Assoc.*, vol. 101, no. 476, pp. 1418–1429, 2006.

[42] D. Angelosante and G. Giannakis, "Rls-weighted lasso for adaptive estimation of sparse signals," in *ICASSP*, 2009, pp. 3245–3248.

[43] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, pp. 1759–1770, 2012.

[44] J.L. Durrieu, G. Richard, B. David, and C. Fevotte, ," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 3, pp. 564–575, March 2010.

[45] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 4, pp. 1462–1469, 2006.

[46] D. FitzGerald and M. Gainza, "Single channel vocal separation using median filtering and factorisation techniques," *ISAST Transactions on Electronic and Signal Processing*, vol. 4, no. 1, pp. 62–73, 2010.

[47] Z. Rafii, F. Germain, D.L. Sun, and G.J. Mysore, "Combining modeling of singing voice and background music for automatic separation of musical mixtures," in *ISMIR*, 2013.

[48] G. E. Poliner, D. P. W. Ellis, F. Ehmann, E. Gómez, S. Streich, and B. Ong, "Melody transcription from music audio: Approaches and evaluation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 15, no. 4, pp. 1247–1256, 2007.

[49] Z. Chen and D.P.W. Ellis, "Speech enhancement by sparse, low-rank and dictionary spectrogram decomposition," in *WASPAA*, 2013.

SEMI-BLIND AUDIO SOURCE SEPARATION OF LINEARLY MIXED TWO-CHANNEL RECORDINGS VIA GUIDED MATCHING PURSUIT

Dimitri Zantalis, *

Audio Lab, Department of Electronics,
University of York
York, UK
dimitri@zantalis.com

Jeremy J. Wells

Music Research Centre, Department of Music,
University of York
York, UK
jez.wells@york.ac.uk

ABSTRACT

This paper describes a source separation system with the intent to be used in high quality audio post-processing tasks. The system is to be used as the front-end of a larger system capable of modifying the individual sources of existing, two-channel, multi-source recordings. Possible applications include spatial re-configuration such as up-mixing and pan-transformation, re-mixing, source suppression/elimination, source extraction, elaborate filtering, time-stretching and pitch-shifting. The system is based on a new implementation of the Matching Pursuit algorithm and uses a known mixing matrix. We compare the results of the proposed system with those from `mpd-demix` of the 'MPTK' software package and show that we get similar evaluation scores and in some cases better perceptual scores. We also compare against a segmentation algorithm which is based on the same principles but uses the STFT as the front-end and show that source separation algorithms based on adaptive decomposition schemes tend to give better results. The novelty of this work is a new implementation of the original Matching Pursuit algorithm which adds a pre-processing step into the main sequence of the basic algorithm. The purpose of this step is to perform an analysis on the signal and based on important extracted features (e.g frequency components) create a mini-dictionary comprising atoms that match well with a specific part of the signal, thus leading to focused and more efficient exhaustive searches around centres of energy in the signal.

1. INTRODUCTION

In the sound engineering field, sometimes the post-processing of an already made stereophonic recording is necessary. For example, in a live studio setting, a system that modifies spatial information contained in a pre-existing two-channel recording could be an invaluable tool to the engineer, saving time and money. The engineer could up-mix [1], [2] the recording making it suitable for reproduction over different formats or apply panning transformation [3], [4], e.g from level panning to delay-based panning. Spatial re-configuration could also benefit consumers in the domestic listening environment. It is a fact that listening trends tend to vary and evolve over time thus it is highly desirable to be able to modify pre-recorded material. Apart from spatial effects other types of processing are source suppression/elimination (e.g. Karakoe system) and individual source modification such as filtering, changing/correcting pitch of single/multiple instrument(s), time stretching/compressing etc.

* This work was funded by EPSRC

All the aforementioned problems and probably many more, could be solved using a three stage approach: source separation followed by post-processing and finally remixing. A simple example system would have a coincident pair stereo recording as its input, separate the sources within the mixture and re-mix the separated sources using a different spatial configuration (e.g. by applying delays to produce time-difference panning) [5], [6]. It is clear that the crux of the system is the source separation step which should produce high quality results as this would most probably affect the quality of any subsequent processing.

Source separation is one of the trickiest types of signal processing and it is a vast field. Of course it would be extremely difficult to devise a solution that handles all cases of source separation and if such a system comes to life it would probably be a hybrid of parametrized and statistical modeling techniques and everything in between. Because of the complexity of the problem we need to state some assumptions, minimize the requirements and design a system that is realizable and scalable.

1.1. Mixing Model

In this work we assume an instantaneous mixing model with no delay-time parameter and no noise term:

$$x_m[n] = \sum_{p=1}^{P} \alpha_{mp} s_p[n] \,, \quad 1 \leq m \leq M \tag{1}$$

where $x_m[n]$ are the mixture signals, $s_p[n]$ the original source signals α_{mp} are the mixing coefficients, M the number of mixtures and P the number of sources. This can be expressed more compactly using matrix notation:

$$\mathbf{x} = \mathbf{A} \cdot \mathbf{s} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^{M \times N}$ are the mixture signals, $\mathbf{A} \in \mathbb{R}^{M \times P}$ is the mixing matrix where each element is a mixing coefficient α_{mp} and $\mathbf{s} \in \mathbb{R}^{P \times N}$ are the source signals. Equation 2 also makes the connection between the problem at hand and linear algebra where, the mixture signals can be seen as linear combinations of the source signals. The problem of source separation is essentially the estimation of the mixing matrix $\mathbf{A}$ and recovery of the sources $\mathbf{s}$ given the mixture $\mathbf{x}$. The problem of estimating both $\mathbf{A}$ and $\mathbf{s}$ is known as 'Blind Source Separation (henceforth BSS) and when audio signals are involved as 'Blind Audio Source Separation' (BASS).

1.2. Number of Mixtures & Sources

Source separation problems are classified based on the number of mixtures and sources. In this work we deal with two-channel

recordings comprising multiple sources, thus $M = 2$ and $P > M$. This leads to the under-determined case of source separation which in general is not a trivial task. In this case the inverse mixing matrix cannot be directly used to estimate the original sources. Instead other ways are employed to estimate the mixing matrix and separate the sources. Usually techniques based on 'Sparse Component Analysis' (henceforth SCA) are used since they produce good results for this source separation case [7]. In this work we use ideas from this field of study.

1.3. Known Mixing Matrix

The proposed system has knowledge of the mixing matrix $\mathbf{A}$. In this case we have a semi-blind source separation problem (SBSS or SBASS for audio signals). Estimation of the mixing matrix is usually treated as a separate problem to recovery of sources and many algorithms can handle this specific task with very good results (TIFROM [8], DUET [9], DEMIX [10], [11], [12] etc.). In fact we could estimate the mixing matrix with a modification of the proposed algorithm but this is outside the scope of this paper and will not be pursued any further. Generally speaking though we can safely assume that the mixing matrix is known or can be estimated accurately.

We also assume that the recording was made using a Blumlein pair, i.e. two 'figure-of-eight' microphones angled at 90°. This is a famous microphone technique that produces very accurate imaging of sources in the front quadrant and is widely used. A source is positioned in space, in-front of the listener, using inter-channel level differences alone, thus it is in accord with the mixing model presented in section 1.1. For the particular case of two-channel mixtures the mixing coefficients are given by:

$$\alpha_{1p} = \frac{\psi_p}{1+\psi_p}, \quad \alpha_{2p} = 1 - \psi_p \tag{3}$$

with

$$\psi_p = \frac{1+tan(\theta_p)}{1-tan(\theta_p)}, \quad -45^\circ \leq \theta_p \leq 45^\circ \tag{4}$$

where θ_p is the direction of the p^{th} source.

1.4. Sparsity of Sources

Another important assumption is that the source signals can be sparsely represented in a suitable domain. A signal is considered sparse in a domain if only few coefficients are needed to represent that signal in that domain. For example speech signals can be sparse in the time domain (e.g. two speakers talking in turns) whereas music cannot. Music exhibits sparsity in different domains such as the time-frequency domain. The notion of sparsity plays a central role in many signal processing fields including BSS and SCA techniques. In fact, regarding BSS, it has been shown that better separation can be achieved by exploiting sparse representations of signals [13]. By representing the signals in a sparse domain we hope that the coefficients of individual sources will be much more distinguishable (i.e we can see time-frequency regions where a source dominates) thus much easier to separate. After separation in the sparse domain, usually performed using frequency masks or clustering algorithms, we invert the separated sources back into the time domain to get the estimates. This is the main idea behind many SCA techniques.

Based on these assumptions we propose a semi-blind audio source separation algorithm that deals with linear, instantaneous mixtures and uses a new software implementation of Matching Pursuit (MP)[14] as its front-end. Similar algorithms that use MP for source separation are the 'Stereo Matching Pursuit'[15], the algorithms proposed in [16] which work with knowledge of the mixing matrix and the algorithm in[17].

Some other systems that are designed for active listening applications but with some overlapping goals are DReaM [18] and MPEG Spatial Audio Object Coding (SAOC) [19]. The fundamental difference against our algorithm is that these systems are based on encoder/decoder schemes. In particular inaudible meta-parameters are embedded within a mixture during the encoding stage and then used in the decoding stage for post processing such as re-mixing and respatialisation. Source separation algorithms based on encoder/decoder schemes are referred to as 'Informed Source Separation' and differ significantly from BASS and SBASS algorithms.

The rest of the paper is organized as follows: In section 2, we describe the original MP algorithm. In section 3, we introduce a modified MP version and make comparisons with basic MP algorithms. In section 4, we show how to apply the new proposed MP implementation in a source separation context. The final sections are for results obtained from our experiments, discussion on future work and conclusion.

2. BASIC MATCHING PURSUIT

Matching Pursuit is a recursive, adaptive algorithm for sparse signal decompositions. It belongs to a family of techniques known as 'Atomic Decompositions' (aka sparse decompositions or sparse atomic decompositions) that aim to decompose a given signal $\mathbf{x}$ as a linear combination of elementary waveforms $(g_\gamma)_{\gamma\in\Gamma}$, called atoms, taken from a dictionary $\mathbf{D}$. This can be formally expressed as [20]:

$$\mathbf{x} = \sum_{\gamma\in\Gamma} c_\gamma g_\gamma \tag{5}$$

where γ is a set of parameters characterizing each atom, g_γ are the individual atoms and c_γ are the expansion coefficients. We can also get an approximate decomposition for a fixed number of atoms m:

$$\mathbf{x} = \sum_{i=1}^{m} c_{\gamma i} g_{\gamma i} + R^{(m)} \tag{6}$$

where $R^{(m)}$ is a residual after an *m-term* decomposition. Matching Pursuit and similar algorithms, aim to find a sub-optimal solution to (5).

For basic MP we let $\mathbf{D} = \{g_\gamma | \gamma \in \Gamma\}$ be a dictionary comprising atoms of unit norm, $\|g_\gamma\| = 1$, for all $g_\gamma \in \mathbf{D}$. We also let the set of atoms in $\mathbf{D}$ to be redundant, i.e we have an over-complete dictionary. Decompositions in over-complete dictionaries are not unique since some atoms might be linearly dependent. MP will recursively build the approximation signal, one atom at a time, choosing at every iteration step the atom that minimizes $\|R^m\|$ in (6). In basic MP we first choose/create a dictionary $\mathbf{D}$, initialize $R^0 = \mathbf{x}$ and for each iteration step i we proceed as follows:

1. Compute inner products $\langle R^{i-1}, g_\gamma \rangle$, for all $g_\gamma \in \mathbf{D}$.
2. Select best atom $g_{\gamma i} = \arg\max_{g_\gamma \in \mathbf{D}} |\langle R^{i-1}, g_\gamma \rangle|$.
3. Get expansion coefficient $c_i = \langle R^{i-1}, g_{\gamma i} \rangle$.
4. Update the residual $R^i = R^{i-1} - c_i g_{\gamma i}$.
5. Check for exit conditions. If none is met continue to next iteration, otherwise stop decomposition.

We see that the standard inner product is used to compare the signal with the atoms in the dictionary. Thus the atom that maximizes the inner product is the one that minimizes the residual. MP is said to be a greedy algorithm in a sense that at every iteration it chooses the atom that removes the most energy from the residual[21]. We should note that MP can be configured to use other atom selection criteria and we will mention some of them in our proposed method. For a more detailed mathematical explanation of MP the reader is referred to [14].

Another important aspect of MP, and similar algorithms for that matter, is the choice or creation of the dictionary $\mathbf{D}$. The classic dictionary proposed in [14] is the Gabor dictionary which is parametric in nature; that is a set of parameters are needed to describe or create atoms of that type. A real Gabor atom is given by [15]:

$$g_{s,u,\xi,\phi} = K_{s,\xi,\phi} w(\frac{t-u}{s}) \cos(2\pi\xi(t-u) + \phi) \qquad (7)$$

where s is the scale parameter (i.e length of the atom), u the location parameter (i.e. location within the signal), ξ the frequency and ϕ the phase of the atom. $w(t)$ is generally any normalized window but for Gabor atoms a Gaussian window is used and can be defined as [22]:

$$w(t) = (\pi\sigma_s)^{-0.25} e^{-\frac{t^2}{2\sigma_s}} \qquad (8)$$

where σ_s is the variance of the window:

$$\sigma_s = (\frac{4}{\pi}) 2^{2(s_0 - s)}, \quad s = 0, 1, 2, ...s_0 \qquad (9)$$

and s_0 depends on the application. In this work we set $s_0 =$ `nextpow2(N)-2`, where `N` is the maximum length of an atom in samples. Finally $K_{s,\xi,\phi}$ in (7) is a normalizing constant, set so that the atom has unit norm. Other parametric dictionaries are the Fourier dictionary, Chirplet dictionary, DCT and DST dictionaries, Gamma-tone and Gamma-chirp dictionaries etc. each having its own set of parameters. Other methods for creating non-parametric dictionaries exist but for the proposed MP implementation we are mostly interested in parametric ones.

Creating a parametric dictionary covering all possible parameter values would be impractical so usually the parameter space of a dictionary is discretised. For example for the Gabor dictionary we could include all atoms with scales $N = 2^s$ with $s = 1, ...S$, frequencies $\omega_k = 2\pi k/N$ for $k = 1...N/2$ and shift locations u every $N/4$ samples. Even so such dictionaries can become very large, especially when we consider joining multiple dictionaries, making the realization of MP almost impossible. The state of the art, of a publicly available MP implementation, is the 'Matching Pursuit ToolKit' (MPTK) [23] which is very fast and has support for multi-channel signals and multiple dictionaries. We use the MPTK in this work as a reference system.

3. GUIDED MATCHING PURSUIT

A modification of the basic MP algorithm is proposed where a pre-processing step is included as the first step in the main sequence of events. At every iteration, the pre-processing step performs some kind of analysis to the residual and extracts important information which is then used to create a mini-dictionary $\mathbf{D}_i$ containing a fraction of the atoms that exist in the original dictionary $\mathbf{D}$. For example the pre-processing step could be a Fourier analysis of the residual, where the frequency components with the maximum magnitude can be used to create the atoms in $\mathbf{D}_i$. In this particular example we choose the frequency components with maximum magnitude since these are most likely to contain a big portion of the signal energy. The idea is that the newly created atoms will correlate well with the corresponding frequency components of the residual. The pre-processing step acts as a guide for creating atoms that might best correlate with the features of the signal we are interested in, therefore we term this new approach as 'Guided Matching Pursuit' (henceforth GMP). Although this is a simple modification of basic MP, this approach has some interesting properties and allows for novel signal decompositions and transformations.

For this work we use the fast Fourier transform (FFT) (or short-time Fourier transform (STFT) for long signals) as the analysis of the pre-processing step, since this is a very simple and fast analysis we can perform on a signal and, as already mentioned, can give us information about the frequency of important partials in the residual. Note that the analysis part can be more elaborate; for example a phase vocoder analysis step or a re-assigned magnitude spectrum could be used to get the instantaneous frequencies of partials instead of the frequencies corresponding to the frequency bins of the FFT. Also other information could be used for the creation of atoms such as the phase obtained from the complex spectra. Some of these options have been tested and in some cases lead to much better results than when a simple FFT is used, but these are not consistent. This issue requires further study. Although we use steps similar to classic sinusoidal analysis systems, GMP differs significantly from these in that the resulting decomposition goes beyond the sinusoidal plus transient plus residual model usually proposed by these systems. The modeling of the underlying audio signal depends on the selection of the dictionary which can contain many different types of atoms (e.g. Gabor atoms, chirp atoms, harmonic atoms, wavelets, learned atoms etc.).

A good property of this method is that every $\mathbf{D}_i$ contains a set of atoms which is much smaller than an ordinary dictionary implementation. To put it in perspective a normal Gabor dictionary, discretized as mentioned earlier, would contain tens of thousands of atoms whereas with our method each $\mathbf{D}_i$ can contain as few as 5 atoms per iteration (e.g. $1\ frequency\ \times 5\ scales$). This is a big reduction in the number of atoms we need to correlate with, which for a 'textbook' implementation of MP is a huge reduction in computation. Also with our method it is much easier to create dictionaries comprising different atom types. Of course the atoms should be mathematically defined (i.e. have parameters that describe them) but this should not pose a problem since many interesting dictionaries exist that can represent a wealth of signal features and share a similar structure and parameter space. For example some possible atoms that can be used are Gabor atoms, Fourier atoms (i.e. Sine and Cosine atoms), DCT and DST atoms, Gaussian chirps, damped sinusoids, Gamma-tones, Gamma-chirps and FM atoms.

Another trick that we employ is the computation of groups of inner products using the FFT, as mentioned in [23]. We know that the inner product of two real, square-integrable functions $f(x)$ and $g(x)$, on an interval $[a, b]$ is given by:

$$\langle f, g \rangle = \int_a^b f(x)g(x)dx \tag{10}$$

We also know that the cross-correlation between two continuous functions f and g is the 'sliding inner product' of those two functions:

$$f(t) \star g(t) = \int_{-\infty}^{+\infty} f(\tau - t)g(t)d\tau \tag{11}$$

Finally comparing the cross-correlation with the convolution of two continuous functions f and g:

$$f(t) * g(t) = \int_{-\infty}^{+\infty} f(t - \tau)g(\tau)d\tau \tag{12}$$

we see that the only difference is the reversal of f for the convolution operation. Thus cross-corralation and convolution are related by:

$$f(t) \star g(t) = f(-t) * g(t) \tag{13}$$

and in the case where f is Hermitian (which implies a symmetric real part) then the cross-correlation and convolution operations are the same. By taking advantage of the convolution theorem and using a fast linear convolution algorithm we can compute groups of inner products really fast. Also since fast convolution computes the inner products between the atoms and the residual for every sample, the need to shift atoms along the residual is eliminated; that is the atom location parameter u is implied by the location of the correlation coefficient with the maximum absolute value (step 2 in basic MP).

Let $\mathbf{x} \in \mathbb{R}^N$ be a short duration, mono-channel input signal, N be the length of the signal in samples, $R(t)$ be the residual after the decomposition, $R(\omega)$ the Fourier transform of the residual, k a frequency bin index, $\mathbf{D}_i$ a mini-dictionary, g_γ the atoms in the dictionary, $\mathbf{C} \in \mathbb{R}^{N \times M}$ a matrix holding the cross-correlations between the residual and each atom in the dictionary, then the steps for a GMP implementation are as follows:

1. Initialise $R^0(t) = \mathbf{x}$, $i = 1$.
2. Compute FFT of residual: $R^{i-1}(\omega) = FFT\left(R^{i-1}(t), N\right)$ with N being the size of the FFT.
3. Select frequency bin with maximum magnitude $k = \arg\max_{k \in R(\omega)} \left|R^{i-1}(\omega)\right|$.
4. Create mini-dictionary $\mathbf{D}_i$ comprising real atoms (of possibly different types) with normalized frequency k/N and different scales.
5. Compute cross-correlations of the residual with all atoms in $\mathbf{D}_i$: $\mathbf{C}_i = XCORR\left(R^{i-1}(t), \mathbf{D}_i\right)$.
6. Select best atom $g_{\gamma i} = \arg\max_{g_\gamma \in \mathbf{D}_i} |\mathbf{C}_i|$.
7. Get expansion coefficient $c_i = \left\langle R^{i-1}(t), g_{\gamma i} \right\rangle$.
8. Update the residual $R^i(t) = R^{i-1}(t) - c_i g_{\gamma i}$.
9. Check for exit conditions. If none is met increase iteration number i by one and go to step 2, otherwise stop decomposition.

Steps 2, 3 and 4 collectively form the analysis step we discussed earlier in its simplest form. Note that these steps could be different depending on the information we want to extract from the residual. Also in this case we assume that the residual is a short signal (e.g. $N = 2048$ samples). If the residual is very long then we should apply the STFT and step 3 would select the frequency bin with the maximum magnitude from a single time frame or maybe select the frequency bins with maximum magnitude from each time frame, for multiple atom extraction (in contrast to single atom extraction of original MP). It is clear that adding a preprocessing step to the basic MP opens up new ways of looking for specific features in a given signal.

We compare the proposed algorithm against MATLAB's ® `wmpalg` [24] (hencefoth WMP) and MPTK [23]. A 2048 samples long snippet of the 'o' vowel, is decomposed using a Fourier dictionary for 20 iterations (i.e. 20 atoms in the decomposition). Table 1 shows the residual energy and the signal to residual ratio at the last iteration and the time taken for each MP implementation to complete. We can see that GMP and MPTK perform better compared to WMP with GMP giving better results overall. MPTK is faster but we should take into account that MPTK is written in C++ whereas GMP and WMP are written in MATLAB's ® `mcode` thus they are not optimised for speed. Having said that we can see that GMP performs much faster than WMP. Figure 1 shows how the energy of the residual decays with every iteration and figure 2 shows how the SRR increases with every iteration. Again we can see GMP performing better compared to MPTK and WMP.

A MATLAB® implementation of GMP with all files that produce these and subsequent results can be found in [25].

Table 1: *Metrics for different MP algorithms after 20 iterations.*

Algorithm	Res. enrg. (dB)	SRR (dB)	time (sec)
GMP	0.6441	13.83	0.89
MPTK	0.9748	12.03	0.39
WMP	1.8315	9.29	1.75

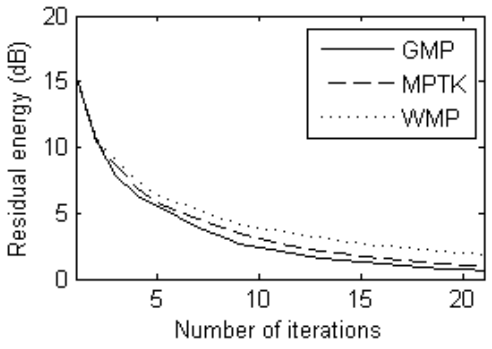


Figure 1: *Residual energy decay curves for three different MP implementations.*

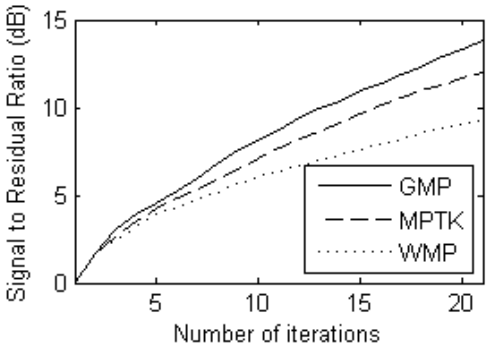


Figure 2: *Signal to residual ratio (SRR) for three different MP implementions.*

4. APPLICATION OF GMP IN SBASS

So far we have described how GMP works and we have shown that we get desirable results when the algorithm is used for signal decomposition. In this section we explain how to modify GMP to work with a source separation problem.

First of all we assume that we deal with two-channel recordings so the first modification regards an extension of the algorithm that works with multi-channel signals. In GMP this is easily achieved by applying the analysis steps to all channels of a signal. In particular we alter step 2 of GMP to compute the Fourier transform of both channels, then we add the resulting spectra and step 3 selects the frequency bin with the maximum magnitude from the new combined magnitude spectrum. Another approach could be to select frequency bins from each channel spectrum and the resulting dictionary would contain atoms with corresponding frequencies. The former approach was used since it makes sure that the frequency selection step is not biased by a particular source direction. The algorithm continues with the creation of the dictionary $\mathbf{D}_i$ and the cross-correlation of $\mathbf{D}_i$ with each channel of the residual.

