

Montréal, Québec, Canada
September 18–20, 2006

Proceedings
of the 9th International Conference
on Digital Audio Effects

DAFx-06 was organized by
the Sound Processing and Control Laboratory
Music Technology Area
and CIRMMT



McGill



Schulich School of Music
École de musique Schulich



Centre for Interdisciplinary Research
in Music Media and Technology

Published by:

McGill University
Schulich School of Music
Music Technology Area

ISBN: 0-7717-062705

Credits:

Cover design: Cédric Van Eenoo
Logo photo: Aurore Corominas
Cover photo, logo design: Vincent Verfaille

Printed in Montreal by repro-UQAM — August 2006

Copies may be ordered from:

Prof. Philippe Depalle
Music Technology Area
Schulich School of Music
McGill University
Email: info@dafx.ca

All copyrights remain with the authors.

DAFx-06 homepage: <http://www.dafx.ca>

Local Organizing Committee

Schulich School of Music – McGill University

Philippe Depalle Conference Chair, Paper Selection
Vincent Verfaille General Management, Proceedings Edition

Sean Ferguson Concert Organization

David Birnbaum	Registrations
Julien Boissinot	Technical Management
Darryl Cameron	Technical Support
Joe Malloch	Technical Support
Bertrand Scherrer	Websites
Andrey da Silva	Papers Edition
Doug Van Nort	Local Arrangements
Marcelo M. Wanderley	Paper Selection

DAFx Conferences

DAFx is an acronym for **digital audio effects**, and refers nowadays to the name of the International Conference on Digital Audio Effects and to the related book ‘*DAFx – Digital Audio Effects*’ edited by Udo Zölzer. It was initiated from a European research project for cooperation and knowledge diffusion (EU-COST-G6 “Digital Audio Effects”, 1997-2001). Since then DAFx has been running on its own feet as a self funded event. Papers of all the conferences are available online at:

DAFx98	Barcelona, Spain	Nov. 19–21, 1998	http://www.iua.upf.es/dafx98/
DAFx99	Trondheim, Norway	Dec. 9–11, 1999	http://www.notam02.no/dafx99/
DAFx-00	Verona, Italy	Dec. 7–9, 2000	http://profs.sci.univr.it/~dafx/
DAFx-01	Limerick, Ireland	Dec. 6–8, 2001	http://www.csis.ul.ie/dafx01/
DAFx-02	Hamburg, Germany	Sept. 26–28, 2002	http://www2.hsu-hh.de/ant/dafx2002/dafx2002.html
DAFx-03	London, UK	Sept. 8–11, 2003	http://www.elec.qmul.ac.uk/dafx03/
DAFx'04	Naples, Italy	Oct. 5–8, 2004	http://dafx04.na.infn.it/
DAFx'05	Madrid, Spain	Sept. 20–22, 2005	http://dafx05.ssr.upm.es/
► DAFx-06	Montreal, Quebec, Canada	Sept. 18–20, 2006	http://www.dafx.ca/
DAFx-07	Bordeaux, France	Sept. 2007	http://www.dafx.u-bordeaux.fr/
DAFx-08	Helsinki, Finland		2008
DAFx-09	Como / Milan, Italy		2009
DAFx	general website		http://www.dafx.de/

Scientific Committee

Daniel Arfib	CNRS-LMA, Marseille, France
Nicola Bernardini	Media Innovation Unit — Firenze Tecnologia, Florence, Italy
F. Javier Casajús	ETSI Telecommunicación — Universidad Politécnica de Madrid, Spain
Laurent Daudet	Laboratoire d’Acoustique Musicale, Université Pierre et Marie Curie (Paris VI), France
Philippe Depalle	McGill University, Montreal, Canada
Giovanni De Poli	University of Padova, Italy
Markus Erne	Scopein Research, Aarau, Switzerland
Gianpaolo Evangelista	Federico II University of Naples, Italy
Emmanuel Favreau	Institut National de l’Audiovisuel, Paris, France
Simon Godsill	Cambridge University, UK
Robert Höldrich	Institute of Electronic Music and Acoustics, Graz, Austria
Jean-Marc Jot	Creative Labs, USA
Sylvain Marchand	Université Bordeaux 1, France
Sören Nielsen	TC Electronics, Denmark
Luis Ortiz	EUIT Telecommunicación — Universidad Politécnica de Madrid, Spain
Rudolf Rabenstein	Erlangen-Nuremberg University, Germany
Davide Rocchesso	University of Verona, Italy
Jöran Rudi	NOTAM, Oslo, Norway
Mark Sandler	Queen Mary, University of London, UK
Xavier Serra	Pompeu Fabra University, Barcelona, Spain
Julius O. Smith	Stanford University, USA
Todor Todoroff	ARTeM, Bruxelles, Belgium
Soledad Torres	ETSI Telecommunicación — Universidad Politécnica de Madrid, Spain
Jan Tro	Norwegian University of Science and Technology, Trondheim, Norway
Udo Zölzer	Helmut-Schmidt University, Hamburg, Germany

Welcome from Don McLean, Dean of the Schulich School of Music

As Dean of the Schulich School of Music of McGill University and Chair of the Board of CIRMMT (the Centre for Interdisciplinary Research on Music Media and Technology) it gives me great pleasure to welcome participants in the 9th International Conference on Digital Audio Effects (DAFx-06) to Montreal and to McGill. As I prepared this note, having taken it through that familiar process of moving it from the next to-do list to the ought-to-have-been-done list, I was subject to my own Fx moment: the power went out. Our family tries to spend at least some time each summer at our cottage in Georgian Bay (part of Canada's Great Lakes system). The area is subject to extraordinary summer storms, one of which took down the regional power system for a couple of days. Beyond the usual opportunity this gave to reflect on our dependence on power, particularly computerized power, for most of our work, it also allowed the sounds of wind and surf, in all their extraordinary complexity of detail to emerge afresh from a background bereft of its normal electro hum and babble.

This is of course why we all signed up: the challenges of modeling and manipulating sound for scientific and artistic purposes are always inspired and reinvigorated by the beauty and sophistication of sound itself. And it is one of our collective goals to focus public attention on the value and quality of the sound around us, both natural and synthetic. The program of the Conference looks delightful, with its balance of stimulating paper and poster sessions, its more-than-honorific bows to historic names like Fender and Moog alongside the most recent efforts in non-linear dynamic modeling, and with the presence of so many distinguished colleagues in the field today—I note even the cocktail party is brought right into the program mix! From audio application coding and software design problems to understanding the human-interface issues of perception and gesture; from sound synthesis to spatialization—such interdisciplinary, international, and infrastructural developments are the key to our future here at McGill.

We count on you to help us forge the links within the research community and to industry that will ensure the future of the art of music and sound.

Here is to wishing the DAFX-06 Conference and your time at McGill and in Montreal every success.

Don McLean

Dean of the Schulich School of Music
Chair of the Board of CIRMMT
McGill University



Schulich School of Music
École de musique Schulich

Welcome from Stephen McAdams, Director of CIRMMT

It is with enormous pleasure that our research community welcomes the DAFX-06 delegates to Montreal, to McGill University, to the Schulich School of Music and to the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT). It is an exciting moment for CIRMMT to receive an international gathering of such excellent scientists, engineers and artists in the realm of digital audio, the domain which is of course the primary concern of our Centre.

CIRMMT was founded in 2000 and has sought over the last few years to create an environment where the kind of work exemplified by DAFX-06 takes place in an interdisciplinary setting uniting experts and students in acoustics, signal processing, motion capture and gesture analysis, gestural control, sound recording and diffusion technologies, psychoacoustics, cognitive psychology, neuroscience, music theory, music performance and music composition.

It is particularly the interaction among fields, and most importantly between artists and scientists/technologists, that is our *raison d'être*, and we are very happy to be able to share this ethos with the attendees of DAFX-06. I personally hope that many of our colleagues visiting from other cities and countries will take the opportunity to interact with the members of CIRMMT who are participating in this conference, that we may all come away fulfilled and enriched by the exchange.

Once again, *bienvenue à Montréal!*

Stephen McAdams, PhD

Professor and Canada Research Chair in Music Perception and Cognition

Director, CIRMMT

smc@cirmmt.mcgill.ca

<http://www.music.mcgill.ca/cirmmt/>



Centre for Interdisciplinary Research
in Music Media and Technology

Welcome from Philippe Depalle, DAFX-06 Conference Chair

The members of the organizing committee are pleased to welcome you to the 2006 International Conference on Digital Audio Effects (DAFx-06). This ninth edition of the conference, which originally started as part of a European COST action, is being held for the first time outside of Europe. It is a great honor to host it at the Schulich School of Music and the Center for Interdisciplinary Research in Music Media and Technology (CIRMMT), at McGill University in Montreal, Canada.

The major objective of this year's conference is to present the most important advances in digital audio as well as digital music processing, and to provide the opportunity for delegates to share their knowledge in an open and friendly atmosphere.

The conference schedule has been designed to keep paper presentations in non-parallel sessions, with extended time periods reserved for poster sessions around noon. Sixty-three works will be presented this year — thirty-nine papers and twenty-four posters — from a total of ninety-two submissions. Our thanks go to the researchers who submitted their papers, to the DAFX scientific committee and to reviewers for their hard work and diligence in helping evaluate the submissions. We are confident that the quality of the selected presentations will contribute to the success of the conference and maintain the high level of excellence established by the eight previous DAFX conferences.

This year's conference schedule will be complemented by three keynote talks given by leading specialists sharing their expertise:

- Prof. Bruno Torrésani, from Université de Provence, Marseille;
- Prof. Julius O. Smith, from Stanford University, Stanford;
- Prof. Stephen McAdams, from McGill University, Montreal.

We thank the keynote speakers and all delegates for making this conference a major scientific event.

Finally, I would like to express my deepest gratitude to the local organizing committee at the Sound Processing and Control Laboratory, McGill University. Thanks also to the Digital Composition Studios, the Centre for Interdisciplinary Research in Music Media and Technology, and the Schulich School of Music for the unique support and energy to help this event take place.

Welcome to Montreal and enjoy DAFX-06!

Philippe Depalle, PhD

Associate Professor, Music Technology Area
depalle@music.mcgill.ca

<http://www.music.mcgill.ca/musictech/>



MUSIC TECHNOLOGY

Contents

Filter Design for Audio Applications

- 1 Discretization of the '59 Fender Bassman Tone Stack
David T. Yeh and Julius O. Smith
- 7 On the Use of Volterra Series for Efficient Real-Time Simulations of Weakly Nonlinear Analog Audio Devices: Application to the Moog Ladder Filter
Thomas Hélie
- 13 Robust Design of Very High-Order Allpass Dispersion Filters
Jonathan S. Abel and Julius O. Smith

Performance and Gestural Control

- 19 Consistency of Timbre Patterns in Expressive Music Performance
Mathieu Barthet, Richard Kronland-Martinet, and Sølvi Ystad
- 25 Real-Time Detection of Finger Picking Musical Structures
Dale E. Parson
- 31 Using Visual Textures for Sonic Textures Production and Control
Jehan-Julien Filatriau, Daniel Arfib, and Jean-Michel Couturier

Poster Session 1

- 37 Graphic Equalizer Design Using Higher-Order Recursive Filters
Martin Holters and Udo Zölzer
- 41 Performance-Driven Control for Sample-Based Singing Voice Synthesis
Jordi Janer, Jordi Bonada, and Merlijn Blaauw
- 45 Parameterized Morphing as a Mapping Technique for Sound Synthesis
Chinmay Pendharkar, Michael Gurevich, and Lonce Wyse
- 49 Examining Design Goals of Digital Musical Instruments
Joseph Malloch, David Birnbaum, Elliot Sinyor, and Marcelo M. Wanderley
- 53 The Modified Chamberlin and Zölzer Filter Structures
Duane K. Wise
- 57 Smooth and Safe Parameter Interpolation of Biquadratic Filters in Audio Applications
Victor Kalinichenko
- 61 Identifying and Analyzing Relevant Characteristics of Dynamic Range Compression
Andrés Cabrera
- 67 X-Micks – Interactive Real-Time Content Based Audio Processing
Norbert Schnell, Diemo Schwarz, and Remy Müller

Sound Synthesis 1

- 71 Dispersion Modeling in Waveguide Piano Synthesis Using Tunable Allpass Filters
Jukka Rauhala and Vesa Välimäki
- 77 Prepared Piano Sound Synthesis
Stefan Bilbao and John ffitch

- 83 Digital Synthesis Models of Clarinet-Like Instruments Including Nonlinear Losses in the Resonator
Philippe Guillemain and Jonathan Terroir
- 89 Modeling Acoustic Impedance with Digital Waveguides
Gary P. Scavone and Julius O. Smith

Audio Effects

- 95 The Mellin Pizzicato
Antonio De Sena and Davide Rocchesso
- 101 Fractal Modulation Effects
Gianpaolo Evangelista
- 107 An Interdisciplinary Approach to Audio Effect Classification
Vincent Verfaille, Catherine Guastavino, and Caroline Traube

Perceptual Issues and Psychoacoustics

- 115 On the Dynamics of the Harpsichord and its Synthesis
Henri Penttinen
- 121 Musical Sound Timbre: Verbal Description and Dimensions
Jan Štěpánek
- 127 Musical Key Estimation of Audio Signal Based on Hidden Markov Modeling of Chroma Vectors
Geoffroy Peeters

Audio Analysis and Low Level Features 1

- 133 Onset Detection Revisited
Simon Dixon
- 139 A New Analysis Method for Sinusoids+Noise Spectral Models
Guillaume Meurisse, Pierre Hanna, and Sylvain Marchand
- 145 Adaptive Noise Level Estimation
Chunghsin Yeh and Axel Röbel

Poster Session 2

- 149 Categories of Perception for Vibrato, Flange, and Stereo Chorus: Mapping Out the Musically Useful Ranges of Modulation Rate and Depth for Delay-Based Effects
William Martens and Marui, Atsushi
- 153 Inter Genre Similarity Modeling for Automatic Music Genre Classification
Ulaş Bağci and Engin Erzin
- 157 Variable Pre-Emphasis LPC for Modeling Vocal Effort in the Singing Voice
Karl I. Nordstrom and Peter F. Driessens
- 161 Frequency-Dependent Boundary Condition for the 3-D Digital Waveguide Mesh
Antti Kelloniemi
- 165 Some Physical Audio Effects
Edgar Berdahl and Julius O. Smith
- 169 Table Lookup Oscillators Using Generic Integrated Wavetables
Günter Geiger
- 173 Error Compensation in Modeling Time-Varying Sinusoids
Wen Xue and Mark Sandler

- 177 A Stochastic State-Space Phase Vocoder for Synthesis of Roughness
Doug Van Nort and Philippe Depalle

Sound Synthesis 2

- 181 Fast Additive Sound Synthesis Using Polynomials
Matthias Robine, Robert Strandh, and Sylvain Marchand
- 187 Synthesis by Arcs on the Unit Circle
Vittorio Cafagna and Domenico Vicinanza
- 193 Circle Maps as a Simple Oscillators for Complex Behavior: II. Experiments
Georg Essl
- 199 Feedback Implementation Within a Complex Event Generation System for Emergent Sonic Structures
Sinan Bökesoy

Spatialization

- 203 Joint Acoustic Source Location and Orientation Estimation Using Sequential Monte Carlo
Maurice Fallon, Simon Godsill, and Andrew Blake
- 209 On the Use of Irregularly Spaced Loudspeaker Arrays for Wave Field Synthesis, Potential Impact on Spatial Aliasing Frequency
Étienne Corteel
- 215 Detection of Room Reflections From a Binaural Room Impulse Response
Sampo Vesa and Tapio Lokki
- 221 Archaeological Acoustic Space Measurement for Convolution Reverberation and Auralization Applications
Damian T. Murphy

Source Separation

- 227 Improved Cocktail-Party Processing
Alexis Favrot, Markus Erne, and Christof Faller
- 233 A Source Localization/Separation/Respatialization System Based on Unsupervised Classification of Interaural Cues
Joan Mouba and Sylvain Marchand

Audio Analysis and Low Level Features 2

- 239 Assessing the Quality of the Extraction and Tracking of Sinusoidal Components: Towards an Evaluation Methodology
Mathieu Lagrange and Sylvain Marchand
- 247 Sinusoidal Extraction Using an Efficient Implementation of a Multi-Resolution FFT
Karin Dressler
- 253 High Accuracy Frame-by-Frame Non-Stationary Sinusoidal Modelling
Jeremy J. Wells and Damian T. Murphy

Poster Session 3

- 259 Granular Resynthesis for Sound Unmixing
Gunnar Eisenberg and Thomas Sikora
- 263 Extraction and Removal of Percussive Sounds from Musical Recordings
John Usher

- 267 Representations of Audio Signals in Overcomplete Dictionaries: What is the Link Between Redundancy Factor and Coding Properties?
Emmanuel Ravelli and Laurent Daudet
- 271 A Spatial Interface for Audio and Music Production
Mike Wozniewski, Zack Settel, and Jeremy R. Cooperstock
- 275 Streaming Frequency-Domain DAFx in Csound 5
Victor Lazzarini, Joe Timoney, and Tom Lysaght
- 279 Real-Time Corpus-Based Concatenative Synthesis With CataRT
Diemo Schwarz, Grégoire Beller, Bruno Verbrugghe, and Sam Britton
- 283 Multichannel Signal Representation in PWGLSynth
Mikael Laurson and Vesa Norilo
- 287 Using Faust for FPGA Programming
Robert Trausmuth, Christian Dusek, and Yann Orlarey

Audio Coding & Audio for Multimedia

- 291 Parametric Coding of Stereo Audio Based on Principal Component Analysis
Manuel Briand, David Vurette, and Nadine Martin
- 297 Exact Discrete-Time Realization of a Dolby B Encoding/Decoding Architecture
Federico Avanzini and Federico Fontana
- 303 Real-time Bayesian GSM Buzz Removal
Han Lin and Simon Godsill
- 309 Application of Raster Scanning Method to Image Sonification, Sound Visualization, Sound Analysis and Synthesis
Woon Seung Yeo and Jonathan Berger

Software Implementation

- 315 Cecilia and TclCsound
Jean Piché and Victor Lazzarini
- 319 A New Paradigm for Sound Design
Ananya Misra, Perry R. Cook, and Ge Wang
- 325 Sound Processing in OpenMusic
Jean Bresson

331 Full Bibliography

349 Index of Authors

DISCRETIZATION OF THE '59 FENDER BASSMAN TONE STACK

David T. Yeh, Julius O. Smith

Center for Computer Research in Music and Acoustics (CCRMA)
 Stanford University, Stanford, CA
 {dtyeh|jos}@ccrma.stanford.edu

ABSTRACT

The market for digital modeling guitar amplifiers requires that the digital models behave like the physical prototypes. A component of the iconic Fender Bassman guitar amplifier, the tone stack circuit, filters the sound of the electric guitar in a unique and complex way. The controls are not orthogonal, resulting in complicated filter coefficient trajectories as the controls are varied. Because of its electrical simplicity, the tone stack is analyzed symbolically in this work, and digital filter coefficients are derived in closed form. Adhering to the technique of virtual analog, this procedure results in a filter that responds to user controls in exactly the same way as the analog prototype. The general expressions for the continuous-time and discrete-time filter coefficients are given, and the frequency responses are compared for the component values of the Fender '59 Bassman. These expressions are useful implementation and verification of implementations such as the wave digital filter.

1. INTRODUCTION

1.1. Motivation

The guitar amplifier is an essential component of the electric guitar sound, and often musicians collect several amplifiers for their tonal qualities despite the space they occupy. As digital signal processors (DSP) continue to improve in performance, there is great interest in replacing expensive and bulky vacuum tube guitar amplifiers with more flexible and portable digital models. A digital model of a guitar amplifier allows a variety of sounds associated with different amplifiers to be selected from a single amplifier unit. One company, Line 6 bases its main product line upon this concept, and other companies such as Roland (Boss), Korg (Vox), Harman International (Digitech) have competing products.

Most commercially viable digital guitar processing products use simplified models of the distortion and filters to reduce DSP power consumption and reduce manufacturing costs. The distortion is typically a nonlinear transfer curve, accompanied by digital filtering that is manually tuned to match the sound of a famous guitar amplifier.

With no pressure to produce a commercially successful product, this research takes a different approach. The goal of this research is to see how accurate a sound can be achieved through careful physical modeling of the vacuum tube amplifier and to provide a physical basis for the digital model and parameters. Because the tone stack is a passive, linear component, it is a straightforward starting point.

1.2. Properties of the tone stack

Commonly found in many guitar amplifiers, especially those that derive from the Fender design, the tone stack filters the signal of

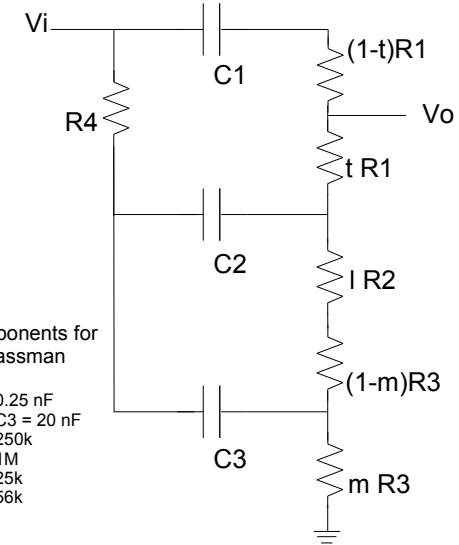


Figure 1: Tone stack circuit with component values.

the guitar in a unique and non-ideal way. The user can adjust Treble, Middle, and Bass controls to modify the gain of the respective frequency bands. However, these controls are not orthogonal, and changing some controls affects the other bands in a complex way.

The full Bassman schematic can be easily found online [1] and in guitar amplifier books. While other guitar amplifiers may vary slightly, in the Bassman type designs, the tone stack is found after the preamplifier stages and before the phase splitter. In good designs, the tone stack is preceded by a cathode follower to buffer the input and reduce variations in frequency response due to loading. Typically this presents a $1k\Omega$ load to the input and the phase splitter stage presents a $1M\Omega$ load to the output.

The Fender '59 Bassman tone stack circuit is shown in Fig. 1. The Treble, Middle, and Bass knobs are potentiometers, which have been modeled here as parameterized resistors. The Treble and Middle controls use linear potentiometers, while the Bass control uses a logarithmic taper potentiometer. In this paper, t and m correspond to the Treble and Middle controls and range in value from $[0, 1]$. The Bass control, l , also ranges from $[0, 1]$, but is swept logarithmically.

1.3. Related work

Fender Musical Instruments has a patent to simulate various tone stacks using an active analog filter and an interpolation scheme to extract the filter coefficients [2]. Line 6 also models the behavior of the Bassman tone stack as indicated in the BassPODxt manual. However, their implementation is proprietary knowledge. An open source guitar effects plug-in suite for Linux, CAPS [3], uses shelving filters instead of the tone stack.

Previous works have analyzed the tone stack using numerical circuit analysis techniques. This involves setting up the nodal equations as a matrix and inverting it or performing Gaussian elimination to find the solution. For example, the Tone Stack Calculator from Duncan Amps will plot the frequency response of various tone stacks given the control settings [4]. Kuehn in his book analyzed the mesh equations of the tone stack, using low frequency and high frequency circuit approximations [5]. He also compares these simplified equations to the numerical solutions solved by inverting the matrix of the mesh equations. While the approximations make the circuit analysis more tractable, they do not reduce the order of the equations and do not make the discretization of the filter any easier.

Because the tone stack is a third-order passive network of resistors and capacitors (RC), its filter coefficients can be derived and modeled exactly in the digital domain as shown later. The approach taken here is to find the continuous time transfer function of the circuit analytically and to discretize this by the bilinear transformation. This provides a means of updating the digital filter coefficients based upon changes to the tone controls.

The passive filter circuit also is suited to implementation as a wave digital filter (WDF)[6]. This approach can easily model standard components such as inductors, capacitors, and resistors. The analytical form derived here can be used for comparison with and verification of the WDF implementation.

2. DISCRETIZATION PROCEDURE

2.1. Symbolic Circuit Analysis

Because this is a relatively simple circuit, it is amenable to exact symbolic analysis by mathematical Computer Aided Design (CAD) software such as Mathematica (Wolfram Research, Inc., Champaign, IL). The filter coefficients can thus be found without any approximations. Performing symbolic nodal analysis on this circuit yields the following input/output transfer function $H(s) = V_o(s)/V_i(s)$, where V_o is the output and V_i is the input as in Fig. 1.

$$H(s) = \frac{b_1 s + b_2 s^2 + b_3 s^3}{a_0 + a_1 s + a_2 s^2 + a_3 s^3}, \quad (1)$$

where

$$b_1 = tC_1R_1 + mC_3R_3 + l(C_1R_2 + C_2R_2) + (C_1R_3 + C_2R_3),$$

$$\begin{aligned} b_2 = & t(C_1C_2R_1R_4 + C_1C_3R_1R_4) - m^2(C_1C_3R_3^2 + C_2C_3R_3^2) \\ & + m(C_1C_3R_1R_3 + C_1C_3R_3^2 + C_2C_3R_3^2) \\ & + l(C_1C_2R_1R_2 + C_1C_2R_2R_4 + C_1C_3R_2R_4) \\ & + lm(C_1C_3R_2R_3 + C_2C_3R_2R_3) \\ & + (C_1C_2R_1R_3 + C_1C_2R_3R_4 + C_1C_3R_3R_4), \end{aligned}$$

$$\begin{aligned} b_3 = & lm(C_1C_2C_3R_1R_2R_3 + C_1C_2C_3R_2R_3R_4) \\ & - m^2(C_1C_2C_3R_1R_3^2 + C_1C_2C_3R_3^2R_4) \\ & + m(C_1C_2C_3R_1R_3^2 + C_1C_2C_3R_3^2R_4) \\ & + tC_1C_2C_3R_1R_3R_4 - tmC_1C_2C_3R_1R_3R_4 \\ & + tlC_1C_2C_3R_1R_2R_4, \end{aligned}$$

$$a_0 = 1,$$

$$\begin{aligned} a_1 = & (C_1R_1 + C_1R_3 + C_2R_3 + C_2R_4 + C_3R_4) \\ & + mC_3R_3 + l(C_1R_2 + C_2R_2), \end{aligned}$$

$$\begin{aligned} a_2 = & m(C_1C_3R_1R_3 - C_2C_3R_3R_4 + C_1C_3R_3^2 \\ & + C_2C_3R_3^2) + lm(C_1C_3R_2R_3 + C_2C_3R_2R_3) \\ & - m^2(C_1C_3R_3^2 + C_2C_3R_3^2) + l(C_1C_2R_2R_4 \\ & + C_1C_2R_1R_2 + C_1C_3R_2R_4 + C_2C_3R_2R_4) \\ & + (C_1C_2R_1R_4 + C_1C_3R_1R_4 + C_1C_2R_3R_4 \\ & + C_1C_2R_1R_3 + C_1C_3R_3R_4 + C_2C_3R_3R_4), \end{aligned}$$

$$\begin{aligned} a_3 = & lm(C_1C_2C_3R_1R_2R_3 + C_1C_2C_3R_2R_3R_4) \\ & - m^2(C_1C_2C_3R_1R_3^2 + C_1C_2C_3R_3^2R_4) \\ & + m(C_1C_2C_3R_3^2R_4 + C_1C_2C_3R_1R_3^2 \\ & - C_1C_2C_3R_1R_3R_4) + lC_1C_2C_3R_1R_2R_4 \\ & + C_1C_2C_3R_1R_3R_4, \end{aligned}$$

where t is the Treble (or “top”) control, l is the Bass (or “low”) control, and m is the “middle” control.

2.2. Verification with SPICE circuit simulation

To verify the correctness of this expression, Figs. 2 and 3 compare the frequency response with the result from the AC analysis of SPICE¹ simulation at the settings $t = m = l = 0.5$. The plots show an exact match, verifying that Eqn. 1 is a complete and exact expression for the transfer function of the tone stack. SPICE simulation also determined that the frequency response was unaffected by the typical loading of $1k\Omega$ at the input and $1M\Omega$ at the output.

2.3. Discretization by Bilinear Transform

The continuous time transfer function was discretized by the bilinear transformation. Substituting $s = \frac{1-z^{-1}}{1+z^{-1}}$ in (1) using Mathematica yields

$$H(z) = \frac{B_0 + B_1z^{-1} + B_2z^{-2} + B_3z^{-3}}{A_0 + A_1z^{-1} + A_2z^{-2} + A_3z^{-3}} \quad (2)$$

where

$$\begin{aligned} B_0 &= -b_1c - b_2c^2 - b_3c^3, \\ B_1 &= -b_1c + b_2c^2 + 3b_3c^3, \\ B_2 &= b_1c + b_2c^2 - 3b_3c^3, \\ B_3 &= b_1c - b_2c^2 + b_3c^3, \\ A_0 &= -a_0 - a_1c - a_2c^2 - a_3c^3, \\ A_1 &= -3a_0 - a_1c + a_2c^2 + 3a_3c^3, \\ A_2 &= -3a_0 + a_1c + a_2c^2 - 3a_3c^3, \\ A_3 &= -a_0 + a_1c - a_2c^2 + a_3c^3. \end{aligned}$$

¹<http://bwrc.eecs.berkeley.edu/Classes/lcBook/SPICE/>

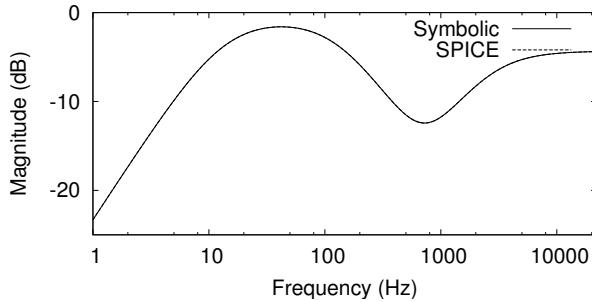


Figure 2: Comparison of magnitude response between analytical expression and SPICE for $t = l = m = 0.5$.