Let us focus on step 5 of GMP. Assuming $\mathbf{D}_i$ holds M atoms then $\mathbf{C}$ is an $N \times M$ matrix where each column holds the N samples long cross-correlation of an atom with the residual. In the multi-channel case $\mathbf{C}$ becomes a $N \times M \times J$ matrix where the third dimension represents channels with $J = 2$ for a two-channel signal. Also remember that cross-correlation can be thought of as a 'sliding inner product', so every sample in each column of $\mathbf{C}$ is effectively the inner product between an atom and the residual signal at a particular time instance n , $\forall n \in \{1..N\}$. Thus the correlation samples in $\mathbf{C}$ are all potential expansion coefficients. We will therefore refer to that matrix as the coefficient matrix $\mathbf{C}$. Because of our instantaneous mixing model assumption in section 1.1 we can use the coefficient matrix to calculate the estimated directions of each expansion coefficient pair (i.e. left and right channel coefficients at same time instance) using:

$$\mathbf{\Theta} = \arctan\left(\frac{|\mathbf{C}_{n,m,2}|}{|\mathbf{C}_{n,m,1}|}\right) - \frac{\pi}{4}\,, \forall n \in \{1..N\},\ \forall m \in \{1..M\}. \tag{14}$$

where $\mathbf{\Theta} \in \mathbb{R}^{N \times M}$ and the constant $\pi/4$ is subtracted in order to bring the estimated directions in the range of $-\pi/4 \leq \mathbf{\Theta}_{n,m} \leq \pi/4$ which stems from our assumption in section 1.3. What we are interested in, is the distribution of the values in each column of $\mathbf{\Theta}$. Figure 3 shows the histograms of three columns of $\mathbf{\Theta}$ (which implies a mini-dictionary with three atoms) obtained by an example mixture with four sources equidistantly spaced in the front quadrant. We can clearly see that most values are clustered around specific directions; in this example around -11.25° and 11.25° which are two of the known mixing directions.

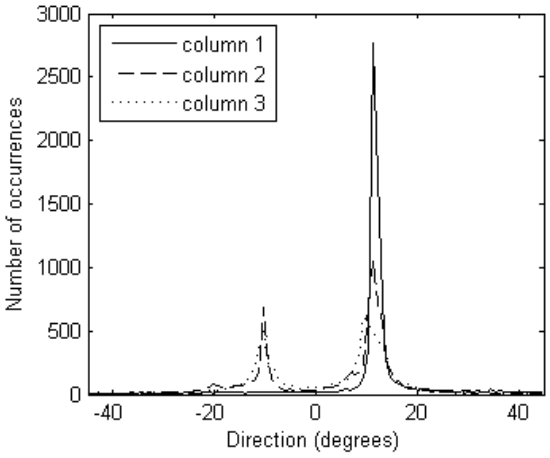


Figure 3: *Histograms of values in columns of* $\mathbf{\Theta}$

Having this information at hand we can proceed with the modification of step 6 of GMP. As already mentioned the original MP algorithm selects at each iteration, the atom that minimizes the energy of the residual, but this is not a strict requirement. What we are after are alternative selection criteria that take advantage of our known mixing matrix and the estimated atom directions. In this paper we test three different atom selection criteria.

In the first case we select the pair of expansion coefficients that are closest to a known direction and have the maximum absolute magnitude. The idea is that if a pair of coefficients is close to a known direction then there is a higher chance that the corresponding atom belongs to the source indicated by that direction. Also selecting an atom with the maximum magnitude (i.e. high energy) makes sure that the algorithm will converge fast. Let N be the maximum length of an atom (in samples), M be the number of atoms in the mini-dictionary and P the number of sources. Also let $\theta \in \mathbb{R}^P$ be a vector of P known directions. Then we calculate:

$$\mathbf{M} = \sum_{j=1}^{2} |\mathbf{C}_{n,m,j}| \tag{15}$$

$\forall n \in \{1..N\}\,, \forall m \in \{1..M\}$ and

$$\mathbf{d}_{n,m,p} = |\mathbf{\Theta}_{n,m} - \theta_p| \tag{16}$$

$\forall n \in \{1..N\}\,, \forall m \in \{1..M\},\ \forall p \in \{1..P\}$,
where $\mathbf{d} \in \mathbb{R}^{N \times M \times P}$ is a matrix holding the distances of each estimated direction in $\mathbf{\Theta}$ from the known directions θ. Then we calculate:

$$\mathbf{E} = \frac{\mathbf{M}}{\mathbf{d}} \tag{17}$$

where $\mathbf{E} \in \mathbb{R}^{N \times M \times P}$. The indices of the maximum value in $\mathbf{E}$ indicate the extraction location in the signal (in samples), the atom to choose from the dictionary (which implies the atom parameters such as type, scale, frequency, phase etc.) along with the

expansion coefficients (obtained from $\mathbf{C}$) and the source the atom belongs to. At this point we should mention that because of our sparsity assumption in section 1.4 each atom is allocated to only one source.

In the second case we favor atoms that appear to be coming from a dominant source. In order to find the dominant source we first find the maximum value of each histogram in $\mathbf{\Theta}$ and then subtract from the known directions θ. The minimum absolute value of that result indicates the source to allocate to. Then we use equations 15, 16 and 17 to obtain the atom to extract, the expansion coefficients and the location to extract from. Note that in this case $\mathbf{d}$ and $\mathbf{E}$ become $N \times M$ matrices since the source to allocate to has already been calculated, thus the third dimension reduces to 1.

In the final selection criterion we start by choosing the 'best' histogram for obtaining the expansion coefficients. The choice is based on the shape of the histograms obtained from $\mathbf{\Theta}$. In particular we favor histograms with values concentrated on one direction only. For example looking at figure 3 we see that the values of column 1 of $\mathbf{\Theta}$ are concentrated around direction 11.25° degrees whereas columns 2 and 3 produce peaks at two different directions. The idea is that if all or most of the estimated directions of a particular column of $\mathbf{\Theta}$ are clustered around one direction only then this is a strong indication that the corresponding atoms belong to the source indicated by that direction. Thus in this particular example the algorithm will select the expansion coefficients that correspond to column 1 of $\mathbf{\Theta}$ as candidates. We select the 'best' histogram h as follows:

$$h = \max\left(\frac{\sum_{n=1}^{N}\mathbf{M}_{n,m}}{\min\left(\frac{\sum_{n=1}^{N}|\mathbf{\Theta}_{n,m}-\theta_p|}{N}\right)}\right) \tag{18}$$

$\forall m \in \{1..M\}$, $\forall p \in \{1..P\}$. Having found the histogram to operate upon we use equations 15, 16, 17 with fixed $m = h$ and obtain $\mathbf{E}$ which is now a vector of length N. The index corresponding to the maximum value of $\mathbf{E}$ will give us the atom extraction location (in samples). Finally using the found location and h we can obtain the expansion coefficients from $\mathbf{C}$.

All three selection criteria presented here, provide us with the expansion coefficients and parameters of the atom to extract, the extraction location in the signal and finally the source that the atom belongs to. A new step is introduced where the selected atom is added to an approximated source. Note that the number of approximated sources will be the same as the number of mixing directions. Finally the algorithm proceeds with updating the residual and checking for exit conditions before continuing to the next iteration.

5. EXPERIMENTS

For the simulation a mixture comprising four sources was used. The sources were obtained from [26] and are anechoic recordings of a clarinet, a violin, a soprano and a viola. The sources are sampled at 44.1kHz and segments of 2^{17} samples (approx. 3 seconds) were used. The mixture was created using the mixing model in section 1.1 with a mixing matrix produced using equations 3 and 4. The sources were mixed so that they were equidistantly spaced in the front quadrant; a situation similar to a string quartet recording.

The same mixture was processed using three different source separation algorithms. All algorithms use the mixing matrix as prior information. We should also mention at this point that these algorithms have many parameters that can affect the outcome of the separation. In this experiment we tested all algorithms using various configurations and the best results are presented. The first algorithm used can be found in [5]. It uses the STFT as its front-end and performs source separation based on the directions estimated by the magnitude spectra of the mixture channels. It was found that a good setting for the particular mixture was an FFT size of 4096 with 75% overlap using a hanning window. The second algorithm we test against can be found in [16] and uses MP as its front-end. For this algorithm we used a Gabor dictionary (similar to that expressed by equation 7) comprising atoms with six scales, from $s = 512$ until 16384 samples with a 50% window-shift between atoms (see [23] for how to setup a dictionary in MPTK). The third algorithm is the one proposed in this paper. In order to be as fair as possible the proposed algorithm was set-up to use a Gabor dictionary comprising atoms up to six different scales. The algorithm also operated in the 'Short-Time Matching Pursuit' (STMP) mode were the signal is split into frames and each frame is processed separately in a similar fashion to the STFT. This is in contrast to MPTK where the signal is processed as a whole. Also for this example, GMP produced the 'best' results using the second atom selection criterion that was described in section 4

Because we are interested in high quality source separation for audio post-processing we used the PEASS toolkit [27], [28] for evaluating the performance of each algorithm. The toolkit produces the standard SDR,SIR,SAR and SNR measures but most importantly it calculates a set of perceptually motivated subjective measures which correlate better with human assessments. In particular it calculates the Overall Perceptual Score (OPS), Target related Perceptual Score (TPS), Interference related Perceptual Score (IPS) and Artifact related Perceptual Score (APS). Table 2 shows the SDR and SIR measures and table 3 the OPS, TPS and IPS scores produced by PEASS toolkit. The best scores are marked in bold.

Table 2: SDR and SIR performance measures (values in dB).

		SDR			SIR	
Src	STFT	MPD	GMP	STFT	MPD	GMP
1	5.83	**8.93**	7.16	**14.23**	10.93	12.16
2	2.75	2.98	**3.91**	**7.28**	5.23	6.09
3	5.77	10.24	**13.83**	**18.41**	18.13	17.21
4	4.97	9.96	**9.98**	**15.54**	12.96	14.51

By quick inspection of the tables there is no clear 'winner' algorithm. Having said that there are some interesting points we can talk about. First of all we can see that overall the algorithms that use MP as their front-end perform better. Also in the SIR case we see that the STFT algorithm produces better results by a small margin. This verifies to some extent the claim that algorithms which use adaptive decomposition schemes as their front-end tend to produce better results. We should also take into account that the mpddemix and GMP algorithms were used with their most basic settings, that is they use only one dictionary with limited number of atom variations. We expect the results to get better if we let the MP based algorithms run with multiple dictionaries comprising various atoms. These tests have yet to be performed. We should also mention that the STFT based algorithm was implemented to

Table 3: OPS, TPS and IPS performance scores (all scores out of 100)

		OPS			TPS			IPS	
Src	STFT	MPD	GMP	STFT	MPD	GMP	STFT	MPD	GMP
1	27.09	25.19	**32.57**	46.83	**53.41**	43.14	24.02	26.14	**59.76**
2	**25.81**	22.66	24	21.95	11.44	**30.82**	33.59	47.73	**66.02**
3	47.89	**70.36**	52.93	**80.21**	69.17	67.48	80.1	**80.94**	68.45
4	36.08	35.03	**36.13**	52.14	**57.83**	45.91	46.65	45.89	**51.68**

be used with this particular example. In particular the clustering of the coefficients that is performed in the STFT based algorithm is specifically designed to work with a mixing matrix that evenly places sources in the front quadrant. The MP based algorithms do not have this limitation and can operate using any mixing matrix.

Regarding the perceptually motivated evaluation scores we see again that the MP-based algorithms produce better results. Comparing MPD and GMP again does not show a clear 'winner' because sources obtain high scores in both algorithms. These are encouraging results for the proposed implementation since we test it against a well established source separation algorithm that uses MP. An interesting observation is that the IPS results show that GMP performs better on all sources but one. The interference related perceptual score is very important when we deal with high-quality source separation because it implies that the separated sources do not suffer from bleeding from other instruments. Informal listening tests have verified that to some extent. Audio examples can be found in [25].

6. CONCLUSION AND FUTURE WORK

In this paper we described a new method for decomposing multichannel audio signals using a variant of the basic Matching Pursuit algorithm. The new approach, which we term 'Guided Matching Pursuit', uses a pre-processing step to gather information about the signal and create a mini-dictionary comprising atoms that are expected to correlate well with the signal. We compared the new decomposition method with two accepted MP implementations and showed that we get better results regarding the signal to residual ratio and the residual energy decay rate. We further described how to apply the new decomposition method in a source separation problem by using three different atom selection criteria that take advantage of a known mixing matrix. We tested and compared the proposed algorithm against two available source separation algorithms that work using same principles and showed that we get similar results and in some cases better perceptual evaluation scores.

At the time, only one mixture has been tested. In particular this is an instrumental mixture comprising sources with quasi-periodic content which is expected to be decomposed well using a Fourier or Gabor dictionary. It will be of great interest to try the algorithm using mixtures that contain transients such as percussion content. To that extend we also want to try the algorithm using dictionaries comprising many atom types such as multi-scale Gabor atoms, windowed multi-scale Fourier atoms, damped sinusoids, chirplets, gamma-tones, gamma-chirps, and fm atoms, in which case we expect to see an increase in quality scores. Also at this point the algorithm takes a big amount of time to complete, which for our purposes at the moment does not pose a problem, but a faster implementation should be considered. This will be achieved by optimizing the code and possibly re-writing the algorithm using a faster language such as C. Finally, since our goal is audio post-production, a next step would be to try out the source separation algorithm in that context and perform subjective listening tests.

7. REFERENCES

[1] Avendano and Jot, "Frequency domain techniques for stereo to multichannel upmix," *Audio Engineering Society Conference: 22nd International Conference: Virtual, Synthetic, and Entertainment Audio*, Jun 2002.

[2] Avendano and Jot, "A frequency-domain approach to multichannel upmix," *J. Audio Eng. Soc.*, vol. 52, no. 7, pp. 740–749, 2004.

[3] Avendano C., "Frequency-domain source identification and manipulation in stereo mixes for enhancement, suppression and re-panning applications," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, Oct 2003, pp. 55–58.

[4] Wells J.J., "Modification of spatial information in coincident pair recordings," *J. Audio Eng. Soc.*, 2010.

[5] Jeremy J. Wells, "Directional segmentation of stereo audio via best basis search of complex wavelet packets," *J. Audio Eng. Soc.*, May 13-16 2011.

[6] Jeremy J. Wells, "A comparison of analysis and re-synthesis methods for directional segmentation of stereo audio," in *Proc. of the 14 th Int. Conference on Digital Audio Effects (DAFx-11), Paris, France*, Paris, France, Sept. 19-23 2011.

[7] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent Component Analysis and Applications*, chapter Sparse Component Analysis, pp. 367–412, Academic Press, 2010.

[8] F. Abrard and Y. Deville, "Blind separation of dependent sources using the "time-frequency ratio of mixtures" approach," in *Signal Processing and Its Applications, 2003. Proceedings. Seventh International Symposium on*, 2003, vol. 2, pp. 81–84.

[9] Yilmaz, "Blind separation using time-frequency masks," *IEEE Transactions on Signal Processing*, 2004.

[10] Arberet S. Gribonval R. Bimbot F., "A robust method to count and locate audio sources in a stereophonic linear instantaneous mixture," in *ICA*, 2006, pp. 536–543.

[11] Arberet S. Gribonval R. Bimbot F., "A robust method to count and locate audio sources in a stereophonic linear anechoic mixture," in *Proc. IEEE Intl. Conf. Acoust. Speech*

Signal Process (ICASSP'07), Honolulu, Hawai, Etats-Unis, 2007, pp. 745–748.

[12] Arberet S. Gribonval R. Bimbot F., "A robust method to count and locate audio sources in a multichannel underdetermined mixture," *Signal Processing, IEEE Transactions on*, vol. 58, no. 1, pp. 121–133, 2010.

[13] M. Zibulevsky B.A Pearlmutter P. Bofill and P Kisilev, *Independent Component Analysis: Principles and Practice*, chapter Blind source separation by sparse decomposition, S. J. Roberts and R.M. Everson eds., Cambridge, 2001.

[14] Mallat S.G. and Zhang Zhifeng, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397 –3415, 1993.

[15] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, 2002, vol. 3, pp. 3057–3060.

[16] Sylvain Lesage, Sacha Krstulovic, and Remi Gribonval, *Under-Determined Source Separation: Comparison of Two Approaches Based on Sparse Decompositions*, vol. 3889, pp. 633–640, Springer Berlin Heidelberg, 2006.

[17] P. Sugden and N. Canagarajah, "Underdetermined noisy blind separation using dual matching pursuits," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, 2004, vol. 5, pp. 557–560.

[18] Marchand S. et al, "Dream: A novel system for joint source separation and multitrack coding," *Audio Engineering Society Convention 133*, Oct 2012.

[19] Breebaart J. et al, "Spatial audio object coding (saoc) - the upcoming mpeg standard on parametric object based audio coding," *Audio Engineering Society Convention 124*, May 2008.

[20] Scott Shaobing Chen, David L. Donoho, Michael, and A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33â–61, 1998.

[21] Seema Jaggi, William C. Karl, Stephane Mallat, and Alan S. Willsky, "High resolution pursuit for feature extraction," *Applied and Computational Harmonic Analysis*, vol. 5, no. 4, pp. 428–449, 1998.

[22] Shie Qian and Dapang Chen, "Signal representation using adaptive normalized gaussian functions," *Signal Processing*, vol. 36, no. 1, pp. 1–11, 1994.

[23] Sacha Krstulovic and Rémi Gribonval, "MPTK: Matching Pursuit made tractable," in *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP'06)*, Toulouse, France, May 2006, vol. 3, pp. 496–499.

[24] "Matching Pursuit," `http://www.mathworks.co.uk/help/wavelet/ref/wmpalg.html`, 2014, Last accessed 18-March-2014.

[25] "GMPcoder," `http://d.zantalis.com/gmpcoder`, 2014, Last accessed 18-March-2014.

[26] T. Lokki et al., "Anechoic recordings of symphonic music," `http://auralization.tkk.fi`.

[27] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and objective quality assessment of audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 7, pp. 2046–2057, Sept 2011.

[28] Emmanuel Vincent, *Improved Perceptual Metrics for the Evaluation of Audio Source Separation*, vol. 7191 of *Lecture Notes in Computer Science*, pp. 430–437, Springer Berlin Heidelberg, 2012.

FINITE VOLUME PERSPECTIVES ON FINITE DIFFERENCE SCHEMES AND BOUNDARY FORUMULATIONS FOR WAVE SIMULATION

Brian Hamilton,[*]

Acoustics and Audio Group,
University of Edinburgh
brian.hamilton@ed.ac.uk

ABSTRACT

Time-domain finite difference (FD) and digital waveguide mesh (DWM) methods have seen extensive exploration as techniques for physical modelling sound synthesis and artificial reverberation. Various formulations of these methods have been unified under the FD framework, but many discrete boundary models important in room acoustics applications have not been. In this paper, the finite volume (FV) framework is used to unify various FD and DWM topologies, as well as associated boundary models. Additional geometric insights on existing stability conditions provide guidance into the FV meshing pre-processing step necessary for the acoustic modelling of irregular and realistic room geometries. DWM "1-D" boundary terminations are shown, through an equivalent FV formulation, to have a consistent multidimensional interpretation that is approximated to second-order accuracy, however the geometry and wall admittances being approximated may vary from what is desired. It is also shown that certain re-entrant corner configurations can lead to instabilities and an alternative stable update is provided for one problematic configuration.

1. INTRODUCTION

Finite difference (FD) methods applied to time-domain partial differential equations have a long history [1] and have, more recently, become popular techniques for wave simulation in musical acoustics and room acoustics modelling. FD methods have appeared in the acoustics literature in various forms, including the digital waveguide mesh (DWM) [2, 3], and the "finite difference time domain" (FDTD) [4, 5] and transmission line matrix (TLM) methods [6] adapted from electromagnetics [7, 8, 9]. The DWM and TLM methods were originally expressed in terms of scattering variables, but they are equivalent to FD methods when expressed solely in terms of nodal quantities [2, 8, 10].

Finite element (FE) methods comprise another major family of numerical methods that can be used for time-stepping wave simulations [11]. FE methods are based on unstructured grids of cells and interpolants defined over points within those cells. One advantage of FE methods over FD methods is that they are well-suited to irregular geometries (boundaries), whereas FD methods are suited to simple geometries, or "staircase" approximations to irregular boundaries. FE methods have been applied to musical acoustics problems [12], though generally not for sound synthesis.

Finite volume (FV) methods can be seen as a medium between these two methods [13], as they are based on grids of volumetric cells and differences between points within those cells. FV methods on Cartesian grids were introduced to room acoustics and shown to be equivalent to the Cartesian FDTD method [14, 4], but they have not seen the same popularity as the finite difference based methods. Recently, FV methods have been reintroduced for acoustical applications, but on unstructured grids and with rigorous energy-based stability analyses, allowing for the modelling of irregular geometries [15].

FV schemes can reduce to FD schemes on the interior domain when the tiling is regular, providing computationally efficient calculations appropriate for large domains such as in 3-D room acoustics modelling. This was shown for the Cartesian case and the 2-D hexagonal case [15] and later on the 3-D face-centered cubic (FCC) grid with rhombic dodecahedral cells [16]. Equivalences such as these remain to be determined for the rest of the DWM topologies. This is the first contribution of this study, which is presented in Section 4, after the introduction of the model equations and finite volume formulations in Sections 2 and 3. The second contribution of this study is a geometrical interpretation of passivity conditions at boundary cells (given certain constraints on the tiling), which provides insight towards meshing of irregular domains (Section 5). Various boundary models have been presented for FD/DWM methods [3, 17, 18], but their FV interpretations have yet to be determined. This is the third contribution of this study (Section 6). Conclusions and future work are given in Section 7.

2. MODEL EQUATIONS

2.1. Second-order wave equation

There have generally been two departure points for time-domain finite difference schemes in acoustics. The first is the second-order wave equation:

$$\partial_t^2 \Psi - c^2 \Delta \Psi = 0\,, \tag{1}$$

where $\Psi := \Psi(t, \mathbf{x})$ represents a velocity potential field, $t \in \mathbb{R}^+$ is time, $\mathbf{x}$ is a position vector within a d-dimensional closed volume $\mathcal{V} \subset \mathbb{R}^d$, Δ is the d-dimensional Laplacian operator, $\Delta := \sum_{i=1}^{d} \partial_{x_i}^2$, and c is the wave speed, assumed to be constant. The notation ∂_t denotes the partial derivative with respect to t, and similarly for spatial directions. The most common FD scheme for this equation was provided by Courant et al. [1], and the multidimensional extensions ($d \geq 3$) can be found in [19]. It was later applied to acoustic modelling for seismology [20], with Ψ as the variable of interest. Another wave equation can be found in the pressure field $p := p(t, \mathbf{x})$ through the use of the first of the two following relationships:

$$p = \rho \partial_t \Psi\,, \quad \mathbf{v} = -\nabla \Psi\,, \tag{2}$$

[*] This work was supported by the European Research Council, under grant StG-2011-279068-NESS, and by the Natural Sciences and Engineering Research Council of Canada.

where ρ represents the density of air and $\mathbf{v} := \mathbf{v}(t, \mathbf{x})$ is the particle velocity field. The pressure wave equation is then

$$\partial_t^2 p - c^2 \Delta p = 0\,. \tag{3}$$

Eqs. (1) and (3) are ultimately equivalent, but one must take care in choosing the appropriate source, boundary conditions, and output for (3) as it is one time-derivative higher than (1). FD methods were also applied to Eq. (3) for seismology [21], and later, DWM methods derived from networks of "acoustic tubes" were also expressed as FD schemes for this wave equation in 2-D [2] and in 3-D [3].

2.2. Conservation equations

The second departure point has been the following linear hyperbolic system, i.e. the conservation equations:

$$\frac{1}{\rho c^2}\partial_t p = -\nabla \cdot \mathbf{v} \qquad \text{(cons. of momentum)}\,, \tag{4a}$$

$$\rho \partial_t \mathbf{v} = -\nabla p \qquad \text{(cons. of mass)}\,, \tag{4b}$$

where $\nabla\cdot$ and ∇ are the d-dimensional divergence and gradient operators respectively. The FDTD[1] method, a popular technique for simulating Maxwell's equations on staggered grids, was adapted to the acoustics equations (4a) and (4b) for the purposes of vocal tract simulation [5] and room acoustics modelling [4]. These equations were also approached using TLM methods adapted from electromagnetics [6], and FV methods [14]. It is straightforward to check that these are different forms for the same underlying system, i.e. (4a) and (4b) give (3), and (1) is recovered using (2). Similarly, it has been shown that the various discretised forms (FD, DWM, TLM, FDTD, FV) are equivalent when implemented on Cartesian grids [24, 23, 2, 3, 10, 22, 15].

3. FINITE VOLUME APPROXIMATION

3.1. Cells and tiling

The following describes a general notation for a tiling of the volume $\mathcal{V}$ by cells in d-dimensions, which will be used to derive a finite volume approximation for the model equations. A tiling of $\mathcal{V}$ is made up of closed cells $\mathcal{C}_i$ indexed by $i \in I$, whose interiors are pairwise disjoint, i.e. $\bigcup_{i\in I} \mathcal{C}_i = \mathcal{V}$, and the boundary of each cell is denoted by $\partial\mathcal{C}_i$. A $(d-1)$-dimensional face (or side) of the cell $\mathcal{C}_i$ is denoted by $\mathcal{S}_{ij} := \mathcal{C}_i \cap \mathcal{C}_j = \mathcal{S}_{ji}$ and its $(d-1)$-dimensional volume is S_{ij}. Any cell $\mathcal{C}_j$ such that $\mathcal{S}_{ij} \neq \{\}$ is a neighbouring cell of $\mathcal{C}_i$. Let N_i be the index set of the neighbouring cells of $\mathcal{C}_i$ and let $K_i := |N_i|$, where $|N_i|$ denotes the cardinality of the set. A boundary face is denoted by $\mathcal{S}_{i(b)} := \mathcal{C}_i \cap \partial\mathcal{V}$ and its $(d-1)$-dimensional volume is denoted by $S_{i(b)}$. Finally, let $I_{(i)}$ and $I_{(b)}$ represent the index sets where $S_{i(b)} = 0$ and $S_{i(b)} > 0$ respectively. Note, in 1-D cell-faces are just points, so it suffices to set all measures $S_{ij} = 1$ when $\mathcal{S}_{ij} \neq \{\}$ and likewise, $S_{i(b)} = 1$ when $\mathcal{S}_{i(b)} \neq \{\}$ for the case $d = 1$. The tiling will also have an associated grid of points $\mathbb{G} := \{\mathbf{x}_i \in \mathcal{V}, i \in I\}$ such that $\mathbf{x}_i$ is not a point shared with other cells, i.e. $\mathbf{x}_i \in \mathcal{C}_i \setminus (\bigcup_{j\in N_i} \mathcal{S}_{ij})$.

[1] In this study, the acronym "FDTD" will refer only to staggered schemes for the system (4) which were adapted from the electromagnetics literature [7, 9]. Second-order wave equation schemes, which are much older [1], are more commonly (and efficiently [22]) employed will simply be called "FD methods" to be more consistent with the larger body of literature on numerical methods for such equations [1, 19, 23] and their applications in various fields [24, 20, 21]. The additional "time-domain" distinction is not necessary here, since this is implied by the model equations.

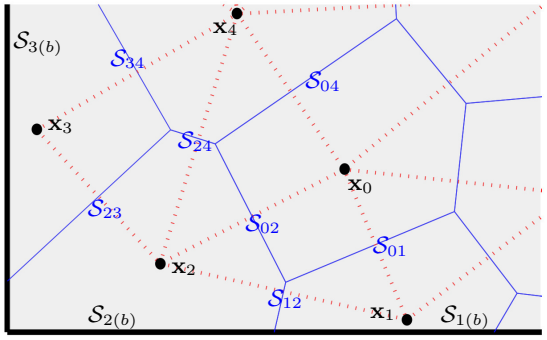


Figure 1: A section of a 2-D tiling, and the triangulation of its grid of points $\mathbb{G}$. The interior is shaded and part of the boundary $\partial\mathcal{V}$ is denoted by the thick line.

For the purposes of this paper, it will be assumed that $\mathcal{C}_i$ is the Voronoi cell of $\mathbf{x}_i$ for $i \in I_{(i)}$ and it will be further assumed that the line segment with endpoints $\mathbf{x}_i$ and $\mathbf{x}_j$ (denoted by $\overline{\mathbf{x}_i\mathbf{x}_j}$) for $j \in N_i$ is oriented normal to $\mathcal{S}_{ij}$ and passes through that side. Not all Voronoi tessellations have this "line of sight" property; those that do are the dual tilings of *Pitteway triangulations* (of $\mathbb{G}$) [25]. Such a tiling is illustrated in Fig. 1.

3.2. Integral form

A finite volume formulation can be derived starting from the acoustic velocity potential wave equation and integrating over the volumetric cell $\mathcal{C}_i$,

$$\int_{\mathcal{C}_i} \partial_t^2 \Psi \, dV = c^2 \int_{\mathcal{C}_i} \Delta\Psi \, dV\,. \tag{5}$$

Using $\Delta\Psi = \nabla \cdot \nabla\Psi$ and the divergence theorem gives

$$\int_{\mathcal{C}_i} \partial_t^2 \Psi \, dV = c^2 \int_{\partial\mathcal{C}_i} \mathbf{n} \cdot \nabla\Psi \, d\sigma\,, \tag{6}$$

where $\mathbf{n}$ denotes the outward normal vector at $\mathbf{x} \in \partial\mathcal{C}_i$. Then using $\partial\mathcal{C}_i = (\bigcup_j \mathcal{S}_{ij}) \cup \mathcal{S}_{i(b)}$, (6) can be written as

$$\int_{\mathcal{C}_i} \partial_t^2 \Psi \, d\mathbf{x} = c^2 \sum_{j\in N_i} \int_{\mathcal{S}_{ij}} \mathbf{n} \cdot \nabla\Psi \, d\sigma - c^2 \int_{\mathcal{S}_{i(b)}} \mathbf{n} \cdot \mathbf{v} \, d\sigma\,. \tag{7}$$

The last term isolates the velocity field pointing out of the boundary, which can be used to implement impedance boundary conditions.

3.3. Finite volume schemes

Let $\hat{\Psi} = \hat{\Psi}(t, \mathbf{x})$ represent an approximation to the acoustic velocity potential $\Psi(t, \mathbf{x})$, and the following abbreviated notation will be used: $\hat{\Psi}_i^{\pm} := \hat{\Psi}(t \pm k, \mathbf{x}_i)$ for some time-step k. The difference operators that will provide a discrete approximation to (7) are

Table 1: Isohedral cells and FD/DWM/FDTD equivalents (found in the literature) for pointwise FV approximations

cell ($\mathcal{C}_i$)	dim. (d)	K_i	$\frac{2d}{K_i}$	$\delta_\Delta = \Delta + O(h^q)$	grid	FD/DWM scheme	staggered (FDTD) scheme for (4)
line segment	1	2	1	$q=2$	integer lattice	standard FD scheme [1]	[24, 23]
regular triangle	2	3	4/3	$q=1$	honeycomb grid	"hexagonal DWM" [26]	-
square	2	4	1	$q=2$	square grid	standard FD scheme [1]	Yee's 2-D scheme (FDTD) [7]*
hexagon	2	6	2/3	$q=2$	hexagonal grid	hexagonal FD scheme [27]	hexagonal FDTD [28]*
cube	3	6	1	$q=2$	cubic grid	standard FD scheme [19]	Cartesian FDTD [4, 5]
regular tetrahedron†	3	4	3/2	$q=1$	diamond grid	"tetrahedral DWM" [26]	-
octahedron†	3	8	3/4	$q=2$	BCC grid	"octahedral FD" scheme [10]	-
rhombic dodecahedron	3	12	1/2	$q=2$	FCC grid	"dodecahedral DWM" [29]	-

†does not tile space *see Footnote 2

defined as follows

$$\delta_{t\pm}\hat{\Psi}_i := \pm\frac{1}{k}(\hat{\Psi}_i^{\pm} - \hat{\Psi}_i)\,, \tag{8a}$$

$$\delta_{tt} := \delta_{t+}\delta_{t-} = \delta_{t-}\delta_{t+}\,, \tag{8b}$$

$$\delta_{ij}\hat{\Psi}_i := \frac{1}{h_{ij}}(\hat{\Psi}_j - \hat{\Psi}_i) = -\delta_{ji}\hat{\Psi}_i\,, \tag{8c}$$

where $h_{ij} = \|\mathbf{x}_{ij}\|$ and $\mathbf{x}_{ij} := \mathbf{x}_j - \mathbf{x}_i$.

Eq. (7) can be approximated by replacing Ψ with $\hat{\Psi}$ and then applying difference operators in the place of continuous derivatives. It must also be assumed that the approximation $\hat{\Psi}$ is constant (averaged) over cells and boundary sides [15]. This results in the following

$$\frac{V_i}{c^2}\delta_{tt}\hat{\Psi}_i = \sum_{j\in N_i} S_{ij}\delta_{ij}\hat{\Psi}_i - S_{i(b)}\hat{v}_{i(b)}\,, \tag{9}$$

where $\hat{v}_{i(b)}$ represents the boundary term averaged over $\mathcal{S}_{i(b)}$, to be specified by the boundary conditions. Due to the imposed restrictions on the Voronoi tessellation, $\delta_{ij}\hat{\Psi}_i$ can be seen as a centered difference about some point on the face $\mathcal{S}_{ij}$, and the approximation will be second-order accurate.

A finite volume approximation of the conservation equations can also be recovered by defining approximations to the pressure and velocity field, denoted by $\hat{p}$ and $\hat{\mathbf{v}}$ respectively, on staggered grids in space and time. Consider the following spatial and temporal half-step shift operators, applied to some function $u(t, \mathbf{x})$:

$$e_{t\pm}u_i := u(t \pm k/2, \mathbf{x}_i)\,, \tag{10a}$$

$$e_{ij\pm}u_i := u(t, \mathbf{x}_i \pm \mathbf{x}_{ij}/2) = e_{ji\mp}u_i\,. \tag{10b}$$

The pressure and velocity approximations are then staggered as

$$\hat{p}_i := e_{t-}\hat{p}(t, \mathbf{x}_i)\,, \tag{11a}$$

$$\hat{v}_{ij} := \mathbf{x}_{ij}\cdot e_{ij+}\hat{\mathbf{v}}(t, \mathbf{x}_i)\,, \tag{11b}$$

and these are related to $\hat{\Psi}_i$ using

$$\hat{p}_i = \rho\delta_{t-}\hat{\Psi}_i\,, \tag{12a}$$

$$\hat{v}_{ij} = -\delta_{ij}\hat{\Psi}_i\,, \tag{12b}$$

resulting in the equivalent (to (9)) staggered scheme:

$$\frac{V_i}{\rho c^2}\delta_{t+}\hat{p}_i = -\sum_{j\in N_i} S_{ij}\hat{v}_{ij} - S_{i(b)}\hat{v}_{i(b)}\,, \tag{13a}$$

$$\rho\delta_{t-}\hat{v}_{ij} = -\delta_{ij}\hat{p}_i\,. \tag{13b}$$

When $\mathcal{C}_i$ is a hexahedron, that is, the Voronoi cell of a "quasi-Cartesian" grid ($d = 3$), (13) defines the FV scheme proposed in [14]. The FV framework presented in [15] generalises these FV schemes to unstructured grids of polytopes ($d \geq 1$).

Finally, by applying $\rho\delta_{t-}$ to both sides of (13a) and substituting in (13b), or simply by applying (12a) to (9), a wave equation scheme for the pressure can be recovered, with an isolated boundary term:

$$\frac{V_i}{c^2}\delta_{tt}\hat{p}_i = \sum_{j\in N_i} S_{ij}\delta_{ij}\hat{p}_i - \rho S_{i(b)}\delta_{t-}\hat{v}_{i(b)}\,. \tag{14}$$

4. EQUIVALENCES WITH FD/DWM/FDTD SCHEMES

4.1. Pyramidal decomposition

In this section, only the interior cells will be considered ($i \in I_{(i)}$), so $\mathcal{S}_{i(b)} = \{\}$, or $S_{i(b)} = 0$. Taking into account the previously imposed restrictions ("line of sight"), an interior cell can then be divided into K_i d-dimensional pyramids $\mathcal{P}_{ij}$ with bases $\mathcal{S}_{ij}$, heights $h_{ij}/2$, and a shared apex $\mathbf{x}_i$. As is commonly known, the area of a 2-D pyramid (a triangle) is the area of the base times the height divided by two. This extends to d-dimensions [30], giving

$$V_i = \sum_{j\in N_i} V_{ij}\,, \quad V_{ij} := S_{ij}h_{ij}/(2d)\,. \tag{15}$$

A cell $\mathcal{C}_i$ will be called "isohedral" (face-transitive) when the parameters S_{ij} and h_{ij} are constants ($S_{ij} = S_i$, $h_{ij} = h_i$ for $j \in N_i$). The volume of an isohedral cell is then simply

$$V_i = K_i S_i h_i/(2d)\,. \tag{16}$$

This pyramidal decomposition is illustrated in Fig. 2 for square and hexagonal cells ($d = 2$), and various isohedral cells are listed in Table 1 for $d \in \{1, 2, 3\}$.

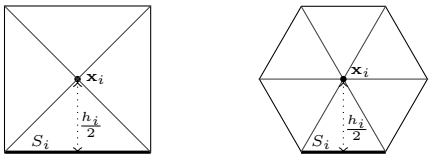


Figure 2: Square and hexagonal cells divided into pyramids. S_i denoted by thick line, $h_i/2$ denoted by dotted line.

4.2. Discrete Laplacians and wave equation FD/DWM schemes

Various FV schemes can be simplified using the pyramidal decomposition. The wave equation scheme in (14), for the isohedral cells

listed in Table 1, can be written as

$$\delta_{tt}\hat{p}_i = c^2\delta_\Delta\hat{p}_i\,, \quad \delta_\Delta\hat{p}_i := \frac{2d}{K_i h_i^2}\sum_{j\in N_i}(\hat{p}_j - \hat{p}_i) = \Delta\hat{p}_i + O(h^q)\,. \tag{17}$$

The spatial operator δ_Δ represents a discrete Laplacian with first ($q = 1$) or second-order ($q = 2$) accuracy, and (17) represents one of the various FD/DWM schemes that have appeared in the literature, as summarised in Table 1. Note, the regular tetrahedron and the octahedron do not tile space so they do not permit a full finite-volume approximation over $\mathcal{V}$. Nevertheless, the FD schemes derived from their single cell approximations are equivalent to the pointwise approximations from their associated FD/DWM schemes.

4.3. Discrete divergence and FDTD staggered schemes

Staggered (FDTD) schemes that employ isohedral cells can also be recovered from FV schemes. Using the pyramidal decomposition, (13a) becomes

$$\frac{1}{\rho c^2}\delta_{t+}\hat{p}_i = -\delta_{\nabla\cdot}\hat{\mathbf{v}}_i\,, \quad \delta_{\nabla\cdot}\hat{\mathbf{v}}_i := \frac{2d}{K_i h_i}\sum_{j\in N_i}\hat{v}_{ij} \tag{18}$$

The spatial operator $\delta_{\nabla\cdot}$ is essentially a discrete divergence operator, but this is more clear in the following special case. When the cell $\mathcal{C}_i$ is made up of pairs of parallel faces (K_i is even), the set N_i can be divided into two non-overlapping complementary sets $N_{i,1}$ and $N_{i,2}$ each with $K_i/2$ elements such that for every $j \in N_{i,1}$ there is a complementary index $j^* \in N_{i,2}$ where $\mathbf{x}_{ij} = -\mathbf{x}_{ij^*}$. Making use of the skew-symmetry $\hat{v}_{ij} = -\hat{v}_{ji}$ results in

$$\delta_{\nabla\cdot}\hat{\mathbf{v}}_i = \frac{2d}{K_i}\sum_{j\in N_{i,1}}\frac{1}{h_i}(e_{ij+} - e_{ij-})(\mathbf{x}_{ij}\cdot\hat{\mathbf{v}}_i) \approx \nabla\cdot\hat{\mathbf{v}}_i\,, \tag{19}$$

which is, more clearly, a sum of centered spatial differences in the directions of $\mathbf{x}_{ij}$ for $j \in N_{i,1}$. This defines the discrete divergence used in the Cartesian case ($\mathbf{x}_{ij}$ for $j \in N_{i,1}$ become the standard unit vectors in $\mathbb{R}^d$), leading to the staggered schemes adapted from Yee's scheme for Maxwell's equations [5, 4]. Furthermore, (18) leads to a hexagonal staggered scheme for acoustics similar in principle to one derived for Maxwell's equations [28].[2] In 3-D this leads to staggered schemes on the face-centered cubic (FCC) and body-centered cubic (BCC) grids. These equivalences are also summarised in Table 1. Clearly, these staggered schemes are all equivalent to their second-order wave equation counterparts, which generalises the link that has been established in the Cartesian case [22]. The staggered FDTD formulation is known to be less efficient than the wave equation formulation in the Cartesian case [22], and this can be shown for the non-Cartesian FDTD extensions for acoustics (left out for brevity). Wave equation schemes are also more efficient than their DWM forms in $d > 1$ [10].

5. STABILITY/PASSIVITY CONDITIONS

In this section, additional geometric insights will be provided on the stability conditions derived in [15], but the full energy analysis provided in [15] will be omitted for brevity. First, consider the lossless case, where the boundary velocity is set to zero,

[2] Maxwell's equations in polarised form (2-D) can be adapted to the acoustics equations (4) with a simple change of variables [31]. It is straightforward to show that this also applies to the 2-D FDTD schemes.

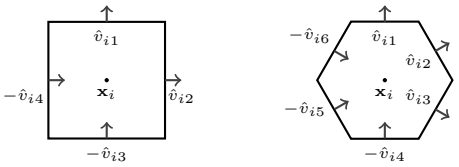


Figure 3: Square and hexagonal cells for staggered grid (FDTD) schemes

i.e. $\hat{v}_{i(b)} = 0$, $i \in I_{(b)}$. In the lossless case it can be shown that the total numerical energy remains bounded (stability) when the following condition is satisfied for each cell ($i \in I$) [15]:

$$k^2 \le \frac{2V_i}{c^2\sum_{j\in N_i} S_{ij}/h_{ij}}\,. \tag{20}$$

This appears to be the condition derived in [4] (considering only quasi-Cartesian grids) but, in fact, it is different. The condition derived in [4] does not distinguish between interior sides $\mathcal{S}_{ij}$ and boundary sides $\mathcal{S}_{i(b)}$, whereas the boundary sides are not counted in the denominator of (20). As such, condition (20) becomes more relaxed at the boundaries for Cartesian cells. The importance of this detail will be pointed out in Section 6.2.

5.1. A sufficient condition for interior isohedral cells

Further insight on condition (20) can be obtained when it is assumed that a "locally irregular" tiling is used, which means that the cells are isohedral and congruent when $i \in I_{(i)}$, but they may vary for $i \in I_{(b)}$. Thus, $K_i = K$, $S_{ij} = S$, $h_{ij} = h$, and $V_i = V$ for $i \in I_{(i)}$. For an interior cell ($i \in I_{(i)}$) the stability condition then becomes

$$k^2 \le \frac{2V}{c^2 K(S/h)}\,. \tag{21}$$

Using $V = KSh/(2d)$, the above reduces to

$$k^2 \le h^2/(c^2 d)\,, \tag{22}$$

or more simply, $\lambda \le \sqrt{1/d}$ where $\lambda := ck/h$ is the Courant number. This can be recognised as the condition obtained from von Neumann analysis for many schemes, including the Cartesian one [19]. In certain cases the von Neumann bound is more relaxed, such as in the hexagonal [10] and FCC schemes [16], as well as many parameterised schemes [10]. It is perhaps more appropriate to call (22) the "passivity condition" [10], since it implies that a DWM/TLM network implementation made up of concretely passive elements is possible, from which stability follows [32].

5.2. A sufficient condition for boundary cells

Now condition (20) will be simplified for boundary cells ($i \in I_{(b)}$). It will be assumed that $h_{ij} = h$, which implies that $\mathcal{C}_i$ has been cut or enlarged, but the point $\mathbf{x}_i$ has not moved from its regular position (for $i \in I_{(b)}$). It is straightforward to prove that $\lambda = \sqrt{1/d}$ is sufficient for stability at boundary cells that are congruent to interior cells (e.g., "staircase boundaries"); the denominator in (20) decreases but the total volume V_i does not, so the local condition at the boundary cell is more relaxed than $\lambda \le \sqrt{1/d}$.

Next, consider the case where the boundary cells are not congruent to the interior cells, which amounts to special cases of "fitted boundaries". Removing the volumes of the interior pyramids $\mathcal{P}_{ij}$

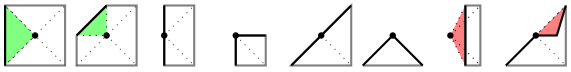


Figure 4: Some boundary cell types possible for a locally irregular square tiling. Green denotes positive V_i^*, red denotes negative V_i^*. Black line denotes $\mathcal{S}_{i(b)}$. Dotted lines denote pyramid divisions. (a)-(f) are stable with $\lambda = \sqrt{1/2}$, (g) and (h) are not.

from the volume of the cell $\mathcal{C}_i$ leaves a quantity denoted by V_i^*,

$$V_i^* := V_i - \sum_{j \in N_i} S_{ij} h/(2d)\,. \tag{23}$$

Then starting from (20), a bound on k^2 is

$$k^2 \leq \frac{2V_i}{c^2 \sum_{j\in N_i} S_{ij}/h} \leq \frac{\sum_{j\in N_i} S_{ij} h + V_i^*}{c^2 d \sum_{j\in N_i} S_{ij}/h} \tag{24a}$$

$$\leq h^2/(c^2 d) \quad \text{when } V_i^* \geq 0\,. \tag{24b}$$

Thus, $\lambda = \sqrt{1/d}$ is sufficient as long as $V_i^* \geq 0$. This condition is illustrated in Fig. 4 with some example 2-D cells in a locally irregular square tiling. Figs. 4g and 4h illustrate the types of cells that should be avoided since they will enforce more strict stability conditions than what is obtained from interior cells. This can be detrimental in audio applications as it reduces the overall temporal bandwidth of the output [33] and reduces the efficiency in minimising dispersion error [34].

It is worth pointing out a link with "conformal methods" for irregular boundary modelling in acoustical FDTD simulations [35, 36, 37]. These techniques are, ultimately, staggered schemes on locally irregular Cartesian grids, and more specifically, the "fitted boundaries" considered here. It was empirically found that a certain minimum volume had to be kept for locally conforming boundary cells, otherwise instabilities would be experienced [35, 36, 37], but a stability condition was not obtained. The condition that needs to be satisfied in such conformal methods is indeed the geometric one described here ($V_i^* \geq 0$), or more generally, condition (20).

6. BOUNDARY MODELS

Some commonly used discrete boundary models in FD/DWM methods can be shown to have equivalent FV formulations. To be considered are the simplest frequency independent lossy boundary conditions:

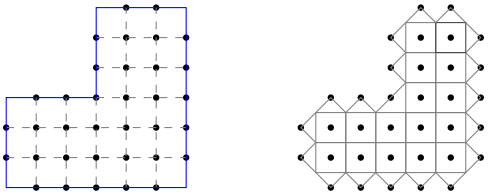
$$\mathbf{n} \cdot \mathbf{v} = (\gamma/\rho c) p\,, \quad \mathbf{x} \in \partial\mathcal{V}\,. \tag{25}$$

where $\gamma \geq 0$ represents the specific acoustic admittance. Applying one temporal derivative and employing (4b) gives the pressure boundary condition

$$-\,\mathbf{n} \cdot \nabla p = (\gamma/c)\partial_t p\,. \tag{26}$$

Frequency-independent lossy boundaries are achieved in the FV framework by discretising (25), while (26) has been the preferred condition in FD-based studies [38, 18, 33]. It will be seen that the FV and FD discretisations are essentially the same. To this end, the following average and difference operators will be necessary:

$$\mu_{t+}\hat{p}_i = \frac{1}{2}(\hat{p}_i^+ + \hat{p}_i)\,, \qquad \delta_{t\cdot}\hat{p}_i = \frac{1}{2k}(\hat{p}_i^+ - \hat{p}_i^-)\,. \tag{27}$$



(a) DWM modelling (b) FV equivalent

Figure 5: Desired L-shaped domain in blue, modelled using 1-D DWM boundaries (left) and equivalent FV interpretation (right). Dashed lines indicate connectivity, or DWM delay lines. Notice that adjacent boundary nodes are not interconnected, in both cases.

In the FV formulation from [15], (25) is discretised on $\mathcal{S}_{i(b)}$ with

$$\hat{v}_{i(b)} = (\gamma/\rho c)\mu_{t+}\hat{p}_i\,. \tag{28}$$

Applying $\rho\delta_{t-}$ and using the identity $\delta_{t\cdot} = \mu_{t+}\delta_{t-}$ gives

$$\rho\delta_{t-}\hat{v}_{i(b)} = (\gamma/c)\delta_{t\cdot}\hat{p}_i\,, \tag{29}$$

and now defining a fictitious "ghost point" $\mathbf{x}_g := \mathbf{x}_i + \mathbf{n}/2$ outside of the domain (as employed in the FD framework, but unnecessary in the FV framework), one could write (29) as

$$-\,\delta_{ig}\hat{p}_i = (\gamma/c)\delta_{t\cdot}\hat{p}_i\,, \tag{30}$$

which is a (centered) FD discretisation of (26). This is just a starting point for equivalences between FV boundaries and FD boundaries. Still to consider are specific boundary geometries and boundary cell types.

It can be shown that as long as the interior energy remains bounded, the only extra condition for stability is $\gamma \geq 0$ (passivity) [15]. In other words, if the lossless case is stable then the lossy boundaries will remain stable. Thus, when $\gamma \geq 0$, the stability of the scheme is ensured with $\lambda \leq \sqrt{1/d}$ as long as $V_i^* \geq 0$ for $i \in I_{(b)}$.