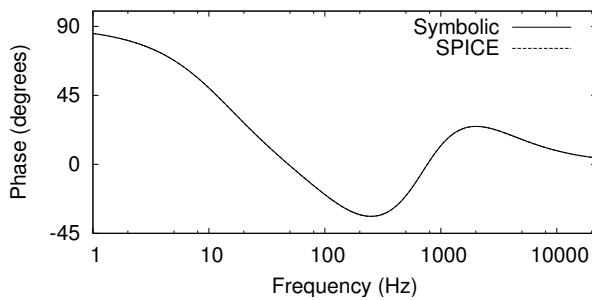


Figure 3: Comparison of phase response between analytical expression and SPICE for $t = l = m = 0.5$.

We used $c = 2/T$, which is ideal for frequencies close to DC.

3. ANALYSIS OF RESULTS

3.1. Comparison of continuous- and discrete-time responses

Figs. 4–6 show the discrete- and continuous-time transfer functions compared for various settings of t , m , and l . Each figure shows a different setting of l , and each sub-figure shows a different setting of m . In each plot, the treble control, t , was swept from 0.0001 to 0.5 to 0.9999 and can be distinguished by the corresponding increase in high frequency response.

The discretized filter used a sampling frequency of 44.1 kHz as typical for audio systems. The plots for $f_s = 44.1$ kHz show an excellent match through 10 kHz. The discrete and continuous plots are practically indistinguishable, with some deviations at the higher frequencies, as expected with the bilinear transform.

Because commercial guitar processing units use a lower sampling rate for cost savings, Figs. 7–9 show the same plots as above with f_s reduced to 20 kHz. These curves deviate slightly more from $H(s)$ at high frequencies, but exhibit the same trends as before.

The errors, defined as the difference between the dB values of $H(s)$ and $H(z)$ at each frequency, are plotted in Fig. 10 for $f_s = 20$ kHz and $f_s = 44.1$ kHz (abbreviated as 44k) for the settings of t , m , and l that give the worst case results. The error is only meaningful for frequencies up through $f_s/2$.

The curves for $t = 0.5, m = 0, b = 1$ are characteristic of tone settings that give a high pass response and have error within 0.5 dB for both cases of f_s .

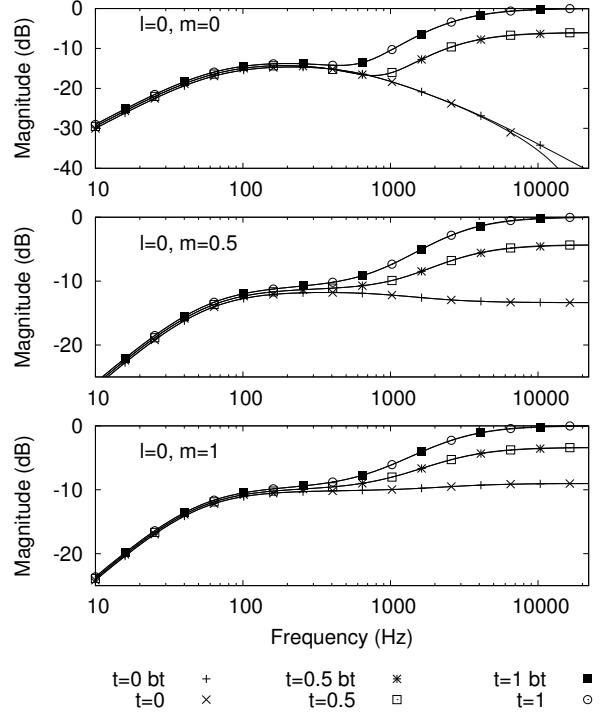


Figure 4: Comparison of filter magnitude response between original and discretized ($f_s = 44.1$ kHz) filters, $l = 0$.

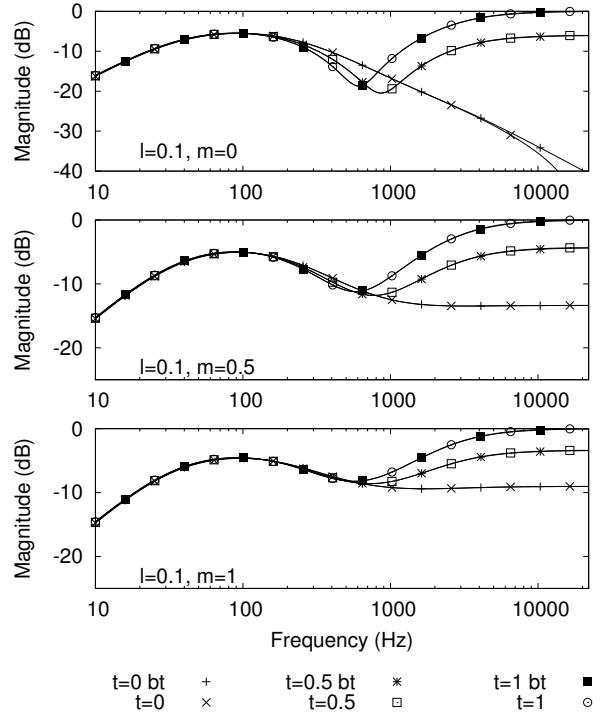


Figure 5: Comparison of filter magnitude response between original and discretized ($f_s = 44.1$ kHz) filters, $l = 0.1$.

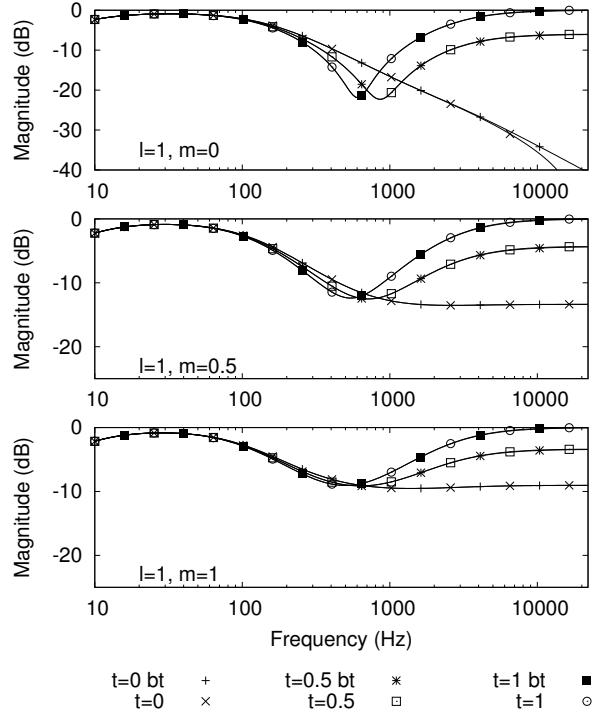


Figure 6: Comparison of filter magnitude response between original and discretized ($f_s = 44.1 \text{ kHz}$) filters, $l = 1$.

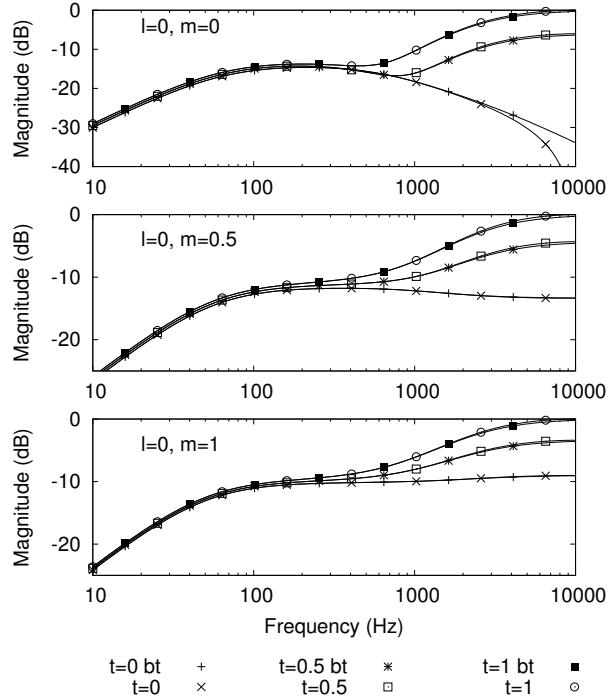


Figure 7: Comparison of filter magnitude response between original and discretized ($f_s = 20 \text{ kHz}$) filters, $l = 0$.

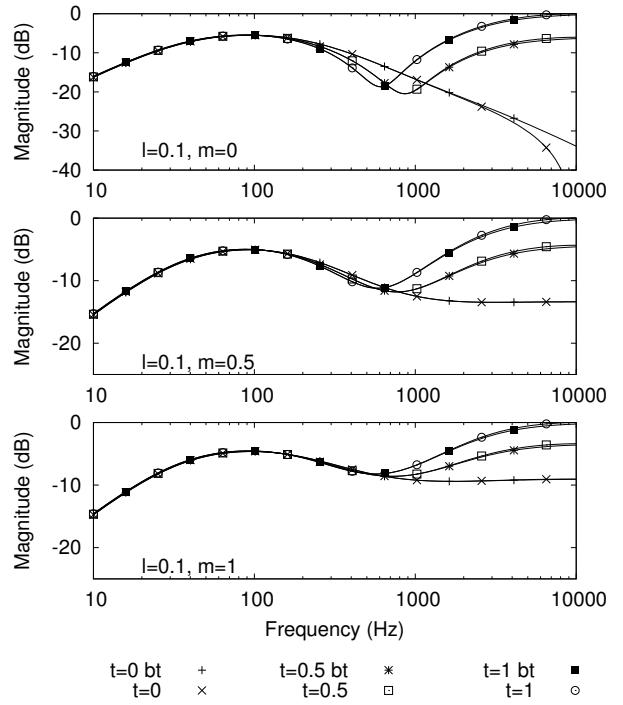


Figure 8: Comparison of filter magnitude response between original and discretized ($f_s = 20 \text{ kHz}$) filters, $l = 0.1$.

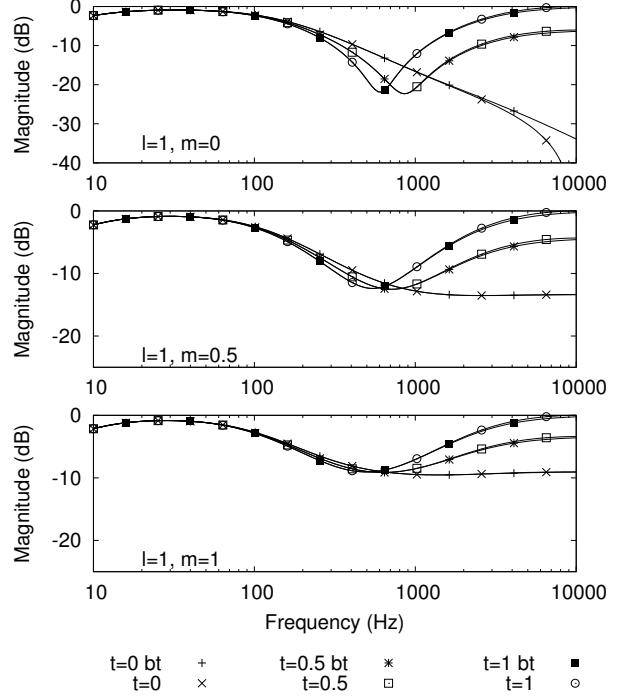


Figure 9: Comparison of filter magnitude response between original and discretized ($f_s = 20 \text{ kHz}$) filters, $l = 1$.

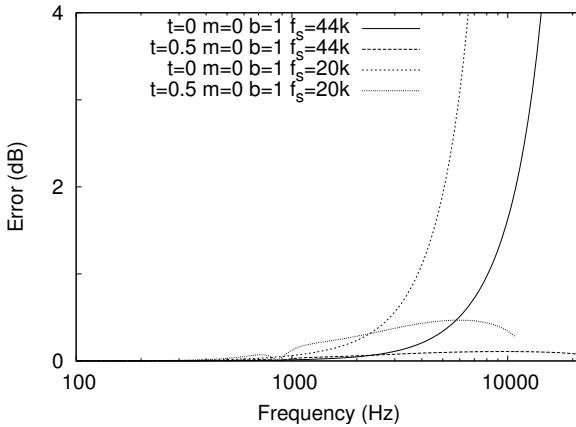


Figure 10: Error as difference between dB values of $H(s)$ and $H(z)$, for $f_s = 20$ and 44.1 kHz , and the noted tone settings.

The curves for $t = 0, m = 0, b = 1$ are characteristic of settings that give a low pass response and exhibit a rapidly increasing error as frequency increases because the bilinear transform maps the null at infinite frequency to $f_s/2$. The error rises to 3 dB at roughly 6 kHz for $f_s = 20\text{ kHz}$, and at 13 kHz for $f_s = 44.1\text{ kHz}$. Because of the low pass nature of these responses, the errors occur at frequencies where the magnitude is at least 10-20 dB lower than its peak value, making them perceptually less salient. Also, given that the frequency response of a typical guitar speaker is from 100 Hz to 6000 Hz, the deviations at higher frequencies would be inconsequential.

3.2. Implications of system poles and zeros for filter implementation

The plots exhibit the complex dependence of the frequency response upon the tone controls. The most obvious effect is that changes in the *Middle* control also affect the treble response. The analytical form of the transfer function provides a way to find the poles and zeros of the system as the settings are varied and gives insight into how the filter could be simplified to facilitate the implementation while maintaining accuracy.

Note that the tone stack is an entirely passive circuit composed of resistors and capacitors. This implies that the three poles of this system are all real. There is a zero at DC, leaving a pair of zeros that may be complex depending on the control settings. This also implies that the tone stack cannot be a resonant circuit although the pair of imaginary zeros can set up an anti-resonance as evident in the notch seen in the frequency response plots.

Also note from Eqn. (1) that none of the coefficients of the denominator depends on the treble control, t . The treble control therefore does not control the modes of the circuit but only adjusts the position of the zeros. This circuit can be decomposed into a weighted sum of terms that correspond to each mode by the partial fraction expansion. From this perspective, the treble control only affects the weighting of the different modes, but not the pole location of each mode. The poles are controlled exclusively by the bass and middle knobs.

This insight suggests possible alternate filter topologies. Instead of implementing the filter directly as a single third-order filter, one could equivalently use series and parallel combinations of

lower order filters. Understanding the poles and zeros of the system, one could make simplifying assumptions, ignoring terms that have little impact on the locations of the poles and zeros.

One implementation would be to find the partial fraction expansion of the transfer function using the expression given and precompute the poles, residues, and direct terms based upon the three-dimensional input space of the tone controls. These terms can be interpolated in the input space and used in the parallel filter structure that arises from the partial fraction expansion.

The existence of an analytical expression for the poles and zeros also informs the choice of c in the bilinear transform. The analytical expression allows the computation of frequency domain features such as local maxima or anti-resonance notches to be matched in the discrete-time domain.

4. CONCLUSIONS

This work shows that the Fender tone stack can be parameterized exactly in the discrete-time domain and that the bilinear transform provides an outstanding frequency mapping for reasonable sampling rates. The transfer function for the physical tone stack was found as a function of its control parameters and component values using symbolic math software. This analysis provides a formula for updating the digital tone stack coefficients in a way that exactly emulates the physical circuit. The symbolic form of the transfer function also allows easy determination of the poles and zeros of the system and guides the design of a filter with simplified coefficients.

Further work remains to factor the expression for the tone stack frequency response and find a structure with simpler expressions for updating the filter. One possible implementation is the wave digital filter. A real-time implementation of the tone stack is also in progress.

5. ACKNOWLEDGEMENTS

David Yeh is supported by the NDSEG fellowship. Thanks to Tim Stilson for help with root loci.

6. REFERENCES

- [1] Ampwares, “5F6-A schematic,” Retrieved June 29th, 2006, [Online] http://www.ampwares.com/ffg/bassman_narrow.html.
- [2] D. V. Curtis, K. L. Chapman, C. C. Adams, and Fender Musical Instruments, “Simulated tone stack for electric guitar,” United States Patent 6222110, 2001.
- [3] T. Goetze, “caps, the C Audio Plugin Suite,” Retrieved June 29th, 2006, [Online] <http://quitte.de/dsp/caps.html>.
- [4] Duncan Amps, “Tone stack calculator,” Retrieved June 29th, 2006, [Online] <http://www.duncanamps.com/tsc/>.
- [5] R. Kuehnel, *Circuit Analysis of a Legendary Tube Amplifier: The Fender Bassman 5F6-A*, 2nd ed. Seattle: Pentode Press, 2005. [Online]. Available: <http://www.pentodepress.com/contents.html>
- [6] A. Fettweis, “Wave digital filters: Theory and practice,” *Proc. IEEE*, vol. 74, pp. 270–327, Feb. 1986.

ON THE USE OF VOLTERRA SERIES FOR REAL-TIME SIMULATIONS OF WEAKLY NONLINEAR ANALOG AUDIO DEVICES: APPLICATION TO THE MOOG LADDER FILTER

Thomas Hélie

Ircam - CNRS - STMS UMR 9912

Équipe Analyse/Synthèse
1, place Igor Stravinsky
F-75004 Paris, France
Thomas.Helie@ircam.fr

ABSTRACT

In this paper, we show how the formalism of the Volterra series can be used to represent the nonlinear Moog ladder filter. The analog circuit is analyzed to produce a set of governing differential equations. The Volterra kernels of this system are solved from simple algebraic equations. They define an exact decomposition of the system. An identification procedure leads to structures composed of linear filters, sums and instantaneous products of signals. Finally, a discrete-time realization of the truncated series, which guarantees no aliasing, is performed.

1. INTRODUCTION

Most of the analog audio devices used in electro-acoustic music have been simulated in numerous softwares thanks to digital implementations. Nevertheless, many musicians still prefer original devices rather than their digital versions. One of the main reasons is that analog circuits involve nonlinearities, responsible for perceptible characteristic distortions. Even for weak nonlinearities, the distortion is progressively activated with respect to the signal amplitude so that playing on the dynamics makes the sound “live”. Including such phenomena in audio implementation is difficult to tackle since nonlinearities naturally creates aliasing.

In this paper, we show that the Volterra series formalism can be used to represent weakly nonlinear analog audio devices as input-output systems, from which efficient digital implementations can be deduced. Volterra series define exact representations of such systems on given amplitude ranges. If the equations which govern the circuit are differential, each kernel of the series is deduced in the Laplace domain from simple algebraic equations. One kernel isolates a sub-system attached to a monomial nonlinearity of order n and monitors the exact associated sub-dynamics. In practice, even a low order truncated version of the series yields realistic distortions while it allows to overcome the problem of aliasing. In order to concentrate on the method rather than a “new complex circuit”, we choose to consider a well-known and deeply-studied circuit, the Moog ladder filter [1, 2, 3, 4].

The paper is structured as follows. In section 2, the analog circuit of the Moog ladder filter is recalled and analyzed to produce a set of governing differential equations. This nonlinear differential system is re-casted, for dimensionless variables. Section 3 introduces the Volterra series and some of their fundamental properties. Section 4 establishes the equations satisfied by the Volterra kernels of the Moog ladder filter: first in § 4.3 for a one stage filter, second

in § 4.3 for a four-stages filter, third in § 4.3 for the complete Moog ladder filter with a loop. Analytic expressions of these kernels are detailed for the orders $n = 1, 2, 3$. Section 5 presents a low-cost numerical simulation in the time domain: in § 5.1, the kernels are identified as structures composed of linear filters, sums and instantaneous products of signals in the continuous time-domain; a state-space representation is given in § 5.2; a digital implementation is derived in § 5.3 such that the pole mapping of the linear part is exact and the aliasing due to the nonlinearities is rejected. The validity of the approximated structure is discussed in section 6. Finally, conclusions are given in section 7.

2. ELECTRONIC CIRCUIT AND NONLINEAR DIFFERENTIAL EQUATIONS

2.1. The Moog ladder filter circuit

The Moog ladder filter is a circuit composed of a driver and a cascade of four filters involving capacitors C and differential pairs of NPN-transistors (see Figure 1).

2.1.1. Transistors

The NPN-transistors (see Figure 1a) are configured such that the base currents I_B can be neglected. Indeed, $I_B = I_C/\beta$ with $\beta > 100$ so that $I_E = I_C + I_B \approx I_C$. Moreover, the PN-junction BE is governed by

$$I_C = I_E = I_s \left[e^{\frac{V_B - V_E}{V_T}} - 1 \right] \approx I_s e^{\frac{V_B - V_E}{V_T}}, \quad (1)$$

where the thermal voltage is $V_T = kT/q \approx 25.85$ mV and the saturation current is $I_s \approx 10^{-14}$ A for the temperature $T = 300$ K, and where $k = 1.38 \cdot 10^{-23}$ J/K is the Boltzmann constant, and $q = 1.6 \cdot 10^{-19}$ C is the electron charge.

2.1.2. Driver

From (1), the ratio $\frac{I_1}{J_1}$ is $\frac{I_1}{J_1} = e^{-\frac{U_0}{V_T}}$ (see Figure 1b). Moreover,

$$I_1 + J_1 = I_c \quad (2)$$

$$I_1 - J_1 = I_c \frac{I_1/J_1 - 1}{I_1/J_1 + 1} = -I_c \tanh \frac{U_0}{2V_T} \quad (3)$$

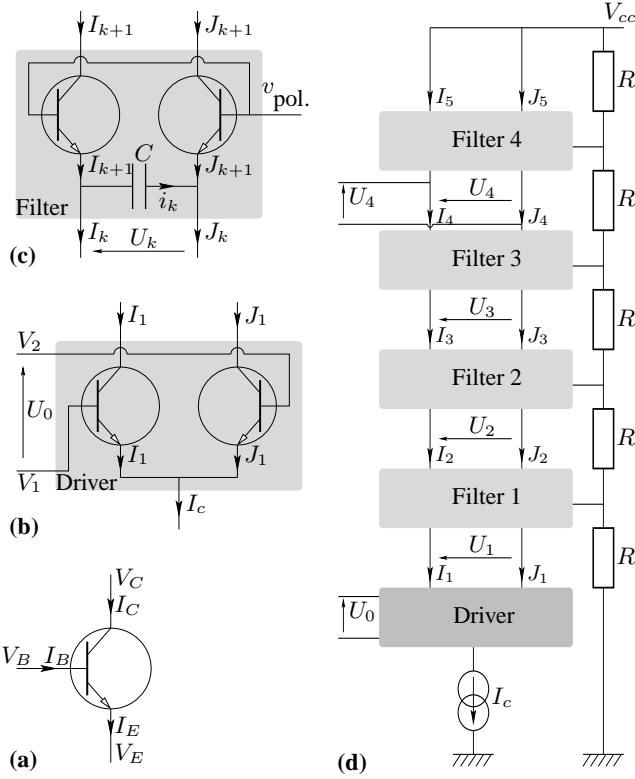


Figure 1: Circuits: (a) NPN transistor, (b) driver, (c) one-stage filter, and (d) four-stages Moog ladder filter.

2.1.3. One-stage filters for $k=1,2,3,4$

From (1), the ratios $\frac{I_{k+1}}{J_{k+1}}$ are $\frac{I_{k+1}}{J_{k+1}} = e^{-\frac{U_k}{V_T}}$ (see Figure 1c). Moreover,

$$I_{k+1} = I_k + i_k \quad (4)$$

$$J_{k+1} = J_k - i_k \quad (5)$$

The sum and the difference of (4) and (5) yield

$$I_{k+1} + J_{k+1} = I_k + J_k \quad (= I_1 + J_1 = I_c) \quad (6)$$

$$I_{k+1} - J_{k+1} = I_k - J_k + 2i_k \quad (7)$$

Now, the differential pair of transistors yields

$$I_{k+1} - J_{k+1} = I_c \frac{I_{k+1}/J_{k+1} - 1}{I_{k+1}/J_{k+1} + 1} = -I_c \tanh \frac{U_k}{2V_T} \quad (8)$$

and the capacitor law yields

$$i_k = C \frac{dU_k}{dt} \quad (9)$$

2.1.4. Four-stages filter and loop

Rewriting the terms of (7) for $k = 1, 2, 3, 4$ thanks to (3), (8) and (9), leads to the voltage equations

$$-I_c \tanh \frac{U_k}{2V_T} = -I_c \tanh \frac{U_{k-1}}{2V_T} + 2C \frac{dU_k}{dt}. \quad (10)$$

In practice, the Moog ladder filter includes the circuit in Figure 1d, a voltage input which controls I_c , some voltage adders, and a loop with a controlled feedback gain [1], [5, p46]. This feedback writes

$$U_0 = U_{in} - 4r U_4 \quad (11)$$

where U_{in} is the input and $r \in [0, 1]$ controls the feedback gain.

2.2. Dimensionless model

A dimensionless version of the problem is given by

$$\frac{1}{\omega_c} \frac{du_k}{dt} + \tanh u_k = \tanh u_{k-1}, \quad k = 1, 2, 3, 4, \quad (12)$$

$$\text{with } u_0 = u_{in} - 4r u_4, \quad (13)$$

where $\omega_c = I_c/(4CV_T)$, $u_k = U_k/(2V_T)$ and $u_{in} = U_{in}/(2V_T)$.

In this paper, parameters ω_c and r are supposed quasi-constant so that the global system is quasi-stationary. Nevertheless, the method presented below could be adapted to non-stationary problems, using non-stationary Volterra series [6, 7].

3. INTRODUCTION TO VOLTERRA SERIES

3.1. Definitions and notations

A system is described by a Volterra series of kernels $\{h_n\}_{n \in \mathbb{N}^*}$ for inputs $|u(t)| < \rho$ if the output $y(t)$ is given by the multi-convolutions

$$y(t) = \sum_{n=1}^{+\infty} \int_{\mathbb{R}^n} h_n(\tau_1, \dots, \tau_n) u(t-\tau_1) \dots u(t-\tau_n) d\tau_1 \dots d\tau_n, \quad (14)$$

where ρ is the convergence radius of the characteristic function

$$\varphi_h(x) = \sum_{n=1}^{+\infty} \|h_n\|_1 x^n, \quad (15)$$

and $\|h_n\|_1 = \int_{\mathbb{R}^n} |h_n(\tau_1, \dots, \tau_n)| d\tau_1 \dots d\tau_n$ is the L^1 -norm of h_n .

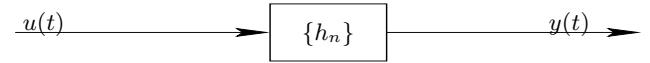


Figure 2: System represented by Volterra kernels.

For a causal system, h_n are zero for $\tau_k < 0$. Their monolateral [8, (29.1.2)] Laplace transforms are denoted with capital letters $H_n(s_1, \dots, s_n)$. For stable systems, the kernels H_n are analytic for $s_k, \Re(s_k) > 0$.

Notation: These systems are usually represented with their kernels, either in the time domain $\{h_n\}$ as displayed in Figures 2 and 3, either in the Laplace domain $\{H_n\}$ as displayed in Figure 4.

Remark 1: Volterra series embed systems described by: (a) linear filters ($h_n = 0$ for $n \geq 2$) ; (b) instantaneous nonlinear functions $y = h(u)$ with $h(0) = 0$ which admits a series expansion $h(u) = \sum_{n=1}^{+\infty} \alpha_n u^n$; (c) their various combinations (sum, product, cascade, as detailed in § 3.2).

Remark 2: For the case (b), the (convolution) kernels are given by $h_n(t_1, \dots, t_n) = \alpha_n \delta(t_1, \dots, t_n)$ in the time domain (δ denotes the Dirac distribution), and by the constant functions $H_n(s_1, \dots, s_n) = \alpha_n$ in the Laplace domain.

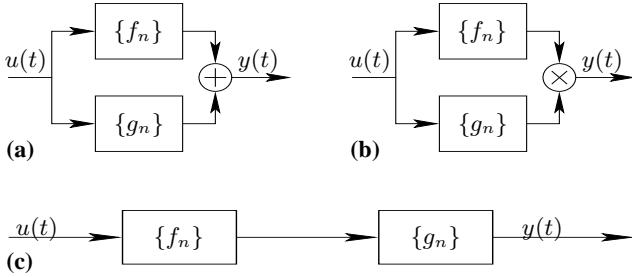


Figure 3: Sum (a), product (b), and cascade (c) of two systems.

3.2. Interconnection laws

Let \mathbb{N}^* denote the strictly positive integers. The kernels $\{H_n\}_{n \in \mathbb{N}^*}$ of the systems in Figures 3a, 3b, and 3c are given respectively by [9, p. 34,35]

$$H_n(s_1, \dots, s_n) = F_n(s_1, \dots, s_n) + G_n(s_1, \dots, s_n), \quad (16)$$

$$H_n(s_1, \dots, s_n) = \sum_{p=1}^{n-1} F_p(s_1, \dots, s_p) G_{n-p}(s_{p+1}, \dots, s_n), \quad (17)$$

$$H_n(s_1, \dots, s_n) = \sum_{p=1}^n \sum_{(i_1, \dots, i_p) \in \mathbb{I}_n^p} F_{i_1}(s_1, \dots, s_{i_1}) \dots F_{i_p}(s_{i_1+ \dots + i_{p-1}+1}, \dots, s_n) \\ \cdot G_p(s_1 + \dots + s_{i_1}, \dots, s_{i_1+ \dots + i_{p-1}+1} + \dots + s_n) \quad (18)$$

where $\mathbb{I}_n^p = \{(i_1, \dots, i_p) \in (\mathbb{N}^*)^p \text{ s.t. } i_1 + \dots + i_p = n\}$. Note that \mathbb{I}_n^p is the singleton $\{(1, \dots, 1)\}$ and that $\mathbb{I}_n^p = \emptyset$ when $p > n$.