6.1. DWM "1-D boundaries"

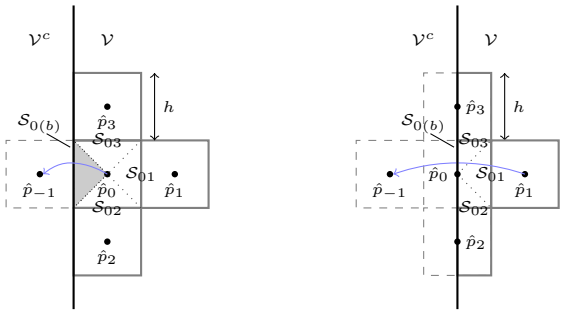
Using the DWM paradigm, boundaries in $d > 1$ are set via 1-D connections from boundary nodes to interior nodes. This is somewhat of an ambiguous way to set the boundaries, but it has the advantage of ensuring stability by virtue of a concretely passive network. For convenience, the DWM boundary node update, connected to K_i interior nodes, will be expressed in the FD equivalent formulation (the so-called "Kirchoff DWM"). Adapted from [17], the update is

$$(\gamma' + K_i)\hat{p}_i^+ = 2\sum_{j\in N_i} \hat{p}_j + (\gamma' - K_i)\hat{p}_i^-\,, \quad i \in I_{(b)}\,, \tag{31}$$

where γ' is meant to represent the desired γ in (26), but its precise value is to be determined. Dividing (31) through by K_i gives

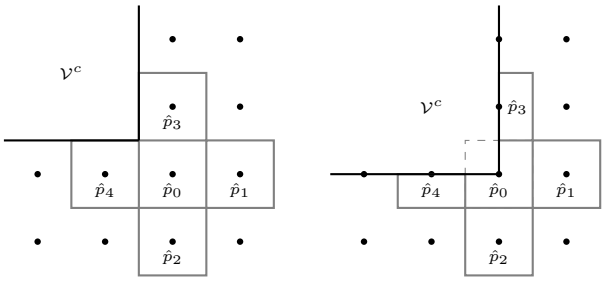
$$\left((\gamma'/K_i) + 1\right)\hat{p}_i^+ = (2/K_i)\sum_{j\in N_i} \hat{p}_j + \left((\gamma'/K_i) - 1\right)\hat{p}_i^-\,. \tag{32}$$

The Courant number is set to $\lambda = \sqrt{1/d}$ for a DWM (by construction). In order to obtain the familiar DWM cancellation of the $\hat{p}_i$ term that would have appeared in the summation (see (17)) with the term $2\hat{p}_i$ left over from the expansion of $\delta_{tt}\hat{p}_i$, the following



(a) Backwards difference (b) Centered difference

Figure 6: Backwards and centered spatial differences, in terms of FV cells. Dotted lines show pyramidal decomposition of $\mathcal{C}_0$. Volume pertaining to V_0^* is shaded. $\mathcal{V}^c$ denotes the exterior.



(a) "Non-centered" (b) "Centered"

Figure 7: Re-entrant corner configurations in 2-D with Cartesian grid

condition would be required from an equivalent FV formulation:

$$\lambda^2 K_i \frac{Sh}{V_i} = 2 \qquad \Rightarrow \qquad V_i = K_i \frac{Sh}{2d} \tag{33}$$

Thus, an equivalent, and multidimensionally consistent to second-order accuracy, FV interpretation of the DWM boundary cell is the union of K_i pyramids $\mathcal{P}_{ij}$ with a shared apex $\mathbf{x}_i$, bases $\mathcal{S}_{ij}$ (with $S_{ij} = S$) and heights $h/2$. This is illustrated in Fig. 5 for the Cartesian DWM on an L-shaped geometry. Note that the pyramidal boundary cells result in a crude approximation to the desired domain. Also note, the so-called "1-D ambiguity" problem [18] was avoided here, which is to interpret the Courant number as being set to unity locally at the boundary. It was suggested that this may lead to stability issues [38], but the DWM boundary model remains passive since it will always be the case that $V_i^* = 0$.

Finally, the precise value of γ' is found to be

$$\gamma' = \gamma \sqrt{d} (S_{i(b)}/S) \,. \tag{34}$$

Thus, γ' is not necessarily the desired acoustic admittance γ. It is consistent with γ in at least two special cases: $d = 1$ (as expected) and when $\gamma = 0$. For the boundary cells depicted in Fig. 5 one has $\gamma' = 2\gamma$. It should be pointed out that, while only the 2-D Cartesian case was illustrated, this FV interpretation extends to all other multidimensional DWM topologies, as long as the associated isohedral cell tiles d-dimensional space, which excludes the tetrahedral DWM and the octahedral DWM.

6.2. Centered and non-centered boundaries

Here, the basic 2-D FD scheme for the wave equation, also known as "standard leapfrog" (SLF), will be used to give an interpretation of boundaries commonly implemented for a half-plane. Consider the half-plane terminations featured in Fig. 6. Based on these two configurations, the spatial derivative in (26) can be approximated using one of the following spatial differences:

$$\frac{1}{h}(\hat{p}_0 - \hat{p}_{-1}) \approx -\mathbf{n}\cdot\nabla\hat{p}_0 \,, \qquad \frac{1}{2h}(\hat{p}_1 - \hat{p}_{-1}) \approx -\mathbf{n}\cdot\nabla\hat{p}_0 \,. \tag{35}$$

The former is a backwards "non-centered" difference and the latter is a centered difference. The centered difference has been preferred in the literature because it is, at first glance, second-order accurate [18]. However, the first operator is also centered if one redefines the boundary to lie between $\mathbf{x}_0$ and $\mathbf{x}_{-1}$, as in Fig. 6a. The temporal derivative (26) is usually approximated by $\delta_{t\cdot}$ to give an overall second-order accuracy. From the FD perspective, the update at the boundary node $\hat{p}_0$ is then:

$$\hat{p}_0^+ = \lambda^2(\hat{p}_{-1} + \hat{p}_1 + \hat{p}_2 + \hat{p}_3 - 4\hat{p}_0) + 2\hat{p}_0 - \hat{p}_0^- \,, \tag{36}$$

where $\hat{p}_{-1}$ represents the "ghost node" to be eliminated using the discretised boundary conditions. After eliminating the ghost node, the boundary update using the backwards spatial difference is

$$\hat{p}_0^+ = \frac{1}{1 + \frac{\gamma\lambda}{2}} \left(\lambda^2(\hat{p}_1 + \hat{p}_2 + \hat{p}_3 - 3\hat{p}_0) + 2\hat{p}_0 + \left(\frac{\gamma\lambda}{2} - 1 \right) \hat{p}_0^- \right. \tag{37}$$

and the boundary update using the centered spatial difference is

$$\hat{p}_0^+ = \frac{1}{1 + \gamma\lambda} \left(\lambda^2(2\hat{p}_1 + \hat{p}_2 + \hat{p}_3 - 4\hat{p}_0) + 2\hat{p}_0 + (\gamma\lambda - 1)\, \hat{p}_0^- \right) \tag{38}$$

On the other hand, both cases are summarised by the FV update:

$$\hat{p}_0^+ = \frac{1}{1 + \gamma\sigma} \left(\frac{c^2 k^2}{V_0 h} \sum_{j=1}^{3} S_{0j}(\hat{p}_j - \hat{p}_0) + 2\hat{p}_0 + (\gamma\sigma - 1)\, \hat{p}_0^- \right) , \tag{39}$$

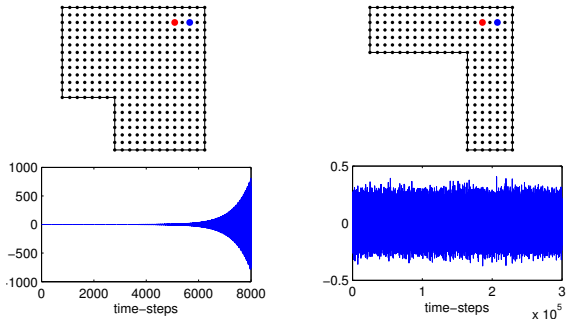
where $\sigma := ckS_{0(b)}/(2V_0)$. For the full cell, one has $S_{0(b)} = S_{0j} = h$ for $j \in \{1, 2, 3\}$ and $V_0 = h^2$. Thus $\sigma = \lambda/2$ and the "backwards" update (37) is obtained ($c^2k^2S_{0j}/(V_0 h) = \lambda^2$). For the half cell, one has $S_{0(b)} = S_{01} = h$ and $S_{02} = S_{03} = h/2$, meanwhile $V_0 = h^2/2$, so $\sigma = \lambda$ and the "centered update" (38) is obtained.

In terms of stability, the full-cell has $V_0^* = h^2/4$ remaining after subtracting interior pyramids (seen in Fig. 6a), and the half-cell has $V_0^* = 0$. Thus, both conditions remain stable with $\lambda = \sqrt{1/2}$, which is consistent with the literature [18, 33]. On the other hand, using the stability condition derived in [14] one would obtain a restriction of $\lambda \leq \sqrt{1/3}$ at the half-cell, which is too strict.

It is straightforward to extend this type of analysis to the corner case in 2-D (quarter-cell, Fig. 4d) and further, to wall, edge, corner FD conditions with cubic cells in 3-D [18]. These turn out to be cubic cells that are cut into one-half, one-quarter, and one-eighth respectively.

6.3. Re-entrant corners

Re-entrant corners are simple examples of irregular geometries to be modelled. An example re-entrant corner is displayed in Fig. 7, in two different configurations with respect to a square grid. In



(a) Domain A (b) Domain B

Figure 8: Top: two test domains. The scheme is initialised by setting $\hat{p}$ to zero at $t = 0$, and $\hat{p}$ is set to -1 at the point marked by blue dot and 1 at the point marked by the red dot at $t = k$. Bottom: outputs from simulations, read at a point on the interior of the domain.

the FD paradigm, one proceeds with a specialised boundary update only when there is a ghost node to be eliminated. In this case, there is no ghost node at the interior corner node, so one might proceed at $\mathbf{x}_0$ with the usual interior update

$$\hat{p}_0^+ = \lambda^2(\hat{p}_1 + \hat{p}_2 + \hat{p}_3 + \hat{p}_4 - 4\hat{p}_0) + 2\hat{p}_0 - \hat{p}_0^- \,. \qquad (40)$$

The first configuration, seen Fig.7a, is the implied geometry when "non-centered" boundary updates are applied at the points adjacent to the corner node ($\mathbf{x}_3$ and $\mathbf{x}_4$). There is no problem with this case as the re-entrant corner cell is a regular interior cell.

The second configuration, seen Fig.7b, requires more consideration. It has been shown that when centered boundary updates are applied at points $\mathbf{x}_3$ and $\mathbf{x}_4$ they correspond to half-cells. Simply applying (40) to the interior corner node [18] would imply that $\mathcal{C}_0$ is no different from an interior cell, but this would not agree with the half-cell neighbours, since the side lengths do not match. A more coherent update at $\mathbf{x}_0$ for this configuration would take into account the volume and sides of this "three-quarter cell". It would be difficult to arrive at such an update using only the FD framework since there are no accessible side and volume parameters to set. On the other hand, the following update, which is consistent with the 90-degree interior corner, can be obtained from a FV perspective:

$$\hat{p}_0^+ = \left(\frac{2\lambda^2}{3} (2\hat{p}_1 + 2\hat{p}_2 + \hat{p}_3 + \hat{p}_4 - 6\hat{p}_0) + 2\hat{p}_0 + \sigma_2 \hat{p}_0^- \right) / \sigma_1 \,, \qquad (41)$$

where $\sigma_1 = 1 + \frac{2\gamma\lambda}{3}$ and $\sigma_2 = \frac{2\gamma\lambda}{3} - 1$. This update ensures stability because $V_0^* = 0$ for the three-quarter cell.

It remains to be seen if the usual update, (40), applied at the interior corner node in the "centered" configuration leads to instabilities. To this end, consider the following experiments on the two L-shaped geometries depicted in Fig. 8. The boundaries are made lossless ($\gamma = 0$) and the usual interior update (40) is applied to the interior corner node. The Courant number is set to the usual $\lambda = \sqrt{1/2}$. The input and output locations are marked in the figures, and the scheme is excited by a non-zero initial condition. The two outputs are shown in Figs. 8a and 8b. In the first case the scheme is clearly unstable as it exhibits exponential growth. However, the same growth is not seen in the second case, even after 3×10^5 time-steps. Thus, employing the usual update at a re-entrant corner adjacent to "centered" boundaries does not always

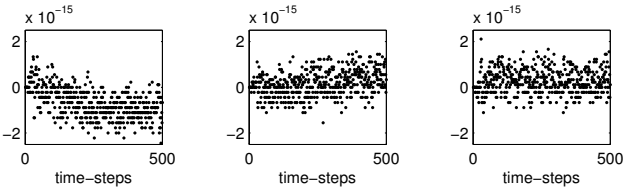


Figure 9: Energy variation for re-entrant corner tests on Domain A using three-quarter cell (left), full cell (middle), and DWM boundaries (right).

lead to instabilities and appears to depend on the overall geometry of the domain. Nevertheless, it would be wise to employ (41) or a full-cell variant since stability will be ensured. The problematic domain (Domain A) was also simulated with the three-quarter cell update (41) and, as expected, this simulation was found to be stable, and more precisely, energy-conserving to machine accuracy, as shown in Fig. 9. The energy was calculated using the expression given in [15] (left out for brevity). Employing the full-cell configuration and the 1-D DWM boundaries on the problematic domain also ensures numerical stability and energy-conservation to machine accuracy. The code used for these simulations will be available at: `http://www2.ph.ed.ac.uk/~s1164563/dafx14`.

7. CONCLUSIONS AND FUTURE WORK

In this study, finite volume equivalences were established for certain FD and DWM schemes for the wave equation and staggered (FDTD) schemes for conservation equations. It was shown that the underlying connection between pointwise FD methods and finite volume methods is a pyramidal decomposition, given certain constraints on the volumetric cells and their neighbours. Additional geometrical interpretations were provided for stability conditions. The "centered" and "non-centered" half-plane boundary updates were shown to be special cases of a FV update. The "1-D" DWM boundary model was shown to have a consistent interpretation in $d > 1$, but that there may be discrepancies with the desired acoustic admittance and domain geometry. Commonly used re-entrant corner updates combined with centered edge conditions in the 2-D Cartesian scheme were shown to have geometrical inconsistencies, leading to unpredictable instabilities. A stable and energy-conserving re-entrant corner update was proposed for the 2-D Cartesian case.

It is straightforward to extend the re-entrant corner analysis to more general schemes in 2-D [39] and 3-D [34], of which the standard Cartesian schemes are special cases. As such, the geometric inconsistencies and possible instabilities reported here extend to proposed re-entrant corner boundary updates for nine-point schemes in 2-D [39] and 27-point schemes in 3-D [34], since the volume and side parameters of re-entrant cells were not taken into account when centered conditions were applied to adjacent cells. Such instabilities have been observed in practice using the proposed interior edge conditions [40]. A matrix eigenvalue analysis of the re-entrant edge in the 3-D SLF scheme can also be found in [41] and conclusions were obtained that are similar to those presented here.

One could proceed and work out the correct re-entrant edge and re-entrant corner updates for the 3-D SLF scheme, to be combined with centered half-plane (half-hyperplane) and quarter-plane terminations, but such boundary updates will still amount to "staircase" boundaries for irregular (curved, slanted) geometries, so they will be no more valid than full-cell staircase boundaries, for which

stability is guaranteed (this was proved in Section 5.2). A FV meshing pre-processing step, successfully implemented, should provide coefficients for all the specialised boundary updates. Strategies for meshing will invariably need to take into account stability conditions, and this will present new challenges.

More generalised impedance boundary models, such as those in [18], which differ in the time-varying components, were not featured, but similar impedance boundary conditions can be found in [15]. The analysis presented here extends to those boundary models, since the condition to bound the total interior energy, (20), does not change as long as the discrete temporal operators are chosen properly and the impedance parameters are non-negative [15].

8. REFERENCES

[1] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen der mathematischen physik," *Mathematische Annalen*, vol. 100, no. 1, pp. 32–74, 1928.

[2] S. A. van Duyne and J. O. Smith III, "Physical modeling with the 2-D digital waveguide mesh," in *Proc. Int. Computer Music Conf. (ICMC)*, Tokyo, Japan, 1993.

[3] L. Savioja, T. J. Rinne, and T. Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. Int. Computer Music Conf. (ICMC)*, Danish Institute of Electroacoustic Music, Denmark, 1994, pp. 463–466.

[4] D. Botteldooren, "Finite-difference time-domain simulation of low-frequency room acoustic problems," *J. Acoustical Society of America*, vol. 98, pp. 3302–3308, 1995.

[5] J. G. Meloney and K. E. Cummings, "Adaptation of FDTD techniques to acoustic modeling," in *11th Annual Review of Progress in Applied Computational Electromagnetics*, vol. 2, 1995, pp. 724–731.

[6] A. Saleh and P. Blanchfield, "Analysis of acoustic radiation patterns of array transducers using the TLM method," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 3, no. 1, pp. 39–56, 1990.

[7] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas and Propagation*, vol. 14, no. 3, pp. 302–307, 1966.

[8] P. B. Johns, "Simulation of electromagnetic wave interactions by transmission-line modelling (TLM)," *Wave motion*, vol. 10, no. 5, pp. 597–610, 1988.

[9] A. Taflove, "Application of the finite-difference time-domain method to sinusoidal steady-state electromagnetic-penetration problems," *IEEE Trans. Electromagnetic Compatibility*, vol. EMC-22, no. 3, pp. 191–202, 1980.

[10] S. Bilbao, "Wave and scattering methods for the numerical integration of partial differential equations," Ph.D. thesis, Stanford University, 2001.

[11] G. Cohen, *Higher-order numerical methods for transient wave equations*. Springer-Verlag, 2002.

[12] L. Rhaouti, A. Chaigne, and P. Joly, "Time-domain modeling and numerical simulation of a kettledrum," *J. Acoustical Society of America*, vol. 105, pp. 3545–3562, 1999.

[13] B. Heinrich, "Boundary value problems and irregular networks," in *Finite Difference Methods on Irregular Networks*. Springer, 1987, pp. 17–39.

[14] D. Botteldooren, "Acoustical finite-difference time-domain simulation in a quasi-Cartesian grid," *J. Acoustical Society of America*, vol. 95, pp. 2313–2319, 1994.

[15] S. Bilbao, "Modeling of complex geometries and boundary conditions in finite difference/finite volume time domain room acoustics simulation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1524–1533, Jul. 2013.

[16] B. Hamilton and C. J. Webb, "Room acoustics modelling using GPU-accelerated finite difference and finite volume methods on a face-centered cubic grid," in *Proc. Digital Audio Effects (DAFx)*, Maynooth, Ireland, Sep. 2013, pp. 336–343.

[17] D. T. Murphy and M. Beeson, "The KW-boundary hybrid digital waveguide mesh for room acoustics applications," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 2, pp. 552–564, 2007.

[18] K. Kowalczyk and M. van Walstijn, "Formulation of locally reacting surfaces in FDTD/K-DWM modelling of acoustic spaces," *Acta Acustica united with Acustica*, vol. 94, no. 6, pp. 891–906, 2008.

[19] G. E. Forsythe and W. R. Wasow, *Finite-difference methods for partial differential equations*. New York: Wiley, 1960.

[20] R. M. Alford, K. R. Kelly, and D. M. Boore, "Accuracy of finite-difference modeling of the acoustic wave equation," *Geophysics*, vol. 39, no. 6, pp. 834–842, 1974.

[21] M. A. Dablain, "The application of high-order differencing to the scalar wave equation," *Geophysics*, vol. 51, no. 1, pp. 54–66, 1986.

[22] J. Botts and L. Savioja, "Integrating finite difference schemes for scalar and vector wave equations," in *Proc. IEEE ICASSP*, Vancouver, Canada, 2013, pp. 171–175.

[23] R. D. Richtmyer and K. W. Morton, *Difference methods for initial-value problems*, 2nd ed. Interscience Publishers, 1967, ch. 10: Sound Waves.

[24] G. W. Platzman, "The lattice structure of the finite-difference primitive and vorticity equations," *Monthly Weather Review*, vol. 86, no. 8, pp. 285–292, 1958.

[25] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, 2009, vol. 501.

[26] S. A. van Duyne and J. O. Smith III, "The 3D tetrahedral digital waveguide mesh with musical applications," in *Proc. Int. Computer Music Conf. (ICMC)*, Hong Kong, 1996, pp. 9–16.

[27] J. Tuomela, "Fourth-order schemes for the wave equation, Maxwell equations, and linearized elastodynamic equations," *Numerical Methods for PDEs*, vol. 10, no. 1, pp. 33–63, 1994.

[28] Y. Liu, "Fourier analysis of numerical algorithms for the Maxwell equations," *J. Computational Physics*, vol. 124, no. 2, pp. 396–416, 1996.

[29] J. A. Laird, "The physical modelling of drums using digital waveguides," Ph.D. thesis, University of Bristol, 2001.

[30] M. Kendall, *A Course in the Geometry of n Dimensions*. Hafner, New York, 1961.

[31] L. Kelders, J. F. Allard, and W. Lauriks, "Ultrasonic surface waves above rectangular-groove gratings," *J. Acoustical Society of America*, vol. 103, no. 5, pp. 2730–2733, 1998.

[32] W. J. Karplus, "An electric circuit theory approach to finite difference stability," *Trans. of the AIEE, Part I: Communication and Electronics*, vol. 77, no. 2, pp. 210–213, 1958.

[33] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. Wiley, 2009.

[34] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, 2011.

[35] J. B. Schneider, C. L. Wagner, and R. J. Kruhlak, "Simple conformal methods for finite-difference time-domain modeling of pressure-release surfaces," *J. Acoustical Society of America*, vol. 104, no. 6, pp. 3219–3226, 1998.

[36] J. G. Tolan and J. B. Schneider, "Locally conformal method for acoustic finite-difference time-domain modeling of rigid surfaces," *J. Acoustical Society of America*, vol. 114, no. 5, pp. 2575–2581, 2003.

[37] J. Häggblad and B. Engquist, "Consistent modeling of boundaries in acoustic finite-difference time-domain simulations," *J. Acoustical Society of America*, vol. 132, no. 3, pp. 1303–1310, 2012.

[38] K. Kowalczyk and M. van Walstijn, "Formulation of a locally reacting wall in finite difference modelling of acoustic spaces," in *Proc. Int. Symp. on Room Acoustics*, 2007, pp. 1–6.

[39] K. Kowalczyk and M. van Walstijn, "Wideband and isotropic room acoustics simulation using 2-D interpolated FDTD schemes," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 1, pp. 78–89, 2010.

[40] N. Borrel-Jensen, "Real-time auralisation of the lower frequency sound field using numerical methods on the GPU," Master's thesis, RWTH Aachen University and University of Copenhagen, 2012.

[41] J. Botts and L. Savioja, "Spectral and pseudospectral properties of room acoustic finite difference simulation," *IEEE Trans. Audio, Speech, and Language Processing*, 2014, to appear.

A CROSS-ADAPTIVE DYNAMIC SPECTRAL PANNING TECHNIQUE

Pedro D. Pestana, *
CITAR - UCP,
R. Diogo Botelho, 1327, 4169-005 Oporto
(also ILID - ULL and CEAUL - UL)
ppestana@porto.ucp.pt

Joshua D. Reiss, †
C4DM,
Queen Mary University of London,
Mile End Road, London E1 4NS
josh.reiss@eecs.qmul.ac.uk

ABSTRACT

This work presents an algorithm that is able to achieve novel spatialization effects on multitrack audio signals. It relies on a cross-adaptive framework that dynamically maps the azimuth positions of each track's time-frequency bins with the goal of reducing masking between source signals by dynamically separating them across space. The outputs of this system are compared to traditional panning strategies in subjective evaluation, and it is seen that scores indicate it performs well as a novel effect that can be used in live sound applications and creative sound design or mixing.

1. INTRODUCTION

Recent work on adaptive digital audio effects has seen the emergence of a new class of cross-adaptive systems that aim to do automatic or computer assisted mixing (see [1] for a review). The architecture is one that allows for a mapping of all concurrent signals in a mix to determine the processing parameters on each of the individual inputs in order to optimize either a perceptual (e.g. loudness balance, see [2]) or objective (e.g. release from masking, see [3]) characteristic of the mixed output signal. In the present work we focus on panning as a tool to overcome masking problems, as done in [4] and others, but following a radically different approach that focuses in the creation of a novel type of tool.

Previous approaches have relied on grounded theory, trying to mimic the decisions that a human sound engineer would undertake. We propose that an interesting area of exploration for intelligent systems is to strive for processing techniques that are prohibitively difficult to achieve with traditional means and by human practitioners. Being non-standard, there is a good chance that results will be unconventional and careful analysis and subjective evaluation is paramount.

The approach we are suggesting follows close on [5], which looks at the possibilities of adaptive digital audio effects, but not specifically the problem of mixing. In that work the authors proposed a method for panning different frequency bins of a signal into different azimuthal positions, where the mapping decisions for panning placement may come from the analysis of a separate signal. We extend the idea so that the azimuthal mapping optimizes masking constraints arising from the need to sum multiple individual tracks in a mix, and that this can be extended to a time-varying system that will achieve a new approach to spatialization and spatial unmasking optimization. A similar approach is done in terms of frequency-unmasking in [6], and DirAC [7] touches on similar concepts, though it aims at 'transparent' reproduction, instead of acting as a cross-adaptive effect.

In Section 2 we elaborate on the theoretical reasons for why a spectral audio panner can be a viable tool, and the concepts on which spatial audio is built upon. Section 3 presents a detailed description of our cross-adaptive algorithm, while Section 4 presents an objective analysis of results in selected samples. In Section 5 we summarize a subjective evaluation performed on examples. Finally, Section 6 points to further directions and application, and provides an overview of our results.

2. THEORETICAL MOTIVATION

Spatialization depends on Interaural Level Difference (ILD) and Interaural Temporal Difference (ITD) cues, but the relative importance of each is still not fully understood. The former works mainly at high frequencies, where the head casts an acoustic shadow that is large enough to attenuate the level reaching the contralateral ear. On the other hand, the latter is predominantly an active mechanism for low-mid frequencies, where the phase difference can be resolved by the brain. At really low frequencies, sounds seem to have an enveloping place of origin. ITDs and ILDs are often distorted and conflicting, and the auditory mechanism uses the most consistent cue; the one that suggests the same direction over a broad spectral band [8].

According to Griesinger [9], with broadband sources cues are ambiguous, and human hearing appears to simply average over the various possible sound directions to determine the best-guess position for the source, while weighting-in past history. The brain seems to use mechanisms other than localization to stream sounds together and only afterward does it assign a common place of origin. Furthermore, Blauert [10] argues that in principle, within a critical band, all sound can only be perceived as a single source of wider or narrower character. To this extent, Pulkki [11] concludes that spatial realism is not needed in audio for the resulting image to have any verisimilitude.

Modern audio production relies on amplitude panning techniques almost exclusively for the creation of azimuthal cues out of monophonic source signals. In this work we shall ignore cues stemming from signal delay, though a translation of the technique could trivially be achieved. It is typical to distribute sound sources among the reproduced stage, as the spatial release from masking (SRM) that is achieved improves clarity and intelligibility. The relationship of ITD and ILD to SRM is not fully studied, but it seems level is panning is a sensible choice [12].

* The author P.D. Pestana was sponsored by national funds through the Fundação para a Ciência e a Tecnologia, Portugal, in projects: ''PEst-OE/EAT/UI0622/2014' and 'PEst-OE/MAT/UI2006/2014'.

† The author is supported by EPSRC grant EP/K007491/1, 'Multisource audio-visual production from user-generated content'

3. ALGORITHM AND CONSTRAINTS

Our algorithm is based on the typical phase vocoder implementation, and can be outlined as follows:

1. Perform a Short-Time Fourier Transform (STFT) on all the input tracks of an audio mix.
2. Pan each resulting time-frequency (t-f) bin on each track independently, placing the heaviest (magnitude-wise) bins of each track to non-colliding locations.
3. Perform the Inverse Fourier Transform to reconstruct the time-domain signals of the mix.

An audio mix is the result of a summation of an arbitrary number J of input tracks. Let us call each individual track x_j and assume out of simplicity that it is monophonic (single-channel) and that all tracks are equal in length. Let us define our notation and establish that the mix's time-frequency representation is then given by the STFT [13]:

$$X_j\left(n,k\right)=\sum_{m=-\infty}^{\infty}x_j\left(m\right)h\left(n-m\right)e^{-i2\pi km/N}, \quad (1)$$

with n indexing discrete time, k the discrete frequency bin and $h(n-m)$ a window function of length N. This results in a complex number at each element, which can then be decomposed into magnitude and phase angle. Consider:

$$Xmag_j\left(n,k\right)=\sqrt{\mathrm{Re}[X_j\left(n,k\right)]^2+\mathrm{Im}[X_j\left(n,k\right)]^2} \quad (2)$$

to be the 3-dimensional matrix of individual magnitudes for each t-f bin n-k of each track j. The phase angle will actually not be important for our application, but it is given by the inverse tangent of the ratio of imaginary to real part, for each element of the matrix. It is both prohibitive and irrelevant to perform the STFT at every point in discrete time, so one uses a fixed window length (N) overlapping with a hop size (I) which is a subdivision of N, and uses time frames that start at nI and are N samples long. A Hamming-windowed STFT with a hop size of $N/2$ will yield perfect reconstruction upon performing the inverse transform and adding all the individual frames. It is now clear that the matrix whose elements are described by equation 1 is $J\times G\times N$ elements long, where G is the signal length zero-padded to the next multiple of I and divided by I, and N, being the window length, is also the number of bins in the spectral domain.

Prior to reconstruction, our goal in the spectral domain is to perform a readjustment of each bin so that it yields two different results for a left (L) and a right (R) channel:

$$X_j\left(n,k\right)\longrightarrow\left\{\begin{array}{l}{Y_j}^{(L)}\left(r,k\right)\\{Y_j}^{(R)}\left(r,k\right)\end{array}\right. . \quad (3)$$

with r the outbound time-frame. For all practical purposes, we shall have $r=n$, since our analysis hop size is equal to the synthesis hop size. Reconstruction of channel Z (either L or R) can then be performed by overlap-adding the windows [13]:

$${y_j}^{(Z)}\left(n\right)=\sum_{r=-\infty}^{\infty}h\left(n-r\right)\frac{1}{N}\sum_{k=0}^{N-1}\left[e^{i2\pi rk/N}{Y_j}^{(Z)}\left(r,k\right)\right]e^{i2\pi nk/N} , \quad (4)$$

and scaling the output so that the overlap of a large number of hops does not increase the synthesized amplitude too much. Note that while the notation quickly becomes heavy, this is simply the strategy behind the well-known phase vocoder approach to spectral domain processing and equation 4 simply represents the summation of the Inverse Fourier Transform of all individual time-frames. The core part of our research is not the analysis-synthesis process, but how to implement the mapping in (3).

A viable reconstruction of a spectrally processed signal depends upon a careful choice of windowing parameters. For our case, with a short window one would likely encounter amplitude modulation artifacts, while a long window would cause temporal aliasing and smearing. A similar balance results from the hop size; allowing for no overlap or little overlap results in clicking noises and too much overlap will cancel out the desired effect, restricting panning azimuth to a very narrow area. An heuristic approach convinced us that a window length of 2^{16} (working at a $44.1\ kHz$ sample rate; one update approximately every 1.49 seconds) with a hop size of one sixteenth the window length yields sonically acceptable results. There is temporal smearing that makes signals sound as if they had been put through a nonlinear reverb, but the amount is subtle enough so that it is still pleasing.

Each track's t-f bins can now be placed at position $p_j(n,k)\in[-1,1]$, where -1 represents full left and 1 full right. This position will then be converted to a specific azimuthal angle dependent on the position of the speakers (the standard stereophonic situation is defined by speakers at $\pm30^\circ$ from the median plane). The sine-cosine rule for amplitude panning states that we should have gains for the left and the right channel that follow:

$$g^{(L)}=\cos\left(\frac{\left(1+p_j(n,k)\right)\pi}{4}\right), \quad (5)$$

$$g^{(R)}=\sin\left(\frac{\left(1+p_j(n,k)\right)\pi}{4}\right). \quad (6)$$

We know from [10] that azimuthal discrimination is more accurate near the origin at the median plane than when we drift to the sides. It is thus sensible to envisage a position distribution scheme that mimics this perceptual phenomenon. We chose to allow for J discrete possible placements (as many as the track count) and balance them according to a variation on the roots of the first order Chebyshev polynomials, which gives us a well-behaved distribution. The root calculation is done through:

$$P_g=\mathrm{sign}\left[\cos\left(\frac{2g-1}{n}\pi\right)\right]-\cos\left(\frac{2g-1}{n}\pi\right), g=1,2...n, \quad (7)$$

Notice we are shifting the nodes so that they are center-heavy instead of tail heavy (the sign function is -1 for negative values, 1 for positive values and 0 for the value zero). We then transform $\{P_1,P_2,...,P_J\}\longrightarrow\{P_{1:n},P_{2:n},...,P_{J:n}\}$ by simply sorting the results, where $t_{k:n}$ denotes the k-th ascending order statistic. This gives us fixed position to which we shall map our J tracks differently for each t-f bin.

Figure 1 shows the constrained positions for four possible track counts. Notice that positions at the extreme left or right are never allowed, something that is considered a good practice by some sources [14]. Two other possible positional distributions were considered: equidistant spacing and the non-constrainment to discrete angles (relying instead on the energy distribution of each bin to achieve symmetrical balance). Informal testing showed that our

choice yields better perceptual envelopment than equidistant spacing and more stability than not having fixed discrete points (panning positions are less prone to large changes from frame to frame).

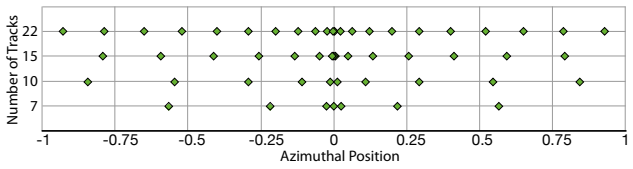


Figure 1: Allowable azimuthal positions P for four possible track counts J, according to our Chebyshev-roots based constraint.

The decision regarding which position to choose for each bin must be bound by some constraints, following [4, 14, 15]:

1. For each bin the sum of the individual track amplitudes contributing to the left and right signal must be balanced in terms of magnitude:

$$\sum_{j=1}^{J} p_j(n,k)\,|X_j\,(n,k)| = 0, \forall n,k. \tag{8}$$

2. The sum of the magnitudes of all weighted frequency bins for a fixed time-frame must be equal at all chosen azimuthal steps:

$$\sum_{k=0}^{N-1} p_j(n,k)\,|X_j\,(n,k)| = \alpha, \forall j,n, \tag{9}$$

where α is a time-frame-varying constant.

3. Successive time frames cannot allow for a frequency-azimuth pair to shift more than a maximum azimuthal step-size. We have restricted the step size to two after some informal listening tests.

4. Bins representing frequencies below a cut-off point should be centrally panned, in keeping with most industry practices. We have chosen that frequency to lie in the vicinity of $150\ Hz$.

It is impossible to enforce constraints 1 and 2 when bounded by discrete positions and the need to fulfill constraint 3, so we use those as lax rules that provide a per-frame platonic ideal, that is then revised by the need to restrict sudden movement. We establish the positions for the first time-frame by applying a palindromic Siegel-Tukey type ordering [16] to each bin across tracks, according to ordered magnitude, and continue in the following time-frames with a greedy algorithm. The starting point choice and iterative proceeding rules are best explained by example:

Suppose we have a five track mix and that the first time-frame results in the following magnitude vectors:

$$\begin{cases} \mathbf{Xmag}_1\,(1,k) = \{0.2, 0.3, 0.25...\} \\ \mathbf{Xmag}_2\,(1,k) = \{0.3, 0.2, 0.15...\} \\ \mathbf{Xmag}_3\,(1,k) = \{0.4, 0.1, 0.35...\} \\ \mathbf{Xmag}_4\,(1,k) = \{0.6, 0.02, 0.6...\} \\ \mathbf{Xmag}_5\,(1,k) = \{0.8, 0.01, 0.5...\} \end{cases} \tag{10}$$

For the first bin (ignoring the fact that constraint 4 would force us not to use panning) track 5 has the highest magnitude value, followed by 4, 3, 2, 1. These would then be panned respectively to p_1, p_5, p_4, p_2, p_3. This ordering would place heavier track-bins towards the extremes, yet would tend to enforce constraint 1.

For the second bin, track 1 has the highest magnitude value, followed by 2, 3, 4, 5. They would be panned respectively to p_3, p_2, p_4, p_5 and p_1. This reverse Siegel-Tuckey ordering will help enforce constraint 2. We can move to bin 3 and simply shift the order so that it starts with the heaviest bin at p_5 and bin 4 will start with p_3 but move to p_4 and bin 5 will have equal positioning possibilities to bin 1. Thus, this part of the algorithm works as a 4-step mapping climbing up the frequency bins.

The second time-frame is first planned in a similar fashion, but we do not allow for individual bin shifts of more than two positions between consecutive time-frames. The target goal will sometimes become unattainable, and in such cases we try to minimize the least square errors between the intended position vector and the possible position vector. There is one special feature that we can use to optimize which is the symmetry of ordering: symmetrical azimuths about zero are equal terms of weight for our constraints 1 and 2. So if we see an intended shift of a specific bin from p_1 to p_4 in two successive time frames, constraint 3 would tell us to move no farther than p_3 (two steps), but given that p_4 is symmetrical to p_2, thus p_2 would present a better move. The programmatic implementation of the greedy algorithm is consequently messy and more prone to be described then notated. Since for each new frame we always calculate our pseudo-optimal positions, the actual overall placement never diverges too far from the intended one.

Finally, with the intended position for each t-f bin in mind, there are two possible approaches to performing the calculations: multiplying $X_j(n,k)$ (the complex spectrum as a whole, not the individual magnitudes) by the gains that were obtained ($g^{(L)}$ and $g^{(R)}$) or zeroing all t-f bins that are not going to be panned to a specific position for each track, doing the Inverse Fourier Transform of the result, and panning on the time domain. We chose to do the former for our examples, but the latter will be explored in the future. There are some audible examples of the algorithm's working at http://www.stereosonic.org/phd/specPanning/, alongside spectrograms that illustrate the change it brings about.

4. RESULT ANALYSIS

Figure 2 shows the intended positions for a segment of a five-track mix. The five discrete azimuthal positions are described in shades of gray from maximum-left (white) to maximum-right (black).

This is an illustrative example of something that is quite hard to visualize but serves to show 1) that pan position is different for each frequency bin of each track, when considering a fixed time-frame (if one looks at any column on any track), 2) that pan positions of each bin change over time (if one looks at any row on any track), 3) that there is some degree of inertia for each time frame, frequency bin (i.e, a row or column will not change position too drastically) and 4) Drums and Synth tend to be either full-left or full-right, the other tracks are much more inert.

An algorithm that proposes to work as a novel audio effect is quite hard to assess objectively. It is important to understand how closely the constraints 1 and 2 are met, in light of the fact that our rules are lax. We have looked at four multitrack songs in order to understand deviations from what is expected. We are dealing with multi-dimensional phenomena so there is little tangible feeling for how big a deviation can be and how to measure it. We use the following two metrics to determine how well we match the constraints. For constraint 1, we find the discrete frequency and

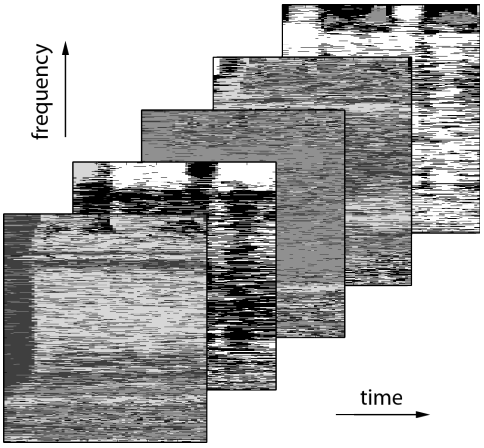


Figure 2: Intended pan positions for 5 tracks (front-to-back: bass, drums, guitar, vocals, synth. The time segments are always the same 5-seconds. White means left and black right.

discrete time mean of the absolute weighted azimuth deviations from zero:

$$\frac{\sum_n \sum_k \left| \sum_j p_j(n,k) \left| X_j(n,k) \right| \right|}{G \times N}, \tag{11}$$

where, as before, N represents the window length, and here is used as the number of bins, and G is the number of time windows used on the STFT.

For constraint 2, we find the mean over discrete time of the standard deviation between azimuthal magnitudes:

$$\frac{\sum_n s_{dev} \left(\sum_k |X_j(n,k)|_{|p_j(n,k)=P_g} \right)}{G} \tag{12}$$

Table 1 shows the results for the four song segments ($\approx 20\ s$), for three different approaches. The benchmark approach is random panning of the t-f bins. We compare that with following our algorithm with a window length of 2^{15} and a hop size of $1/16$ and a window length of 2^{10} with a hop size of $1/2$.

Approach	Song #1	Song #2	Song #3	Song #4
Random panning	4.957	6.148	4.469	6.065
Window 2^{15}, hop 1/16	0.024	0.019	0.018	0.024
Window 2^{10} hop 1/2	0.077	0.101	0.992	0.140

Approach	Song #1	Song #2	Song #3	Song #4
Random panning	1.664	2.167	1.857	1.935
Window 2^{15}, hop 1/16	0.325	0.143	0.464	0.656
Window 2^{10} hop 1/2	1.431	0.542	1.161	1.548

Table 1: Approximation to constraints 1 (top) and 2 (bottom). Top table values are multiplied by 10^3, bottom table by 10^{16}.

The values show that constraint 2 will always be approached as a result of the law of large numbers, even for the case of random panning. However, both our approaches achieve a better result than a random strategy. A large window size will yield more frequency bins, which will likely smooth the variations at each position, so its relatively better score comes as no surprise. As for constraint 1, the algorithm shows a clear improvement against random panning. Here the larger window also seems to work consistently better, most likely because of the larger overlap, stemming from the hop size.

5. SUBJECTIVE EVALUATION

In order to understand whether the approach is worthwhile we performed a subjective multi-stimulus evaluation. Twenty subjects of moderate experience with audio engineering took place. The test signals were presented via headphones at a consistent listening level ($83\ dB$) through a steady signal chain. The use of headphones was a compromise, as the technique is better suited for a loudspeaker test; however, repeatability in different settings was important, so a compromise was chosen. Tests were double blind, using 4 different versions of 4 different songs (20 second segments). Versions of one such song can be heard online at http://www.stereosonic.org/phd/specPanning/. Procedures followed closely [17] and the different stimuli were:

1. A monophonic version that served as the base for the algorithm. Loudness balance done by a mixing engineer. This balance choice was kept throughout (M).
2. A traditional static stereophonic version, where panning decisions were done by a professional mixing engineer (S).
3. A spectrally panned version with our best time constants: window length of 2^{15} with a $1/16$th hop (Lg, for 'long window').
4. An anchor, a spectrally panned version with a time constant that some preliminary tests had shown problematic: window length of 2^{10} with a $1/2$ hop (Sh, for 'short window').

The order of presentation was randomized, a fact that was explained to the listeners in advance. Subjects were asked to rank the versions according to three parameters (one per listening run for a total of 3×4 runs with 4 versions per run): 'clarity', 'production value' and 'excitement'. The concept of clarity should be associated with the ability to segregate sources, thus with release from masking. Production value is inherently ambiguous and subjective, but because subjects were studying or had studied audio engineering, the meaning should be clear: technical quality of the mix. Excitement was expected to reveal a dimension that is not coupled to the evaluation of quality but of a rawer reaction to the mix.

Results are summarized in Figure 3, showing mean and confidence interval bounds for the three parameters. The professional version is superior in terms of production value perception, yet our algorithm competes in terms of clarity. This is an encouraging result, as we are comparing a trained professional approach to an experimental algorithm that goes dangerously against what would be considered accepted by traditional standards. Our algorithm also yields the best score for excitement, beyond the error bounds, which validates the hypothesis of using it on a mix for its 'special effect' character. The anchor version is perceived as unpleasing, yet it scores fairly well in terms of excitement. A Friedman test for evaluation consistency among users only finds evidence for randomness in terms of production value, which might be explainable by the difficulty in having consensus on what "Production Value" means. Further investigation reveals that subjects are evaluating songs differently in that case. Separating by song, our algorithm is perceived as excelling in production value for one of the four examples (exactly the one that can be found online).

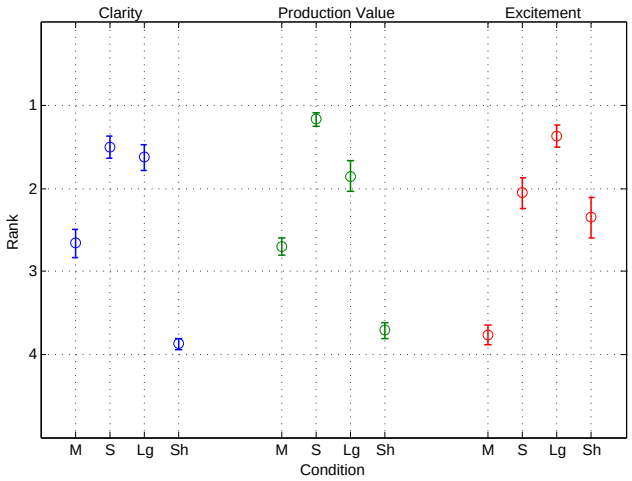


Figure 3: Evaluation results for Clarity (left), Production Value (middle) and Excitement (right), averaging both in terms of songs and subjects. See main text for conditions.

6. CONCLUSIONS AND FURTHER WORK

A technique of spectral unmasking through cross-adaptive dynamic panning is explored. It is to be understood as an experimental effect and not a method that can be used in traditional mixing. An absolute constraint equilibrium cannot be met constantly, and slow variations through time are shown to produce better results. Subjective evaluation confirms the algorithm can be used as a novel effect, resulting in an appreciated level of excitement. It is shown to compete with a professional mix in terms of clarity, so the purpose of unmasking is to some degree accomplished (particularly if one remembers the evaluators will always understand the algorithm as "strange"). Such a radical panning strategy could very well be understood as nonsensical, but results show otherwise, at least when considering the longer time-constant version. It is safe to assume that a listener will not understand that there is an explosion of panned events, and is still very much able to identify each sound source as being one.

A technique such as this can be useful in several scenarios:

1. Whenever amplitude panning of whole sources is frowned upon, such as large outdoor live shows, where elements are kept in the middle. The algorithm would allow audience members on the center to feel a sense of spaciousness while allowing everyone to still have the perception of a full mix.
2. In song sections, where the producer wishes to add a subtle other-worldly feeling to a part.
3. In excessively dense mixes, where the mixing engineer is struggling for clarity.

Several extensions of the idea can be researched in the future if one considers that the azimuthal positional choices on this work were mainly heuristic. There are many explorations on the nature of the constraints, and the reasons for their choice, that can result in clearer and more effective results. The research for a real-time strategy is in the works. A comparison with techniques involving filter-bank methods either in the analysis or synthesis is also an important step.

7. REFERENCES

[1] Joshua D. Reiss, "Intelligent Systems for Mixing Multichannel Audio," in *17th Intl Conf on Digital Signal Processing (DSP)*, 2011.

[2] Enrique Perez Gonzalez and Joshua D. Reiss, "Automatic Gain and Fader Control For Live Mixing," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2009.

[3] Enrique Perez Gonzalez and Joshua D. Reiss, "Improved Control for Selective Minimization of Masking Using Inter-Channel Dependancy Effects," in *Proc of the 11th Intl Conf on DAFx*, 2008.

[4] Stuart Mansbridge, Saoirse Finn, and Joshua D. Reiss, "An Autonomous System for Multi-track Stereo Pan Positioning," in *Proc of the 133rd AES Convention*, 2012.

[5] Vincent Verfaille and Udo Zölzer, "Adaptive Digital Audio Effects (A-DAFx): A New Class of Sound Transformations.," *IEEE Transactions on Audio, Speech, and Language.*, vol. 14, no. 5, pp. 1817 – 1831, 2006.

[6] Piotr Kleczkowski and Adam Kleczkowski, "Advanced Methods for Shaping Time-Frequency Areas for the Selective Mixing of Sounds," in *Proc of the 120th AES Convention*, 2006.

[7] Ville Pulkki, "Spatial Sound Reproduction with Directional Audio Coding," *Journal of the Audio Engineering Society*, vol. 55, no. 6, pp. 503–516, 2007.

[8] Ville Pulkki, "Localization of Amplitude-panned Virtual Sources I: Stereophonic Panning," *Journal of the Audio Engineering Society*, vol. 49, no. 9, pp. 739–752, 2001.

[9] David Griesinger, "Stereo and Surround Panning in Practice," in *Proc of the 112th AES Convention*, 2002.

[10] Jens Blauert, *Spatial Hearing: the Psychophysics of Human Sound Localization*, MIT Press, Cambridge, 1997.

[11] Ville Pulkki, T. Lokki, and Davide Rocchesso, "Spatial Effects," in *DAFx*, Udo Zölzer, Ed., chapter 5, pp. 139–180. John Wiley & Sons, Chichester, second edition, 2011.

[12] J. M. Dillon H. Cameron S. Glyde, H. Buchholz and L. Hickson, "The importance of interaural time differences and level differences in spatial release from masking," *Journal of the Acoustical Society of America*, vol. 2, no. 134, pp. 147–152, 2013.

[13] R.E. Crochiere, "A Weighted Overlap-Add Method of Short-Time Fourier Analysis/Synthesis," *IEEE Trans. on Speech and Aud. Proc.*, vol. 281, no. 1, pp. 99–102, 1980.

[14] Pedro D. Pestana, *Automatic Mixing Systems Using Adaptive Digital Audio Effects*, Phd, Universidade Católica Portuguesa, 2013.

[15] Enrique Perez Gonzalez and Joshua D. Reiss, "A Real-Time Semiautonomous Audio Panning System for Music Mixing," *EURASIP Journal on Advances in Signal Processing*, 2010.

[16] S. Siegel and John W. Tukey, "A nonparametric Statistics for the Behavorial Sciences," *Journal of the American Statistical Association*, vol. 55, pp. 429–445, 1960.

[17] S. Bech and N. Zacharov, *Perceptual Audio Evaluation — Theory, Method and Application*, John Wiley & Sons, Chichester, 2006.

LOW-DELAY ERROR CONCEALMENT WITH LOW COMPUTATIONAL OVERHEAD FOR AUDIO OVER IP APPLICATIONS

Marco Fink, Udo Zölzer

Dept. of Signal Processing and Communications
Helmut-Schmidt-Universität
Hamburg, Germany
marco.fink@hsu-hamburg.de

ABSTRACT

A major problem in low-latency Audio over IP transmission is the unpredictable impact of the underlying network, leading to jitter and packet loss. Typically, error concealment strategies are employed at the receiver to counteract audible artifacts produced by missing audio data resulting from the mentioned network characteristics. Known concealment methods tend to achieve only unsatisfactory audio quality or cause high computational costs. Hence, this study aims at finding a new low-cost concealment strategy using simplest algorithms. The proposed system basically consists of an period extraction and alignment module to synthesize concealment signals from previous data. The audio quality is evaluated in form of automated measurements using PEAQ. Furthermore, the system's complexity is analyzed by drawing the computational costs of all required modules in all operating modes and comparing its computational load versus another concealment method based on auto-regressive modeling.