The radii of convergence are such that $\rho_h \geq \min(\rho_f, \rho_g)$ for the cases (a,b) and $\rho_h \geq \min(\rho_f, \varphi_f^{-1}(\rho_g))$ for the case (c).

4. VOLTERRA KERNELS OF THE MOOG LADDER FILTER

4.1. Kernels of a single stage \mathcal{F}

Let $\{F_n\}_{n \in \mathbb{N}^*}$ be the unknown kernels of a single stage filter with input u_{k-1} and output u_k . They describe the dimensionless system (12) which corresponds to the circuit in Figure 1c, for a given k . Let $\{T_n\}_{n \in \mathbb{N}}$ be the coefficients of the series expansion of \tanh . They are given by $T_{2p} = 0$ for $p \in \mathbb{N}$, $T_1 = 1$, $T_3 = -1/3$ and, more generally, by $T_{2p-1} = (-1)^{p-1} 2(2^{2p}-1)B_{2p}/(2p)!$ for $p \geq 1$ where B_n denotes the n^{th} Bernoulli numbers (see [8, (4.5.64)]). According to the remark 2 in § 3.1, the coefficients T_n also define the constant kernels $\{T_n\}$ of the system $y(t) = \tanh(u(t))$, in the Laplace domain.

Now, we describe (12) through a block diagram involving the Volterra kernels $\{F_n\}$ and $\{T_n\}$ which define the null-system detailed in Figure 4, where

$$Q_1(s_1) = \frac{s_1}{\omega_c} \quad (19)$$

defines to the linear operator $\frac{1}{\omega_c} \frac{d}{dt}$ in the Laplace domain. The kernels of this null-system can be derived from the interconnection laws (16) and (18). Writing that the kernels of the null system are zero yields, for $n \in \mathbb{N}^*$,

$$F_n(s_1, \dots, s_n) Q_1(s_1 + \dots + s_n) + \sum_{p=1}^n \sum_{(i_1, \dots, i_p) \in \mathbb{I}_n^p} F_{i_1}(s_1, \dots, s_{i_1}) \dots F_{i_p}(s_{i_1+ \dots + i_{p-1}+1}, \dots, s_n) T_p = T_n. \quad (20)$$

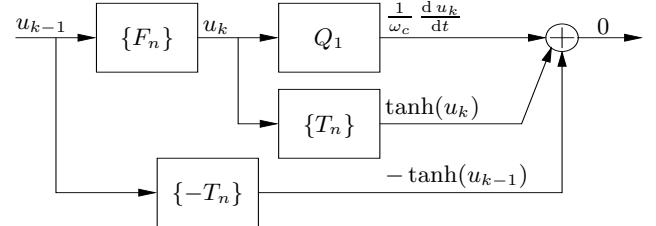


Figure 4: Canceling system for \mathcal{F} .

The first term in (20) represents the cascade $\{F_n\} \rightarrow Q_1$ in Figure 4. It is derived from (18) in which only the term with $p = n$ is not zero. The second term represents the cascade $\{F_n\} \rightarrow \{T_n\}$. Note that the index $p = 1$ is associated to $F_n(s_1, \dots, s_n) T_1$ while the indexes $p \geq 2$ only involve F_k with $k \leq n-1$. The second member stands for $\{-T_n\}$. Equation (20) rewrites, for $n \in \mathbb{N}^*$,

$$F_n(s_1, \dots, s_n) = [T_1 + Q_1(s_1 + \dots + s_n)]^{-1} \cdot \left[T_n - \sum_{p=2}^n T_p \sum_{(i_1, \dots, i_p) \in \mathbb{I}_n^p} F_{i_1}(s_1, \dots, s_{i_1}) \dots F_{i_p}(s_{i_1+ \dots + i_{p-1}+1}, \dots, s_n) \right]. \quad (21)$$

This yields recursive algebraic equations: for each n , the second member of (21) is a finite sum composed of kernels F_{i_k} which have been yet computed since $i_k < n$. The kernels for $n = 1, 2, 3$ are given by,

$$F_1(s_1) = [T_1 + Q_1(s_1)]^{-1} = \left[1 + \frac{s_1}{\omega_c} \right]^{-1} \quad (22)$$

$$F_2(s_1, s_2) = 0, \quad (23)$$

$$F_3(s_1, s_2, s_3) = T_3 [1 - F_1(s_1) F_1(s_2) F_1(s_3)] F_1(s_1 + s_2 + s_3). \quad (24)$$

Thus, including the nonlinear effect in the application requires to consider the kernels at least until $n = 3$.

4.2. Kernels of a complete four-stages filter \mathcal{F}^4

Let $\{F_n^k\}_{n \in \mathbb{N}^*}$ denote the kernels of the cascade of k systems $\{F_n\}_{n \in \mathbb{N}^*}$. The kernels $\{F_n^4\}$ are derived from (18) in two steps: first, the cascade of $\{F_n\}$ and $\{F_n\}$ yields $\{F_n^2\}$; second, that of $\{F_n^2\}$ with $\{F_n^2\}$ yields $\{F_n^4\}$.

As $F_2(s_1, s_2) = 0$, this leads to, for $n = 1, 2, 3$,

$$F_1^2(s_1) = [F_1(s_1)]^2, \quad (25)$$

$$F_2^2(s_1, s_2) = 0, \quad (26)$$

$$F_3^2(s_1, s_2, s_3) = F_3(s_1, s_2, s_3) F_1(s_1 + s_2 + s_3) + F_1(s_1) F_1(s_2) F_1(s_3) F_3(s_1, s_2, s_3), \quad (27)$$

and for the second step,

$$F_1^4(s_1) = [F_1^2(s_1)]^2 = [F_1(s_1)]^4, \quad (28)$$

$$F_2^4(s_1, s_2) = 0, \quad (29)$$

$$\begin{aligned} F_3^4(s_1, s_2, s_3) &= F_3^2(s_1, s_2, s_3)F_1^2(s_1+s_2+s_3) \\ &\quad + F_1^2(s_1)F_2^2(s_2)F_3^2(s_3)F_3^2(s_1, s_2, s_3) \\ &= \sum_{k=0}^3 [F_1(s_1)]^k [F_1(s_2)]^k [F_1(s_3)]^k \\ &\quad \cdot F_3(s_1, s_2, s_3) [F_1(s_1+s_2+s_3)]^{3-k}. \end{aligned} \quad (30)$$

4.3. Kernels of the Moog ladder filter \mathcal{L} with a loop

Let $\{L_n\}_{n \in \mathbb{N}^*}$ be the kernels of the four-stages filter with the loop, fed by the input u_{in} and with output u_4 . They describe the dimensionless system (12-13) which corresponds to the circuit in Figure 1d. This system is such that the block diagram in Figure 5 defines the null system. In this block diagram, the kernels of the

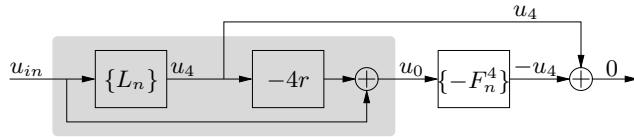


Figure 5: Canceling system for \mathcal{L} .

sub-system inside the gray box are $4r L_n(s_1, \dots, s_n) + \delta_{1,n}$ where $\delta_{1,n}$ denotes the Kronecker symbol ($\delta_{1,n}=1$ if $n=1$ and $\delta_{1,n}=0$ otherwise). Writing from (18) that the cascade of this system with $\{F_n^4\}_{n \in \mathbb{N}^*}$ is $\{L_n\}_{n \in \mathbb{N}^*}$ yields

$$\sum_{p=1}^n \sum_{(i_1, \dots, i_p) \in \mathbb{I}_n^p} [\delta_{1,i_1} - 4r L_1(s_1, \dots, s_{i_1})] \dots [\delta_{1,i_p} - 4r L_{i_p}(s_{i_1+ \dots + i_{p-1}+1}, \dots, s_n)] \cdot F_p^4(s_{i_1+ \dots + i_{p-1}+1} + \dots + s_n) = L_n(s_1, \dots, s_n), \quad (31)$$

so that, for $n = 1, 2, 3$,

$$L_1(s_1) = [1 - 4r L_1(s_1)] F_1^4(s_1), \quad (32)$$

$$L_2(s_1, s_2) = 0, \quad (33)$$

$$\begin{aligned} L_3(s_1, s_2, s_3) &= -4r L_3(s_1, s_2, s_3) F_1^4(s_1+s_2+s_3) \\ &\quad + [1 - 4r L_1(s_1)] [1 - 4r L_1(s_2)] \\ &\quad \cdot [1 - 4r L_1(s_3)] F_3^4(s_1, s_2, s_3). \end{aligned} \quad (34)$$

Finally, the kernels are given by

$$L_1(s_1) = R(s_1) F_1^4(s_1), \quad (35)$$

$$L_2(s_1, s_2) = 0, \quad (36)$$

$$\begin{aligned} L_3(s_1, s_2, s_3) &= R(s_1)R(s_2)R(s_3) F_3^4(s_1, s_2, s_3) \\ &\quad \cdot R(s_1+s_2+s_3), \end{aligned} \quad (37)$$

$$\text{with } R(s) = [1 + 4r F_1^4(s)]^{-1}. \quad (38)$$

5. SIMULATION

5.1. Identifying structures composed of filters, sums and products

The Volterra kernels of order 1 given in (22), (28) and (35) correspond to standard linear filters. Those of order 3 given in (24),

(30) and (37) are sums of terms with general expression

$$A_1(s_1)B_1(s_2)C_1(s_3)D_1(s_1+s_2+s_3).$$

From (18), each term defines an elementary system of order 3 presented in Figure 6, where A_1 , B_1 , C_1 and D_1 are linear filters. For

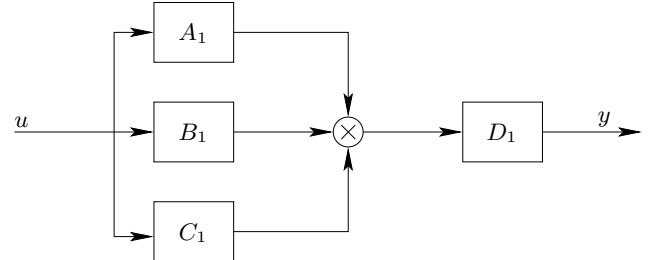


Figure 6: Elementary system of order 3.

instance, in (24), F_3 can be decomposed into two elementary systems as in Figure 3a: one corresponds to $A_1 = B_1 = C_1 = 1$ and $D_1 = T_3 F_1$ and is the cascade of an instantaneous cube power and the filter $T_3 F_1$ where $T_3 = -1/3$; the second corresponds to $A_1 = B_1 = C_1 = F_1$ and $D_1 = -T_3 F_1$ and is the cascade of a filter F_1 , an instantaneous cube power and the filter $-T_3 F_1$.

Thus, by identification, (22-24), (28-30) and (35-37) lead to the structures given in Figures 7, 8 and 9 for the third order structures of \mathcal{F}^1 , \mathcal{F}^4 and \mathcal{L} , respectively. Note that third order approximations of (12-13) would involve instantaneous loops whereas these structures have no loops and yield realizable systems composed of causal linear filters, sums and products in the time domain. The structure \mathcal{L}_3 makes the resonant filter $R(s)$ appear

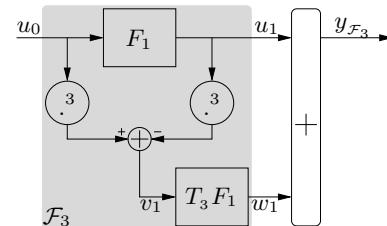


Figure 7: Third-order structure \mathcal{F}_3 of the system \mathcal{F} .

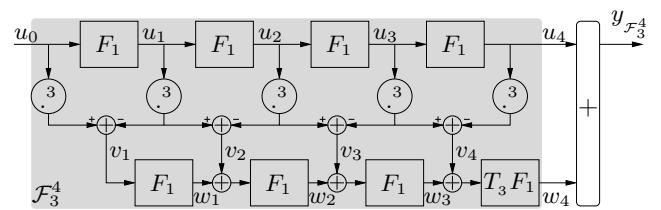


Figure 8: Third-order structure \mathcal{F}_4 of the system \mathcal{F}^4 .

only through an encapsulation of the four-stages system \mathcal{F}_3^4 . This corroborates the remark given in [5, p.51] even for the third order nonlinear case: the loop does not modify the low-pass properties of the filter.

Indeed, controlling the resonance (Q-factor) through the feedback-gain r modifies the filter R but does not affect the structure \mathcal{F}_3^4

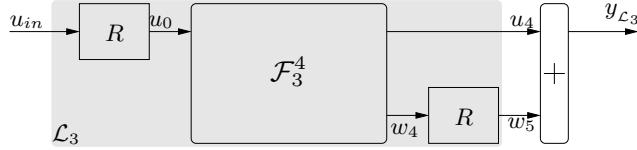


Figure 9: Third-order structure \mathcal{L}_3 of the system \mathcal{L} .

since the cut-off pulsation ω_c is controlled through I_c but not r . Now, the filter R is resonant but has not a low-pass behavior. Bode diagrams for $r \in \left\{0, \frac{1}{3}, \frac{2}{3}, 1\right\}$ are displayed in Figure 10.

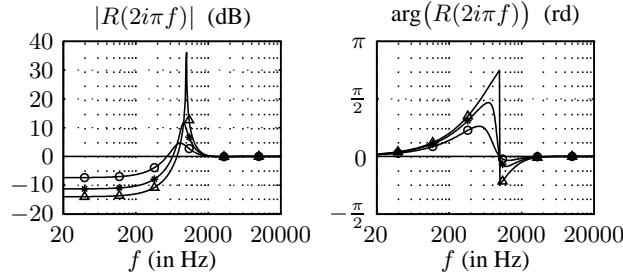


Figure 10: Bode diagrams of R for $r = 0$ (—), $r = \frac{1}{3}$ (○), $r = \frac{2}{3}$ (*) and $r = 1$ (△).

5.2. State-space representation

In this section, linear filters involved in the structure \mathcal{L} are reshaped into state-space representations

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t) \quad (39)$$

$$y(t) = Cx(t) + Du(t) \quad (40)$$

which define stationary linear systems with P inputs u , Q outputs y and the state x of dimension N . The vectors u , y and x have dimensions $P \times 1, Q \times 1, N \times 1$, respectively. The matrices A , B , C and D have dimensions $N \times N$, $N \times P$, $Q \times N$ and $Q \times P$, respectively.

5.2.1. Cascade of four filters \mathcal{F}_1^4

The cascade of four linear filters \mathcal{F}_1 with one input $u_F = u_0$ and four outputs $y_F = [u_4, u_3, u_2, u_1]^t$ (see Figure 8) admits the representation (39-40) with the state $x_F = y_F$ and

$$A_F = \omega_c \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad (41)$$

$$B_F = \omega_c [0 \ 0 \ 0 \ 1]^t, \quad (42)$$

$$C_F = I_4, \quad (43)$$

$$D_F = [0 \ 0 \ 0 \ 0]^t, \quad (44)$$

where I_4 denotes the 4×4 identity matrix.

5.2.2. Filter R

The filter R defined in (38) with one input u_R and one output y_R admits the representation (39-40) with the state

$x_R = [x, dx/dt, d^2x/dt^2, d^3x/dt^3]^t$ and

$$A_R = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\omega_c^4(1+4r) & -4\omega_c^3 & -6\omega_c^2 & -4\omega_c \end{bmatrix}, \quad (45)$$

$$B_R = [0 \ 0 \ 0 \ 1]^t, \quad (46)$$

$$C_R = [-4r\omega_c^4 \ 0 \ 0 \ 0], \quad (47)$$

$$D_R = 1. \quad (48)$$

5.2.3. Linear processing

The linear part of the Moog ladder filter corresponds to the upper stage of Figure 9, that is, the cascade of the filter R and the linear four-stage filter \mathcal{F}_1^4 . It admits a state-space representation, with $u_L = u_{in}$, $y_L = [u_4, u_3, u_2, u_1, u_0]^t$, $x_L = [u_4, u_3, u_2, u_1, x, dx/dt, d^2x/dt^2, d^3x/dt^3]^t$, and

$$\frac{dx_L(t)}{dt} = \left[\begin{array}{c|cc} A_F & B_F \cdot C_R \\ \hline \mathbf{0}_{4,4} & A_R \end{array} \right] x_L(t) + \left[\begin{array}{c} B_F \cdot D_R \\ \hline B_R \end{array} \right] u_L(t), \quad (49)$$

$$y(t) = \left[\begin{array}{c|cc} I_4 & \mathbf{0}_{4,4} \\ \hline \mathbf{0}_{1,4} & C_R \end{array} \right] x_L(t) + \left[\begin{array}{c} \mathbf{0}_{4,1} \\ \hline D_R \end{array} \right] u_L(t). \quad (50)$$

5.2.4. Processing of order 3

This part is composed of the intermediate and the lower stages in Figures 8-9. The intermediate stage is nonlinear but memoryless. It computes

$$v = [(u_3)^3 - (u_4)^3, (u_2)^3 - (u_3)^3, (u_1)^3 - (u_2)^3, (u_0)^3 - (u_1)^3]^t. \quad (51)$$

The lower stage is a cascade of four linear filters \mathcal{F}_1 with adders, a gain $T_3 = -1/3$ and a linear filter R . It admits a state-space representation with $u_{NL} = v$, $y_{NL} = w_5$, and

$x_{NL} = [w, dw/dt, d^2w/dt^2, d^3w/dt^3, w_4, w_3, w_2, w_1]^t$ where w is involved in the state-space representation of R , and

$$\frac{dx_{NL}(t)}{dt} = \left[\begin{array}{c|cc} A_R & B_R \cdot [T_3, 0, 0, 0] \\ \hline \mathbf{0}_{4,4} & A_F \end{array} \right] x_{NL}(t) + \left[\begin{array}{c} \mathbf{0}_{4,4} \\ \hline \omega_c I_4 \end{array} \right] u_{NL}(t) \quad (52)$$

$$y_{NL}(t) = [C_R \cdot D_R \cdot [T_3, 0, 0, 0]] x_{NL}(t) \quad (53)$$

5.3. Digital simulation without aliasing and results

The state-space representation of \mathcal{L}^3 is given by equations (49-53). The digital implementation of its linear parts is very standard. Methods such as bilinear or backward difference transforms and even redesigned versions of \mathcal{L}^1 have been deeply studied in [2]. Another way to preserve important features such as the *exact pole mapping with (r, ω_c)* consists in deriving the exact free-regime dynamics from the solution of (39), namely, $x(t) = \int_0^t \exp((A(t-\tau)).B.u(\tau)d\tau + \exp(A\tau).x(0))$ so that, denoting $x_n = x(nT)$ for the sampling period T ,

$$x_{n+1} = \exp(AT).x_n + \int_{nT}^{(n+1)T} \exp(A(t_{n+1}-\tau)).B.u(\tau)d\tau. \quad (54)$$

Finite dimensional approximations of $\mathbf{u}(t) = \sum_{n \in \mathbb{Z}} \mathbf{u}_n h(t-nT)$ with $h(t) = \sin(\pi t)/(\pi t)$ will yield digital filters. As a low order example, the approximation $h_1(t) = (1 - \frac{|t|}{T}) \mathbf{1}_{[-T, T]}(t)$ leads to

$$\mathbf{x}_{n+1} = \exp(AT) \cdot \mathbf{x}_n + \mathbf{B}_1 \cdot \mathbf{u}_{n+1} + \mathbf{B}_0 \cdot \mathbf{u}_n, \quad (55)$$

where $\mathbf{B}_1 = TE_1(T) - E_2(T)$, $\mathbf{B}_0 = E_2(T)$ with $E_1(t) = T^{-1} \int_0^t \exp(A(T-\tau)) \mathbf{B} d\tau$ and $E_2(t) = \int_0^t E_1(\tau) d\tau$. The output \mathbf{y}_n is computed from (40). The approximation due to h_1 means that the exact system is fed with a modified input with spectrum $\text{TF}[\mathbf{u}](f) [\text{sinc}(Tf)]^2$ rather than $\text{TF}[\mathbf{u}](f)$, where TF denotes the Fourier transform and $[\text{sinc}(Tf)]^2 = \text{TF}[h_1](f)$.

Now, the aliasing due to the cube powers in (51) can be rejected by encapsulating the digital system with an oversampling process at the input and an under-sampling process at the output. Here, the oversampling factor is 3. This factor improves the approximation due to h_1 since $[\text{sinc}(\xi)]^2$ decreases from 0 dB at $\xi = 0$ to only -0.8 dB at $\xi = 1/6$ rather than -7.8 dB at $\xi = 1/2$.

Results are presented in Fig. 11 for a sum of 2 square waves (437Hz, 443Hz) with a linear attack (0.5s) and a linear decay (0.3s). Parameters are $\omega_c = 2\pi f_c$ with $f_c = 1500$ Hz, $r = 0.15$ and $T = 1/44100$ s.

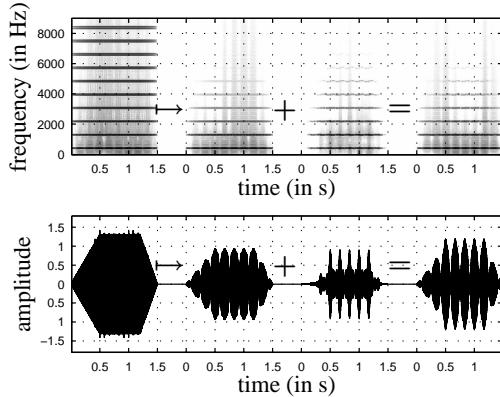


Figure 11: Spectrograms and signals of $u_{in} \mapsto u_4 + w_5 = y_{L3}$.

6. DISCUSSION

The validity of the third order structure is conditioned by that of the series expansion of $\tanh(u)$. Typical valid ranges for orders 1, 3 and 5 are $|u| < 0.5$, $|u| < 0.75$, $|u| < 1$. Compared to Fig. 7, a $(2N+1)$ -order structure \mathcal{F}_{2N+1} involves $N+1$ elementary filters F_1 and also instantaneous operators (powers, products, sums). Moreover, structures \mathcal{F}_{2N+1}^4 and \mathcal{L}_{2N+1} are built from \mathcal{F}_{2N+1} . Now, a way to improve the validity for a fixed order $2N+1$ consists in modifying coefficients T_{2k+1} ($1 \leq k \leq N$) so that they minimize a distance between $\tanh(u)$ and its $(2N+1)$ -order polynomial approximation $P_{2N+1}(u)$, globally on a u -range rather than near $u = 0$. This will introduce some ripples on $P_{2N+1}(u)$ but which do not affect the global behavior if sufficiently small (in particular, $P_{2N+1}(u)$ must preserve the sign of $\tanh(u)$ over the considered u -range).

7. CONCLUSION

In this paper, the Volterra series have been used to model a weakly nonlinear analog audio device. This formalism helps to transform

nonlinear differential systems (including loops) into an infinite set of algebraic equations from which the Volterra kernels are deduced. Each kernel isolates a sub-system attached to a monomial nonlinearity and monitors the exact associated sub-dynamics. In practice, keeping the very first kernels suffices to capture the distortion in a significant amplitude range, which characterizes the warmth of analog devices.

Structures which admits a realization in the time domain can be deduced from the Volterra kernels. In this paper, for each kernel, elementary and low-cost sub-systems have been identified, but other systematic identification procedures are also available, see e.g. [10]. Moreover, a truncated version of the series allows to reject aliasing for digital implementations. In practice, using lower oversampling factors can be sufficient, especially for naturally low-pass systems.

This formalism also proves to be useful for solving weakly nonlinear partial differential equations, see e.g. [11] for the nonlinear propagation in a brass.

8. REFERENCES

- [1] R. A. Moog, "A voltage-controlled low-pass high-pass filter for audio signal processing," in *17th Conv. Audio Eng. Soc.*, New York, USA, 1965, pp. 1–12.
- [2] T. Stilson and J. Smith, "Analyzing the Moog VCF with considerations for digital implementation," in *Proc. Int. Comp. Music Conf. (ICMC'96)*, Hong Kong, 1996, pp. 398–401.
- [3] A. Huovilainen, "Non-linear digital implementation of the Moog ladder filter," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 61–64.
- [4] T. E. Stinchcombe, "Derivation of the transfer function of the Moog ladder filter," [Online] http://mysite.wanadoo-members.co.uk/tstinchcombe/synth/Moog_ladder_tf.pdf, Tech. Rep., 2005.
- [5] J. Smith, "Virtual acoustic musical instruments: Review of models and selected research," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 2005, presentation Overheads.
- [6] F. Lamnabhi-Lagarrigue, *Analyse des Systèmes Non Linéaires*. Editions Hermès, 1994, ISBN 2-86601-403-0.
- [7] A. Isidori, *Nonlinear control systems*, 3rd ed. Springer Verlag, 1995.
- [8] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions*. New York: Dover, 1970.
- [9] M. Hasler, *Phénomènes non linéaires*. École Polytechnique Fédérale de Lausanne, Jan. 1999.
- [10] W. J. Rugh, *Nonlinear System Theory. The Volterra/Wiener approach*. Baltimore: The Johns Hopkins University Press, 1981.
- [11] T. Hélie and M. Hasler, "Volterra series for solving weakly non-linear partial differential equations: application to a dissipative Burgers' equation," *Int. Journal of Control*, vol. 77, no. 12, pp. 1071–1082, 2004.

ROBUST DESIGN OF VERY HIGH-ORDER ALLPASS DISPERSION FILTERS

Jonathan S. Abel*

Universal Audio, Inc.
Santa Cruz, CA 94060 USA
abel@uaudio.com

Julius O. Smith

CCRMA, Stanford University
Stanford, CA 94305 USA
jos@ccrma.stanford.edu

ABSTRACT

A nonparametric allpass filter design method is presented for matching a desired group delay as a function of frequency. The technique is useful in physical modeling synthesis of musical instruments and emulation of audio effects devices exhibiting dispersive wave propagation. While current group delay filter design methods suffer from numerical difficulties except at low filter orders, the technique presented here is numerically robust, producing an allpass filter in cascaded biquad form, and with the filter poles following a smooth loop within the unit circle.

The technique was inspired by the observation that a pole-zero pair arranged in allpass form contributes exactly 2π radians to the integral of group delay around the unit circle, regardless of the (stable) pole location. To match a given group delay characteristic, the method divides the frequency axis into sections containing 2π total area under the desired group-delay curve, and assigns a pole-zero allpass pair to each. In this way, the method incorporates an order selection technique, and by adding a pure delay to the desired group delay, allows the trading of increased filter order for improved fit to the frequency-dependent group delay. Design examples are given for modeling the group delay of a dispersive string (such as a piano string), and a dispersive spring, such as in a spring reverberator.

1. INTRODUCTION

In many media, waves propagate *dispersively*, with different frequencies traveling at different speeds. For instance, as described in Fletcher and Rossing [1], high frequencies slightly outrun low frequencies on “stiff” strings, making the higher overtones somewhat sharp. A pulse propagating on such a string is eventually “smeared out” along the string, as illustrated in Figure 1, with its high frequencies leading the way, and its low frequencies lagging.

To simulate such a system, waveguide synthesis models [2] are often used for their accuracy and efficiency. A waveguide model of a dispersive string segment is shown in Figure 2; propagation losses and dispersion are lumped into filtering at the waveguide ends. For musical strings such as for the piano, the attenuation filter $A(z)$ can be low-order and minimum-phase, yet perceptually exact, because there is very little loss in the string itself. The dispersion filter, in contrast, needs to be a high-order allpass whose job it is to delay different frequencies by different amounts according to the frequency-dependent propagation speed.

There are several approaches to designing allpass filters to achieve a prescribed delay vs. frequency. We now give a brief overview of some prior literature on this problem.

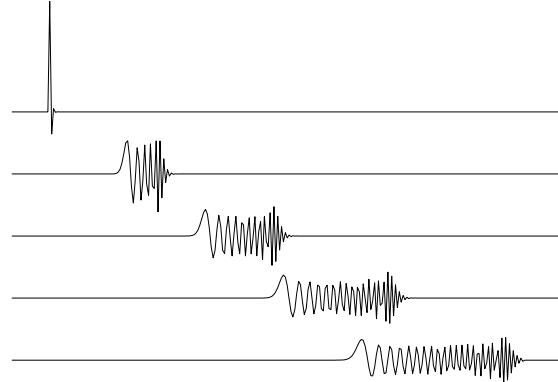


Figure 1: *Dispersive string propagation.*

Hilbert-transform methods, such as [3, 4], make use of the fact that the log-magnitude and phase of a minimum-phase spectrum form a Hilbert transform pair [5]. A group-delay filter can then be designed by integrating the group delay to form a desired phase response. An all-pole filter is fit to a minimum-phase frequency response having log magnitude equal to the Hilbert transform of half the desired phase response. The allpass filter is then formed by inverting the poles to generate corresponding maximum-phase zeros.

These nonparametric Hilbert-transform methods have a few drawbacks. They are not optimal, and they use the FFT in a manner that suffers significant time aliasing when the poles and zeros get too close to the unit circle in the complex plane. Perhaps a more serious difficulty is that the filter designed by these methods is in direct form, as opposed to factored form (which is often preferred in applications). It is often costly and numerically difficult to factor a high-order allpass filter form for implementation as cascade second-order sections. In addition, because the round-off error in direct-form filter coefficients can have a very large effect on the locations of poles and zeros, these methods can break down at extremely high model orders. Moreover, the internal all-pole filter-design methods used (such as LPC in [3]) can itself suffer numerical difficulties at the needed very high orders. Finally, there is no built-in model-order selection in these methods.