1. INTRODUCTION

The transmission of audio material over packet-switched networks experienced increasing popularity in many application areas over the last decade. To comply with the bound of available data rates it was necessary to develop audio codecs to decrease the data rate of audio material without affecting the audible quality in a severe way. In other words, the massive distribution of digital music content was initiated by the development of codecs.

Nowadays, the usage of the internet tends to change from file-based multimedia exchange to streaming-based and interactive scenarios, like *Networked Music Performances* (NMP) [1, 2]. To allow the experience of a real-time system the overall delay between sender and receiver should be minimal. Unfortunately, there are many delay contributors between source and sink. Besides the obvious non-deterministic network delay, the blocking delay of the audio hardware, the delay introduced by receiver buffers, and the algorithmic delay of underlying codecs have to be considered. Apparently, only the blocking delay and the algorithmic delay can be reduced with the help of signal processing. Therefore, the focus of optimization in codecs has changed from data rate to delay. State of the art audio codecs [3, 4] feature algorithmic delays less than 5 ms to allow the aforementioned interactive scenarios.

A major problem in interactive online applications is the diminished audio quality caused by non-optimal network conditions leading to jitter and packet loss. Commonly, receivers apply error concealment techniques to reduce the impact of audible artifacts by replacing the gap, resulting from packet loss, in various ways. Many different concealment strategies are known [5, 6] but the majority of them requires an unpractical amount of processing power or can not satisfy a certain quality level. Popular strategies are often based on Overlap and Add techniques, like *Waveform Similarity Overlap and Add* (WSOLA) [7], or model-based extrapolation [8, 9, 10].

For *Audio over IP* (AoIP) implementations on embedded devices [11] like the Raspberry Pi [12], the concealment is even the systems's limiting module. Hence, this study aims at finding a low-cost alternative and still allow a reasonable audio quality. Since the concealment is to be applied in low-delay AoIP applications, it is optimized for very short blocks.

The proposed concealment strategy aims at extracting periods from past data blocks and utilize them to fill the gap produced by packet loss. In contrast to the before mentioned techniques, the computation of an auto-regressive model and/or autocorrelation shall be avoided to save computations.

A crucial property of high-quality audio error concealment strategies is to guarantee correct phase alignments between frames. Hence, the fade-in and fade-out of the concealed block to the original data receives much attention. Several methods to realize this important property are proposed and analyzed. The actual period extraction is accomplished with the help of zero-crossings and matched pre-processing. The quality of the proposed concealment strategy was ranked using large-scale automated measurements with the *Perceptual Evaluation of Audio Quality* (PEAQ) [13] algorithm. Additionally, the complexity was compared to a high-quality concealment strategy [8], based on extrapolation using auto-regressive models [14].

The paper is structured as follows. The overall system and all its sub-modules are explained in Section 2. The result of the quality measurements can be found in Section 3, whereas the complexity analysis is located in Section 4. Section 5 concludes this paper.

2. SYSTEM

The concealment system is structured module-wise as shown in Fig. 1. Previous data $x_p(n)$ is optionally enhanced in the Pre-Processing block to improve the performance of the Zero-Crossing Analysis module, which generates the zero-crossings vector $z_c(n)$. The resulting $z_c(n)$ is then used to identify extraction boundaries, allowing the cutting out of multiple periods of the unaltered previous data $x_p(n)$ within the Extraction block. Subsequently, the alignment module is supposed to identify the phase offset between the last data block and the extracted concealment signal $x_e(n)$. The actual phase alignment can then be performed within the same

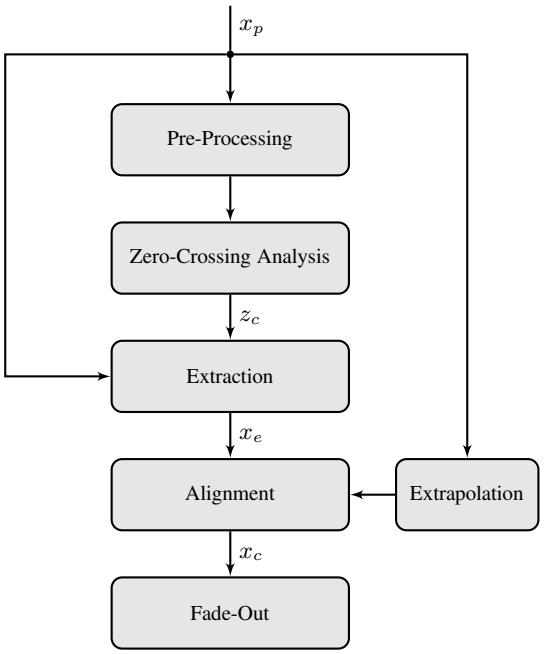


Figure 1: System overview

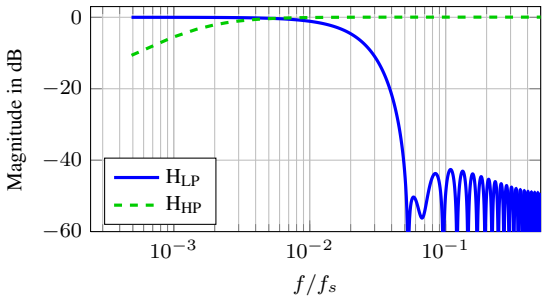


Figure 2: Pre-processing filters

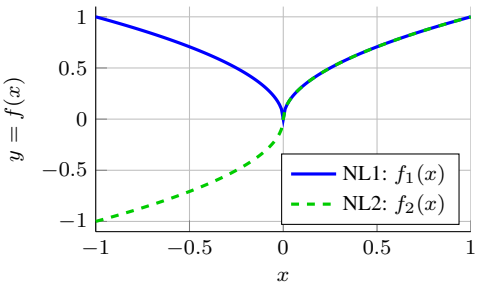


Figure 3: Odd and even pre-processing non-linearities

module. To allow smooth transitions from the previous data block into the concealment data the extrapolation module is applied. It computes several continuative samples and cross-fades them with the concealment data resulting in x_c. Lastly, the Fade-Out module guarantees a certain continuity from the concealed data x_c into the next data block. Hence, the length of x_c should exceed a single block length.

2.1. Pre-processing

Two different pre-processing steps are considered to improve the system's overall performance. Only the fundamental harmonic content of the input signal should be suspect to the zero-crossing analysis. Therefore, a FIR lowpass H_{LP} with a normalized frequency of $\omega_n = .01 \cdot 2\pi$ and order 20 is utilized. In order to guarantee a DC-free signal a first-order recursive filter $H_{\text{HP}} = \frac{1-z^{-1}}{1-0.99z^{-1}}$ is additionally applied. The corresponding transfer functions are illustrated in Fig. 2. To conserve the position of the zero-crossings, it is crucial to apply zero-phase filtering. Therefore, the FIR lowpass is applied in forward and reverse direction [15]. The IIR filter phase response is close to zero in the passband and hence, is not affecting the signal.

Besides restricting the bandwidth of the input, the signal was subject to non-linear processing to enhance the first harmonic as proposed in [16]. The characteristic curves of the considered non-linear functions

$$f_1(x) = \sqrt{|x|} \tag{1}$$

$$f_2(x) = \sqrt{|x|} \cdot \text{sgn}(x) \tag{2}$$

are depicted in Fig. 3. Apparently, $f_1(x)$ and $f_2(x)$ are odd and even non-linear functions, respectively.

2.2. Zero-Crossing Detection

Zero-crossing positions are a meaningful low-level feature, mostly utilized in speech processing. For example, the detection of voiced and unvoiced parts in speech recordings using the zero-crossing rate (ZCR) is very common [17]. It can also be beneficial in the context of AoIP since received audio streams primarily consist of single instruments or speakers and therefore, should mainly contain harmonic signals featuring a strong periodicity. Zero-crossings can be stored in a binary vector

$$z_c(n) = \begin{cases} 1 & \text{if} \quad \text{sgn}(x(n) \cdot x(n-1)) < 0 \\ 0 & \text{else} \end{cases} \tag{3}$$

of length N_i, indicating a zero-crossing at samples $[n_1, \ldots, n_{N_i}]$.

A zero-crossing is similar to sign changes between two consequent samples $x(n-1)$ and $x(n)$. Whenever their product turns negative a zero-crossing can be assumed. Applying the XOR operation to the sign bits of $x(n-1)$ and $x(n)$ is an efficient way to achieve the same result.

The following processing steps require the actual position index vector i_{zc} containing the N_i sample indexes $[n_1, \ldots, n_{N_i}]$, where $z_c(n)$ is non-zero. The resulting vector i_{zc} can be refined by defining a lower bound δ_{zc} for the distance between two zero crossings. This value should be set according to the maximal expected fundamental frequency f_{max} of the input signal and the sampling frequency f_s

$$\delta_{\text{zc}} = \left\lceil \frac{f_s}{2 \cdot f_{\text{max}}} \right\rceil . \tag{4}$$

Whenever the inter-zero-crossing-interval falls below δ_{zc}, the corresponding zero-crossing candidate index is removed from i_{zc}.

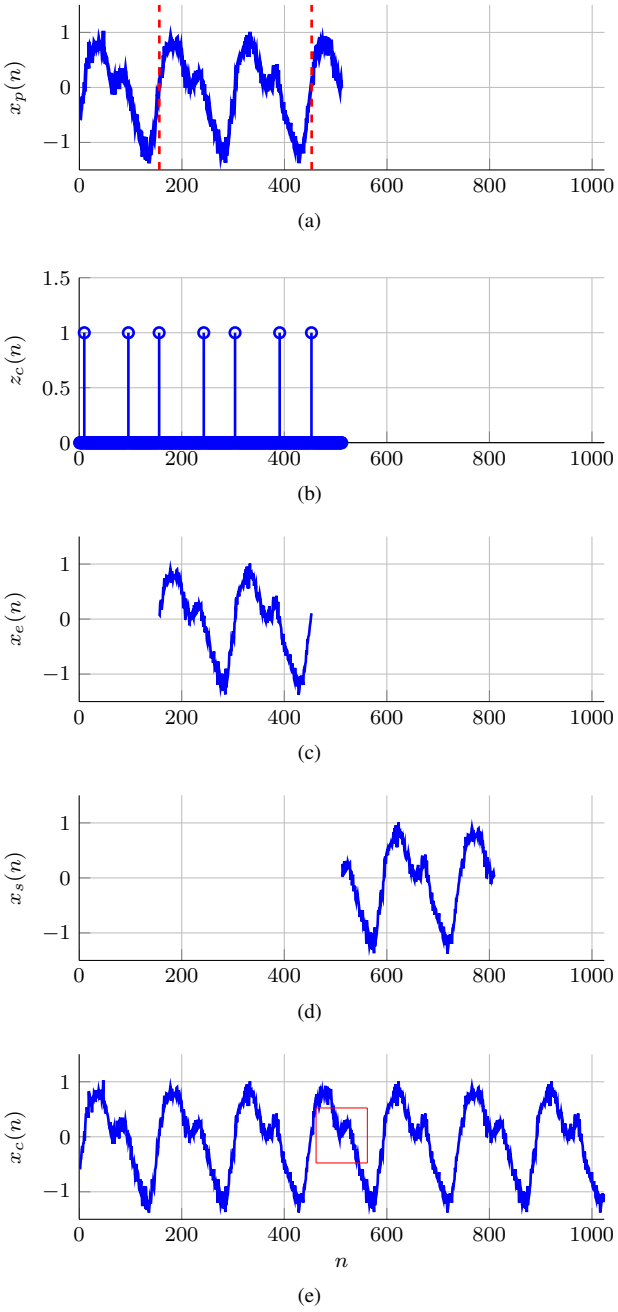


Figure 4: Relationship between x_p(a), z_c(b), x_e(c), x_s(d), and x_c(e)

2.3. Period Extraction and Alignment

The period extraction module receives a block of previous data x_p and a vector i_{zc}, containing the indexes of zero crossings in x_p, to extract one or P periods. The process is illustrated in Fig. 4. Fig. 4a) shows a periodic waveform $x_p(n)$, while Fig. 4b) depicts the corresponding zero-crossings, which are used to define the extraction boundaries, defined by the zero crossing indexes n_{N_i-2P} and n_{N_i}, which are marked with red stripes in the Fig. 4a). The extracted signal

$$x_e(n) = x_p(n), \quad \text{for} \quad n = [n_{N_i-2P}, \ldots, n_{N_i}] \tag{5}$$

of length N_e is shown in Fig. 4c). The quantity of extractable periods depends on the amount of identified zero crossings. To extract at least a single period one has to allow a certain length N_p for x_p. N_p can be restricted when a minimal frequency f_{min} for the concealment process is defined. From this it follows that

$$N_p = \left\lceil \frac{\beta \cdot f_s}{f_{min}} \right\rceil, \tag{6}$$

where β is an optional safety margin.

Since the extracted periods are concatenated until they show a certain length N_c, the beginning and end have to be matched to allow smooth transitions between the repeated periods. In the implementation this feature was realized by replacing n_m samples at the front and the end with a linear series between $x_e(N_i - n_m + 1)$ and $x_e(n_m)$ of length $2 \cdot n_m$.

The block $x_e(n)$ is zero-phased due to its cutting at the zero crossings. To align the phase of the extracted periods to the end of the last block x_p, a circular shift by l samples is applied to obtain extracted and shifted sequence

$$x_s(n) = x_e(n - l \bmod N_e), \tag{7}$$

which is plotted in Fig. 4d).

Several ways to estimate the shifting offset l are possible. In this study three methods are chosen and analyzed:

1. Zero-crossing distance: The simplest reviewed method is to compute the offset of the last zero-crossing in i_{zc} and the length of the past data block $x_p(n)$, which is determined by the amount of blocks in the buffer M and the block size N

   $$l_1 = NM - n_{N_i}.$$

   Apparently, the estimation accuracy mainly depends on the zero-crossing's precision.

2. Slope and amplitude matching: If the slope and amplitude of two periodic signals with the same frequency are similar, one can assume a similar phase at that point. Hence, it is necessary to obtain the slope at the end of $x_p(n)$ and the slope of $x_e(n)$ by differentiating both. The ten closest values are assumed as candidates. In a second step, the candidate showing the smallest deviation in amplitude to the last sample of $x_p(n)$ is determined and its index is used for the shift operation.

3. Cross-correlation: It is well-known that the delay between two signals can be computed using the cross-correlation [18]. The time delay is the offset between the location of the cross correlation's maximum and the zero lag index. Since the alignment of $x_e(n)$ to the end of $x_p(n)$ is desired, the cross correlation from the end of the signals in inverse direction is computed.

Subsequently, the concealment signal $x_c(n)$ is synthesized by concatenating $x_s(n)$ until it exceeds a certain length $N_c > N$ to allow the replacement of a complete data block of length N. The phase-matched concatenation of $x_p(n)$ and $x_c(n)$ is illustrated in Fig. 4e). The actual block transition is highlighted with the red box.

2.4. Extrapolation and Fade-In

The last section exposed a strategy to produce the actual concealment waveform $x_c(n)$ that is supposed to show similar characteristics as the waveform of previous blocks. However, the concealment quality can be improved by extrapolation of previous data and fading the resulting N_f samples into $x_c(n)$. Alike the estimation of the offset l, multiple strategies of extrapolation were compared and shall be described in the following.

1. Reflection: Applying the point reflection to the signal's tail can be performed by reverse indexing, inversion, and an offset with the last value times 2

$$x_{f1}(n) = 2 \cdot x_p(N_p) - x_p(N_p - n).$$

2. Weighted Slope Continuation: The slope of previous data is linearly weighted to avoid a constant slope in the extrapolation

$$x_{f2}(n) = x_p(N_p) + (x_p(N_p - n) - x_p(N_p - 1 - n)) \cdot n.$$

3. Linear: A linear extrapolation can be achieved by determining the slope at the end of x_p and accumulating it N_f times to the last value of x_p.

$$x_{f3}(n) = x_p(N_p) + (x_p(N_p) - x_p(N_p - 1)) \cdot n.$$

4. Polynomial: The extrapolation with a polynomial can be described as the attempt to find a function $f(x)$ describing previous data and feed it with following x values to obtain new data. The polynomial p can be found in the least-square sense by solving

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{N_f-1} \\ 1 & x_2 & \cdots & x_2^{N_f-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N_e} & \cdots & x_{N_f}^{N_f-1} \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N_f-1} \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{Nf}) \end{pmatrix},$$

where $[x_1, \ldots, x_{N_f}]$ is assumed to be a linear series from 1 to N_f. The actual extrapolated signal can then be computed the following way

$$x_{f4}(n) = p_0 + p_1(n+N_f)^1 + \cdots + p_{N_f-1}(n+N_f)^{N_f-1}$$

or using the Horner's method.

An overview of the utilized extrapolation is illustrated in Fig. 5. It shows an arbitrary waveform which is extrapolated at $n = 8$ using the four presented methods.

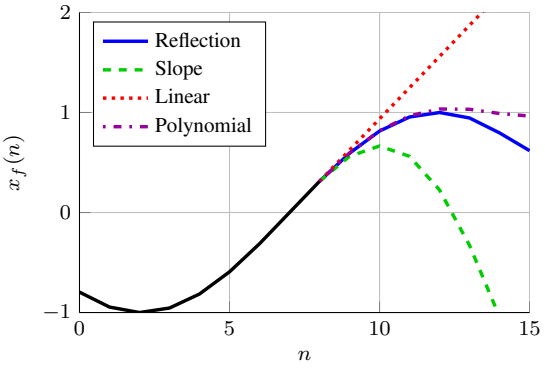


Figure 5: Different simple extrapolation algorithms for $N_f = 8$

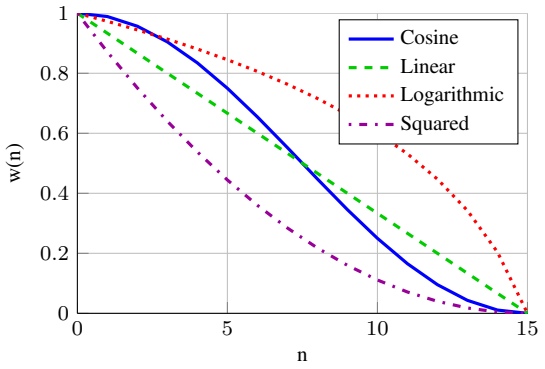


Figure 6: Fading windows for $N_w = 16$

2.5. Fade-Out

As already mentioned, the signal quality of error concealment strategies depends crucially on smooth transitions between extrapolated and actual received data. Hence, the transition from conceal data to the subsequent intact audio block has to be assured. As already denoted in [8] it is beneficial to choose the extrapolated block's length N_c longer than the actual block size N and apply a cross-fade with the next block. Several window forms $w(n)$ and lengths N_w were investigated and it turned out that only cross-fades leading to a constant amplitude

$$w(n) + w(N_w - n) = 1 \tag{8}$$

are of benefit. Although, constant power windows

$$w^2(n) + w^2(N_w - n) = 1 \tag{9}$$

are widely used in audio processing, and especially mixing, they can only be optimally used in the case of uncorrelated signals, like when mixing different tracks. In this study, four different constant amplitude cross-fade windows are applied. Namely, a cosine, a linear, a logarithmic, and a squared. All those windows are illustrated in Fig. 6. The influence of the cross-fade window length and form shall be evaluated in the following sections.

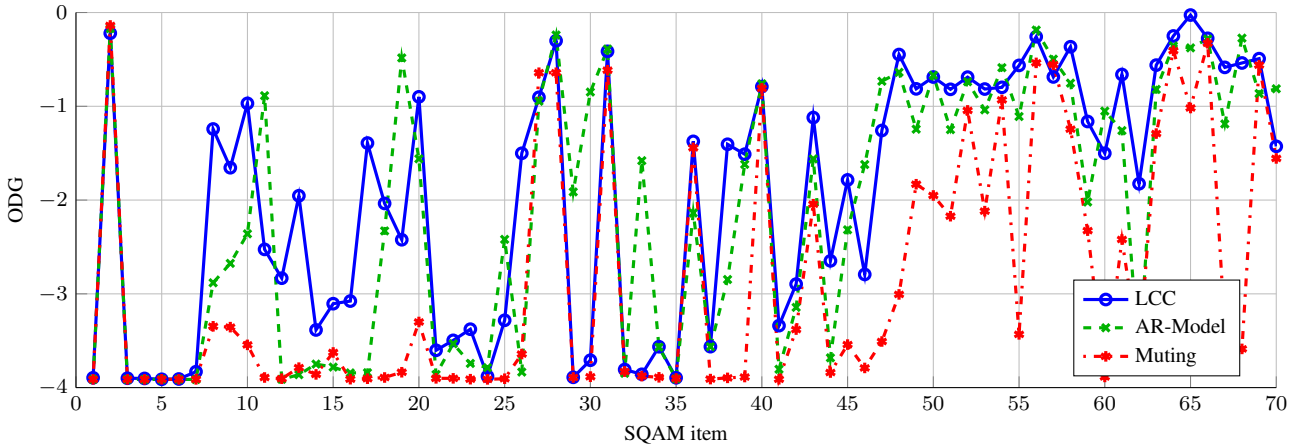


Figure 7: ODG score over SQAM test files

3. EXPERIMENTS

3.1. PEAQ

The quality of the proposed system was initially evaluated objectively using automated measurements in MATLAB, similar to the experiments in [8]. Although the *Perceptual Evaluation of Audio Quality* (PEAQ) method [13] was actually developed to rate audible artifacts caused by audio codecs, it turned out to be an advantageous tool to judge the quality of audio error concealment. The experiments in [8] revealed a significant correlation between the automated PEAQ measurements and a listening test. The result of the PEAQ algorithm is the so-called *Objective Difference Grade* (ODG). It ranges from -4 to 0, covering the identified audio quality impairment from "very annoying" to "imperceptible". The sound data base used for the measurement is the *Sound Quality Assessment Material* (SQAM) [19], consisting of 70 high-quality test items featuring a large variance of sound sources and tonal characteristics.

The actual test procedure was designed as follows: Initially, a SQAM item is loaded and down-mixed to mono. Using the test item's amount of samples and the current block size N allows to calculate the corresponding amount of frames. For every frame a random value is computed, that indicates a lost frame if its larger than the currently simulated packet loss rate assuming that one network packet contains a single audio frame. To have an error reference for following comparisons the input signal is copied once and all erroneous frames are set to 0. In other words, muting is applied as a worst-case concealment for the comparison.

The concealment routine was then called for the erroneous frame with N_P previous samples, the amount of required samples N_c, and the modes for the pre-processing, alignment, and extrapolation modules, respectively. The first N samples of its result are used to replace the erroneous frame and the remaining samples are cross-faded with the following audio frame. The overall result is written to a wave file. To obtain the ODG score, the self implemented PEAQ tool is fed with the resulting wave file and the corresponding original one. This test procedure is repeated for different parameters of the block size N, packet loss rate e, pre-processing mode, alignment mode, extrapolation mode, cross-fade length N_w, the cross-fade window function w_{cf}, and for every SQAM test item. Note, that the sample rate f_s of all test items is 44.1 kHz, the minimal frequency f_{min} is chosen to be 80 Hz, and the safety margin β is set to 1.2. Hence, the search window length N_p, which is used to find zero-crossings, is restricted to 662 samples.

Note, that the authors verified the correctness of the self implemented PEAQ tool by comparing it with two different freely available implementations peaqb and PQevalAudio [20, 21]. A test run over the SQAM data set using the three implementations and using a random concealment setting resulted in a mean absolute deviation of 0.09 for peaqb and 0.07 for PQevalaudio in comparison to the ODG scores of the own implementation.

First of all, it shall be shown how well the presented concealment strategy, denoted as Low-Cost Concealment (LCC) in the following, performs for the different SQAM items. Fig. 7 shows the ODG score over the SQAM items for a block size of 64 samples, packet loss rate $e = 0.01$, a cosine-shaped cross-fade window of length 32. Additionally, the corresponding ODG score of another concealment method [8], based on auto-regressive modeling (AR-Model) using the Burg method, and the muting error reference are shown. The AR-Model produces very good result if the models are computed using a large amount of previous data. Nevertheless, the quality of LCC and AR-Model are similar for very short blocks, although the AR-model requires much more computational effort. LCC even shows a slightly improved average score $\mu_{\text{LCC}} = -1.93$ vs. $\mu_{\text{AR}} = -2.11$ for the explained simulation parameters. Both proposed methods outperform the muting concealment clearly ($\mu_{\text{Mute}} = -2.88$). It comes apparent that the results are strongly signal-dependent. Interestingly, the overall trend is quite similar for LCC and AR-Model. For example, the SQAM items $1 - 7$, which are different synthetic signals, don't show any improvement while the string instruments, voice, and singing items work very well. Also items $26 - 34$ lead to very bad results since these are percussive and bell-like sounds without strong periodicity. As mentioned in the introduction, the system is designed to conceal natural, harmonic sounds. Hence, the previously described items are removed from the test set to evaluate the proposed system in the context of its potential application.

As a next step, the measurement was repeated using the limited SQAM data set but with varying block length N and error rate e. The results of the individual SQAM test items were averaged.

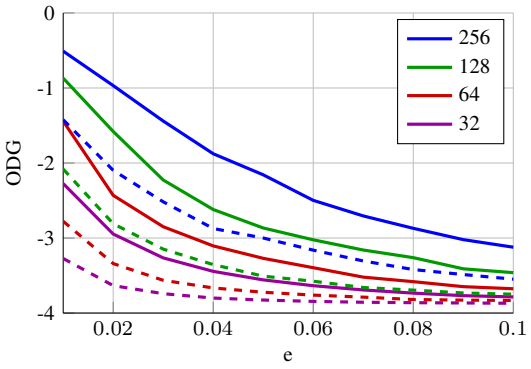


Figure 8: ODG score over error rate using different block lengths for the proposed method (solid) and the AR reference method (dashed)

As expected, the quality decreases for increased e and hence, the ODG curves in Fig. 8 decrease monotonically. The solid curves represent the LCC results, whereas the dashed curves show the results for the error reference. The ODG scores are degraded more severely for shorter blocks even when the same amount of previous data is used for the concealment. This behavior allows two conclusions. On the one hand the replacement and fading using longer blocks is smoother and causes less abrupt signal changes or on the other hand, short signal deviations have stronger negative impact on the ODG score.

In the following, the influence of the combination of the different operating modes are analyzed. Therefore, the block size N and error rate e were set to 64 and 0.01, respectively. Only the operating modes are varied and the average ODG score using the limited SQAM data set was computed. First of all it should be noticed that all LCC modes, besides the combination of the first non-linear function and the filter with arbitrary alignment and extrapolation, improved the average ODG score significantly. Most combinations show an improvement of more than 1 ODG score in comparison to the muting error reference, which was rated $\mu_{\text{Mute}} = -2.8361$ by PEAQ. The mode combinations, yielding the ten best results, are shown in Table 1. Apparently, the pre-processing improves the performance of the zero-crossing analysis since the four best combinations use the filter and/or the second non-linear function. The alignment, based on matching of slopes and amplitudes, outperforms the cross correlation and zero-crossing distance estimations to estimate the phase offset. The simple linear extrapolation seems to be the favorable method for the fade-in process since the six best results are obtained using the linear extrapolation.

As mentioned before, also the effect of the cross-fade window's form and length on the PEAQ results shall be demonstrated. The previous test setup was repeated with fixed parameters ($N = 64$, $e = 0.01$) and the best-performing concealment mode (see Table 1), presented in the previous section. The limited SQAM set was used for this experiment again. As it is apparent from Fig. 9, showing the ODG score for the windows described in Sec. 2.5 and relative window lengths, the centered cross fades (Cosine and Linear) clearly outperform the non-centered ones (Logarithmic and Squared). This implies that neither extrapolated data nor the next intact block should be emphasized in the cross-fade process. The linear cross-fade window is slightly beneficial, especially for larger N_w. There is only minor improvement for window lengths $N_w > 0.5N$ but if it falls below, strong quality degradation occurs.

Table 1: 10 best-performing LCC modes

Pre-Processing	Alignment	Extrapolation	ODG score
NL2+F	SA	L	−1.5924
F	SA	L	−1.5948
F	ZC	L	−1.6171
NL2+F	ZC	L	−1.6247
None	SA	L	−1.6452
NL2	SA	L	−1.6452
NL2+F	SA	P	−1.6648
F	SA	P	−1.6661
NL2+F	SA	S	−1.6740
F	SA	S	−1.6744
Muting			−2.8631

F: Filter, NL1: Non-linearity 1, NL2: Non-linearity 2,
SA: Slope and Amplitude Matching, ZC: Zero-Crossing Distance,
L: Linear Extr., P: Polynomial Extr., S: Slope Extr.

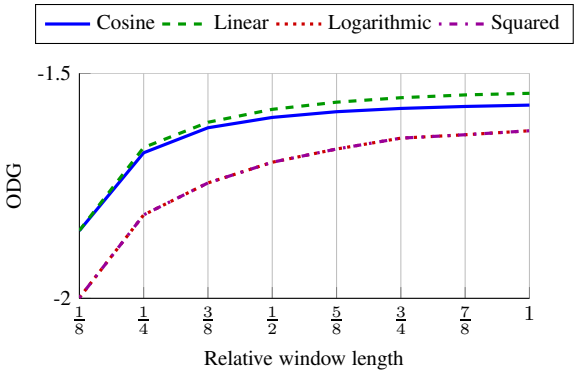


Figure 9: ODG score over different window lengths and forms

4. COMPLEXITY

Since the LCC complexity significantly depends on the operation mode and parametrization it is non-trivial to express an overall complexity. Hence, the complexity of the included modules in different configurations shall be discussed. The following expressions (see Table 2) represent the complexity to conceal a block of length N. Non-complex multiplications (MULs) and additions (ADDs) are listed in the corresponding MUL and ADD column, whereas important conditionals, memory operations, and functions, which significantly depend on the implementation and target platform like the root square function, are listed in the misc column.

1. Preprocessing: The twice-executed FIR filter of order 20 (21 taps) consumes $2 \cdot 21 \cdot N$ MULs and ADDs, whereas the IIR filter requires 4 MULs and 5 ADDs. Both non-linear functions consist of the abs function (e.g. conditional and assignment) and the square root function, which is strongly implementation-dependent and hence can not be described generically.
2. Zero-Crossing Analysis: The zero-crossing analysis can be implemented by comparing the sign of two consecutive sam-

Table 2: Complexity Overview

Module	MUL	ADD	misc
Preprocessing			
FIR	$2 \cdot 21 \cdot N$	$2 \cdot 21 \cdot N$	
IIR	$4 \cdot N$	$5 \cdot N$	
NL1	0	0	$N\cdot$ sqrt, $N\cdot$ abs
NL2	N	0	$N\cdot$ sqrt, $N\cdot$ abs
Zero-Crossing	N	0	$N\cdot$ sign
Extraction	0	0	memcpy(N_e)
Alignment			
ZC	1	0	
SA	0	$2 \cdot N_e + 11$	sort(N_e) + sort(10)
XC	0	0	conv(N_e) + sort($2 \cdot N_e - 1$)
Extrapolation			
Mirror	0	5	
Slope	4	8	
Linear	5	8	
Polynomial	9	12	solve(4)
Fading	8	4	

ples by multiplying them and comparing the resulting sign bit.

3. Extraction: The actual extraction is a simple copy operation (memcpy) of previous data in a new processing buffer.
4. Alignment: The alignment itself is simply a rearrangement of the extracted buffer which can be realized by adapted indexing, e.g. with a pointer offset. If the alignment offset index l is computed using the zero-crossing distance one can subtract the last zero-crossing index from the length of the buffer used for the zero-crossing analysis.

   For the case of the Slope and amplitude matching, the derivative of the extracted concealment signal (N_e ADDs) needs to be computed. Then, the derivative of the last two samples (1 ADD) of the last block is subtracted from the derived extracted block (N_e ADDs). The result is sorted (sort(N_e)) to find the 10 smallest values, corresponding to the closest matches. The closest amplitude is obtained by subtracting the amplitude of the last sample of the last block (1 ADD) from the 10 candidates and again searching the maximum value by sorting (sort(10)).

   Finding the offset index l using the cross correlation (XC) is the most complex method since it requires to convolve the extracted periods with the last samples of previous data (conv(N_e)) and then find the maximum in the result by sorting (sort($2 \cdot N_e - 1$)).
5. Extrapolation: The result of the extrapolation is always cross faded with the aligned extracted block to allow smooth transitions. The fading window length is fixed and hence the actual fading consumes 8 MULs and 4 ADDs on top of every extrapolation method. The mirrored extrapolation is implemented by subtracting the inversely indexed previous samples (4 ADDs) from the last sample times 2 (1 ADD). The weighted slope continuation requires 4 ADDs to compute the slope, 4 MULs for the the weighting, and 4 ADDs for offsetting the result. The simple linear extrapolation only utilizes the last derivative of previous data (1 ADD), the linear weighting (4 MULs), plus the offset (4 ADDs). To compute the polynomial extrapolation a linear equation system with 4 unknown variables has to be solved (solve(4)). Computing the polynomial using the Horner's scheme requires $3 \cdot 3$ MULs and $3 \cdot 4$ ADDs.

Comparing the performance of the different LCC modes (Table 1) and their corresponding complexities (Table 2) reveals an extraordinary situation. The typical quality versus complexity tradeoff doesn't hold true in these measurements. The best-performing LCC alignment mode was SA which clearly outperforms the XC mode but is clearly less complex. The same holds true for the extrapolation mode. The best performing linear mode only requires a fraction of the polynomial complexity but yields better results.

To roughly determine the computation time difference of the LCC in contrast to the AR-Model, a test run over the complete SQAM test set was performed and the execution times of the corresponding conceal function calls were measured. Both concealment strategies were fed with the same data, a common block size N of 64, and error rate of 0.01. LCC, in its best working configuration, required about 233 ms to conceal a complete file in average, whereas the AR-Model computed about 1269 ms. In other words, the current implementation of LCC requires only about 18 % of the AR-Model's execution time.

5. CONCLUSIONS

The goal of this study was to find a low-cost error concealment which is suited for AoIP applications requiring lowest latency, like Distributed Network Performances, on low-power platforms, like embedded devices. The proposed system basically consists of an period extraction and alignment module. The simple extraction, based on zero-crossings and matched pre-processing, is sufficient to extract periods, which can then be aligned to prior data and concatenated to synthesize a concealment waveform. For the purpose of smooth phase transitions from previous frames into the concealment frame and back into following audio frame, cross-fades are applied. The samples, extending the previous audio frame to allow the cross-fade, are obtained by extrapolation. The audio quality of

the presented concealment system, and all of its operating modes, is evaluated using a large-scale automated test using PEAQ and the SQAM data set. The improvement of the average ODG constitutes more than 1 ODG score in comparison to muting as the simplest concealment method. The best-performing operating modes were identified using the same test setup.

Apparently, the repetition of periods from prior audio frames, combined with proper fade-in and fade-out, allows a significant quality improvement in comparison to simplest methods like muting or frame repetition. The resulting audio quality was found to be even slightly superior then for computationally demanding concealment methods based on auto-regressive modeling. The initial goal of this study, the reduction of computational complexity of concealment strategies, was accomplished since the new proposed system only consumes 18 % computation time while offering at least comparable quality.

6. REFERENCES

[1] A. Carôt and C. Werner, "Network music performance-problems, approaches and perspectives," in *Proceedings of the Music in the Global Villag*, Budapest, Hungary, 2007.

[2] C. Chafe, S. Wilson, and R. Leistikow, "A simplified approach to high quality music and sound over IP," in *Proc. of the COST G6 Conference on Digital Audio Effects (DAFx-00)*, Verona, Italy, pp. 1–5.

[3] J.M. Valin, G. Maxwell, T. Terriberry, and K. Vos, "High-Quality, Low-Delay Music Coding in the Opus Codec," in *Audio Engineering Society Convention 135*. Audio Engineering Society, 2013.

[4] J.M. Valin, T. Terriberry, C. Montgomery, and G. Maxwell, "A High-Quality Speech and Audio Codec With Less Than 10-ms Delay," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 1, Jan. 2010.

[5] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *Network, IEEE*, pp. 40–48, 1998.

[6] B.W. Wah, "A survey of error-concealment schemes for real-time audio and video transmissions over the Internet," *Proceedings International Symposium on Multimedia Software Engineering*, pp. 17–24, 2000.

[7] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod, "A new technique for audio packet loss concealment," in *Global Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity*, Nov 1996, pp. 48–52.

[8] M. Fink, M. Holters, and U. Zölzer, "Comparison of Various Predictors for Audio Extrapolation," *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*, 2013.

[9] S. Preihs, F.R. Stöter, and J. Ostermann, "Low delay error concealment for audio signals," in *Audio Engineering Society Conference: 46th International Conference: Audio Forensics*, Jun 2012.

[10] J.M. Valin, K. Vos, and T. Terriberry, "RFC 6716: Definition of the Opus Audio Codec," Website, Available online at `http://tools.ietf.org/html/rfc6716` visited on May 20th 2014.

[11] F. Meier, M. Fink, and U. Zölzer, "The JamBerry - A Stand-Alone Device for Networked Music Performance Based on the Raspberry Pi," *Linux Audio Conference 2014*, 2014.

[12] The Raspberry Pi Foundation, "Raspberry Pi Homepage," `www.raspberrypi.org`.

[13] International Telecommunication Union, "BS.1387: Method for objective measurements of perceived audio quality," Website, Available online at `http://www.itu.int/rec/R-REC-BS.1387` visited on March 7th 2014.

[14] I. Kauppinen and K. Roth, "Audio signal extrapolation - theory and applications," in *Proc. of the 5th Conference on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002.

[15] A.V. Oppenheim and R.W. Schafer, *Discrete-Time Signal Processing*, Pearson Education, Limited, 3rd edition, 2010.

[16] W. Hess, "Time-domain pitch period extraction of speech signals using three nonlinear digital filters," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '79.*, Apr 1979, vol. 4, pp. 773–776.

[17] A. Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*, A John Wiley & Sons, Inc., publication. Wiley, 2012.

[18] C. Knapp and G. Clifford Carter, "The generalized correlation method for estimation of time delay," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 24, no. 4, pp. 320–327, Aug 1976.

[19] European Broadcast Union, "EBU SQAM CD," Website, Available online at `tech.ebu.ch/publications/sqamcd` visited on March 7th 2014.

[20] Giuseppe Gottardi, " Perceptual Evaluation of Audio Quality - beta," `http://sourceforge.net/projects/peaqb/files/peaqb/1.0.beta/`.

[21] P. Kabal, "PQevalAudio," `http://www-mmsp.ece.mcgill.ca/Documents/Software/Packages/AFsp/PQevalAudio.html`.

CHARACTERISATION OF DISTORTION PROFILES IN RELATION TO AUDIO QUALITY

Alex Wilson, Bruno Fazenda

Acoustics Research Centre,
School of Computing, Science and Engineering
University of Salford
Salford, UK
a.wilson1@edu.salford.ac.uk

ABSTRACT

Since digital audio is encoded as discrete samples of the audio waveform, much can be said about a recording by the statistical properties of these samples. In this paper, a dataset of CD audio samples is analysed; the probability mass function of each audio clip informs a feature set which describes attributes of the musical recording related to loudness, dynamics and distortion. This allows musical recordings to be classified according to their "distortion character", a concept which describes the nature of amplitude distortion in mastered audio. A subjective test was designed in which such recordings were rated according to the perception of their audio quality. It is shown that participants can discern between three different distortion characters; ratings of audio quality were significantly different ($F(1,2) = 5.72, p < 0.001, \eta^2 = 0.008$) as were the words used to describe the attributes on which quality was assessed ($\chi^2(8, N = 547) = 33.28, p < 0.001$). This expands upon previous work showing links between the effects of dynamic range compression and audio quality in musical recordings, by highlighting perceptual differences.

1. INTRODUCTION

While a single, consistent definition for quality has not yet been offered, it has an accepted meaning when applied to certain restricted circumstances, such as audio reproduction systems. Measurement standards exist for the assessment of audio quality [1, 2] however such techniques typically apply to the measurement of quality with reference to a golden sample; what is in fact being ascertained is the perceived reduction in quality due to destructive processes. One such example is in the case of lossy compression codecs in which the audio being evaluated is a degraded copy of the reference and the deterioration in quality is measured [3].

This study is concerned with the audio quality of "produced" music where there is no fixed reference and quality is evaluated by comparison with all other samples heard. In this manner, "audio quality" is perhaps better related to "product quality", as considered in consumer research, food science and sensory profiling in general. In these cases quality is based on multi-modal perception - partly influenced by objective parameters, such as sugar level in drinks, and partly by issues such as branding and packaging [4].

The assessment of audio quality in musical recordings, especially that of popular music, is therefore thought to be based on both subjective and objective considerations. The weighting of these two factors can vary by individual, depending on experience and expertise [5].

1.1. Signal statistics

The distribution of sample amplitudes in digital audio signals has been shown in a number of previous studies - such works have displayed the probability mass function (PMF) for a number of digitised recordings of popular music. A PMF shows the probabilities of a discrete random variable occurring at discrete values. Particular characteristics can be observed in the PMF, such as clipping of the signal and errors in the analogue-to-digital conversion [6]. Often, this distribution is represented as an "amplitude histogram", where bins are chosen based on decibel increments [7, 8]. Some summary features have been suggested [9, 10], however such a logarithmic approach lacks the required detail in high-amplitude values. A detailed investigation of high-amplitude distributions is particularly relevant due to the fact that signal levels increased in recent decades, in what is often described as a "loudness war" [11].

1.2. PMF distortion

Throughout the literature there is rarely much attempt to analyse this distribution in the required detail and provide summary features. Previous work by the authors has provided one such summary feature of the PMF, which relates to the level of distortion and the perception of audio quality [5].

Hard-limiting and dynamic range compression have been studied in relation to listener preference [12, 13]. Since these parameters are encompassed by the PMF of an audio segment, the previous study by the authors attempted to gather them into a higher-level feature. Since the PMF describes many possible states (here, it is the 2^{16} quantisation levels in a 16-bit audio signal), a histogram was generated with 201 bins, providing a good balance of runtime, accuracy and clarity of visualisation. In order to evaluate the shape of the distribution, particularly the slope and the presence of localised peaks, the first derivative was determined. For the ideal distribution this had a near-Gaussian form so the goodness-of-fit to a Gaussian distribution was obtained for each sample (ratio of the sum of squares of the regression and the total sum of squares). This was used as a feature describing loudness, dynamic range and related distortions, referred to previously and herein as 'Gauss'.

This feature was shown to relate significantly to subjective impressions of audio quality, when participants were asked to rate "the audio quality of the sample" [5]. While this feature can distinguish between distorted and non-distorted audio signals and give an approximation of the amount of distortion, the difference between different types and causes of distortion is not clear from this feature alone. This paper will describe a method which can be used to determine other aspects of distorted audio in addition to level.

2. TYPICAL DISTORTIONS IN MASTERED AUDIO

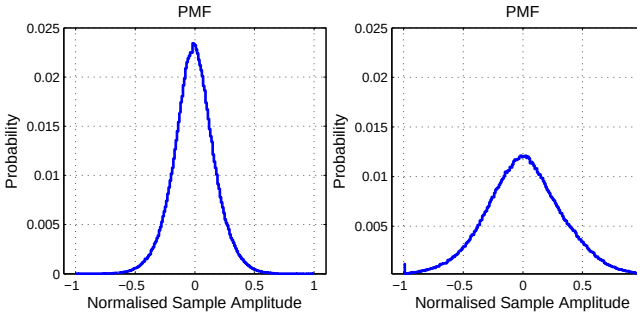
An examination of commercial music samples was undertaken in order to identify typical outputs of the mastering process and its visible imprints on the PMF. The nature of the sample amplitude distribution is influenced by the aforementioned loudness war, in which the perceived loudness of digital music signals has increased since the launch of the CD in 1982, at the expense of reduced micro-dynamics, achieved using dynamic range compression [10]. Despite the popular term, this may be thought of more as a "loudness race", as this increase takes place primarily in the 1990's and has remained at an escalated level since, in a state of *détente*.

Shown in Figure 1 is the PMF for a selection of audio samples. While the area under the curve is identical by definition, the shape varies. Figure 1a shows a distribution which is typical of its time. Due to the nature of its dynamic range, a distribution of this shape is often considered to be an ideal, neutral distribution, in relation to issues of loudness and dynamic range compression.

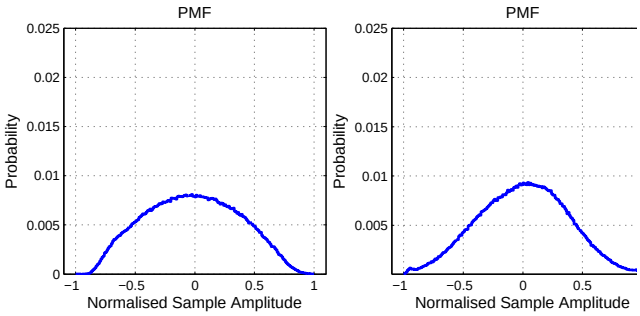
While hard-clipping of the waveform becomes increasingly popular during this "loudness race", as in Figure 1b, it becomes less common in recent years. Other PMF distributions have become popular, featuring a similar loudness level while avoiding hard-clipping. This can be achieved in a number of ways, one of which is to apply limiting to individual instrument groups during the mix process, or the use of multi-band limiters in the mastering chain. The awareness of inter-sample peaks has also lead some engineers to avoid the implementation of hard-clipping [14].

If a mix has been clipped the subsequent processing in the mastering stage, including equalisation, further dynamics processing, stereo-enhancement and downsampling from high sample rates, can cause this clipping to be spread out over a wider amplitude range, in regions around the maximal values, as in Figure 1d.

Figure 1c shows an example of a distribution highly warped in comparison to typical distributions and therefore the Gauss value is very low. Distortion, across the full mix, is evident on audition. It was worth noting that this album involved the same producer, mix-engineer and mastering-engineer as 'Death Magnetic', the 2008 album by Metallica which was responsible for a popular, if at times ill-informed, backlash against the loudness war [15, 16]. This demonstrates how a team of engineers can impart a distortion characteristic on productions and that this characteristic can be identified by listeners.



(a) *Europe, 'The Final Countdown', 1986* (b) *Prodigy, 'Firestarter', 1996*



(c) *Slipknot, 'Duality', 2004* (d) *Katy Perry, 'Roar', 2013*

Figure 1: *Examples of probability mass functions for digital music, showing a variety of production outcomes*

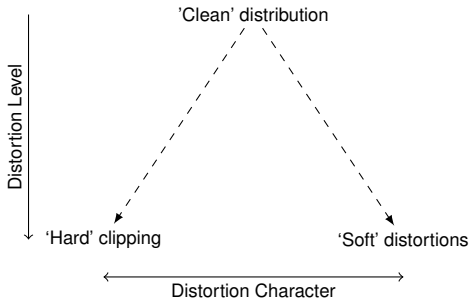


Figure 2: *2D distortion paradigm demonstrating three discrete characters*

2.1. Distortion character

It becomes apparent that hard-clipping is one of a number of possible outcomes when attempting to maximise the perceived loudness of digital music signals. Incorporating this distortion type into a two-dimensional paradigm with distortion amount introduces the notion of distortion character, as illustrated in Figure 2.

This is referred to as distortion character by the authors since the problem is essentially one of character recognition. For example, while the letter W can be defined as such, **W** and $\mathcal{W}$ are still recognised as equivalent symbols. In this case, the shape of the PMF curve is the 'character' in the problem and comparable PMFs are considered to contain similar amplitude characteristics.

2.2. Test hypotheses

This paper describes an investigation into the perception of different distortion profiles in relation to audio quality. In constructing a subjective test and its subsequent analysis, the following null hypoth are used as a basis for the design.

- #1 There is no difference in quantitative quality ratings of the different audio clips.
- #2 There is no difference in quantitative quality ratings of the different distortion character groups.
- #3 There is no difference in how words are used to describe the quality ratings of different distortion character groups.

Test hypothesis #1 was rejected in previous work by the authors featuring a similar experiment [5] but stands as the basic null hypoth in this work.

2.3. Audio dataset

The dataset of audio examples is comprised of 321 songs by 229 different bands or artists. There is a mean of ten samples from each year between 1982 and 2013. The clips used for analysis are 20-second excerpts centred about the second chorus.

The audio is being collected as part of a larger study into the nature of quality-perception. While other studies have included digital audio files representing music from earlier periods [8, 9, 10] there is usually not much explanation as to how they have been sourced. In particular, what is often not acknowledged is that samples which represent music made prior to the advent of digital media would have been remastered for release on digital media. When these remasters were made is important; remastered audio does not typically retain the amplitude characteristics of the original release. To address this issue, only audio from original CD releases is considered here, i.e. from 1982 onwards.

There are only two samples from 1982, due in part to the fact that many of the earliest CD releases were re-issues of material recorded in previous years. Both of these releases feature an emphasis system, designed to compensate for deficiencies in the A/D conversion process, which at this time was based on earlier, 14-bit technologies. The signal had been subject to pre-emphasis, and de-emphasis was to be performed on-board the player. For this study, a de-emphasis filter was designed in order to retrieve representative amplitude statistics for any samples featuring pre-emphasis. Based on an available circuit analysis [17], the filter was an IIR design, constructed by use of the Yule-Walker method.

2.4. Labelling of distortion character

For simplification, only three categories of distortion character are considered - clean, hard and soft, as in Figure 2. The clean and hard categories are quite well-defined analytically, however the soft character is a set of PMF shapes having high dynamic range compression but without hard-clipping, such as Figure 1d, where small deformations in the PMF can be seen just below the extreme levels. Two options were available for labelling the dataset;

1. The samples could be labelled analytically, since hard clipping, or lack thereof, can be determined by the values of the PMF in its extreme values, after normalisation.
2. The samples could be labelled by an expert panel, by simultaneous audition of the signal and visual inspection of the PMF.

Method 2 was used due to the subjective nature of the problem and the fact that method 1 makes assumptions about the nature of the soft distortion character, which is harder to categorise. As a result of this labelling approach a classifier was designed to blindly label samples, as learned by the initial classification of the expert panel.

3. CLASSIFICATION

3.1. Feature extraction

The designing of such a classifier, in this case, has two objectives.