In [6], high quality stiff-string sounds were demonstrated using high-order allpass filters to simulate dispersion in a digital waveguide model. In [7], this work was extended by applying a least-squares allpass-design method [8] and a spectral Bark-warping technique [9] to the problem of calibrating an allpass filter of arbitrary order to recorded piano strings. They were able to correctly tune the first several tens of partials for any natural piano

*Jonathan Abel is also a consulting professor at CCRMA.

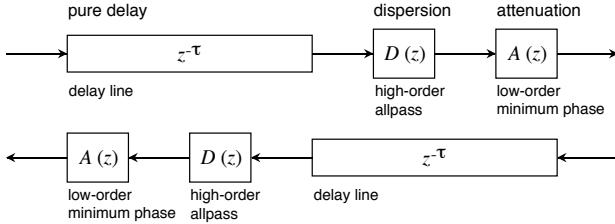


Figure 2: Dispersive waveguide section.

string with a total allpass order of 20 or less. Additionally, minimization of the L^∞ norm [10] has been used to calibrate a series of allpass-filter sections [11], and a dynamically tunable method, based on Thiran allpass filters, has recently been proposed [12].

Optimal allpass design methods are limited by numerical precision and computational cost. In particular, Bensa [11] reports difficulty in matching a piano-string dispersive delay using a model order greater than five. In a least-squares method for allpass design [8], the maximum order was 15 in five design examples, and numerical difficulties were noted when designing a particular delay equalizer at orders greater than 10. A detailed analysis of numerical issues is given in [10], along with techniques for roughly doubling the order that can be reliably designed. In particular, the numerical improvements enabled an order increase from 10 to 20 for a delay equalizer for a 7th-order lowpass filter. In [13], a particular order 80 lowpass delay equalizer was designed using linear programming methods; the order of another example in that paper was extended from 25 to 150 using the numerical improvements of [10], but this was classified as a “special case”, and is the highest order we have seen reported to date by such methods. In all of these methods, the allpass order is set *a priori*, rather than being automatically determined by the design method in some way.

The technique described in this paper, first presented at [14], is essentially nonparametric, with the allpass filter computed in *factored biquad form* directly from the desired group delay function. The method is based on an invariant feature of allpass group delays: A (complex) pole-zero pair arranged as an allpass filter will generate a group delay that peaks at the pole-zero frequency, and that has a constant area of 2π , irrespective of the pole location within the unit circle.

The group delay of a cascade of filter sections is the sum of the section group delays. So, to match a given group-delay characteristic, the frequency axis may be divided into *bands* such that the area under the desired group delay curve in every band is 2π radians. A pole-zero allpass section may then be assigned to each band, and the pole radius adjusted so that a specified portion of its group-delay area occurs within the band. An allpass filter approximating the desired group delay characteristic is formed by cascading a number of such allpass sections, covering the desired total frequency band.

Details of the method are given in § 3, with various design tradeoffs discussed in § 4. Applications to physical modeling synthesis of a stiff string and to emulation of a spring reverberator are considered as design examples. Finally, § 6 contains our summary and conclusions.

We begin by reviewing properties of the group delay of a pole-zero pair arranged as an allpass filter.

2. FIRST-ORDER ALLPASS PROPERTIES

Consider a pole p in the z -plane at normalized radian frequency¹ θ and radius ρ , with a complementary zero at $\zeta = 1/\bar{\rho}$. That is, the zero is at the same frequency θ , but inverse radius $1/\rho$. Then the pole $p = \rho \exp(j\theta)$ and zero $\zeta = (1/\rho) \exp(j\theta)$ form an allpass pair

$$G(z) = \frac{-\rho e^{-j\theta} + z^{-1}}{1 - \rho e^{j\theta} z^{-1}}, \quad (1)$$

as shown in Figure 3. The group delay of any linear time-invariant filter with transfer function $G(z)$ is defined as the negative derivative of the phase response $\angle G[\exp(j\omega)]$ with respect to frequency [15], i.e.,

$$\tau(\omega) \triangleq -\frac{d}{d\omega} \Im\{\log G(z)\}|_{z=\exp(j\omega)}, \quad (2)$$

where $\omega \in [-\pi, \pi]$, and is given in this case by

$$\tau(\omega) = \frac{1 - \rho^2}{1 + \rho^2 - 2\rho \cos(\omega - \theta)}. \quad (3)$$

As illustrated in Figure 3, the group delay $\tau(\omega)$ is symmetric in frequency about the pole-frequency $\omega = \theta$, with a peak delay of

$$\max_{\omega} \{\tau(\omega)\} = \frac{1 + \rho}{1 - \rho}, \quad (4)$$

which is achieved at the pole frequency $\omega = \theta$, and a minimum delay of

$$\min_{\omega} \{\tau(\omega)\} = \frac{1 - \rho}{1 + \rho}, \quad (5)$$

which is attained half way around the unit circle at $\omega = \theta + \pi$.

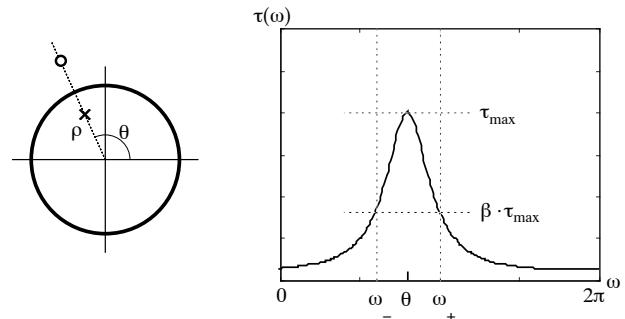


Figure 3: Allpass filter pole, zero. Group delay.

Now, what happens to the group delay $\tau(\omega)$ as the pole radius ρ is varied? Referring to Figure 4, as the pole moves towards the unit circle, the peak delay increases, while the bandwidth over which the delay is large decreases, concentrating around the pole frequency θ .

We might expect the group-delay peak to increase as the pole approaches the unit circle, but why does it also become narrow? What’s going on is that there is only so much group delay to “go around”—more precisely, the integral of the group delay around the unit circle is always 2π . Since the group delay is the negative derivative of the phase response with respect to frequency, its

¹We will assume the sampling rate is 1 so that the normalized radian frequency θ ranges between $-\pi$ and π .

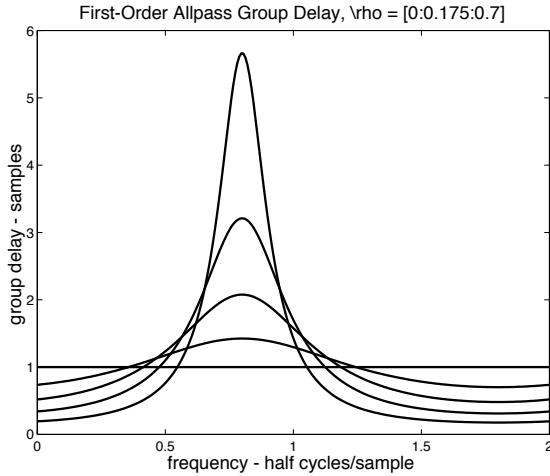


Figure 4: First-order allpass group delay, various ρ .

integral around the unit circle is simply minus the phase accumulated during one traversal of the unit circle, which is 2π per pole, regardless of where the pole is inside the unit circle:

$$\int_0^{2\pi} \tau(\omega) d\omega = \varphi(0) - \varphi(2\pi) = 2\pi \quad (6)$$

Note that only the pole contributes to this integral because the zero lies outside the unit circle. These facts can be readily seen by considering the graphical method for evaluating the phase response of a digital filter [15].

3. DISPERSION FILTER DESIGN

Recall that the goal is to design an allpass filter $D(z)$ to match a desired frequency-dependent delay, $\delta(\omega)$. The approach taken here constructs the allpass filter from (complex) first-order sections, taking advantage of the fact that each section has a 2π integrated delay, and a delay peak which may be arbitrarily located and scaled. The filter is formed by dividing the desired group delay into frequency bands, each having area 2π , as shown in Figure 5, and then modeling each delay band with its own allpass section.

The design procedure is as follows:

1. Add a constant delay to the desired frequency-dependent delay $\delta(\omega)$ so that it integrates to a desired multiple of 2π , call it N , where N is the desired allpass order.
2. Starting at DC, divide $\delta(\omega)$ into 2π -area frequency bands, as illustrated in Figure 5.
3. Fit a first-order (complex) allpass section $G_n(z)$ to each band as described below.
4. Cascade the first-order sections to form the allpass filter,

$$D(z) = \prod_{n=1}^N G_n(z). \quad (7)$$

What remains is to find the pole location for each band. The idea is to have the pole-zero pair delay $\tau(\omega)$ approximate the group

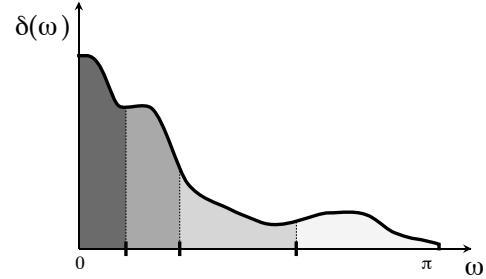


Figure 5: Segmenting $\delta(\omega)$ into 2π -area bands.

delay $\delta(\omega)$ in that band, and be small outside the band. The pole frequency is taken to be the band midpoint,

$$\theta = (\omega_+ + \omega_-)/2, \quad (8)$$

where the frequencies ω_{\pm} denote the left and right band edges. The pole radius is chosen so that the group delay at either band edge is a fraction β of the peak group delay, as illustrated in Figure 3. This controls the width of the group delay peak in each band, and determines a tradeoff between smoothness of fit to the desired group delay and the maximum “slew rate” of the allpass group delay (*i.e.*, its ability to follow small-bandwidth features in the desired group delay). Setting

$$\tau(\omega_{\pm}) = \beta \cdot \max_{\omega} \tau(\omega), \quad (9)$$

and using (3) and (4), we find

$$\rho(\beta) = \eta - [\eta^2 - 1]^{\frac{1}{2}}, \quad (10)$$

where

$$\eta = \frac{1 - \beta \cos \Delta}{1 - \beta}, \quad \Delta = (\omega_+ - \omega_-)/2. \quad (11)$$

Note that in the second step of the design procedure above, the initial band edge was set to zero. As the group delay $\delta(\omega)$ is even in frequency for real filters, this choice leads to first-order allpass sections appearing as complex conjugate pairs which may be combined to form biquads having real coefficients. Allpass filters with real coefficients also result by choosing the first band to be centered on DC. This can be done by setting the first band edge frequency to that at which the integral of $\delta(\omega)$ from DC is π . In this case, there will be two first-order allpass sections with real poles (one at DC and one at the Nyquist limit), and the rest will appear as complex conjugate pairs.

The list of band edges encodes all relevant delay information, with the band filters separately computed from their band-edge frequencies and a user-supplied β . As a result the design method is very efficient, requiring little more than an evaluation of (10) and (8) per designed biquad. It is also numerically robust, with numeric requirements nearly independent of model order.

A fair number of bands is often needed to capture the behavior of the desired delay, and in this case, the bands will be everywhere sufficiently narrow that $\rho(\beta)$ may be approximated by

$$\rho(\beta) \approx 1 - \left[\frac{\beta}{1 - \beta} \right]^{\frac{1}{2}} \Delta, \quad \Delta \ll 1. \quad (12)$$

The overlap parameter β is generally supplied by the user and independent of frequency, so that the root $[\beta/(1-\beta)]^{1/2}$ may be pre-computed. Under these conditions, the filter design is less costly than its implementation, and real-time manipulation of the frequency-dependent delay is inexpensive. Furthermore, it is well known that the coefficients of stable biquad sections can be interpolated from one to another without obtaining unstable intermediate filters.

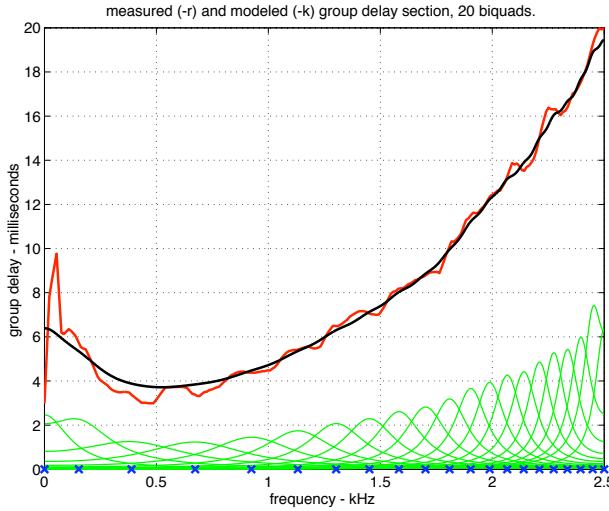


Figure 6: Spring element group delay model.

As an example design, Figure 6 shows a desired delay characteristic $\delta(\omega)$, the frequency-dependent portion of the measured time delay for a single traversal of a spring reverberator element. Frequencies in the neighborhood of 500 Hz arrive first, with delay increasing away from 500 Hz.

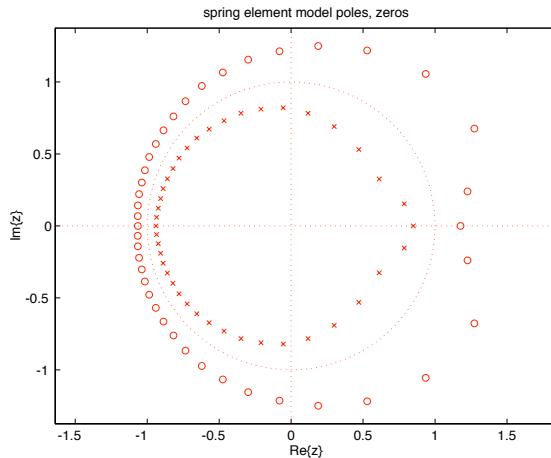


Figure 7: Spring element model poles and zeros.

Also shown in Figure 6 is the group delay of a 20-biquad all-pass filter designed to match the spring element delay; it's essentially a smoothed version of the desired delay. The pole frequencies of the designed allpass are shown on the frequency axis, along

with their associated group delays. In Figure 7, the allpass pole and zero locations are shown in the z -plane, the poles tracing a smooth loop inside the unit circle. Where the delay is large, the poles and zeros are more closely spaced, and are closer to the unit circle. This is consistent with the notion that the greater the delay, the smaller the bandwidth required to achieve a 2π delay integral. That the poles are closer to the unit circle in the presence of narrowly spaced bands can be seen from (12), where the pole distance to the unit circle is roughly proportional to bandwidth.

4. DISPERSION FILTER DESIGN TRADEOFFS

There are several parameters which may be adjusted to provide an improved fit to the desired group delay characteristic.

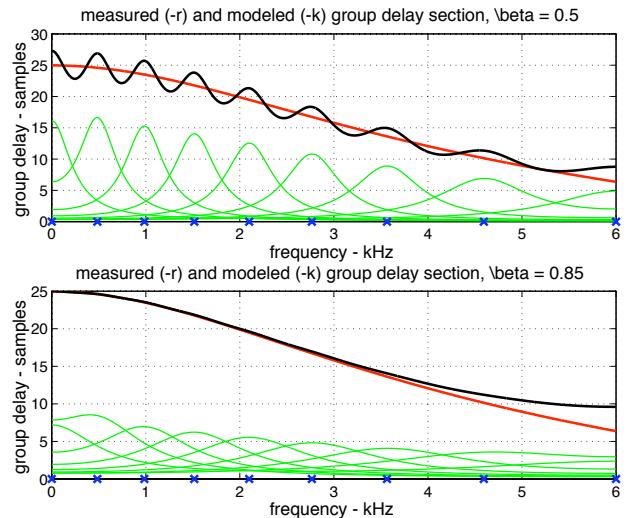


Figure 8: Example designs, $\beta = \{0.5, 0.85\}$.

The parameter β determines the extent to which successive band group delays overlap. Values of β close to one produce smooth group delays, whereas small values of β produce rippled group delays, but allow the designed filter to track narrow-bandwidth delay features such as sharp transitions from one delay to another. This is seen in Figure 8, showing the frequency-dependent delay of a length of piano string modeled using $\beta = 0.5$ and $\beta = 0.85$. Note how the higher β gives a smoother fit, but a greater “tracking error” near the Nyquist limit.

Since the band allpass sections are separately computed, there is no barrier to making β a function of frequency. For instance, β could be adjusted in proportion to a local measure of the smoothness of $\delta(\omega)$.

Adding a pure delay δ_0 to the desired group delay $\delta(\omega)$ allows additional, more closely spaced allpass sections to be used, and provides a more accurate fit. Figure 9 shows a dispersive delay modeled using five biquads (top) and ten biquads (bottom). The additional biquads provided when the pure delay is added result in a better fit across the band.

If computational resources are limited, it may be desired to model the group delay only in a band of interest. For instance, when designing audio dispersion filters for vibrating-string models, it is typically most cost-effective to obtain an allpass filter that

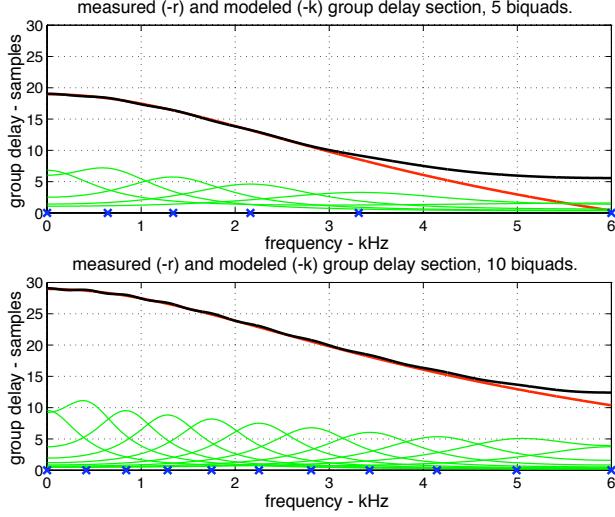


Figure 9: Example designs, $\delta_0 = \{0, 10\}$ samples.

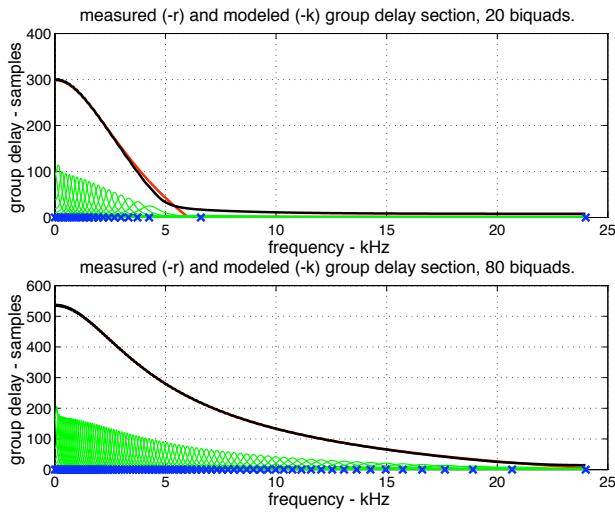


Figure 10: Example designs, $\{6, 24\}$ kHz bandwidth.

correctly tunes only the lowest-frequency partial overtones, where the number of partials correctly tuned is significantly less than the total number of partials present, as in [7]. In Figure 10, we see that 80 biquads are needed to model the delay out to 24 kHz, whereas only 20 biquads are needed to model the delay out to 6 kHz. Such a filter would be efficient for implementing the string dispersion needed for a low note on a piano.

5. PIANO-STRING DISPERSION FILTER

Figure 11 shows the impulse response based on measurements from note F1 of a piano-string (top) and that of the cascade of a minimum phase attenuation filter and an allpass dispersion filter (bottom) designed by the method of this paper using 64 biquad sections (see Figure 2 for string-model context). As can be seen,

the measured and model impulse responses appear virtually identical. Frequency-domain plots are omitted, as there is no visible error, except for a slight divergence (on the order of a tenth of a millisecond) at the Nyquist limit (12 kHz).

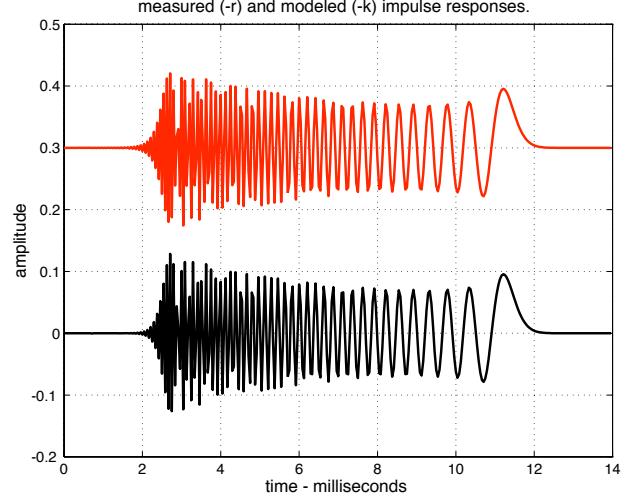


Figure 11: Order 128 impulse-response fit for a piano string, note F1.

6. SUMMARY AND FUTURE WORK

In this paper, we described a new method for allpass dispersion filter design having the following features:

- Simple and fast
- Model order automatically determined
- Filters computed in factored biquad form
- Applicable to variable allpass designs in real time
- Effective at extremely high orders
(numerically robust)

Example designs were shown for stiff string and spring-reverb modeling. Other applications to consider include stiff strings in other instruments (e.g., cello), springs, and dispersive acoustic tubes. More generally, the method can be useful for high-order group delay equalization.

For future work, we think it could be effective to use this method to initialize one of several optimal methods for group-delay allpass design, to see if their convergence and numerical performance can be improved. Another idea is to model the pole locus as points along a parametrized curve, such as a series of spline curves, and to minimize group-delay error with respect to certain parameters of such a “pole locus curve”; while such a constrained locus of poles cannot be expected to be optimal in any global sense, the resulting optimized design may yield a useful improvement over the noniterative starting-point described in this paper. For this and other variations, it can help to carry out optimizations in smaller band slices corresponding to cascaded allpass sections. Finally, we think time-varying group-delay filter applications appear promising.

7. REFERENCES

- [1] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments, 2nd Edition.* New York: Springer Verlag, 1998.
- [2] J. O. Smith, III, *Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects.* [Online] <http://ccrma.stanford.edu/~jos/pasp/>, Mar. 2006.
- [3] B. Yegnanarayana, "Design of recursive group-delay filters by autoregressive modeling," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 30, no. 4, pp. 632–637, Aug. 1982.
- [4] G. R. Reddy and M. N. S. Swamy, "Digital all-pass filter design through discrete hilbert transform," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'90)*, Albuquerque, USA, 1998.
- [5] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.
- [6] A. Paladin and D. Rocchesso, "A dispersive resonator in real-time on MARS workstation," in *Proc. Int. Comp. Music Conf. (ICMC'92)*, San Francisco, USA, 1992, pp. 146–149.
- [7] D. Rocchesso and F. Scalcon, "Accurate dispersion simulation for piano strings," in *Proc. Nordic Acoust. Meeting (NAM'96)*, Helsinki, Finland, June 12-14 1996, pp. 407–414.
- [8] M. Lang and T. I. Laakso, "Simple and robust method for the design of allpass filters using least-squares phase error criterion," *IEEE Trans. Circuits and Systems—I: Fundamental Theory and Applications*, vol. 41, no. 1, pp. 40–48, 1994.
- [9] J. O. Smith III and J. Abel, "Bark and ERB bilinear transform," *IEEE Trans. Speech and Audio Proc.*, vol. 7, no. 6, pp. 697–708, Nov. 1999.
- [10] M. Lang, "Allpass filter design and applications," *IEEE Trans. Sig. Proc.*, vol. 46, no. 9, pp. 2505–2514, 1998.
- [11] J. Bensa, "Stiff piano string modeling: Computational comparison between finite differences and digital waveguide," in *Proc. 148th Meeting Acoust. Soc. Am.*, San Diego, USA, Nov. 2004, nov. 15–19.
- [12] J. Rauhala and V. Välimäki, "Tunable dispersion filter design for piano synthesis," *IEEE Sig. Proc. Letters*, vol. 13, no. 5, pp. 253–256, May 2006.
- [13] Z. Jing, "A new method for digital all-pass filter design," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 11, pp. 1557–1564, Nov. 1987.
- [14] J. S. Abel, J. O. Smith, and J. Bensa, "An allpass filter design method with application to piano string synthesis," in *Proc. 149th Meeting Acoust. Soc. Am.*, Vancouver, Canada, May 2005.
- [15] J. O. Smith, III, *Introduction to Digital Filters.* [Online] <http://ccrma.stanford.edu/~jos/filters/>, Sept. 2005, online book.

CONSISTENCY OF TIMBRE PATTERNS IN EXPRESSIVE MUSIC PERFORMANCE

Mathieu Barthet, Richard Kronland-Martinet, Sølvi Ystad

CNRS - Laboratoire de Mécanique et d'Acoustique
 31, chemin Joseph Aiguier
 13402 Marseille Cedex 20, France
 {barthet|kronland|ystad}@lma.cnrs-mrs.fr

ABSTRACT

Musical interpretation is an intricate process due to the interaction of the musician's gesture and the physical possibilities of the instrument. From a perceptual point of view, these elements induce variations in rhythm, acoustical energy and timbre. This study aims at showing the importance of timbre variations as an important attribute of musical interpretation. For this purpose, a general protocol aiming at emphasizing specific timbre patterns from the analysis of recorded musical sequences is proposed. An example of the results obtained by analyzing clarinet sequences is presented, showing stable timbre variations and their correlations with both rhythm and energy deviations.

1. INTRODUCTION

This article is part of a larger project aiming at analyzing and modelling expressive music performance. To follow the classification made by Widmer and Goebl in [1], we use an "Analysis-by-measurement" approach the first step of which is to define the performer's expressive patterns during the interpretation. Various approaches to identify performance rules have been proposed. Amongst these, the "Analysis-by-synthesis" approach developed at the KTH [2] [3] which relies on musical theory knowledge has led to the establishment of context-based performance rules. They mainly take into account the tempo and the intensity of musical notes or phrases, either to emphasize their similarity (grouping rules), or to stress their differences (differentiation rules). Another approach has been proposed by Tobudic and al. [4], leading to a quantitative model of expressive performance based on artificial intelligence to reproduce the tempo and dynamic curves obtained from performances played by musicians. All these studies have mainly focused on rhythm and intensity variations.

In the present study, an investigation on the consistency of timbre expressive variations in music performance is proposed. A comparison between timbre, rhythmic and intensity expressive variations is also made, since the correlations between these parameters are probably strong. For this purpose, a professional clarinettist was asked to play a short piece of music (the beginning of a Bach's Cello Suite) twenty times. The choice of the instrument was mainly related to the fact that it is self-sustained and that the performer easily controls the sound event after note onset. In addition, earlier studies by Wanderley [5], report that the movements of a clarinettist are highly consistent for various music performances of the same piece. Since these movements seem to be closely linked to the interpretation, we also expect the expressive parameters to be highly consistent. In a previous study [6], the investigation of the performance parameters of a physically modelled clarinet indicates that timbre is involved in musical

expressivity and seems to be governed by performance rules. In this study, we aim at checking if timbre also follows systematic variations on natural clarinet sounds.

We shall first describe a general methodology developed to analyze and compare recorded musical performances in order to point out consistency of timbre, rhythmic and intensity patterns in expressive music performance. An application of this methodology to twenty recorded musical sequences of the same clarinettist is then given. Eventually, we show that timbre, as rhythm and intensity, follows systematic variations, and that correlations exist between these parameters of the expressivity.

2. METHODOLOGY

In this section, we describe a general methodology to analyze and compare musical performances from recorded monophonic sequences.

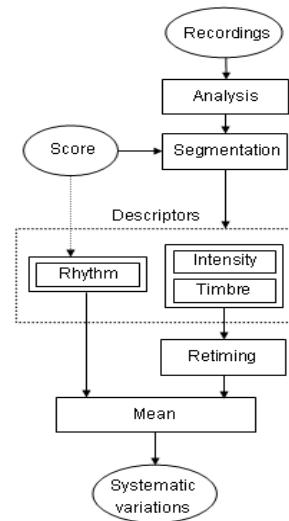


Figure 1: Methodology.

The hypothesis we want to verify is that when a performer plays several times a piece with the same musical intention, patterns of rhythm, intensity, and timbre over the course of the piece, show a high consistency. For that purpose, we derive from the recorded sequences some performance descriptors characterizing the musical expressivity of a performer at a note-level. We then calculate the mean of the performance descriptors to determine if

their variations are systematic. Figure 1 sums up the different steps of the methodology.

2.1. Sound corpus

If the expressive variations introduced by the musician resist an averaging over a large amount of performances played with the same musical intention, they can be considered as systematic. We thus need a large number of recordings of the same musical piece performed as similarly as possible to identify the consistency of musical expressivity patterns.

To avoid influence from room acoustics, the recordings of these performances have to take place in a non-reverberant acoustical environment.

In the following, we will note N , the number of notes of the musical melody, and n will refer to the n^{th} note played. We will note P the number of recorded performances, and p will refer to the p^{th} one.

2.2. Note segmentation

Note segmentation is an intricate task and is slowed down by difficulties such as the detection of two successive notes having the same pitch, or silences between musical phrases. In [7], the author describes a way to determine the timing of the note onsets from musical audio signals. Here the task can be facilitated by the a priori knowledge of the score giving an estimation of the fundamental frequencies. The note segmentation process is composed of two parts, the pitch tracking, consisting in estimating the fundamental frequencies of the recorded sequences, and the segmentation.

2.2.1. Pitch tracking

A lot of studies have been carried out on this subject. A review can be found, for instance, in [8].

In our case, we use the software LEA from the Genesis company to generate filtered sequences from the original recordings which only contain the fundamental frequencies of the notes played during the performances. Since these new sequences only contain a single frequency-varying sinusoidal component, it is pertinent to calculate their analytic signals $Z^p(t)$. Finally, we obtain the instantaneous fundamental frequencies $F0(t)$ due to the following relation:

$$F0^p(t) = \frac{1}{2\pi} \frac{d\phi^p(t)}{dt} \quad (1)$$

where $\phi^p(t)$ is the phase of $Z^p(t)$.

2.2.2. Segmentation

As we have a large number of recordings, we built an automatic note segmentation method. It is also important that the process remains identical for each sequence in order to segment each note in the same way before the averaging of the performance descriptors.

Our method is based on the analysis of the fundamental frequency variations $F0(t)$. As a matter of fact, it presents instabilities at the transitions between notes. A detection of these instabilities gives the timing of the transitions. Hence, we obtain the note timings T_n^p for each note n and for each performance p .

2.3. Performance descriptors

Rhythm descriptors are obtained from the rhythm indications of the score and from data obtained after the note segmentation part. Intensity and timbre performance descriptors are high-level descriptors derived from a time/frequency representation of the recorded sequences.

2.3.1. Rhythm descriptors

We obtain the note durations D_n^p of each performance p from the note timings T_n^p . The rhythm deviation descriptor ΔD_n^p is defined as the difference between the note durations given by the score D_n^{score} (called nominal durations) and the durations of the notes played during the performances D_n^p (called effective durations):

$$\Delta D_n^p = D_n^p - D_n^{\text{score}} \quad (2)$$

It is a discrete time function calculated for each note.

2.3.2. Intensity and timbre descriptors

We derive these descriptors from a time/frequency analysis of the recorded sequences. They are also discrete functions of the time, but depending on the time bins defined by the analysis. In the following, $d^p(t)$ will refer to the descriptors calculated over the entire course of the performance p , and $d_n^p(t)$ will refer to the values of $d^p(t)$ restricted to the duration of the note n .

2.4. Retiming of the performance descriptors

To verify our hypothesis, we have to calculate the average of the performance descriptors $d^p(t)$ over all recorded sequences. As the performances are played by a human musician, the durations D_n^p of the notes are slightly different. In order to synchronize all these performance descriptors, a retiming process is thus necessary. This retiming consists in temporal contractions or dilations. We will denote by Γ these transformations. In our case, we do not need to realize an audio time-stretching keeping the frequency content of the signal as it is described for instance in [9], since the descriptors we derive from the signals are not going to be heard.

The dilation coefficient α_n^p will be chosen so as to adjust the duration D_n^p of the descriptors $d_n^p(t)$ to the mean duration $\overline{D_n}$ of the notes over all the recorded performances. Thus, we will alter the performance descriptors as little as possible. If $\alpha_n^p > 1$, Γ is a dilation, and if $\alpha_n^p < 1$, Γ is a contraction.

The mean note duration $\overline{D_n}$ is given by:

$$\overline{D_n} = \frac{1}{P} \sum_{p=1}^P D_n^p \quad (3)$$

The dilation coefficient α_n^p is then given by:

$$\alpha_n^p = \frac{\overline{D_n}}{D_n^p} \quad (4)$$

Finally, the retiming transformations Γ applied on the note performance descriptors $d_n^p(t)$ can be written as:

$$\Gamma : d_n^p(t) \longmapsto \Gamma[d_n^p(t)] = d_n^p(\alpha_n^p(t)) \quad (5)$$

2.5. Systematic and random variations of the descriptors

Once the synchronization of the note performance descriptors is realized, we calculate their mean to point out systematic behavior, and their standard deviation to characterize random fluctuations. The mean note descriptors $\overline{d_n}(t)$ over all the recorded performances are given by:

$$\overline{d_n}(t) = \frac{1}{P} \sum_{p=1}^P d_n^p(\alpha_n^p(t)) \quad (6)$$

Random fluctuations of the descriptors are characterized by their standard deviation $\sigma_{\overline{d_n}}(t)$.

Hence, if the behavior of the performance descriptors $d_n^p(t)$ is systematic over all the performances, they will be strongly correlated with their mean value, and the standard deviation will be rather low. Furthermore, the mean will be a smoothed version of the descriptors, loosing the random fluctuations. On the contrary, if the behavior of the descriptors is not systematic, then their mean will differ from the descriptors, and the standard deviation will be high.

We also evaluate the consistency of the performance descriptors by calculating the correlation coefficients $r^2(\Gamma[d])$ of the retimed observation p of the descriptor d and the $P - 1$ others. The mean of these correlation coefficients $r^2(\Gamma[d])$ measures the strength of the correlations.

3. AN APPLICATION TO THE CLARINET

3.1. Sound corpus

We asked the professional clarinetist C. Crousier to play the same excerpt of an Allemande of Bach (see Figure 2) twenty times with the same musical intention. This excerpt is destined to be played rather slowly and expressively (its score indication is "Lourd et expressif"). A 48 bpm reference pulsation was given to the musician by a metronome before the recordings. It was then stopped during the performance to give the player the freedom to accelerate or slow down. The reference pulsation allowed us to calculate the notes' nominal durations given by the score D_n^{score} and thus evaluate the performer's rhythmic deviations.



Figure 2: Excerpt of Bach's Suite II B.W.V. 1007 (Allemande).

The recordings of the clarinet were made in an anechoic chamber with a 44100 Hz sample frequency. We used SD System clarinet microphones fixed on the body and the bell of the instrument, avoiding recording problems due to the movements of the instrumentalist while playing. Both microphones have a flat frequency response (+/- 2,5 dB) in the frequency range where the timbre descriptors are calculated [100 - 8000 Hz].

3.2. Performance descriptors extraction

We applied the Short Time Fourier Transform (STFT) on each recorded musical sequences. Hanning windows of 1024 samples

and 75 % of overlap have been used for this purpose. Timbre descriptors were calculated considering $N_{harm} = 15$ harmonics.

3.2.1. Rhythm descriptor

We normalized the rhythm descriptors ΔD_n^p given by the equation (2) according to the notes' nominal durations. Its mean expressed as a deviation percentage is hence given by:

$$\overline{\Delta D_n}(\%) = 100 \cdot \frac{\overline{\Delta D_n}}{D_n^{score}} \quad (7)$$

3.3. Intensity variations

We characterize intensity variations by the Root Mean Square envelopes of the recorded sequences.

3.4. Timbre variations

Three timbre descriptors adapted to clarinet sounds have been chosen to describe the timbre variation during musical performance: the spectral centroid, which can be regarded as the mean frequency of the spectrum, the spectral irregularity correlated to the differences between odd and even harmonics, and the odd and the even descriptors, correlated to the energy of odd and even harmonics in the spectrum. We will present a particular case showing that these timbre descriptors contain complementary information.

3.4.1. The Spectral Centroid

The definition we use for the spectral centroid SCB is the one given by Beauchamp in [10]. It differs from the classical definition by the presence of a term b_0 that forces the centroid to decrease when the energy in the signal is low, avoiding an increase of the spectral centroid at the end of the notes. It has been shown that the spectral centroid is correlated to the brightness of a sound and correlates with the main control parameters of the clarinetist, i.e. the mouth pressure and the reed aperture [11] [12]. It is defined by:

$$SCB_n^p(t) = \frac{\sum_{k=1}^{N_{sup}} k \cdot A_k(t)}{b_0 + \sum_{k=1}^{N_{sup}} A_k(t)} \quad (8)$$

where the $A_k(t)$ are the modulus of the STFT considered up to the frequency bin N_{sup} . The term b_0 is given by:

$$b_0 = \max[A_k(t)], \quad k = 1, \dots, N_{sup} \quad (9)$$

3.4.2. The Spectral Irregularity

Krimphoff has pointed out the importance of the spectral irregularity [13]. We here derived a new definition from the one Jensen gave in [14], including a term b_1 in the denominator for the same reason as for the spectral centroid. The spectral irregularity IRR_B can then be defined by:

$$IRR_B^p(t) = \frac{\sum_{h=1}^{N_{harm}-1} (A_{h+1}(t) - A_h(t))^2}{b_1 + \sum_{h=1}^{N_{harm}} A_h(t)^2} \quad (10)$$

where:

$$b_1 = (\max[A_h(t)])^2, \quad h = 1, \dots, N_{harm} \quad (11)$$

3.4.3. The Odd and Even descriptors

The lack of even harmonics compared to odd ones is characteristic of the clarinet timbre (see for instance [15]), but their energy increases as the breath pressure increases (see [12]). A measure of odd and even harmonics energy compared to the overall energy is given by the Odd and Even descriptors defined below. We will show a particular case where they explain subtle timbre variations of the clarinet.

$$Odd_n^p(t) = \frac{\sum_{h=0}^{N_{odd}-1} A_{2h+1}(t)}{\sum_{h=1}^{N_{harm}} A_h(t)} \quad (12)$$

$$Even_n^p(t) = \frac{\sum_{h=1}^{N_{even}} A_{2h}(t)}{\sum_{h=1}^{N_{harm}} A_h(t)} \quad (13)$$

where N_{odd} is the number of odd harmonics, and N_{even} the number of even harmonics.

4. CONSISTENCY OF THE PERFORMANCE DESCRIPTORS

4.1. Strong correlations between the performances

The mean correlation coefficients of the retimed performance descriptors are given in table 1. The high values of $r^2(\Gamma[d])$ point out a strong consistency of the rhythm descriptor ΔD , the intensity descriptor RMS , and the timbre descriptors SCB , IRR , Odd and $Even$, over the various performances.

d	ΔD	RMS	SCB	IRR	Odd	Even
$r^2(\Gamma[d])$	0.76	0.89	0.84	0.71	0.74	0.74

Table 1: Mean correlations of the performance descriptors

4.2. Rhythmic patterns

Figure 3 both shows the fundamental frequencies and durations of the notes to be played by the performer as indicated on the score and the mean of the measured fundamental frequencies and durations of the notes for the 20 performances. It points out that the total duration of the performances is on average longer than the nominal one (about 1s difference). In order to play expressively, the performer effects rhythmic deviations compared to the rhythm indicated on the score. These rhythmic deviations lead to local accelerandi or decelerandi. In general, certain notes tend to be shortened by the performer (case where $\overline{\Delta D_n} < 0$, see for example notes 10 and 20), whereas certain notes tend to be lengthened (case where $\overline{\Delta D_n} > 0$, see for example notes 5, 11 and 12). From 7s to the end, almost all the notes are played longer, up to twice their nominal durations for some of them. This reveals a slowing down of the tempo by the performer which is very common in endings of musical phrases. These results are in agreement with the "Duration Contrast" and "Final Retard" rules defined by Friberg and colleagues, which model the two rhythmic principles indicated above [2].

4.3. Intensity patterns

As can be seen on figure 5, the phrase begins forte and then there is a progressive decrescendo until the end of the phrase. The energy

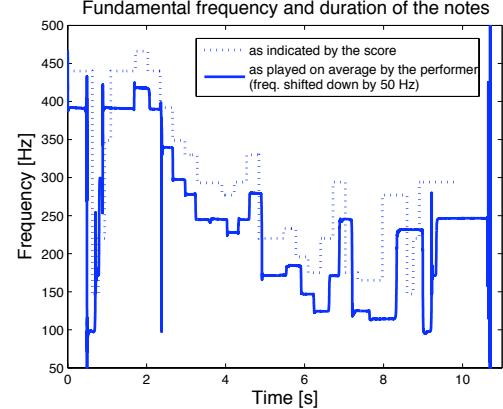


Figure 3: Fundamental frequency and duration of the notes as indicated by the score (dotted) and as played on average by the performer (solid). The measured fundamental frequency (solid) has been shifted down by 50 Hz to point out the rhythmic differences between the nominal and the effective note durations.

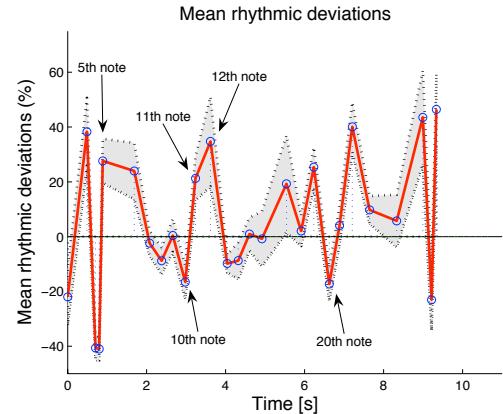


Figure 4: Mean rhythmic deviations (bold), +/- the standard deviation (dotted). The timing of the notes are indicated by circles.

peak at time bin 1600 may be due to the fact that the note played has a very low frequency (147 Hz) and is more radiated by the clarinet.

4.4. Timbre patterns

Figure 6 represents the mean spectral centroid and its standard deviation. It is strongly correlated to the mean intensity variations in a monotonous and increasing way ($r = 0.80$). Further, within the duration of notes, strong changes of spectral centroid can be observed for all the performances. This can easily be seen for the fifth note (around time bin 200), for which the difference between the lowest and the highest value of the mean spectral centroid (at the note onset and close to the note offset), is about 500 Hz. A neat change in the note's timbre is audible (sounds are given at <http://w3lma.cnrs-mrs.fr/~kronland/DAFx06>). It is worth noticing that after the attack of this note the values of the odd descriptor decrease and the values of the even descriptor increase (figure 8) so that the global energy of even harmonics grows faster than the

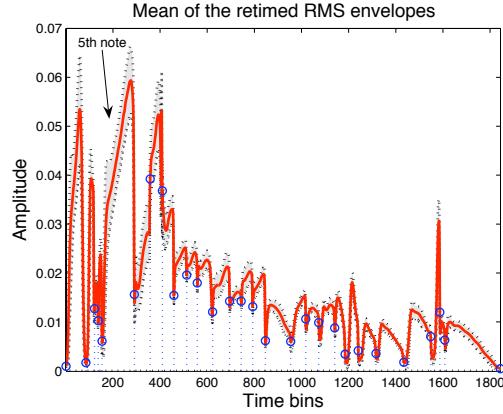


Figure 5: Mean RMS envelope (bold), +/- the standard deviation (dotted). The note transitions are indicated by circles.

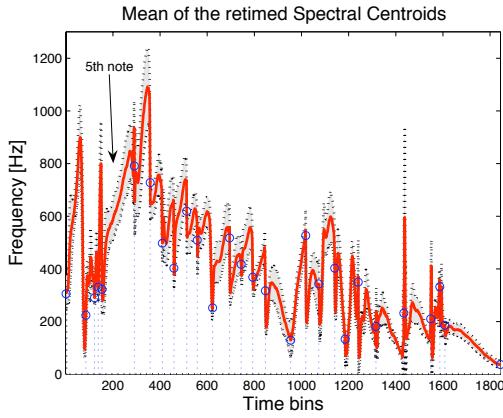


Figure 6: Mean Spectral Centroid (bold), +/- the standard deviation (dotted). The notes transitions are indicated by circles.

global energy of odd harmonics. This does not mean that the phenomenon is equally distributed over the whole spectrum. If it was so, the spectral irregularity would decrease (the energy of even harmonics getting closer to the odd one) but this is not the case (see figure 7). The spectral irregularity remains quite stable within the duration of the fifth note after its attack. This is probably due to the fact that even harmonics grow faster than odd ones in a narrow frequency area. Indeed, we showed in the case of synthetic clarinet sounds that an increase of the breath pressure induces an energy increase of high-order harmonics and more particularly for even harmonics around the reed resonance frequency [12]. This is due to the non-linear coupling between the excitor (the reed) and the resonator (the bore) and explains the increase of the brightness of the sound.

4.5. Timbre and Intensity correlation

Figure 5 and 6 show that there is a strong correlation between the spectral centroid and the envelope. Nevertheless, the spectral centroid of a note depends on its fundamental frequency and this biases the observation. Hence, we have normalized the spectral centroid according to the mean instantaneous fundamental frequency

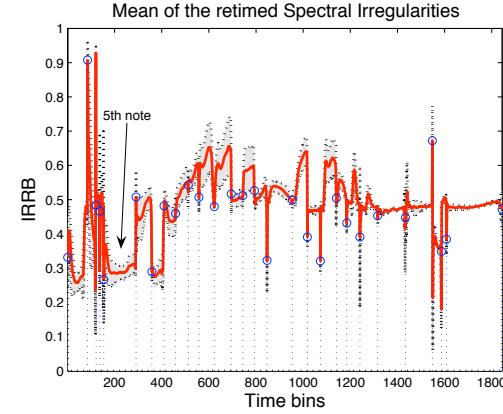


Figure 7: Mean Spectral Irregularity (bold), +/- standard deviation (dotted). The note transitions are indicated by circles.

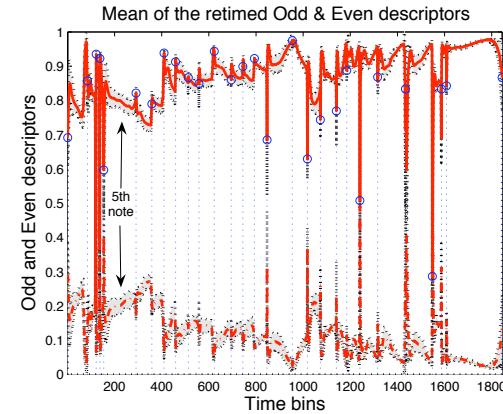


Figure 8: Mean Odd and Even descriptors (bold), +/- the standard deviation (dotted). The note transitions are indicated by circles.

as follows:

$$\overline{SCB'(t)} = \frac{\overline{SCB(t)}}{\overline{F0(t)}} \quad (14)$$

Figure 9 represents the normalized spectral centroid $\overline{SCB'}$ as a function of the normalized mean RMS envelope for two categories of notes, the short and piano ones, and the long and forte ones. It is worth noticing that these two categories of notes seem to follow different kinds of trajectories. Indeed, the spectral centroids of the short and piano notes increases very quickly compared to the envelope, whereas the spectral centroids of the long and forte notes seems to increase less rapidly than the envelope. This should be verified on a longer excerpt including a greater number of long and forte notes to cover a wider range of pitches, since the two trajectories below the diagonal correspond to the same pitch. The correlations we made are only qualitative but proves that there is a link between the variations of intensity, timbre and rhythm during the playing.

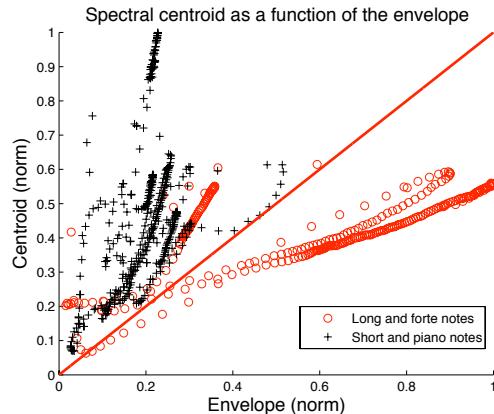


Figure 9: Spectral centroid as a function of the RMS envelope.

5. CONCLUSION AND FURTHER WORKS

Analysis and comparisons of recorded clarinet performances of an excerpt of a Bach Suite played several times by the same performer with the same musical intention, pointed out systematic and random variations of certain timbre descriptors. The present study confirmed former results obtained from performances played on a digital clarinet [6]. Strong correlations have been observed between the spectral centroid and the RMS envelope during the performances. In spite of these correlations, the spectral irregularity seems to be less correlated to the intensity, indicating that timbre changes are not only a consequence of intensity variations. More precise investigations on this topic are however needed to clarify the relations between timbre and intensity. Qualitative results show that timbre and intensity patterns also seem to be related to rhythmic deviations over the course of the musical piece. Multidimensional analysis are to be conducted to better understand the links between timbre, rhythm and intensity variations.

This study represents a first step to show the importance of timbre variations in expressive music performances. We plan in the future to use signal processing techniques to alter the interpretation and to evaluate the importance of variations in rhythm, intensity and timbre by psychoacoustic tests.

6. ACKNOWLEDGEMENTS

We would like to thank C. Crousier for his excellent advice and participation in this project. We are also grateful to the GENESIS company for providing the LEA software.

7. REFERENCES

- [1] G. Widmer and W. Goebel, "Computational models of expressive music performance: The state of the art," *J. New Music Research*, vol. 33, no. 3, pp. 203–216, 2004.
- [2] A. Friberg, "A quantitative rule system for musical performance," Ph.D. dissertation, Department of Speech, Music and Hearing, Royal Institute of Technology, Stockholm, 1995.
- [3] J. Sundberg, *Integrated Human Brain Science: Theory, Method Application (Music)*. Elsevier Science B.V., 2000, ch. Grouping and differentiation two main principles in the performance of music, pp. 299–314.
- [4] A. Tobudic and G. Widmer, "Playing Mozart phrase by phrase," ÖFAI-TR-2003-02, Tech. Rep., 2003.
- [5] M. Wanderley, *Gesture and Sign Language in Human-Computer Interaction: International Gesture Workshop*. Springer Berlin, Heidelberg, 2002, ch. Quantitative analysis of non-obvious performer gestures, p. 241.
- [6] S. Farner, R. Kronland-Martinet, T. Voinier, and S. Ystad, "Timbre variations as an attribute of naturalness in clarinet play," in *Proc. 3rd Computer Music Modelling and Retrieval Conf. (CMMR05)*, Pisa, Italy, 2005, pp. 45–53.
- [7] S. Dixon, "On the analysis of musical expression in audio signals," *Storage and Retrieval for Media Databases, SPIE-IS&T Electronic Imaging*, vol. 5021, pp. 122–132, 2003.
- [8] E. Gomez, "Melodic description of audio signals for music content processing," Pompeu Fabra University, Barcelona, Tech. Rep., 2002, [Online] <http://www.iua.upf.es/mtg/publications/Phd-2002-Emilia-Gomez.pdf>.
- [9] G. Pallone, "Dilatation et transposition sous contraintes perceptives des signaux audio: application au transfert cinéma-video," Ph.D. dissertation, Aix-Marseille II University, Marseille, 2003.
- [10] J. W. Beauchamp, "Synthesis by spectral amplitude and "brightness" matching of analyzed musical instrument tones," *J. Audio Eng. Soc.*, vol. 30, no. 6, pp. 396–406, 1982.
- [11] P. Guillemain, R. T. Helland, R. Kronland-Martinet, and S. Ystad, "The clarinet timbre as an attribute of expressiveness," in *Proc. 2nd Computer Music Modelling and Retrieval Conf. (CMMR04)*, Esbjerg, Denmark, 2004, pp. 246–259.
- [12] M. Barthet, P. Guillemain, R. Kronland-Martinet, and S. Ystad, "On the relative influence of even and odd harmonics in clarinet timbre," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 351–354.
- [13] J. Krimphoff, S. McAdams, and S. Winsberg, "Caractérisation du timbre des sons complexes, II Analyses acoustiques et quantification psychophysique," *Journal de Physique IV, Colloque C5*, vol. 4, 1994.
- [14] K. Jensen, "Timbre models of musical sounds," Ph.D. dissertation, Department of Computer Science, University of Copenhagen, 1999.
- [15] A. Benade and S. Kouzoupis, "The clarinet spectrum: Theory and experiment," *J. Acoust. Soc. Am.*, vol. 83, no. 1, pp. 292–304, Jan. 1988.

REAL-TIME DETECTION OF FINGER PICKING MUSICAL STRUCTURES

Dale E. Parson

Agere Systems, Allentown, Pennsylvania, USA

dparson@agere.com

ABSTRACT

MIDIME is a software architecture that houses improvisational agents that react to MIDI messages from a finger-picked guitar. They operate in a pipeline whose first stage converts MIDI messages to a map of the state of instrument strings over time, and whose second stage selects rhythmic, modal, chordal, and melodic interpretations from the superposition of interpretations latent in the first stage. These interpretations are nondeterministic, not because of any arbitrary injection of randomness by an algorithm, but because guitar playing is nondeterministic. Variations in timing, tuning, picking intensity, string damping, and accidental or intensional grace notes can affect the selections of this second stage. The selections open to the second stage, as well as the third stage that matches second stage selections to a stored library of composition fragments, reflect the superposition of possible perceptions and interpretations of a piece of music. This paper concentrates on these working analytical stages of MIDIME. It also outlines plans for using the genetic algorithm to develop improvisational agents in the final pipeline stage.

1. MIDIME SOFTWARE PIPELINE

MIDIME is a software architecture that distills the structure of finger-picked string music at several levels of abstraction. It accepts MIDI input from a guitar, analyzes playing to determine low level (finger patterns, meter, accents, tempo, root, scale and chords) and high level (composition fragments) musical intent, and generates accompaniment MIDI streams for one or more synthesizers. This paper is about the architecture and algorithms of MIDIME's fully working analysis stages.

Figure 1 illustrates the MIDIME pipeline. MIDI input through Stage 3 and MIDI output are in full working form. Stage 4 is in a working prototype state. Each stage in Figure 1 is a software thread that analyzes data from the preceding stages and writes its interpretations to its output data table. A data table is a memory resident data structure which its writer updates and which downstream writers read. Each data table is a first-in first-out circular queue of rows owned by its writer. A writer controls its data table by locking the single data row that it is working on until it makes a change that it must expose to downstream readers; the writer then releases the lock, locks the following row, and copies the completed row into the new row, where it repeats the cycle. This is a conventional queuing architecture, with the novelty that reader threads entail the overhead of blocking on a lock only when polling determines that they need access to the row under construction.

The contents of a data table depend on its pipeline stage. A row in a Stage 1 data table contains guitar string state for a given time period. A Stage 2 row contains musical structure data for a given time period. A Stage 3 row contains indices showing where the current performance matches the state of a composition map

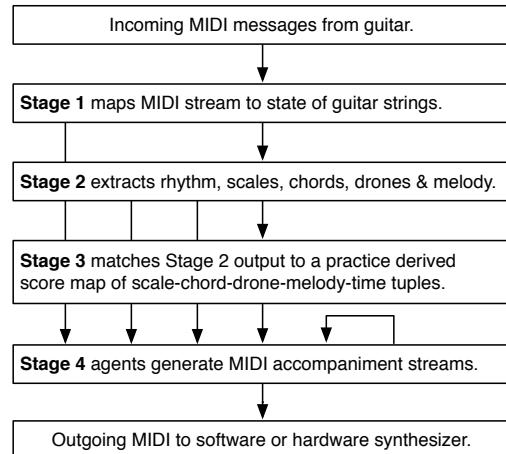


Figure 1: MIDIME pipeline architecture.

derived from practice performances. A Stage 4 row contains outgoing MIDI messages. Each writer is alternately a reader that waits for new data from upstream writers, and a writer that emits another row to downstream readers.

Overall, the pipeline of Figure 1 acts as a reactive system that responds to incoming messages within application time constraints. Each writer has the job of waiting for data tailored to its application requirements within the pipeline, transforming the data to meet the requirements of the next downstream writer, and releasing that data. Analysis steps are sometimes complex, but they are always restricted to a tractable single level of abstraction.

2. GUITAR STRINGS OVER TIME

Stage 1 converts a stream of finger-picked MIDI messages to a two dimensional matrix of guitar string state over time, where the most recent row is the current state of the strings, and each column is a string. Writer constructor parameters establish the number of strings, MIDI channel-to-string mappings, and minimum noteon velocity and duration. Stage 1 discards any note with velocity or duration below its minimum threshold as a transient.

The main jobs of the Stage 1 writer are to map MIDI channels-to-string positions, map pitch bend messages to discrete semitones while discarding extraneous pitch bends, record string / note state in a Stage 1 row, and release that row to downstream readers. The guitar synth uses one MIDI channel per string.

Table 1 shows a short trace of the state of a pluck on the E note at the second fret of the fourth string, followed by a slide to the F at the third fret. The left column shows incoming MIDI messages and the right column shows Stage 1 data rows as they appear in the MIDIME debug graphical user interface. A MIDI trace includes buffer row number, message type, channel, velocity for noteon and

magnitude for bend, and arrival time in milliseconds since the start of the process. A Stage 1 trace shows row number, changing string number, with subsequent lines showing the state of all six strings: musical note and octave on the string (e.g., E4), “~” showing cumulative bend, “^” showing string velocity, “o” showing volume setting, “<” showing note start time in milliseconds, “>” showing duration for notes just terminated, and a hex line for MIDI control signals such as foot pedal messages. The zeroed volume and control lines are elided from most of the Stage 1 trace in Table 1.

0: bend,c=4,b=0x2068	0: string 4, stage1 row 0:
2549	--- --- E4 --- --- ---
1: noteon,c=4,p=E4,v=122	~0 ~0 ~8192 ~0 ~0 ~0
2550	^0 ^0 ^122 ^0 ^0 ^0
2: bend,c=4,b=0x2074	o0 o0 o0 o0 o0
2569	<0 <0 <2550 <0 <0 <0
3: bend,c=4,b=0x2088	0> 0> 0> 0> 0>
2598	0x0 0x0 0x0 0x0 0x0 0x0
4: bend,c=4,b=0x2098	1: string 4, stage1 row 1:
2619	--- --- E4 --- --- ---
5: bend,c=4,b=0x20a2	~0 ~0 ~8258 ~0 ~0 ~0
2729	^0 ^0 ^122 ^0 ^0 ^0
6: bend,c=4,b=0x2092	<0 <0 <2550 <0 <0 <0
2789	0> 0> 329> 0> 0> 0>
7: bend,c=4,b=0x2070	2: string 4, stage1 row 2:
2820	--- --- F4 --- --- ---
8: bend,c=4,b=0x2042	~0 ~0 ~8206 ~0 ~0 ~0
2849	^0 ^0 ^122 ^0 ^0 ^0
9: bend,c=4,b=0x22b8	<0 <0 <2879 <0 <0 <0
2879	0> 0> 0> 0> 0>

Table 1: MIDI & Stage 1 trace of a semitone slide on string.