1. To label unseen samples with the appropriate distortion character, using a consistent metric
2. To provide information on which objective features were used to perform this labelling

Table 1: *Features used in objective analysis*

Feature	*Description*
Crest factor	Ratio of peak amplitude to RMS amplitude
Loudness	According to ITU BS. 1770-3 [19]
Top1dB	Proportion of samples between 0dBFS and -1dBFS
Rolloff	Frequency at which 85% of spectral energy lies below [20]
Harsh energy	Fraction of total spectral energy contained within 2k-5kHz band
LF energy	Fraction of total spectral energy contained within 20-80Hz band
MIRemotion	Objective predictions of emotional response - Happy, Sad, Tender, Anger, Fear, Activity, Valence, Tension [21]
PMF	Evaluated as a histogram with 201 bins - see Section 1.2
Centroid	First moment of PMF
Spread	Square root of second moment of PMF
Skewness	Third moment of PMF
Kurtosis	Fourth standardised moment of PMF
Flatness	Ratio of geometric mean and arithmetic mean of PMF
PMF_d	First derivative of PMF
Gauss	Measure of distortion in PMF_d feature [5]

Table 1 shows features which were extracted from each sample in order to train the classifier. These are mainly amplitude-based features due to the nature of the problem. The evaluation of certain features was aided by the MIRtoolbox [18].

3.2. Classifier design

Statistical analysis was aided by the use of Orange, a data-mining toolbox for Python [22]. Orange can also be used as a visual programming environment and this was used for prototyping. Based on this prototyping stage, the decision was made to use support vector machines (SVM) for classification. This decision was made as it is a well-known method which can address both aims of the classifier, as mentioned in section 3.1, and as described below.

3.2.1. Optimisation

For optimisation and reproducibility, the final classifier was also implemented using Orange but with the Python-scripting interface. The SVM implementation in this package is that of LIBSVM [23]. As the initial set of features extracted contains over 400 features, this number was reduced by means of recursive feature elimination (RFE) [24]. The algorithm is described below.

1. A list of features is provided and a linear SVM is obtained.
2. The features are ranked according to their weights in the SVM solution.
3. The lowest-ranked feature is removed from the list.
4. Repeat these steps until desired number of features remain.

This algorithm was used to return the ten features most relevant to distortion character classification from the initial set shown in Table 1. All 321 audio clips were used in this analysis. The features found to be most important in classifying distortion character

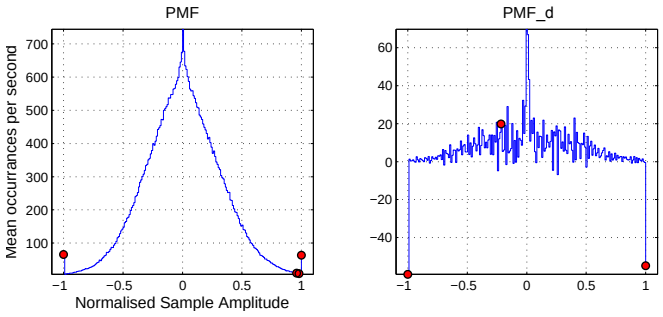


Figure 3: *PMF and (PMF)′ feature sets, in which the features important to distortion character classification are highlighted by circles*

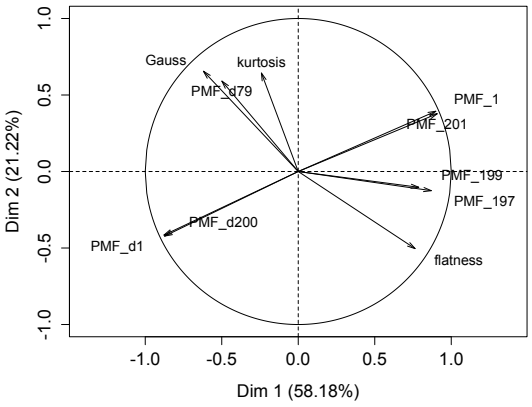


Figure 4: *Correlation of features with principal components*

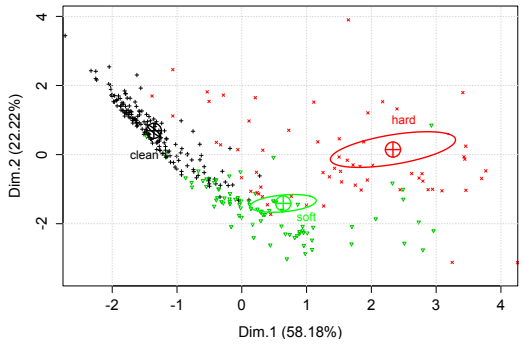


Figure 5: *Individuals factor map, showing the distribution of the three different groups*

were the following: gauss, kurtosis, flatness, the 1st, 197th, 199th and 201st elements of PMF and the 1st, 79th and 200th elements of PMF_d. Features that are a subset of PMF and PMF_d are shown in Figure 3, highlighted in PMF and PMF_d of audio which displays clipping, as evident from the values of these features.

A new SVM implementation was created, using a multi-class configuration. The parameters of the SVM are automatically optimised using LIBSVM's procedures [23].

3.2.2. Performance

This data was randomly divided into two portions; 50% for training and 50% for testing. The trained classifier was tested using 10-fold cross-validation and achieved a classification accuracy of 0.795, with area under ROC curve of 0.888. The confusion matrix for this test is shown in Table 2.

Table 2: *Confusion matrix showing performance of trained classifier on test dataset of 161 samples*

		Predicted			
		clean	hard	soft	*recall*
Real	clean	73	5	4	0.89
	hard	2	28	5	0.80
	soft	5	12	27	0.61
precision		0.91	0.62	0.75	

Both recall and precision is greatest for the 'clean' category. This indicates that there is a conformity between these examples and, as such, they can easily be recognised.

Recall is high for the 'hard' category, as this clipping is recognised easily by the PMF_d features (see Figure 3). However, precision is lower, as samples with hard clipping may have any other general PMF shape as identified by the gauss, kurtosis and flatness features. This leads to misclassification into this category.

Similarly, recall is low for the 'soft' class as this category is composed of a collection of PMFs that could not more easily be labelled as either of the two other groups. Precision is still rather high, as other groups are unlikely to be misclassified as members of this category. The most common misclassification is that of 'soft' as 'hard', likely due to the lack of conformity in the 'soft' group and reasons described above.

Table 3: *Variance explained by first five dimensions*

Dim.	1st	2nd	3rd	4th	5th
% var.	58.18	21.22	7.76	5.72	3.50
Cumulative % var.	58.18	79.39	87.15	92.87	96.37

3.3. Principal component analysis

In order to investigate how the ten identified features vary across the proposed three characters, a Principal Component Analysis (PCA) is performed. This yields dimensionality-reduction; the ten dimensions can be reduced to a combination of orthogonal functions which explain as much of the original variance as possible. PCA is performed with the 'FactoMineR' package, using **R**.

58.18% of the variance in the data is explained by Dim.1, which relates mainly to features associated with hard-clipping. The first two dimensions account for 80%, with Dim.2 describing the kurtosis and general 'peakiness' of the distribution. The variance of each dimension and cumulative variance is shown in Table 3.

Figure 5 shows the individual audio samples, grouped by distortion character. The centroid of each group is shown, with ellipses representing 90% confidence. This shows that each group is distinctly defined in this space. The axis limits were chosen for clarity; some outliers are not visible.

4. RELATION TO SUBJECTIVE IMPRESSION OF AUDIO QUALITY

4.1. Test design and execution

Of these 321 audio samples which were analysed, 62 were used in a listening test in which participants were asked to report their impression of the quality of the recording. This was assessed using the following instructions for each sample.

1. How do you rate the audio quality of this sample?
2. Please choose **two** words which describe the attributes on which you assessed the audio quality.

Participants rated the audio quality of each sample on a 5-point scale, with 5 as highest. For question 2, participants were provided with a list of commonly used terms as a reference but were encouraged to provide their own terms. The list of words provided is shown in Appendix A. In this paper, the frequencies of these words are used in a *post-hoc* investigation of the perception of the three different distortion characters, i.e. to avoid bias, participants were not made aware of the distortion character concept prior to, or during, the listening test.

The test took place in the listening room at University of Salford, a room which conforms to BS.1116-1 [1]. Audio was delivered via Sennheiser HD 800 headphones, the frequency response of which was measured using a Brüel & Kjær Head and Torso Simulator (HATS). Low-frequency rolloff in the response below 110Hz was compensated for using an IIR filter designed using the Yule-Walker method. This then facilitated the addition of a notch filter at 0Hz.

The loudness of all audio samples was normalised, according to current broadcast standards, after headphone compensation had taken place [19]. The presentation level to participants was 82dB(A), as measured using the HATS and sound level meter.

One additional clip was added to the beginning of each test to serve as a trial. A short break was automatically suggested when 40% of trials had been completed. Post-experiment discussion was typically led by the participant and offered valuable insight.

4.2. Participant demographics

The total number of participants was 22 (4 female, 18 male), tested over a period of five days. The participants were 13 experts and 9 non-experts, which was self-reported, based on their level of academic or professional experience in fields relating to acoustics and audio. The mean age of participants was 24.2 years (std.dev = 4.5 years), varying from 19 to 39. Participants were asked to indicate their preferred musical genres and it was observed that the participants had diverse musical tastes.

Test duration varied by participant, with a mean value of 40 minutes (std.dev = 11 minutes). As this contained the option of a short break, the effect of fatigue on the reliability of subjective quality ratings was considered to be negligible, according to suggested guidelines [25].

4.3. Results

With 63 audio samples and 22 subjects, these 1386 auditions were gathered and analysis was performed on this dataset. As this test was also concerned with variables outside the scope of this paper, an n-way ANOVA was performed. This revealed a significant effect of the variables relevant to this paper, in terms of the influence on quality ratings, which were investigated further.

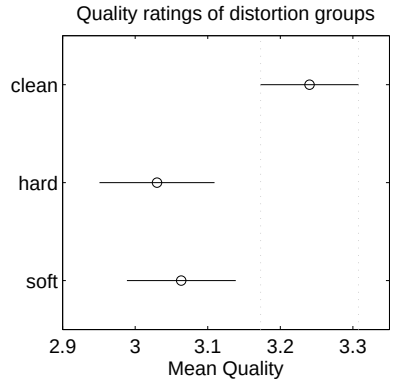


Figure 6: *1-way ANOVA: Quality, grouped by distortion character*

A one-way ANOVA was performed with *post-hoc* multiple comparison and Bonferroni adjustment applied. As shown in Figure 6, the mean quality rating is higher for the 'clean' category compared to the other two, while 'hard' and 'soft' distortion categories are rated similarly ($F(1,2) = 5.72, p = 0.00, \eta^2 = 0.008$). This provides evidence in support of rejecting test hypothese #1 and #2, however the effect size is considered to be small, as $\eta^2 < 0.01$. This is influenced by the narrow use of the scale and contributions from other variables, as seen in earlier tests [5].

4.3.1. Analysis of words used to explain quality ratings

While the provided list contained 41 words, in total, 255 words were used over the course of the 1386 unique auditions, after spelling had been corrected and equivalent terms collated (for example, 'compressed' and 'over-compressed' were equivalent in this context). In this lexicon, many words are not used often, some being unique to a single participant. While this study is comparatively small, connections between a words frequency and rank in a frequency table are found in other studies of linguistic corpora [26].

Figure 7 shows word clouds of the terms used to describe the participants' quality ratings, generated using **R**, along with the packages 'tm' [27] and 'wordcloud'. The five most frequently occurring terms are shown in Table 4 and account for 19.7% of all descriptions requested. In order to determine if there was significant variation in the frequency of each term across the three categories, a Chi-Square analysis was performed. The words chosen to describe the quality of each distortion character differed significantly ($\chi^2(8, N = 547) = 33.28, p = <.001$). This data provides evidence in support of rejecting test hypothesis #3. In Table 4, frequencies highlighted in bold (with '>' or '<') are either significantly greater than (>) or less than (<) the expected counts.

		Groups			
		Clean	Hard	Soft	TOTAL
Words	*Distorted*	**21<**	**47>**	**59>**	127
	Punchy	53	37	**34<**	124
	Clear	49	30	45	124
	Full	29	28	30	87
	Harsh	**42>**	20	23	85

Table 4: *Frequency count (Chi square test analysis) of five most commonly used words*

(a) *'clean' category* (b) *'hard' category* (c) *'soft' category*

Figure 7: *Word clouds, showing the most used terms for each category. Larger/darker text indicates greater frequency.*

5. DISCUSSION

5.1. Feature reduction

The RFE process returned the most important features for classification. Perhaps unsurprisingly, all of these features are directly associated with the PMF. Other features such as crest factor or loudness (see Table 1) are indirectly encoded in the PMF, while spectral features were ranked lower. The emotion features 'Happy' and 'Anger' [21] were found to relate to audio quality in a previous study [5] and were included as they encode amplitude information. However these features were not highly ranked in this case.

The features found to be most useful are those bins of the PMF histogram close to extreme values which can detect the presence or absence of clipping, the gauss feature which can discriminate the 'clean' samples from the other groups, and kurtosis and flatness which can help to isolate the 'soft' category. PCA results in Figure 5 show that the three categories are well-defined by these features.

5.2. Production trends

Figure 8 shows the proportion of samples in each year of analysis which belong to each of the three distortion character categories. The scattered data is smoothed by local regression using weighted linear least squares and a second-degree polynomial model method, with rejection of outliers. As there is an average of ten audio clips per year, this data can be seen as illustrative rather than conclusive. From this plot, a number of discussion points are evident.

1. The transition from more dynamic signals to more compressed signals occurs throughout the 1990's.
2. The percentage of 'clean' samples has remained stable since.
3. Recent years have seen a move away from hard clipping, and towards the use of softer distortions in the final master.

Historical analysis of the gauss feature has shown that values range from an "early digital" phase to a "modern digital" phase, via a transition phase [5], referred to earlier as a loudness race. When distortion character is taken into consideration, a similar three-stage effect is observed in the clean category.

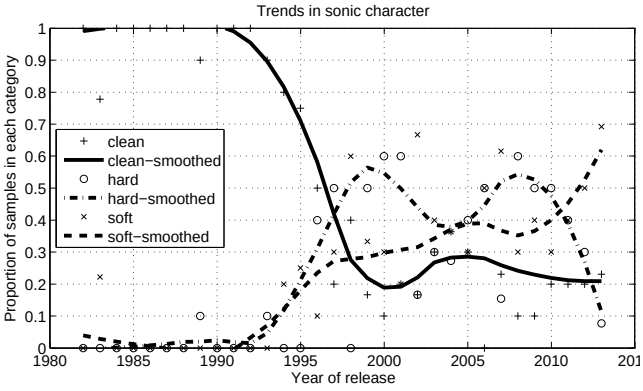


Figure 8: *Timeline showing changes in production trends and relative usage of each distortion character*

5.3. Differences in perception of each distortion character

While participants rated the quality of the clean samples higher than both distorted groups, there was no significant difference found between the two individual distorted groups. There was however, a difference in the way quality was perceived, as the distribution of descriptive terms varies between categories. From Figure 7 it can be seen that the number of words used for the 'clean' category is higher, whereas the word clouds of the other two categories are dominated by a small number of terms. A discussion of the influences of these words is provided below.

5.3.1. Distortion

'*Distorted*' was the most frequently occurring word overall. This indicates it is easily recognised by listeners and is a primary descriptor of quality, or lack therof. Table 4 shows it is used less than statistically expected by chance alone to describe the 'clean' category and more so for both other categories. This indicates that the three distortion characters do represent different levels of distortion, as illustrated in Figure 2 and Figure 5.

5.3.2. Punch and clarity

The adjectives '*punchy*' and '*clear*' are two of the most frequently used terms throughout, however they are used in varying amounts depending on the distortion character. This suggests the relative importance of such terms and also how they may be measured objectively, a task that has been investigated in recent literature [28].

'*Clear*' is used relatively less often for the examples with hard-clipping, although it is not significant. That the frequency of '*punch*' is lower for the soft category may simply be a result of other words being used more frequently. However, objectively, hard clipping would result in inter-sample peaks in subsequent stages of amplification and reproduction which could be interpreted as additional dynamic range, whereas this effect would not be so great for the 'soft' category.

5.3.3. Harshness and Fullness

The description '*full*' was used often but there is little variation in use across these three groups. This indicates that when participants used the word to explain why a particular numerical quality rating was awarded, the decision was not concerned with the distortion character but other factors. '*Harsh*' was often used by participants and there are a number of possible explanations for this.

- Participants' sensitivity to the headphones used, the response of which may have sounded unfamiliar to some participants
- This word was used more often for the 'clean' category, which is the dominant distortion character for the older material used. Changes in the typical spectrum of music recordings since this period may have had an influence [29].
- Additionally, under loudness-normalisation, the more dynamic nature of the 'clean' samples results in higher peak volumes and a transient response that some listeners may be less accustomed to. This is effectively the opposite of the common complaint among audiophiles that compressed music sounds flat and lifeless under loudness-normalised conditions.

5.4. Side-effects of loudness normalisation

This last point came up in post-experiment discussion with some participants and also in certain words used to describe particular songs. A smooth jazz sample was described as '*compressed*' and '*distorted*' by a number of participants (one using the term '*over-compressed*') as, when played at the same perceived loudness as other samples, it sounded unnaturally loud. Also, as the sample featured very subtle percussion the crest factor was lower than its distortion character would suggest. These issues indicate that, with loudness-normalisation, choosing a playback volume that does not bias against any one particular distortion character is difficult.

6. CONCLUSIONS

This work has set out to investigate whether commercial music samples can be categorised according to distortion level and type and does this categorisation further the understanding of audio quality in the context of modern commercial music.

It has been seen that the concept of a distortion character, informed by subjective perception, relates to certain objective measures of the PMF, namely particular regions as dictated by certain bins of the histogram, as well as summary features such as the statistical moments. The quantitative and qualitative aspects of quality ratings varied significantly for the three groups. This relationship between distortion character and quality ratings can contribute towards the understanding of quality-perception in the context of recorded music as well as inform attempts at evaluating the quality of an unknown audio stream.

6.1. Further work

6.1.1. Application to greater bit depth

These findings apply to digital audio with a bit-depth of 16 bits and a sampling rate of 44.1kHz. The ratio of quantisation levels to samples per second is 1.49:1 and this allows the PMF to be sufficiently non-sparse. For 24-bit resolution it would take a sampling rate of 11.25MHz to achieve the same ratio of quantisation levels to sampling rate. Thus, many distortion artefacts present in the 16-bit PMF will take a different form in systems with a greater bit-depth. To the authors knowledge, there have not yet been published studies analysing 24 or 32-bit digital audio in this manner, where even at the highest sampling rates commonly used, the PMF would be highly sparse. Further work would involve testing the methods described in this paper on 24-bit audio where discrete sample level counts are close to zero or equal to zero. The PMF of 32-bit audio, with 2^{32} levels, is likely to be prohibitively large to be able to study a large dataset of audio samples with the techniques described. New techniques are currently under trial.

6.1.2. Modelling distortion in mastered music

While clipping is well defined and evident in waveforms of mastered audio, soft distortions vary in their complexity, with varying dynamic and harmonic stability [30]. Further work is required to determine whether such analytical models can be used to describe how soft distortion appears in mastered commercial releases.

6.1.3. Quality-prediction

This study indicates that distortion character may not contribute greatly to a solely objective model of audio quality but does indicate that subjective elements, such as perceived punch, clarity and harshness, can provide useful information. Regarding the study of perceived audio quality in commercial music, the results related to dynamic range compression will be added to findings in other areas, such as overall balance of instruments and emotional response of the listener, to give a wider picture of how we understand audio quality in this context.

7. REFERENCES

[1] ITU-R BS.1116-1, "Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems," Tech. Rep., International Telecommunications Union, 1997.

[2] ITU-T P.800, "Methods for objective and subjective assessment of quality," Tech. Rep., International Telecommunications Union, 1996.

[3] Amandine Pras, Rachel Zimmerman, Daniel Levitin, and Catherine Guastavino, "Subjective Evaluation of mp3 compression for different musical genres," *Audio Engineering Society Convention 127*, 2009.

[4] M. Ng, C. Chaya, and J. Hort, "The influence of sensory and packaging cues on both liking and emotional, abstract and functional conceptualisations," *Food Quality and Preference*, vol. 29, no. 2, pp. 146–156, Sept. 2013.

[5] Alex Wilson and Bruno Fazenda, "Perception & evaluation of audio quality in music production," in *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, 2013, pp. 1–6.

[6] Eric Benjamin, "Characteristics of musical signals," *Audio Engineering Society Convention 97*, vol. 4, 1994.

[7] Miomir Mijić, Draŝko Maŝović, Dragana Ŝumarac Pavlović, and Milan Petrović, "Statistical properties of music signals," in *Audio Engineering Society Convention 126*, May 2009.

[8] Joan Serrà, Alvaro Corral, Marián Boguñá, Martín Haro, and Josep Ll Arcos, "Measuring the evolution of contemporary western popular music.," *Scientific reports*, vol. 2, pp. 521, Jan. 2012.

[9] D Tardieu, E Deruty, C Charbuillet, and G Peeters, "Production effect: audio features for recording techniques description and decade prediction," in *Proc. of the 14th Int. Conference on Digital Audio Effects (DAFx-11)*, 2011, pp. 441–446.

[10] Emmanuel Deruty and Damien Tardieu, "About Dynamic Processing in Mainstream Music," *Journal of the Audio Engineering Society*, vol. 62, no. 1, 2014.

[11] Earl Vickers, "The loudness war: Background, speculation, and recommendations," *Audio Engineering Society Convention 129*, pp. 1–27, 2010.

[12] Naomi B H Croghan, Kathryn H Arehart, and James M Kates, "Quality and loudness judgments for music subjected to compression limiting.," *The Journal of the Acoustical Society of America*, vol. 132, no. 2, pp. 1177–88, Aug. 2012.

[13] T.J. Cox, B.M. Fazenda, S. Groves-Kirkby, I.R. Jackson, P. Kendrick, and F. Li, "Quality, timbre and distortion: perceived quality of clipped music," in *Proc. Institute of Acoustics - Conference on Reproduced Sound 2013. Manchester, UK*. November 2013, Institute of Acoustics (UK).

[14] Søren H Nielsen and Thomas Lund, "Level control in digital mastering," in *Audio Engineering Society Convention 107*. Audio Engineering Society, 1999.

[15] Ethan Smith, "Even heavy-metal fans complain that today's music is too loud!!!," *Wall Street Journal*, 25 September 2008. Available: http://online.wsj.com/news/articles/SB122228767729272339, accessed 18 March 2014.

[16] S. Michaels, "Death magnetic 'loudness war' rages on," *The Guardian*, 1 October 2008. Available: http://www.theguardian.com/music/2008/oct/01/metallica.popandrock, accessed 18 March 2014.

[17] Gary Galo, "A De-Emphasis Test CD," http://www.audioxpress.com/files/2009/02/galo3025.pdf, accessed December 27, 2013.

[18] Olivier Lartillot and Petri Toiviainen, "A matlab toolbox for musical feature extraction from audio," in *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10-15, 2007, pp. 237–244.

[19] ITU-R BS.1770-3, "Algorithms to measure audio programme loudness and true-peak audio level," Tech. Rep., International Telecommunications Union, 2012.

[20] George Tzanetakis and Perry R. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[21] Tuomas Eerola, Olivier Lartillot, and Petri Toiviainen, "Prediction of multidimensional emotional ratings in music from audio using multivariate regression models," in *International Conference on Music Information Retrieval*, Kobe, Japan, Oct. 26-30, 2009, pp. 621–626.

[22] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan, "Orange: Data mining toolbox in python," *Journal of Machine Learning Research*, vol. 14, pp. 2349–2353, 2013.

[23] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, Mar. 2002.

[25] Raimund Schatz, Sebastian Egger, and Kathrin Masuch, "The impact of test duration on user fatigue and reliability of subjective quality ratings," *Journal of the Audio Engineering Society*, pp. 63–73, 2012.

[26] George Kingsley Zipf, *Human behaviour and the principle of least effort*, addison-wesley press, 1949.

[27] Ingo Feinerer, Kurt Hornik, and David Meyer, "Text mining infrastructure in R," *Journal of Statistical Software*, vol. 25, no. 5, pp. 1–54, March 2008.

[28] S Fenton and J Wakefield, "Objective profiling of perceived punch and clarity in produced music," *Audio Engineering Society Convention 132*, pp. 1–15, 2012.

[29] PD Pestana, Z Ma, and JD Reiss, "Spectral Characteristics of Popular Commercial Recordings 1950-2010," *Audio Engineering Society Convention 135*, pp. 1–7, 2013.

[30] Sean Enderby and Zlatko Baracskai, "Harmonic instability of digital soft clipping algorithms," in *Proc. of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, York, UK, September 2012.

A. APPENDIX 1 - LIST OF WORDS PROVIDED TO PARTICIPANTS

Bright, dark, loud, quiet, mellow, clear, clean, punchy, dull, bland, dense, exciting, weak, strong, sweet, shiny, fuzzy, wet, dry, distorted, realistic, spacious, narrow, wide, deep, shallow, aggressive, light, gentle, cold, hard, synthetic, crunchy, hot, rough, harsh, smooth, thin, full, airy, big

Author Index

ISBN: 978-3-00-046825-4