The most substantial job of the Stage 1 writer is filtering the many pitch bend messages emitted by the guitar synth. The Roland GR-33 guitar synth does not send a new noteon message when sliding into a new semitone. Instead, it emits a series of pitch bend messages centered at 8192 (0x2000), with the GR-33 configured to bend down from 8192 to 0 over an octave range, and to bend up from 8192 to 16383 (0x3fff) over an octave range. This configuration gives a semitone bend value of 8192/12 or about 683 (0x2ab). Table 1 shows the slide over the fret that exceeds the 0x2ab threshold in the last MIDI line with a bend of 0x22b8 at time 2879 ms., about 1/3 of a second after the noteon message showing the string pluck. Stage 1 integrates this MIDI stream into three rows that show the E4 pluck, the end of E4 when crossing the fret at 2550+329 = 2879 ms., and the F4 note starting at 2879 ms. Each “~” bend within a string’s Stage 1 row is residual bend left over after computing the fret crossings. As in the Table 1 example, Stage 1 always interprets *slurs* on a sounding string (*i.e.*, hammering onto a string, pulling off, sliding and stretching with a fretting finger) as new string state with zero time elapsed from the previous string state, e.g., E4 start time 2550 + duration 329 = F4 start time 2879 in this example. Fresh *plucks* with a picking finger always show non-zero time elapsed between note states on a string, *i.e.*, last note time + duration < next note time. Distinguishing plucks from slurs is critical for Stage 2 analysis, since plucks distinguish meter and tempo in finger picking patterns.

Table 1 shows a 3-to-1 reduction in the number of MIDI-to-Stage 1 events delivered to downstream readers. The overall trace for this test run of various left hand slurs shows about a 5-to-1 reduction, which is fairly typical for guitar playing and for GR-33 sensitivity and stability. Transitory pitch bends accumulate in the Stage 1 “~” entries, but only semitone-crossing pitch bends

advance a Stage 1 row. There is no reduction in Stage 1 data size; entries are largely redundant with previous rows, in the interest of making the entire state of all strings at a point in time easily accessible to downstream analysis. The typical 5-to-1 reduction in events does greatly reduce the processing demands on downstream analysis, however. The Stage 2 writer activates for as few as one fifth of the incoming MIDI messages, and when it does, the Stage 1 data rows that appear in Figure 2 are structured to ease analysis of finger picking patterns and string-oriented note analysis.

3. MUSICAL STRUCTURES OVER TIME

3.1. Finger Patterns, Meter and Tempo

When a Stage 2 writer receives a new Stage 1 row, its first step is to analyze rhythm. It starts with rhythm because subsequent analyses of scale, chord, drones and melody depend on performer-established time frames. Scales and drones are long term entities. Chords are usually short term entities that may cycle through long term patterns. Establishing long and short term temporal boundaries requires establishing a temporal reference.

Unlike many computer music systems, MIDIME does not impose an external, metronome-like temporal grid on incoming or outgoing notes. Finger picking is intrinsically a repetitive, rhythmic activity that establishes its own temporal boundaries and cycles. The picking hand plucks the strings in series of repeating rhythmic patterns while the other hand frets, slides, hammers, pulls and stretches the strings in order to select and alter pitch. The timing and force of finger plucks in a pattern show considerable surface variation, but behind this variation is an intent to create a cyclic pattern. The problem for Stage 2 is to find patterns in data representing musical intent, in rhythm or tone, while dealing with seemingly random fluctuations in surface features. There may be more than one possible rhythmic or tonal interpretation of incoming performance data; that is OK, because it gives Stage 4 improvisers the opportunity to go into unexpected directions that are empirically related to the human performance. The approach of Stage 2 is to locate regularities that are abstractions of the incoming notes. This temporary gridding is counteracted by Stage 4 writers who map the Stage 2 abstracted data back onto the real time and note space being played.

The example rhythm analysis of this section uses the finger picking pattern of Figure 2. The thumb (T) picks the first of 8 eighth notes in a 4/4 measure, followed by eighth notes on the index finger (I) on an intermediate string and the middle finger (M) on the first string. This pattern repeats, then the thumb strikes again on the seventh eighth note, and on the eighth there is a rest while the thumb prepares to strike again. This repetition of the thumb is not normally found in bluegrass, which often uses the three fingers in basic triplets. Thumb repetition slows the tempo and places accents on the first, fourth, and seventh eighth notes. Sometimes the index or middle finger dampens its string instead of plucking.



Figure 2: 4/4 Finger pattern with accents at 1, 2.5, and 4.

The algorithm to extract meter and tempo from Stage 1 data works in the following steps.

1. Collect pointers to Stage 1 string plucks into a 24 element array; collect records of the millisecond intervals since the previous respective plucks into a corresponding array of integers. An array size of 24 allows multiple repetitions of all possible three-finger patterns. There may be remainder plucks at the beginning or end of the array when the picking pattern does not divide 24 plucks evenly, or when picking goes through a pattern or tempo transition. Slurs do not contribute because only plucks represent right hand picking patterns. When the array fills, the oldest pluck is discarded at the start of event processing.

2. Build a temporary array of pointers to the elements of the above interval array, and sort this temporary array by the interval time in ascending order, giving access to the shortest interval first.

3. Traverse this sorted array and build a third array of bins. The smallest interval goes into the first bin, along with all succeeding intervals that are not more than 20% longer than the least interval in the bin. This 20% threshold was originally 16% in anticipation that a given unit of time could be subdivided into either 2 or 3 or (2×3) for conventional note subdivisions such as quarter-to-eighth note or quarter-to-triplet subdivisions; $1/(2 \times 3) = 1/6 = 16.7\%$. This attempt was too tight for typical playing; relaxing it to 20% made matching more consistent. As soon as a pluck interval exceeds 20%, it becomes the least value in a new bin; longer intervals go into that bin until exceeding its 20% threshold, creating a new bin, and the process continues until all 24 intervals are placed into bins.

4. Each bin is averaged by dividing its sum by the number of intervals in the bin. This number gives average picking time for that bin, with the bin representing some part of a pattern. The arithmetic division for averaging, like all division in MIDIME, is scaled integer division rather than slower floating point division.

5. Take the entire interval time in the 24 element array of plucks as a conceptual “measure” — this time is entirely empirical and is not tied to any rigid framework — and begin subdividing it into grid boundaries. First divide this total time into half, giving three snap points, one at the start (0 time), one at the end (total time), and one halfway between. For each bin from step 4, build a new bin that moves the value in the step 4 bin to its closest snap point, if and only if this snapping step does not move the bin average more than 20%. If any snap would exceed 20% in either direction, try a new snap by dividing the total time by the next step in the series (2, 3, 4, 6, 8, 12, ...). This series again represents splitting time repeatedly into halves and triplets. Eventually a subdivision of total time is found that allows snapping each average bin interval by less than 20%. In the worst case the total time would be divided down to 1 millisecond snap units, giving no snapping, but any regularity in the performer’s finger picking patterns avoids this degenerate state. Create a snapped interval for each bin when the process converges.

6. It is now possible to divide each snapped interval value by the snap unit to come up a small integer for each bin, and small integer ratios across bins that display repetition representing finger picking patterns. This step is implicit. The snapped bins of step 5 are sufficient for fast, machine level block memory comparisons.

7. Look for repeating patterns in the values of the bins of step 5. Find the shortest repeating pattern of snapped interval values that covers the largest number of elements in the 24 element interval array. For example, repeating pattern Z-X-Y-X-Y Z-X-Y-X-Y would not subdivide further because these two five-element pat-

terns organize ten total intervals, while an X-Y X-Y pattern would organize only the four elements at the end of the array. A pattern such as X-Y-X-Y-X-Y, in contrast, would factor into three adjacent X-Y patterns covering six contiguous intervals.

The algorithm proceeds by starting with a pattern length of one half the 24 elements, using the C library function memcmp to compare adjacent subsequences of snapped intervals. After comparing all adjacent pairs of sequences at a given length, the algorithm reduces the proposed pattern length by 1, and again compares adjacent subsequences of snapped intervals. It proceeds down to a proposed pattern length of 1, always storing the smallest successful pattern length that matches the greatest number of intervals. If it finds no matches, then it treats the entire array of intervals as one, non-repeating pattern. This condition usually indicates that the picking pattern or tempo is changing.

Table 2 makes these steps concrete with an example using real data. *Raw time* shows measured time intervals since previous plucks in milliseconds. The smallest value in each 20% bin is highlighted in the raw time row (*i.e.*, 125, 156, 189 and 333). *Bin average time* averages raw values in each bin, yielding average bin intervals 143, 175, 195 and 341. Splitting the total time of 5.125 seconds for this sample yields an interval snap of 160, a rounding of $(5125/32) = (5125/2^5)$, or 5 halvings of the total concrete “measure.” This is the coarsest splitting of the total time that snaps each average bin time within 20% of its value. Pattern matching then determines that a repeating pattern seven intervals in length covers the last 14 intervals in this example; the pattern is 1-1-1-1-1-1-2 snap units; the final 7 are highlighted. These are the eighth notes of Figure 2 with the unplucked rest residing halfway through the final 2 interval; the first eighth note of Figure 2 is the last pluck of the detected pattern. Rhythm analysis also normalizes pluck MIDI velocities as scaled multiples of the average pluck velocity for each slot in a detected pattern, giving a velocity pattern of 1-1-3-1-0-4-3 for this table’s data. The string numbers are not used in analysis, but for this data they are 2-1-3-2-1-3-3. Finally, the fingers used to pluck this data are I-M-T-I-M-T-T. Note the two-eighth-note relative timing of the adjacent thumbs, and the scaled velocities of 3 and 4 for the thumb. Stage 4 processing can use the final seven slots of this table to determine tempo and accents and to schedule output improvisation timing and velocity.

A player will occasionally dampen a string in this pattern so that no pluck will be detected, sometimes intentionally, and will sometimes hit an extra string by accident. These variations add intervals that match combined intervals or subintervals of the pattern. Rhythm analysis determines if there are any trailing sequences of intervals after the final pattern sequence whose total snapped times add up to the pattern time. If it finds them, it marks them as “tail patterns” that include missing or extra plucks. There are none in Table 2, although there are 6 plucks that add 8 snap units highlighted near the top of the right column, preceding the first occurrence of the pattern. The index finger damped its string for the out-of-place 2. Rhythm analysis treats trailing sequences that sum to pattern time as additional cases of the pattern for purposes of scheduling output. Downstream improvisation has access to snapped and raw times in planning its output note scheduling.

Irregularities in playing and transitions in pattern and tempo cause periods of loss of pattern detection. There are two ways for Stage 4 accompaniment agents to deal with periods of desynchronization. The first is to treat such periods as *rubato*, *i.e.*, intrinsically desynchronized periods of playing. In this approach a Stage 4 agent takes the entire 24 interval pluck time as the mea-

raw time in milliseconds	bin average time	snapped time	snaps
186	175	160	1
358	341	320	2
153	143	160	1
333	341	320	2
201	195	160	1
148	143	160	1
194	195	160	1
348	341	320	2
144	143	160	1
337	341	320	2
203	195	160	1
146	143	160	1
178	175	160	1
179	175	160	1
193	195	160	1
125	143	160	1
339	341	320	2
191	195	160	1
156	175	160	1
177	175	160	1
173	175	160	1
189	195	160	1
141	143	160	1
333	341	320	2

Table 2: Temporal intervals for a seven-pluck finger pattern.

sure, subdividing it for note generation according to the agent's own algorithm. The guitar player's and the agent's note timing weave in and out of overlap in temporal consonance / dissonance transitions. The second, more conservative approach is to program a Stage 4 agent to save a copy of the interval ratios and tempo of the most recent repeating Stage 2 pattern. This agent generates notes that adhere to this pattern and tempo. An example is a bass playing agent emitting a rhythmically stable bass line initialized from repetitive Stage 2 output. The guitar player can follow this emitted bass line, or the guitar player can force transition to a new meter or tempo by playing consistently for rhythmic pattern detection. The guitar player and the Stage 4 agent can pass control of rhythm back and forth as part of improvisation. Prototype MIDIME agents have used both approaches successfully.

Despite having $O(n^3)$ time complexity on the interval array, rhythm analysis and in fact all processing through Stage 3 has a total latency in reacting to a MIDI message that is under the 1 millisecond resolution of the operating system's time function. The combination of machine-level bit and logical operations and the use of scaled integer arithmetic instead of floating point, along with a fast processor and small bounds on the size of data being matched, help to keep pattern matching fast. Stage 1 events have about a 2-to-1 ratio to the number of Stage 2 events because most Stage 2 events are driven by plucks; Stage 1 slurs and note off events contribute tangentially to Stage 2 output.

3.2. Scales, Drones, Chords and Melody

Both scale and chord matching collect recent notes into *12-bit bit vectors*. A vector comprises the bottom 12 bits of a 16-bit C++ integer, with bit position 2^0 representing note C, 2^1 note C#, ..., 2^{11} note B. A bit vector discards octave and repeated note information. Scale matching uses an array of 12 data structures to track pluck times and to sum the number of pluck appearances of each note C .. B in the last four measures, counting a given note only once per

measure, where a measure is the finger pattern time just discussed. Thus a note C .. B can be counted 0 to 4 times in four measures.

After updating pluck counts and retiring any plucks older than four measures, scale analysis starts by using only the notes with the highest number of plucks. For example, if some notes appeared 4 times, only those notes would contribute their bits into a 12-bit scale vector. Scale analysis uses pattern matching described next to match this 12-bit vector to a scale, recording the scale if it matches. It then tries again with all notes with counts of 4 or 3, then again with 4 or 3 or 2, and finally with 4 or 3 or 2 or 1. The intent in starting with 4 is to discard transient, chromatic notes that appear infrequently during initial matching. If this approach discards too many notes to find an effective match, the later attempts with more notes may succeed. All four attempts are always made, but a later scale match replaces an earlier match of fewer notes only if the later match is a better match.

Scale analysis looks for a match in two loops. The first, *biased loop* iterates through optional 16-bit *scale configuration parameters* supplied by the user. If the user knows scales in advance, he or she can configure Stage 2 of the pipeline to try these scales. A 16-bit scale parameter consists of a 12-bit note pattern as before, plus a 4-bit number 0 .. 11 representing the root of the scale, C .. B, in the top 4 bits of the 16-bit vector. The biased loop works by bitwise ORing these configuration parameter bits into the played, empirical bits extracted from the performance as described by the last paragraph, and attempting to match that vector. This is a top down approach.

Scale configuration parameters are optional, and even when they are available, scale matching also uses an *empirical loop* to try to match the empirical bits. This matching loops through an optional set of *tonic configuration parameters*, which are some subset of the values 0 .. 11 representing a tonic of C .. B. If these optional parameters are missing, MIDIME simply tries all possible tonics 0 through 11 in combination with the empirical bits, by shifting the tonic bits into the top 4 bit positions of a 16-bit vector. Scale analysis maps a given 16-bit vector to a possible scale by using the 16-bit vector as an integer index into a 53248-entry table of scales (12 tonics + a key for unknown tonic = 13 keys $\times 2^{12}$ combinations of 12 notes = 53248). Here is a possible match for the note pattern 0x0a5 in the key of D, *i.e.*, notes C-D-F-G

```
const uint16_t MidimeModeTable[53248] = { // lots of entries
    0x2a5, // 0x20a5 --> D minor_pentatonic, distance 1
```

The comment after “//” shows that 0x20a5, C-D-F-G rooted in D, maps to the value 0x2a5, notes C-D-F-G-A, with a distance of 1. This is the D minor pentatonic scale. The distance shows that the actual data misses the table's entry by 1 bit. The total difference in number of bits is the *Hamming distance*. It measures how good the match is, and this table stores mappings only with distances of 2 or less. Invalid mappings appear as 0 in the table, *i.e.*, no notes on.

This table is generated by another program and compiled into C++ MIDIME. Consulting the table takes small constant time to determine a matching scale. Hamming distance is determined by exclusive-ORing the biased or empirical note vector played with the scale vector proposed by the table, giving an integer with 1 in each bit position where the real notes differ from the table's notes. There would be 1 bit for the missing A note in this example. Scale analysis uses this Hamming difference vector as an index into a compiled Hamming weight table that returns the number of bits set to 1 for that entry. The Hamming weight tells how far the match is

from perfect, and scale analysis stores the best match it has seen so far. At the end of its two loops, scale analysis uses the best match from biased and empirical analysis, using empirical to break a tie. Scale analysis also stores the tonic associated with the best match.

In the absence of optional scale configuration parameters, scale matching tends to be conservative, for example preferring major or minor pentatonic scales to more complex scales by using only notes with high pluck counts in the last four measures. Biased matching, in contrast, tends to prefer configuration parameter scales when they fit with real note data. This result works well for improvisation. In the absence of an anticipated scale, downstream improvisers will have conservative reactions in the absence of concrete data; in cases where the user supplies anticipated scales, the improvisers will use them as long as they fit with the real notes.

Drone matching looks to see an open string plucked repeatedly, without pitch change, over at least 4 finger pattern measures. If a string's pitch changes in these measures, it is dropped from the 12-bit drone vector, and if its pitch drops below previously recorded values, that new pitch becomes the baseline open string pitch.

Chord analysis considers only plucked notes currently sounding on the strings. It maintains a note for each sounding string, discarding low velocity notes as possible mistakes, combining the others into a 12-bit vector, then consulting a chord mapping table and computing Hamming distance similar to scale matching. Chord matching uses only empirical note data. It does not interact with long-term scale matching in order to allow for playing "outside" a scale's harmonic structure, a practice common in jazz. When these data are sparse, chord matching makes conservative selections such as simple triads.

Finally, melody matching uses the most recent pluck that is greater than the average velocity within its measure, as determined by meter velocity calculations. This matching aligns well with the practice of using the thumb for melody when finger picking.

4. COMPOSITION MATCHING AND ACCOMPANIMENT

Construction of Stage 3 of the pipeline is recently completed. It uses traces of Stage 2 scale-chord-drone-melody state that a user builds by saving and merging performances of composition fragments. Stage 3 looks for saved scale-chord-drone sequences that match current Stage 2 data within a parameterized Hamming distance limit, similar to Stage 2 Hamming-based matching, by comparing performance to stored fragments. Stage 3 identifies known composition fragments in real time to allow Stage 4 improvisers to make long term, complex improvisation plans.

Stage 4 players are in a prototype state. Agents have chromosomes consisting of long and short term genes of four types: rhythm, melody, harmony and timbre. A gene is a unit of typed code that contributes to a long or short term plan for that gene type. A gene's plan is a reaction to something played, or to a Stage 3 map of where playing appears to be heading, or to another Stage 4 player's performance.

5. RELATED WORK AND CONCLUSION

Roads gives an outline of the tasks required for rhythm analysis, along with a substantial list of references in his chapter on pitch and rhythm recognition [1]. Hamanaka, et. al., have designed a system that uses hidden Markov models to eliminate surface deviations while analyzing note onset times [2], and that uses the gen-

erative theory of tonal music to infer hierarchical metrical structure from MIDI performance data [3]. Their techniques comprise statistical analyses of surface variations in meter in a fixed-tempo, fixed-duration set of performances.

Papers on determining edit distance with insertions and deletions in musical score retrieval, a non-real-time generalization of the bit vector Hamming distances used for MIDIME scales and chords, can be found in [4, 5]. Toussaint shows application of Hamming distance to rhythm analysis [6].

MIDIME differs from meter analysis systems cited above in extracting cyclic rhythmic information that is intrinsic in repeated patterns used by finger picking guitarists. By using the most recent 24 real-time string plucks for temporal background, and by looking for patterns only in the performance data as opposed to matching string plucks to a library of known finger patterns, MIDIME can adapt to changes in tempo, meter and duration and to new finger patterns in real time.

Thom's system uses an off-line learning algorithm to create a probabilistic model applied to rewrite four-measure melodies at performance time [7]. Biles uses the genetic algorithm in creating accompaniment melodies [8]. Both frameworks require scripted chord sequences and a tempos. The goals of MIDIME are different from these systems, both in MIDIME's analysis of all aspects of empirical musical data in preference to scripts, and in creating harmonic, rhythmic and timbral behavior, in addition to the melodic orientation of these systems. The primary goal of MIDIME is exploration of new music within a framework of multi-level improvisation, rather than limited improvisation within fixed musical structures found in other performance analysis and improvisational systems.

6. REFERENCES

- [1] C. Roads, *The Computer Music Tutorial*. MIT Press, 1996.
- [2] M. Hamanaka, M. Goto, H. Asoh, and N. Otsu, "A learning-based quantization: Unsupervised estimation of the model parameters," in *Proc. Int. Comp. Music Conf. (ICMC'03)*, Singapore, 2003, pp. 369–372.
- [3] M. Hamanaka, K. Hirata, and S. Tojo, "Automatic generation of metrical structure based on GTTM," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 53–56.
- [4] K. Lemstrom and S. Perttu, "SEMEX – an efficient music retrieval prototype," in *Proc. Int. Symp. Music Information Retrieval (ISMIR'00)*, Plymouth, Massachusetts, USA, 2000, [Online] http://ismir2000.ismir.net/papers/lemstrom_paper.pdf.
- [5] A. Pienimaki and K. Lemstrom, "Clustering symbolic music using paradigmatic and surface level analyses," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, 2004, pp. 175–178.
- [6] G. Toussaint, "A comparison of rhythmic similarity measures," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, 2004, pp. 134–137.
- [7] B. Thom, "BoB: An improvisational music companion," in *Proc. Fourth Int. Conf. Autonomous Agents*, Barcelona, Spain, 2000, pp. 309–316.
- [8] J. A. Biles, "GenJam," [Online] <http://www.it.rit.edu/~jab/> including numerous publications on GenJam and evolutionary music.

USING VISUAL TEXTURES FOR SONIC TEXTURES PRODUCTION AND CONTROL

Jehan-Julien Filatriau

Communications and Remote Sensing Lab
Université Catholique de Louvain
Louvain la Neuve, Belgium
filatriau@tele.ucl.ac.be

Daniel Arfib

LMA-CNRS
31, Chemin Joseph Aiguier
Marseille, France
arfib@lma.cnrs-mrs.fr

Jean-Michel Couturier

Blue Yeti
68, avenue de la Grande Conche, 17200 Royan, France
jmc@blueyeti.fr

ABSTRACT

This work takes place in the framework of a global research on the synthesis of sonic textures and its control through a gesture-based interaction in a musical practice. In this paper we present different strategies to link visual and sonic textures using similar synthesis processes; theoretical considerations underlying to this problematic are firstly exposed and several personal realizations, illustrating different approaches to design a gesturally controlled audio-visual system, are then described.

1. INTRODUCTION

Sound production and gestural control have their own specificities and belong to a field named “sound and music computing”. It is interesting and useful to make a cross-comparison with other fields such as vision, perception or emotion. This paper is an attempt for the cross fertilization of different fields: visual textures making has an incredible advance in processing techniques, from which we can benefit. But also concepts behind vision can be applied to sound as soon as they are correctly transposed. The main direction of our research is to see if analogies can be done in the design and use of visual textures and the design and use of sonic textures. It completes previous articles [1] [2] which have described the algorithmic part of textures making. Our motivations behind this paper is then twofold: in chapter 2 we describe a personal view of these analogies between visual and sonic textures, supported by examples of the literature or our own prospects; in chapter 3 and 4 we present examples of the applicative research of the three authors, packaged in a way that distinguishes research done on static and dynamic images. As we will see we push forward the notion of a “malleable texture” which is especially handful for gesture control.

2. BASIC FOUNDATIONS

Before describing experiments we have carried out, we give in this section some generic considerations about the link between image and sound: firstly, we will explain why starting from visual textures making techniques could be relevant for the generation of sonic textures. Then, we will show that many techniques have been developed by the computer graphics community, enabling the synthesis of a wide range of visual textures. Finally, we will expose

different strategies that can be carried out in order to link image and sound in a common generative process.

2.1. The bridge between visual and sonic textures

This bridge can be done at a technical level, of course, and next section will provide examples of such kind. But the link is also at a conceptual level of perception and recognition. The fact is that visual textures can be perceived as such once certain statistical facts arise from them; this is the perceptual fact. Among all these textures one can recognise different kinds of textures and label them; this is the recognition part. These two parts are a great importance in presence, a notion developed also by HCI and which deals with the contact with environment. All of this is also true for sonic textures: they have auditory clues that make them be textures, they are distinguishable and are part of the discriminating process of hearing in the environment.

Computer science provides us some techniques, sometimes called procedural, in order to synthesize images and sound. Are these techniques equivalent? There is still a great deal of research on how to transpose a set of methods from a field to another, and in our case this is amplified by the fact that the visual and sonic object or features are not the same. Their common ground is the mobility (a video is a moving image, a sound is a perseverance of a sonic stimulus along time) and the controllability. The malleability of visual or sonic forms can be seen from two points of view: either the object is malleable or it is fixed, but the point of view is different. Such analogies are not only words, but will be demonstrated in the image-to-sound correspondence. Sonic textures have specificities one does not have to miss: they rely on time, and sonic clues are very dependent upon the hearing process. As an example the sound of rain has a rhythmic component associated with a resonance for each drop; a static boring sound is not equivalent to the profusion of a texture image. So one has to be very careful while designing an image-to-sound system.

2.2. How to make visual textures

2.2.1. What is a visual texture?

The multiple definitions found in the literature present visual textures (Fig.1) as spatially homogenous and typically containing re-

peated structures subject to some random variations. Bar-Joseph highlights another important characteristic [3]: using any window of size larger than some critical size, the “information content” exhibited in the window is invariant to the window position within the given sample.

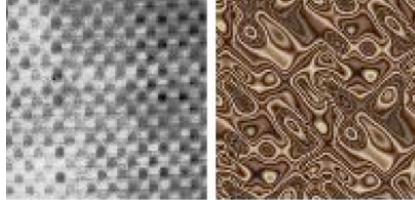


Figure 1: Examples of natural textures (at left) and synthetized textures (at right).

2.2.2. Visual textures synthesis

Visual texture synthesis has received an increasing attention from the computer graphics research community over twenty years. The objective is to generate, from an original sample, an endless output texture perceptually similar to the reference image. First attempts on texture synthesis focused on the development of well-known procedural textures [4]. After these, methods starting from the analysis of an input image have been carried out: those techniques model visual textures as sample from probabilistic distribution [5] and more recently as Markov Random Fields [6]. A lot of techniques dedicated to the synthesis of animated textures have also been developed: some of them extend techniques used for fixed images [3][7], while other methods rely on more or less complex physical models to render dynamic natural phenomena like waterfalls, fire, or waves motion [8]. Advances in these synthesis algorithms coupled to increasing possibilities offered by computer science have made possible high quality synthesis of fixed and time-varying visual textures. These images are potentially interesting candidates for an image-to-sound process based on static image. However those methods are generally computationally intensive and not adapted to real-time constraints, which make them not suitable for a dynamic image-to-sound process. For this aim, an alternative solution may be offered by fractal textures, described in the next section.

2.2.3. Fractal textures

A fractal is a rough or fragmented geometric shape that can be subdivided in an infinity of elements, each of which being, at least approximately, a reduced-size copy of the whole. Fractals are especially used in computer modeling of irregular patterns and structures in nature such as clouds, mountains, or branches of trees that do not correspond to simple geometric shapes. Fractals present some characteristics that make them close to visual textures: indeed fractals are generally self-similar, independent of scale and often referred to as “infinitely complex”.

Iterated Function System (IFS) is one of the simplest ways to generate fractal images [9]. Fractals of this type are created by applying a series of affine transformations to an initial point through a number of iterations. A weighted probability factor is associated to each transformation, in order to favor certain configurations during the iterative process. From a geometrical point

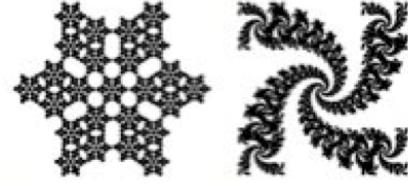


Figure 2: Examples of IFS fractal textures.

of view, this process corresponds to a combination of elementary transformations (translations, rotations, compression and shears) allowing after few iterative cycles to build up a large range of fractal shapes like trees, spirals or snowflakes (Fig.2). The complexity of these algorithms depends on the number of points composing the image and the number of iterations of the process; small values of these parameters allow to compute and render in real-time quite complex fractal shapes. Furthermore, by varying parameters describing affine transformations, one can easily distort these shapes and obtain animated image. This is the strategy we have adopted in one of our image-to-sound instrument, as described in section 4.2.

2.3. How to link visual to sonic textures

There is no one-way procedure to link vision and sound. We have developed three points of view, starting from the “image from a sound” thinking, going to the “pixel image sonification” and ending with the “equivalence of the processes”. Each of them gives a territory for the sound, which can be explored by gesture.

2.3.1. Considering time as a spatial coordinate

We are accustomed to sonograms (Fig.3) and this way we can consider a 2D image being in fact a 1D vector using time as the second coordinate. This means to derive from a static image an evolving sound.

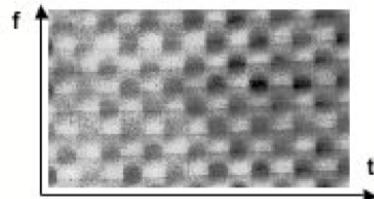


Figure 3: “Images from a sound” thinking.

Using images as potential sonograms is however not straightforward. A sonogram can be considered as the modulus of a sliding short Fourier transform analysis, and the reconstruction can be made according to a phase-vocoder approach [10]. But not every image can be considered as a sonogram. There is a relationship between the points of a sonogram which can be explained this way: a point in a sonogram will give rise to a gaboret (sinusoid enveloped by a window) in the time domain. When taking again a sonogram from this gaboret, one finds the reproducing kernel, which establishes relations between adjacent points. This is not usually fulfilled by arbitrary images. This means that one cannot consider any image as a “valid” sonogram. The other very important point is that not only modulus is necessary, but phases are also important and if not given, they must be estimated. This is not an

easy task, and different methods have been suggested [11] [12], which give only an approximate result. One of them consists in going back and forth from an image to a sound, each time reassigning the value of the target modulus to the image, while others assign phases along spectral lines or transients. It is also possible to use the relation between the derivative of phase and modulus according to frequency and modulus. We did a combination of these techniques, as will be demonstrated in section 3.1.

An alternative to the phase-vocoder is the sum of sinusoids approach, where each line of the image corresponds to the envelope of a sinusoid. This forces the frequencies to be on a scale, a strong fact that can be played with, but cannot be called the reconstruction of a sonogram. However, this approach is musically significant and has been widely used by programs such as Metasynth [13].

The gestures that can be associated to this way of doing depend if we work out of real-time or in real-time. A work in studio will depend on gestures linked to computer processing, and only after the sound is rendered. We will show in section 3.1 a description of such a possibility using a database of natural images. When using this correspondence in real-time, we can have two strategies:

- either we consider an image as fixed, so that the gesture will develop sound according to a time unfolding and some frequency and amplitude transformations,
- or we create new images according to action gestures, and unfold these images with modulation gestures which eventually modulates the parameters of the image to sound transcription.

2.3.2. Considering the temporal evolution of an image

This is a classical way to see things, as far as the video capture is now easily made. Here we consider the external aspect of a visual process — for example motion of a dancer in front of a camera — and derive from it features that can be transposed in a sound [14]. If the image is an evolving visual texture and the sound a sonic one, the idea is to find a mapping for the conversion of one to the other. The concept is to have a malleable image which drives a sonic process. The mapping between the two processes can be very loose, because in one way one tries to capture the dynamism of the image as well as its textural content. Such a mapping will be demonstrated in section 4 with the “Filtering string” (1D version of a texture) and with the “Sonic fern” using a 2D image. A subset of this approach relies on the exploration of this image via different pathways to drive the sonic process (Fig.4). The equivalent would be a moving camera over a fixed image, and the focused area gives the present slice of sound. We can say that the initial image is totally innocent, but the way we position successive images gives a sense. An example of this approach will be demonstrated in the “Texture Scratcher” instrument in section 3.2.

2.3.3. Linking both visual and sonic generative processes

In this approach we start from an algorithm of visual texture synthesis, but instead of focusing on the result of this process, one focusses on the process itself, and this way we can link the destiny of a visual object to the destiny of the sonic object (Fig.5). If a texture destiny is written in the unfolding of a program, the data which enables this unfolding will be the constituents of the sound. This way the spirit of an algorithm can be expressed in two media. This way is very powerful, as soon as one can really use the same algorithm for visual and sonic texture making. We are in front of

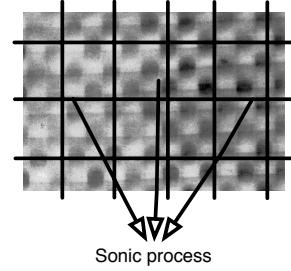


Figure 4: “Pixel image sonification” technique.

the huge problem of the visual feedback of a sonic process, a field addressed by new research thesis works [15][16].

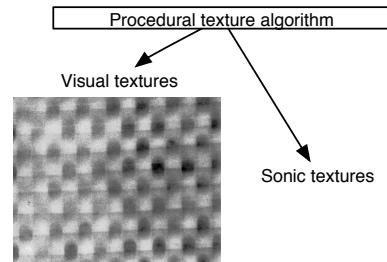


Figure 5: “Equivalence of the processes” technique.

3. APPLICATIONS WITH FIXED IMAGES

After these fundamental considerations, we describe in two following sections four realizations done by the authors involving image and sound in the same synthesis process and driven by a gestural interaction: first examples are based on static images (section 3) whereas following ones rely on dynamic images (section 4).

3.1. Visual textures seen as sonograms

We are tempted to use the wide data banks of visual textures and exploit them as sonograms. It is not an easy game, because in a way textured images are 2D and not concerned with time, whereas sound is 1D and develops with time. Anyway an interesting point of view is to force a visual texture so that it represents a sonogram. Though a brute force approach of a phase reconstruction from the arbitrary image can be used, the sound results are very metallic. So we decided to use an alternative approach; on a sonogram the horizontal lines can be interpreted as sinusoid partials, whereas the vertical lines can be interpreted as transients. In a sinusoidal +transients +noise approach, it is then convenient to separate an original image into the sum of three images representing these components. This can be done by convoluting the image with three 2D filters, each one selecting a particular aspect of the original image (Fig.6). Starting from a sonogram it is then possible to reconstruct the sine waves, transients and additional noise. We applied this technique with an image taken from the widely used Brodatz database of natural textures (Fig.7).

Each of these three images is then independently treated with programs corresponding to the acoustic facts it belongs to:

- for image 1 (left), vertical lines are assigned phases corresponding to transients. This means phases are zero for maxima and turn at natural rate around these maxima.

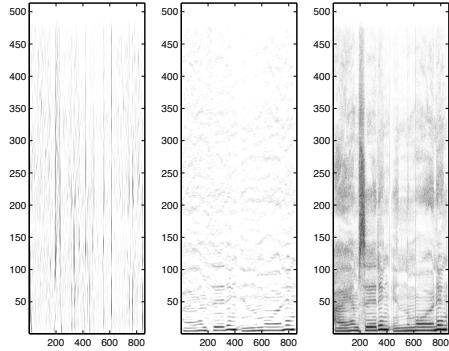


Figure 6: Separation of a sonogram into three subimages.

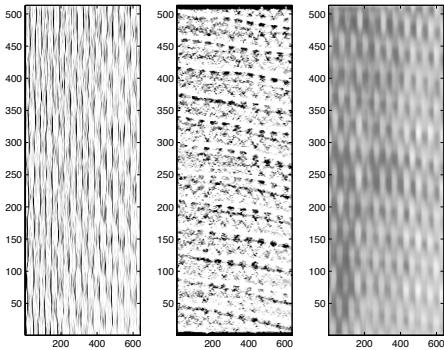


Figure 7: Separation of a Brodatz texture into subimages.

- for image 2 (middle), the phases for horizontal lines are developed corresponding to the spectral lines (maximum of amplitude) using techniques similar to SMS [17] (the spectral line is at the top of a parabola for a Gaussian window) or using the fact that the horizontal derivative of the phase is proportional to the vertical derivative of the modulus.
- for image 3 (right), phases are assigned a random value.

These three images are then rendered and the sum of these sonic signals gives a texture which holds the characteristics of the three images (sine plus transients plus noise) and a good intuitive mixing helps find a good balance between them. Presently we have designed only studio techniques corresponding to this processing. This means we use transformations on the images to obtain digital audio effects. But we do plan experiments where images will be created or retrieved according to gesture, and their temporal development altered by modulation gestures.

3.2. The Texture Scratcher

In this instrument, described in detail in [1], the relation between image and sound is quite different than in the previous “sonogram” approach: here sound synthesis does not consist anymore of a basic translation image-to-sound, but is now based on a gesture-controlled exploration of a visual space. This instrument is a real-time adaptation of the Functional Iteration Synthesis (FIS) [18] implemented in Max-MSP-Jitter environment and using a tablet-screen and a joystick as gestural interfaces. FIS is a special application of the well-known wave terrain synthesis, where an orbit is traced on a three-dimensional surface (the wave terrains) to generate a waveform corresponding to the variation of elevation of the trajectory on the terrain.



Figure 8: The Texture Scratcher.

Specificity of FIS comes from the way the terrains are built: terrains are here computed by iteration of non-linear functions, which make them extremely complex and close to visual textures. Sound is produced by scratching areas of the terrain displayed on the screen with the stylus (Fig.8); terrain scratching is carried out either through a direct way using only the stylus, or through an indirect way by adding a joystick enabling to generate more complex orbits by a parametric control. In this instrument, the resulting sound is not a direct translation of the image. Image is used as a support for the sound synthesis, and though the outline of image has an influence on the sonic result, characteristics of the sound primarily depend on the way to explore the image (orbit shape, velocity, periodicity).

4. APPLICATIONS WITH DYNAMIC IMAGES

4.1. The Filtering String

The “Filtering String” [19] was our first attempt to drive both visual and sound processes by a common gestural control. In this instrument, the shape of a virtual slow-moving string is used to drive a filter bank fed with noise as input. The string shape, based on a spring-mass model, is displayed on the screen and varies the gains of 32 filters. The user acts simultaneously on the image and sound production by interacting with the dynamic system thanks to a graphic tablet and a multi-touch surface: one hand, using the stylus, configures the intrinsic parameters of the string (damping, tension, stiffness) while the other one applies forces on it by exerting pressure on the multitouch surface. This changes equilibrium position of the system and makes the string oscillating, implying dynamic fluctuations in the frequency spectrum of the sound.

4.2. The Sonic Fern

The idea underlying this new instrument is to compute a set of points representing in one hand a graphical object — a fern — and driving in other hand a sonic process. Both processes are conducted by the same gestural control, which makes the fern move and the corresponding sound evolve in a parallel way. The first step for this is to construct an image shape from a set of input parameters specified by the user. The second step is to connect to the output data an algorithm that creates a sonic texture.

4.2.1. Construction of the fern by iterated function system

As described in section 2.2, Iterated Function System (IFS) is a simple way to generate complex image at a low computational cost [9]. For this instrument we use for the control of sound synthesis parameters one of the most famous shape generated by IFS: the Barnsley’s fern (Fig.9).



Figure 9: The classical Barnsley's fern (at left) and deformed ferns (at middle and right).

The iterative algorithm executed to build this kind of fractal image using IFS is described below. Images constructed by this way are composed of P points, each point being the result of N iterations. The iterative process starts with a couple (x_0, y_0) chosen at the origin, and for each iteration n, a couple of coordinates (x_n, y_n) is computed from previous values (x_{n-1}, y_{n-1}) applying one of the four predefined affine transformations chosen randomly. After N iterations, a point of coordinates (x_N, y_N) is drawn on the image, and the whole process restarts for the next point. The recursive nature of the IFS guarantees that the whole is a larger replica of each frond, that is each leaf of the fern is the same as the full fern.

```

for j = 1 : NumOfPts
% set initial position
x = 0; y = 0;
for i = 1 : iterations
% random draw assigning which transformation
% will be applied for this iteration
p = rand(1);
if(p < p1) k = 1;
else if(p < (p1 + p2)) k = 2;
else if(p < (p1 + p2 + p3)) k = 3;
else k = 4;
%Compute x_{n+1} and y_{n+1}
x_{n+1} = a_k * x_n + b_k * y_n + c_k;
y_{n+1} = d_k * x_n + e_k * y_n + f_k;
x_n = x_{n+1};
y_n = y_{n+1};
end;
pts(j, 1) = x_n; pts(j, 2) = y_n;
end;

```

The rendering of the fern needs thus four affine transformations T_k , $k=1,2,3,4$, each of them being characterized by six coefficients [a,b,c,d,e,f] as below:

$$T_k = \begin{cases} x_n = a_k * x_{n-1} + b_k * y_{n-1} + c_k \\ y_n = d_k * x_{n-1} + e_k * y_{n-1} + f_k \end{cases} \quad (1)$$

An image is thus characterized by a matrix M of 24 coefficients [$a_k, b_k, c_k, d_k, e_k, f_k$] with $k=1,2,3,4$. By varying these coefficients, one can change the resulting shape of the image. Small changes in one or few coefficients will have as consequence to deform the fern (rescaling, rotation, inclination) whereas varying all the coefficients will render an image very far from the original fern. Furthermore, a continuous variation of one or several of these parameters allows to obtain a user-controlled malleable fern.

4.2.2. Implementation of IFS object in Max-MSP

As IFS offers a quite simple method to compute set of points representing moving graphical object, we tried to implement this algorithm in Max-MSP environment, with the objective to link output data provided by IFS to a sonic process. Because available Max-MSP libraries are not suited to the execution of iterative process, we programmed in C our proper Max object, that we called IFS. This object enables to compute and render with Jitter any IFS fractal image in real-time (Fig.10). User is able to specify global parameters (number of points and iterations) and vary independently each value of the coefficient matrix M. By this way, it is a very convenient tool for the drawing of a large range of animated fractal shapes in Max-MSP-Jitter environment. In the experiments described below, the fern was composed of 256 points calculated after 15 iterations.

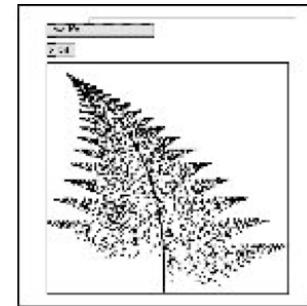


Figure 10: Barnsley's fern construction algorithm.

4.2.3. From image data to sound parameters

Many strategies may be adopted to map the data of the fern shape to sound synthesis parameters. We developed two of them, very simple, based on granular techniques:

- in the first case, each point constituting the fern triggers a sound grain extracted from a sample, as in classical granulation techniques. Characteristics of the grains vary according to the motion of the fern: horizontal coordinate of the point is used to determine the position of the extracted grain in the original sample whereas the vertical coordinate is assigned the length, amplitude and pitch shifting factor to each grain.
- in the second case, grains are not anymore extracted from a sample but synthesized by noise filtering. Each point of the fern is assigned a noisy grain, that is a narrow noise band which parameters are mapped with image data as follow: the central frequency of the band is given by the horizontal coordinate of the point on the image, its bandwidth by the vertical position and the amplitude of the grain is controlled by the motion speed of the point.

4.2.4. Gesture-controlled deformation of the fern

The approach taken here for linking image to sound offers the possibility to use a common gestural control to drive both visual and sonic processes. For the mapping we use the “Max drum”, a gestural interface enabling to track the position of two sticks in a 3D space. The objective of the mapping was to give the user the ability to animate the shape of the fern through his gestures. For this we

chose to vary simultaneously four coefficients (a_2, b_2, d_2, e_2) of the matrix M. We developed two ways to gesturally control these parameters:

- instantaneous deformation : here values of the four coefficients are directly mapped to X and Y position of each stick. This mapping allows the user to actually sculpt the shape of the fern by means of the two sticks.
- oscillating deformation : in this second way, we consider the fern as a pseudo rigid object, able to oscillate around a equilibrium position when forces are applied to it. User stretches the fern in the same way than previously before to release it by getting up the stick in order to make the fern oscillate. The oscillation is achieved by varying the four coefficients following the motion of a spring-mass model. This is done thanks to the scansynth object originally developed for the Filtering String instrument [19].

Gestures associated to each mapping are of different natures: in the first case, user acts instantaneously on the shape of the image by modulation gestures, whereas in the second, an action gesture the stretch and release of the fern triggers the subsequent evolution of the parameters which will be unfolded after that time.

4.2.5. Future works

Here we tried to link visual and sonic textures in the same synthesis process relying on an IFS fractal image. These experiments provided interesting results, especially on the relation between gestures, image and sound. Indeed a strong correspondence exists in the dynamism of resulting image motion and sound fluctuations. Now we need to pursue in this track by investigating other fractal shapes, more “textured” than the fern, or by trying other mapping strategies to enforce correspondence between image and sound features. We are only at the start of the evaluation of the musical possibilities offered by this instrument, which in fact will belong to each composer or experimenter.

5. CONCLUSION

This work gives a major contribution to the following question: in which way available image processing techniques dedicated to visual textures can be exploited for the sonic textures production? We have experimented different strategies to link image and sound in the same process and results exposed here let us think that the visual textures may be involved at several levels in sonic processes. Among the most promising, the track of mapping a malleable graphic object to a sonic process offers a wide range of attractive possibilities in term of gestural interaction.

6. REFERENCES

- [1] J.-J. Filatriau and D. Arfib, “Instrumental gestures and sonic texture,” in *Proc. Sound and Music Computing (SMC) Int. Conf.*, Salerno, Italy, 2005, [Online] http://www.tele.ucl.ac.be/~jjfil/smco5_Filatriau_Arfib.pdf.
- [2] G. Eckel, D. Rocchesso, and G. Strobl, “Sound texture modeling: A survey,” in *Proc. Sound and Music Computing (SMC) Int. Conf.*, Marseille, France, 2006, [Online] <http://assoc.orange.fr/gmem/smco06/papers/8-soundtexturemodelingSMC06.pdf>.
- [3] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, “Texture mixing and texture movie synthesis using statistical learning,” *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 2, pp. 120–135, July 2001.
- [4] K. Perlin, “An image synthesizer,” *Computer Graphics*, vol. 19, no. 5, pp. 287–296, July 1985.
- [5] J. S. De Bonet, “Multiresolution sampling procedure for analysis and synthesis of texture images,” in *Proc. ACM SIGGRAPH*, New Orleans, USA, 1997, pp. 361–368.
- [6] A. Efros and T. Leung, “Texture synthesis by non-parametric sampling,” in *Proc. IEEE Int. Conf. on Computer Vision, Kerkyra, Greece*, 1999, pp. 1033–1038.
- [7] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, “Texture optimization for example-based synthesis,” *ACM Trans. Graphics*, vol. 24, no. 3, pp. 795–802, July 2005.
- [8] S. Thon and D. Ghazanfarpour, “Ocean waves synthesis and animation using real world information,” *Computer Graphics*, vol. 26, no. 1, pp. 99–108, 2002.
- [9] M. Barnsley and H. Rising, *Fractals Everywhere*. Boston Academic Press Professional, 1993.
- [10] M. Dolson, “The phase vocoder: A tutorial,” *Computer Music J.*, vol. 10, no. 4, pp. 14–27, 1986.
- [11] J. Laroche and M. Dolson, “New phase vocoder technique for pitch-shifting, harmonizing and other exotic effects,” in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 1999, pp. 91–94.
- [12] D. Griffin and J. Lim, “A new model-based speech analysis/synthesis system,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'85)*, Tampa, USA, 1985, pp. 513–516.
- [13] U. Software, “Metasynth,” Retrieved June 29th, 2006, [Online] <http://www.uisoftware.com/MetaSynth/>.
- [14] A. Camurri, G. De Poli, A. Friberg, M. Leman, and G. Volpe, “The MEGA project: analysis and synthesis of multisensory expressive gesture in performing art applications,” *J. New Music Research*, vol. 34, no. 1, pp. 5–21, 2005.
- [15] J.-M. Couturier, “Utilisation avancée d’interfaces graphiques dans le contrôle gestuel de processus sonores,” Ph.D. dissertation, University of Aix-Marseille II, 2004.
- [16] S. Jordà, “Digital lutherie: Crafting musical computers for new musics’ performance and improvisation,” Ph.D. dissertation, Universitat Pompeu Fabra, Barcelona, 2005.
- [17] X. Serra and J. Bonada, “Sound transformations based on the SMS high level attributes,” in *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, 1998, pp. 138–142.
- [18] A. D. Scipio, “Synthesis of environmental sound textures by iterated non linear functions and its ecological relevance to perceptual modeling,” *J. New Music Research*, vol. 31, no. 2, pp. 5–21, 2002.
- [19] J.-M. Couturier, L. Kessous, and D. Arfib, “Expressiveness and digital musical instrument design,” *J. New Music Research, special issue on Expressive Gesture in Performing Arts and New Media*, vol. 34, no. 1, pp. 125–136, 2005.

GRAPHIC EQUALIZER DESIGN USING HIGHER-ORDER RECURSIVE FILTERS

Martin Holters, Udo Zölzer

Department of Signal Processing and Communications
Helmut-Schmidt-University, Hamburg, Germany
martin.holters@hsu-hh.de

ABSTRACT

A straight-forward design of graphic equalizers with minimum-phase behavior based on recently developed higher-order band-shelving filters is presented. Due to the high filter order, the gain in one band is almost completely independent from the gain in the other bands. Although no special care will be taken to design filters with complementary edges except for a suitable definition of the cut-off frequencies, the resulting amplitude deviation in the transitional region between the bands will be sufficiently low for many applications.

1. INTRODUCTION

Equalizers are common audio effect devices. With their configurable magnitude response, they allow the signal's spectrum to be shaped, to emphasize or deemphasize certain frequency bands or to compensate non-ideal reinforcement equipment or room acoustics. Equalizers allow the specification of desired gains for specified bands. These bands can either be adjustable by the user in so called parametric equalizers, or can be fixed for graphic equalizers, which we will focus on in this paper.

When the gains of the different bands of a graphic equalizer are controlled with sliders, the positions of the sliders' knobs can be understood as samples of the equalizers magnitude response at different frequencies. Hence, the sliders yield a graphic representation of the magnitude response, which is why this form of equalizer is called "graphic".

The bands are usually distributed logarithmic across the frequency, to match human perception. Common designs include octave and 1/3-octave distribution.

When an equalizer (or any kind of effect device) is used in a live performance, the processing delay is of concern. While the maximum allowable total delay may be a matter of discussion, it is safe to require each individual device to have the lowest possible processing delay, to allow cascading of several devices. This mandates using minimum-phase recursive filters instead of linear-phase FIR filters.

Since a general IIR filter design is computationally intensive, the equalizer is usually implemented as a cascade of sub-filters, each of which realizes the desired gain for one band and ideally has unity gain in the other bands.

In the following, we will first go through the general steps of designing an equalizer out of such sub-filters. While doing so, we will also discuss the properties these sub-filters should ideally have. We will then present a rather straight-forward design for such filters, to finally employ them in two design examples to discuss their specific features.

2. EQUALIZER DESIGN USING SHELVING FILTERS

When designing an equalizer using per-band sub-filters, the main task is to specify the involved frequencies to arrive at a specification for the sub-filters. Let us denote the normalized lower and upper cut-off frequencies of the i -th band with $\Omega_{L,i}$ and $\Omega_{U,i}$, respectively. Furthermore, we define the bandwidth as the difference

$$\Omega_{B,i} = \Omega_{U,i} - \Omega_{L,i} \quad (1)$$

and the center frequency as the geometric mean

$$\Omega_{C,i} = \sqrt{\Omega_{U,i} \cdot \Omega_{L,i}} \quad (2)$$

of the cut-off frequencies. The frequency bands shall be adjacent, that is $\Omega_{U,i} = \Omega_{L,i+1}$. The logarithmic distribution can equivalently be specified using the cut-off frequencies $\Omega_{L,i+1} = R \cdot \Omega_{L,i}$ or center frequencies $\Omega_{C,i+1} = R \cdot \Omega_{C,i}$, from which immediately follows that also $\Omega_{B,i+1} = R \cdot \Omega_{B,i}$. Note that for an octave equalizer $R = 2$ and for a 1/3-octave equalizer $R = \sqrt[3]{2} \approx 1.26$. We shall base our design on the band center frequencies, so that it is convenient to express the other parameters as

$$\Omega_{L,i} = \frac{1}{\sqrt{R}} \Omega_{C,i}, \quad (3)$$

$$\Omega_{U,i} = \sqrt{R} \Omega_{C,i} \quad (4)$$

and

$$\Omega_{B,i} = \left(\sqrt{R} - \frac{1}{\sqrt{R}} \right) \Omega_{C,i}. \quad (5)$$

Ideally, the sub-filter for the i -th band has magnitude of the desired gain g_i inside the band and unity gain outside that band, that is the ideal band-shelving filter which satisfies

$$\left| H_{\text{ideal},i}(e^{j\Omega}) \right| = \begin{cases} g_i & \text{if } \Omega_{L,i} \leq \Omega < \Omega_{U,i} \\ 1 & \text{else.} \end{cases} \quad (6)$$

In practice, the sub-filters will of course have transitional regions around the band edges. In order to have a flat magnitude response in cases where the same gain is used in adjacent bands, we therefore demand the filters to have (at least nearly) complementary band-edges. In particular, we require

$$\left| H_i(e^{j\Omega_{L,i}}) \right| = \left| H_i(e^{j\Omega_{U,i}}) \right| = \sqrt{g_i}, \quad (7)$$

so that at the cut-off frequency, when $g_i = g_{i+1} = g$, we have

$$\left| H_i(e^{j\Omega_{U,i}}) \cdot H_{i+1}(e^{j\Omega_{U,i}}) \right| = g. \quad (8)$$

The filter edges should also be sufficiently steep, so that the inter-band influence is negligible even if two adjacent bands are configured to vastly different gains. To accomplish this, we will use higher-order filters.

3. SHELVING FILTER DESIGN

The design of the individual shelving filters of which the equalizer is to be composed is based on a continuous-time minimum-phase low-shelving prototype derived from a Butterworth low-pass design [1, 2] as given by

$$H_{LS}(s) = \prod_{m=1}^M \frac{s + e^{j\alpha_m} \sqrt[M]{g}}{s + e^{j\alpha_m}}, \quad \alpha_m = \left(\frac{1}{2} - \frac{2m-1}{2M} \right) \pi, \quad (9)$$

where M is the filter order. Note that for $M = 1$ and $M = 2$, this design reduces to the well known first- and second-order low-shelving filters, respectively [3]. The magnitude response of the low-shelving prototype is

$$|H_{LS}(j\omega)| = \sqrt{\frac{\omega^{2M} + g^2}{\omega^{2M} + 1}}. \quad (10)$$

By choosing $|H_{LS}(j\omega_B)| = \sqrt{g}$, we find that for the normalized prototype of (9), the cut-off frequency $\omega_B = \sqrt[2M]{g}$.

Assuming even filter order M , we can rewrite (9) in terms of real-valued second-order sections as

$$\begin{aligned} H_{LS}(s) &= \prod_{m=1}^{M/2} \frac{s^2 + 2 \cos(\alpha_m) \sqrt[M]{g}s + \sqrt[M]{g}^2}{s^2 + 2 \cos(\alpha_m)s + 1} \\ &= \prod_{m=1}^{M/2} \frac{1 + 2V \frac{1 + c_m s}{s^2 + 2c_m s + 1} + V^2 \frac{1}{s^2 + 2c_m s + 1}}{1 + 2V \frac{1 + c_m s}{s^2 + 2c_m s + 1} + V^2 \frac{1}{s^2 + 2c_m s + 1}} \end{aligned} \quad (11)$$

with $c_m = \cos(\alpha_m)$ and $V = \sqrt[M]{g} - 1$.

From this, we obtain a digital filter by applying the bilinear transform [4] and frequency-scaling with

$$s = \frac{1}{K} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \quad (12)$$

yielding

$$\begin{aligned} H_{LS}(z) &= \\ &\prod_{m=1}^{M/2} \left(1 + 2V \frac{K(K+c_m+2Kz^{-1}+(K-c_m)z^{-2})}{1+2Kc_m+K^2+(2K^2-2)z^{-1}+(1-2Kc_m+K^2)z^{-2}} \right. \\ &\quad \left. + V^2 \frac{K^2(1+2z^{-1}+z^{-2})}{1+2Kc_m+K^2+(2K^2-2)z^{-1}+(1-2Kc_m+K^2)z^{-2}} \right). \end{aligned} \quad (13)$$

The frequency-scaling constant K is chosen to map the desired digital cut-off frequency Ω_B to the analog cut-off frequency $\omega_B = \sqrt[2M]{g}$, that is

$$K = \frac{1}{\sqrt[2M]{g}} \tan\left(\frac{\Omega_B}{2}\right). \quad (14)$$

Given the low-shelving filter, a band-shelving filter as required for the equalizer of order $2M$ can be derived by a low-pass to band-pass frequency transformation according to

$$H_{BS}(z) = H_{LS}\left(z \frac{\cos \Omega_M - z}{1 - \cos \Omega_M z}\right), \quad (15)$$

which shifts the filter such that the maximum (or minimum for $g < 1$) of the transfer function is reached at Ω_M , while the bandwidth Ω_B and the minimum-phase behavior are retained [5]. In

practice, the substitution may be performed by replacing the unit-delays of the low-shelving filter with the all-pass

$$A(z) = z^{-1} \frac{\cos \Omega_M - z^{-1}}{1 - \cos \Omega_M \cdot z^{-1}}. \quad (16)$$

One possible realization of the m -th frequency-shifted second-order section of the band-shelving filter $H_{BS}(z)$ which aims at minimizing the number of different coefficients is shown in Figure 1, where

$$a_{0,m}^{-1} = \frac{1}{1 + 2Kc_m + K^2}. \quad (17)$$

More details about this design and realization aspects can be found in [1, 2].

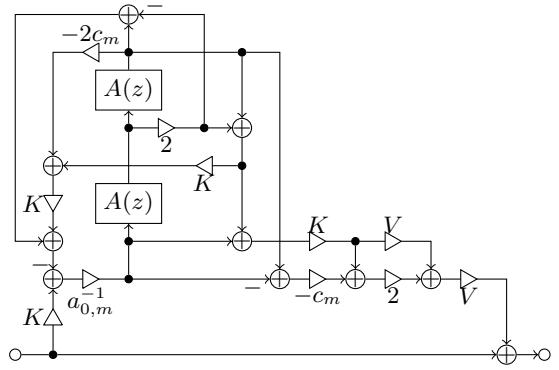


Figure 1: Realization of the m -th fourth-order (frequency-shifted second-order) section of $H_{BS}(z)$.

The magnitude response

$$|H_{BS}(e^{j\Omega})|^2 = \frac{(\cos \Omega_M - \cos \Omega)^{2M} + (K \sin \Omega)^{2M} g^2}{(\cos \Omega_M - \cos \Omega)^{2M} + (K \sin \Omega)^{2M}}, \quad (18)$$

of the filter defined by (15) is depicted in Figure 2 for various maximum-gain frequencies.

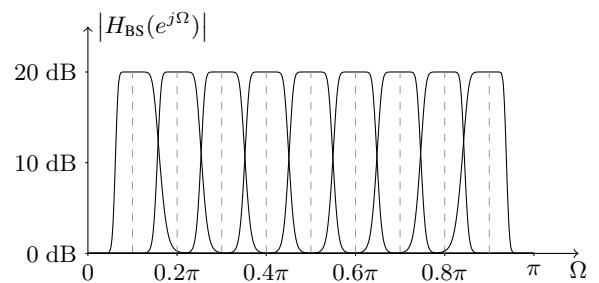


Figure 2: Magnitude responses of band-shelving filters with order $2M = 12$, gain $g = 10$, bandwidth $\Omega_B = 0.1\pi$ and maximum-gain frequencies $\Omega_M = [0.1\pi, 0.2\pi, \dots, 0.9\pi]$.

Comparing these to the required properties stated in section 2, we find that these filters deliver a good approximation of (6), getting even better with higher M of course, as the edges get steeper. To fulfill (7), we have to choose the maximum-gain frequency such that

$$\tan^2\left(\frac{\Omega_{M,i}}{2}\right) = \tan\left(\frac{\Omega_{U,i}}{2}\right) \cdot \tan\left(\frac{\Omega_{L,i}}{2}\right). \quad (19)$$

i	$f_{C,i}/\text{Hz}$	$f_{L,i}/\text{Hz}$	$f_{U,i}/\text{Hz}$	$f_{M,i}/\text{Hz}$	$\cos\left(\frac{2\pi f_{M,i}}{f_s}\right)$
1	30	21	42	30	0.999992
2	60	42	85	60	0.999969
3	120	85	170	120	0.999877
4	240	170	339	240	0.999507
5	480	339	679	480	0.998026
6	960	679	1358	960	0.992110
7	1920	1358	2715	1923	0.968500
8	3840	2715	5431	3861	0.874993
9	7680	5431	10861	7862	0.515600
10	15360	10861	21722	17955	-0.702955

Table 1: Center frequencies $f_{C,i}$, lower cut-off frequencies $f_{L,i}$, upper cut-off frequencies $f_{U,i}$, maximum-gain frequencies $f_{M,i}$ and resulting frequency-shifting coefficients $\cos(\Omega_{M,i})$ of the octave equalizer.

i	K_i	i	K_i	i	K_i
1	0.001168	5	0.018694	9	0.312322
2	0.003300	6	0.052838	10	1.023332
3	0.004673	7	0.074962		
4	0.013201	8	0.213467		

Table 2: Bandwidth coefficients K_i of the octave equalizer with gains alternating between 12 dB and -12 dB.

This is different from known approaches with bi-quad peak filters, where usually $\Omega_M = \Omega_C$. Considering the flatness of the magnitude response in the respective band, it seems however valid to have slightly different Ω_M and Ω_C .

4. DESIGN EXAMPLES

4.1. Octave equalizer

Let us consider a 10-band octave equalizer with center frequencies

$$f_{C,i} = [30 \text{ Hz}, 60 \text{ Hz}, \dots, 15360 \text{ Hz}]$$

at a sampling frequency $f_s = 48 \text{ kHz}$. The resulting lower and upper cut-off frequencies $f_{L,i}$ and $f_{U,i}$ from (3) and (4), as well as the frequencies of maximum gain $f_{M,i}$ from (19) and the resulting coefficients $\cos(\Omega_{M,i})$ of the frequency-shifting all-pass are listed in Table 1.

We use a band-shelving filter of eighth-order (i.e., $M = 4$) for each band. For the resulting two fourth-order sections for each band, $c_0 = \cos(3\pi/8)$ and $c_1 = \cos(5\pi/8)$, respectively.

As a first test case, the equalizer is configured to gains alternating between 12 dB and -12 dB, so that $g_i = 3.98107$ and hence $V_i = 0.412538$ for odd i and $g_i = 0.251189$ and hence $V_i = -0.292054$ for even i . Application of (14) yields the K_i of Table 2. The resulting magnitude response in Figure 3 shows that the inter-band influence is sufficiently small, so that the difference between actual and desired gain in each band is hardly perceptible.

To study the imperfections due to non-complementary filter edges, we now set equal gain of 12 dB in all bands, yielding the same $g_i = 3.98107$ and hence $V_i = 0.412538$ for all bands. By application of (14), we find the K_i of Table 3.

The resulting amplitude response of the equalizer and the individual band-shelving filters is depicted in Figure 4, the deviation from 12 dB in Figure 5. The gain is almost exactly 12 dB at the

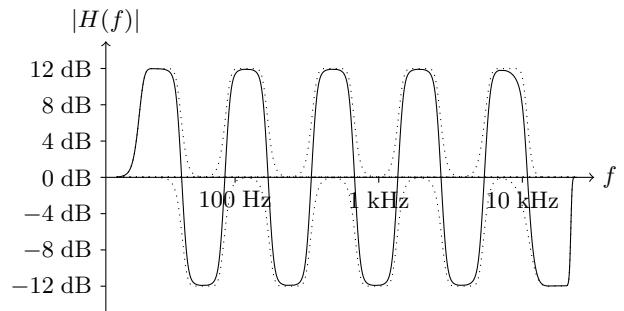


Figure 3: Magnitude response of the octave equalizer with eighth-order ($M=4$) band-shelving filters with gain alternating between 12 dB and -12 dB (solid line) and of the individual filters (dotted).

i	K_i	i	K_i	i	K_i
1	0.001168	5	0.018694	9	0.312322
2	0.002336	6	0.037407	10	0.724464
3	0.004673	7	0.074962		
4	0.009346	8	0.151123		

Table 3: Bandwidth coefficients K_i of the octave equalizer with equal gain (12 dB) in all bands.

center frequencies and at the cut-off frequencies, so it is not necessary to consider the gains configured in other bands when choosing the g_i for one filter, contrary to other sophisticated equalizer design methods [6]. The influence of one filter in other bands is limited to a small transitional region around the cut-off frequencies, where the magnitude response exhibits slight ripple no higher than that of traditional constant-Q equalizers [7].

Only in the transitional region between ninth and tenth band, the deviation from 12 dB exceeds 1 dB. This is an artifact of the bilinear transform, which results in filters with an upper edge close to the Nyquist frequency to have relatively flat lower edges. Thus, the lower edge of the tenth band and the upper edge of the ninth band have significantly different slopes, resulting in the larger ripple. For applications where this is not acceptable, the situation could be remedied to some extent by choosing a lower upper edge frequency for the tenth band, which for the current design lies outside the audible spectrum anyway. For example, by arbitrarily setting $f_{U,10} = 18.5 \text{ kHz}$, the ripple of Figure 6 can be obtained.

4.2. 1/3-Octave equalizer

Next we examine a 30-band 1/3-octave equalizer with center frequencies

$$f_{C,i} = [25 \text{ Hz}, 31 \text{ Hz}, \dots, 20319 \text{ Hz}],$$

again at a sampling frequency of 48 kHz. As for the octave equalizer, we use eighth-order ($M=4$) band-shelving filters.

The magnitude response for the same gain of 12 dB in all bands is depicted in Figure 7, the deviation from 12 dB in Figure 8. Again, the ripple stays below 1 dB up to 10 kHz and grows toward the highest bands. By applying the same trick as for the octave equalizer, reducing the upper edge of the highest band from 22807 Hz to 20.5 kHz, the ripple can be reduced to that shown in Figure 9.

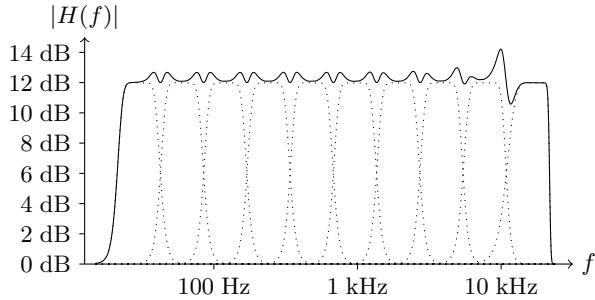


Figure 4: *Magnitude response of the octave equalizer with eighth-order ($M=4$) band-shelving filters with equal gain (12 dB) in all bands (solid line) and of the individual filters (dotted).*

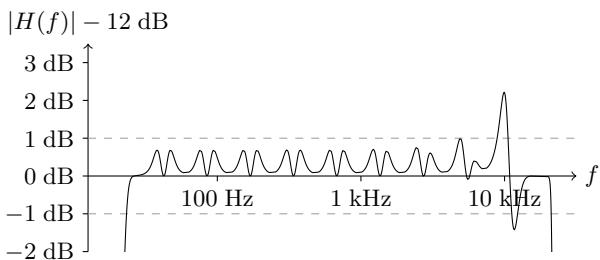


Figure 5: Deviation of the magnitude response of Figure 4 from the ideal constant 12 dB.

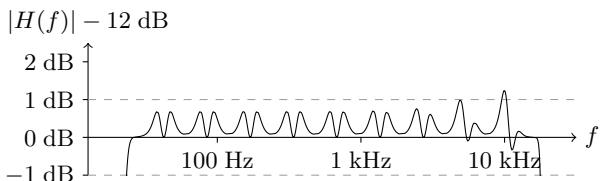


Figure 6: Deviation of the magnitude response from 12 dB for the modified octave equalizer with $f_{U,10} = 18.5$ kHz.

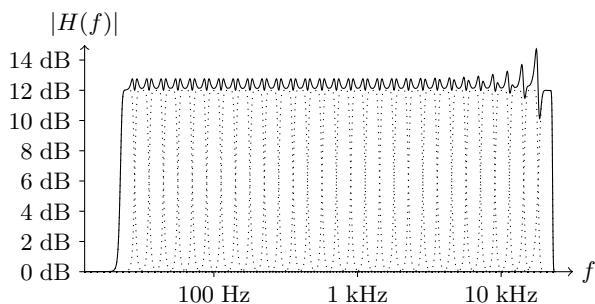


Figure 7: Magnitude response of the 1/3-octave equalizer with eighth-order ($M=4$) band-shelving filters with equal gain (12 dB) in all bands (solid line) and of the individual filters (dotted).

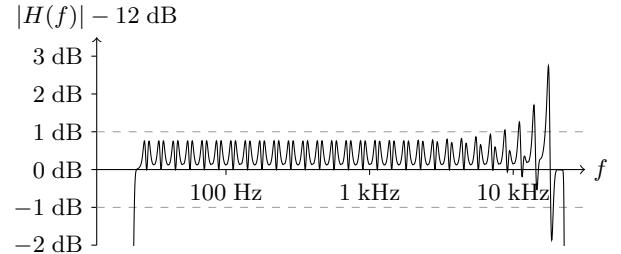


Figure 8: *Deviation of the magnitude response of Figure 7 from the ideal constant 12 dB.*

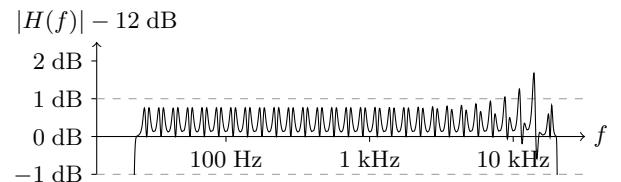


Figure 9: Deviation of the magnitude response from 12 dB for the modified 1/3-octave equalizer with $f_{U,30} = 20.5$ kHz.

5. CONCLUSIONS

We have presented a straight-forward design of graphic equalizers with minimum-phase behavior based on higher-order band-shelving filters. Thanks to the high filter order, the inter-band influence is very small, that is the gain in one band is almost completely independent from the gain in the other bands. Although no special care has been taken to design filters with complementary edges except for a suitable definition of the cut-off frequencies, the resulting amplitude deviation in the transitional region between the bands is very low. Despite a slight increase at high frequencies, the amplitude ripple should be sufficiently low for most applications.

6. REFERENCES

- [1] S. J. Orfanidis, "High-order digital parametric equalizer design," *J. Audio Eng. Soc.*, vol. 53, no. 11, pp. 1026–1046, 2005.
 - [2] M. Holters and U. Zölzer, "Parametric recursive higher-order shelving filters," in *120th Conv. Audio Eng. Soc.*, Paris, France, Paris, 2006, paper 6722.
 - [3] U. Zölzer, *Digital Audio Signal Processing*. New York: J. Wiley & Sons, 1997.
 - [4] R. M. Golden, "Digital filter synthesis by sampled-data transformation," *IEEE Trans. Audio and Electroacoustics*, vol. 16, no. 3, pp. 321–329, 1968.
 - [5] M. N. S. Swamy and K. S. Thyagarajan, "Digital bandpass and bandstop filters with variable center frequency and bandwidth," *Proc. IEEE*, vol. 64, no. 11, pp. 1632–1634, 1976.
 - [6] R. Miller, "Equalization methods with true response," in *116th Conv. Audio Eng. Soc.*, Berlin, Germany, 2004, paper 6088.
 - [7] D. Bohn and T. Pennington, "Constant- Q graphic equalizers," Rane Corporation, RaneNote 101, 1987, [Online] <http://www.rane.com/pdf/note101.pdf>.

PERFORMANCE-DRIVEN CONTROL FOR SAMPLE-BASED SINGING VOICE SYNTHESIS

Jordi Janer, Jordi Bonada, Merlijn Blaauw

Music Technology Group

Universitat Pompeu Fabra, Barcelona

{jjaner|jbonada|mblaauw}@iua.upf.edu

ABSTRACT

In this paper we address the expressive control of singing voice synthesis. Singing Voice Synthesizers (SVS) traditionally require two types of inputs: a musical score and lyrics. The musical expression is then typically either generated automatically by applying a model of a certain type of expression to a high-level musical score, or achieved by manually editing low-level synthesizer parameters. We propose an alternative method, where the expression control is derived from a singing performance. In a first step, an analysis module extracts expressive information from the input voice signal, which is then adapted and mapped to the internal synthesizer controls. The presented implementation works in an off-line manner processing user input voice signals and lyrics using a phonetic segmentation module. The main contribution of this approach is to offer a direct way of controlling the expression of SVS. The further step is to run the system in real-time. The last section of this paper addresses a possible strategy for real-time operation.

1. INTRODUCTION

One of the challenges of artificial voice synthesizers is to generate a realistic, human-like sound. Speech techniques pursue realistically synthesizing a given emotion (“sadness”, “fear”, “surprise”, etc.). In singing voice synthesis, besides emotion, the goal is to achieve natural sounding musical expression. Several systems have addressed this issue. These usually follow a strategy that consists in first training a model from real performances and then to apply this model in the synthesizer. These models will usually only describe one or a limited set of performance styles. Although good results are achieved, there is still an important issue that remains unaddressed: flexible control.

Control plays an important role in musical virtual instruments. In practical implementations of human voice synthesis systems the available amount of control and the voice quality achieved are bound to each other, leading to a trade-off between *explicit modeling techniques* and *sample-based techniques* [1]. The former allows very flexible control while the voice quality will generally be unconvincing. The latter, in contrast achieves a good voice quality, but with limited control. The challenge in the next years is to combine both techniques, permitting a wider control range while maintaining a good and natural voice quality.

Focusing on sample-based synthesizers, their control has an additional trade-off between database size and transformation post-processing required. We might devise different strategies. One could be building a large database with complete phrases of different expressive performances; another could be using only small database of a single set of phoneme samples with a “flat”

expression. Most Singing Voice Synthesizers (SVS) require two types of inputs: a melody, usually a MIDI score, and lyrics. To control the expression, we can either augment this MIDI score with an expressive model or allow the user to manually edit the control parameters using the synthesizer’s GUI. If we use expressive models, the type of control parameters is often very high-level, such as “happy or sad”, “bluesy”. The difficulty lies in defining a useful, preferably small, set of such parameters that allow meaningful combinations and a wide range of expression. Another big difficulty with this approach is mapping these parameters to low-level synthesizer controls such as the timing of phonetics, pitch, dynamics, etc., which are difficult to control directly by the user.

This work uses the sample-based spectral concatenation SVS introduced by Bonada and Loscos in [2]. This synthesizer picks samples from a database and concatenates one after the other, while applying different kinds of transformations. Good results are achieved by using a corpus of diphonemes instead of single phonemes. Using diphonemes, the samples not only contain the phonemes themselves, but also the transitions or articulations between phonemes. The diphoneme samples are further labeled automatically to indicate sustained parts of the samples and transitional parts of the sample to aid synthesis. The SVS reads a musical score, in the form of a MIDI score, a phonetic transcription of the lyrics and additional expressive parameters such as vibrato, type of attack, etc. The experiments use a singer database in Spanish. Although the described system runs off-line, our goal is to adapt it for real-time operation, as briefly addressed in the last section.

Having briefly introduced the limitations of controlling the expression of Singing Voice Synthesizers, we propose to control the expression of a sample-based Singing Voice Synthesizer’s output voice with another voice. We extract a set of descriptors of the input voice, which are then mapped to the synthesizer’s internal parameters. These parameters are: energy, fundamental frequency and vibrato. The phonetic timing of the synthesis output is also derived from the input voice. In the proposed system, the phonetic timing is generated by *automatic phonetic segmentation* using a similar approach as [3].

A potential application of this system is voice impersonation. Methods such as timbre mapping and transposition achieve good results for changing the character of the voice, e.g. gender change. However, with state-of-the-art voice processing techniques, it is difficult to impersonate a particular target singer by audio transformation solely. Although the timbre of the target can be reached in stationary sounds, most of the singer characteristics lie in the phonetic articulations, which are hard to achieve by means of transformation. Another use of this system can be for singing in a foreign

language not mastered by the singer, where a bad pronunciation might induce a negative effect on the singing voice quality. Using sample-based synthesizers we can improve the pronunciation with ease.

On the other hand, this work can be considered as the natural succession of the singing voice morphing system presented by Cano et al. [4]. In this real-time karaoke-oriented system, the user controlled the timing of a given song, while the synthesized voice was a morph between the user and the target professional singer. However, for each song to be performed it was required to record the professional performance for that specific song. Using a SVS based on samples of the target singer and the presented control, we would overcome this issue.

2. CONTROLLING VOICE SYNTHESIS EXPRESSION

Expressive sound synthesis control is mainly tackled from two different viewpoints. The first adds expression to a neutral musical score by means of expression models. The second looks at the interfaces that a performer uses to control Digital Musical Instruments (DMI). In our approach, we use an alternative method for controlling sound synthesis.

2.1. Control with expression models

Part of the research on musical expression deals with characterizing and modeling expressive performances in order to apply a given expression (e.g. “sad”, “happy”, “dark”, etc.) to neutral performance, or even to reproduce the playing style of a given performer [5]. First approaches worked solely at an abstract level, such as MIDI scores. Using machine learning techniques the system learns different types of manually annotated performances for a given score. Then it modifies the neutral performance to another synthesized one that conveys a particular expression. Recent studies tackle the modeling and control of expressive performances at a signal level. In these systems, given a neutral performance the user navigates in a two-dimensional expression space, controlling the mood of the transformed sound [6].

This type of control is very high-level. User actions are therefore limited to select among predefined expressive performances or at most to control the amount of morphing between two of them.

2.2. Control with input interfaces

Another way of applying expression is by means of a musical interface. Musical interfaces, also referred as *Musical Controllers* are mainly employed in performance situations and work in real time. The control of Digital Musical Instruments has been traditionally bound to the limitations of the MIDI protocol. Interfaces capture *musical gestures*; which gestures and how they are parameterized will determine the synthesized sound. A wide variety of musical controllers are available nowadays, ranging from traditional keyboards to sophisticated artificial vision systems.

In commercial singing voice synthesizers such as Yamaha’s Vocaloid system¹, we can use a MIDI interface for recording a MIDI score. In addition, the control parameters of the synthesizer can be manually edited using a GUI. The user can draw pitch and loudness contours, or add high-level expressive gestures such as vibrato. Also, the type of articulation (e.g. “legato”, “strong accent”) can be specified. A problem of this type of control is that

there is no *Musical Controller* we know appropriate for singing voice synthesis, as there are for other types of synthesis such as MIDI wind controllers for wind instrument synthesizers. This fact restricts the generation of expressive MIDI score suited to the voice. On the other hand, manually parameter editing does not solve the problem completely since it is a slow and tedious process, although very flexible.

2.3. Performance-driven control

An alternative way of applying expressive control to a synthesizer is to use an audio-stream as control. This concept is often referred in the literature as “Indirect Acquisition”. Several systems addresses this issue from different perspectives, either using a general audio-signal as control [7, 8], or specifically using the singing voice as control for other sounds [9]. In [3], Meron uses singing performances as input for an automatically trained system that produces high-quality singing voice synthesis.

In our approach the performance descriptors have a direct control of the synthesizer parameters. This type of control avoids the intermediate level of the MIDI score, which reduces the resolution of the musical gestures. With respect to the mapping, it can be kept simple since voice is controlling voice.

3. SYSTEM IMPLEMENTATION

Our implementation consists of three independent modules: *Performance Analysis*, *Phonetic Alignment* and the actual *Singing Voice Synthesizer*. The inputs are a sound file and a phonetic transcription of the lyrics in text format. In this section we introduce the different modules that allow us to control the synthesis with a performance. First, we describe the performance analysis and the phonetic alignment modules. Second, we look at the control layer of the synthesizer, observing how it affects the generated singing.

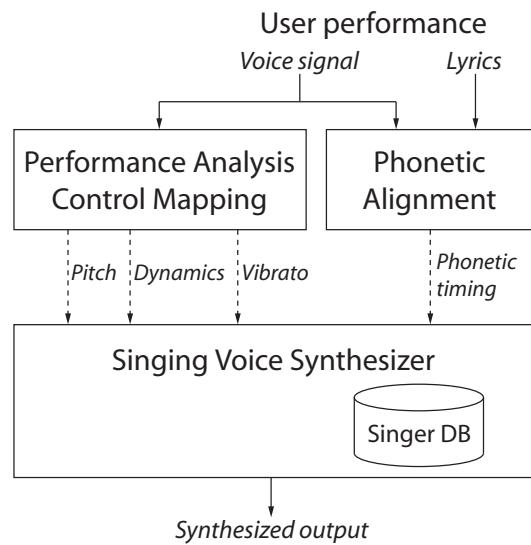


Figure 1: Block diagram of the complete system: *Performance Analysis*, *Phonetic Alignment*, and *Singing Voice Synthesizer*.

¹<http://www.vocaloid.com/>

3.1. Performance analysis and mapping

We analyze the voice signal and extract a set of descriptors that are mapped to the synthesizer's control. Expression in singing voice could be described in a low level by energy, fundamental frequency and timbre, which all vary dynamically along the time axis. Along with the mentioned descriptors, a higher-level analysis allows us to extract the phonetics and musical gestures such as vibrato or type of articulation.

From the set of low-level descriptors, we take energy and fundamental frequency. From the high-level analysis, we use the vibrato. Energy is computed directly from the windowed input voice signal. Several techniques and methods for fundamental frequency estimation have been extensively studied in the past decades. In our implementation we use a frequency-domain method based on work by Cano et al. [10]. Detecting the presence of vibrato in the signal is done by post-processing the fundamental frequency curve. The method implemented gives a measure of the probability of vibrato, and its depth and rate values. In brief, the method searches typical vibrato patterns (i.e. a quasi-sinusoidal component with a frequency in a range from 3 Hz to 8 Hz) after filtering out slow fundamental frequency variations due to intonation. Extracted descriptors are vibrato rate and vibrato depth. This method is still under development and needs to be studied further.

How input parameters are assigned to the synthesis parameters is known as mapping. In our case, the mapping is one-to-one, and the mapping layer consists basically in adapting the energy and fundamental frequency signals to the requirements of the synthesizer control layer. The energy curve is smoothed in order to fit the *dynamics* control of the synthesizer. The fundamental frequency curve also has to be adapted. The first requirement for the pitch control parameter is to be continuous. Thus, in transition and unvoiced phonemes when fundamental frequencies are missing, the curve is filled by smoothly interpolating the boundary values. Also, the rapid and small fluctuations of fundamental frequency not resulting from musical articulations, should ideally be removed. The reason is that these fluctuations are caused by the phonetics and how they are pronounced. Thus, they vary from one singer to another, and will be already present in the database samples. As a result, the computed pitch control is actually a smoothed and continuous version of the fundamental frequency analysis where musical gestures such as articulation and vibrato are kept. In addition, we have two options for controlling the vibrato: either we use the *pitch curve* extracted from the user performance, or a *pitch curve* of an actual vibrato by the recorded singer. The choice will depend on the quality of the user performance, since untrained singers will hardly produce a good vibrato. Finally, the parameters that are sent to the synthesizer are shown in the table 1.

3.2. Phonetic alignment

Up to this point, we have addressed the control of the musical expression described by pitch and dynamics contours. However, an important part of the singing voice expression belongs to phonetic information. In western classical singing, scores assign a phoneme to each note. For long notes, a vowel sound is sustained, while consonant sounds take place in the transitions. This concept is maintained in the used SVS [2]. In the phonetic timing of the internal score (see section 3.3), stationary voiced phoneme samples

Voice Descriptor	Synthesizer Parameter
Fundamental Frequency	Pitch
Energy	Dynamics
Vibrato Depth	Vibrato Depth
Vibrato Rate	Vibrato Rate

Table 1: *Performance descriptors extracted from the voice signal and corresponding synthesis control parameter.*

are looped while the duration of most remaining phoneme samples is kept unaltered.

In our approach, the phonetic timing is also driven by an input voice. Therefore, the duration of the phonemes are derived from those in the input performance. To extract the phonetic timing from the voice signal we use the segmentation module of the *Julius* Automatic Speech Recognition system [11]. The segmentation process is based on Hidden Markov Models (HMM), and uses acoustic models for Japanese in HTK format. Although the performances are in Spanish, the automatic segmentation works good enough since the phonetics of both languages are to some extent similar. However, we need to convert appropriately the Spanish phonetics symbols to the Japanese ones.

At first, we aligned all the phonemes in the lyrics to the corresponding phonemes from the input performance. We observed that with the used SVS, while time-scaling voiced phonemes does not significantly affect the synthesis quality, the time-scaling process of unvoiced sound produced unrealistic and artificial sound. In order to improve the sound quality, we introduced two alternatives modes of generating the phonetic timing. In the first, “voiced onset” mode, only voiced phoneme onsets are aligned, while in the second, “vowel onset” mode, only vowel onsets are aligned. Both options ensure that unvoiced sounds have the same duration as in the database, i.e. they are not time-scaled. In the figure 2, we observe the waveforms of the input performance and the two proposed alternative phonetic alignments. A first listening evaluation showed that the most natural sounding results are achieved with “voiced onset” alignment.

3.3. Synthesizer control layer

Internally the SVS used in this work has a small number of low-level controls that define the synthesis target. The *phonetic timing* defines which phoneme sounds at which time and its duration. *Pitch* and *dynamics* define target pitch and target singing intensity respectively and can be controlled directly or be derived from higher-level controls (MIDI notes and velocities) using models of pitch and dynamics curves. A number of other expressive controls control other sample transformations such as “roughness” or “breathiness” quality of the voice. Finally, vibrato controls, depth and rate, can be used to easily control the vibrato of the synthesized voice using a signal analysis of actual vibratos uttered by the recorded singer.

From these low-level controls an internal score is built which describes a sequence of samples and their transformation. The optimal sequence of samples is chosen so the overall transformation required is minimized. The cost function used is derived from the compression or expansion of samples needed to fit the given phonetic timing, the pitch and dynamics transformations needed to

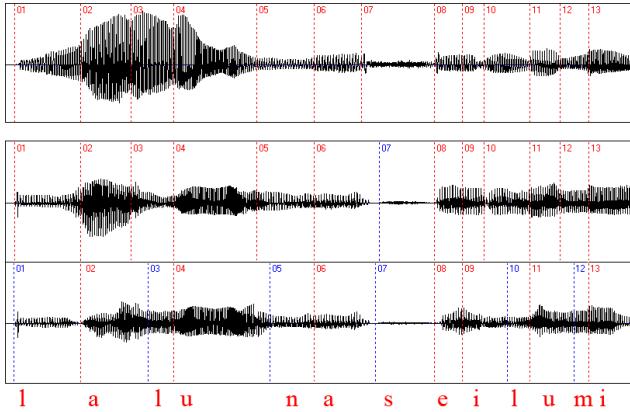


Figure 2: Two different modes of phonetic alignment: The second plot shows the synthesized output with all voiced phonemes aligned to the input segmentation (top plot). The bottom plot shows the synthesized output with only vowels aligned to the input segmentation (top plot).

reach the target pitch and dynamics curves and the continuity of the sample sequence.

4. A STRATEGY FOR REAL-TIME OPERATION

As we have mentioned, the current implementation runs off-line. However, our final aim is to have a system running in real-time. In this section we introduce some of the issues involved in a real-time operation.

The goal is to perform phoneme alignment to a text, which generates the phonetic timing information sent to the synthesizer. Our strategy will be based on the approach presented by Loscos et al. in [12]. This system uses HMM for the phoneme recognition, a Finite State Network (FSN) and a Viterbi algorithm to perform the text alignment.

The main limitation for implementing the approach presented in this paper in real-time is that the SVS uses diphoneme samples. Therefore, the synthesizer has to wait that the rightmost phoneme is detected in the phonetic segmentation, and then play the complete diphoneme sample. Obviously, this will introduce latency in our system. A way to minimize this effect would be to time-compress the left-hand side phoneme of the diphoneme sample, or to start playing the diphoneme sample with an appropriate offset.

5. CONCLUSIONS

We have introduced an approach that uses a singing performance to control the expression of a Singing Voice Synthesizer. It aims at improving the process of generating expressive synthetic singing, for which most state-of-the-art systems rely on modeling techniques or direct manipulation of low-level parameters. We have introduced the performance analysis module of our proposed system that extracts *pitch*, *dynamics* and *vibrato* information from an input voice signal. Given lyrics and a voice signal, the system can automatically perform phonetic segmentation from which the phonetic timing of the synthesis can be derived. Due to sound quality

limitations of the SVS used, we suggested two alternatives for the phonetic alignment besides a one-to-one alignment with the input voice. First experiments showed that best sounding quality and natural sounding singing is achieved aligning voiced phonemes only. Although our final aim is to implement a real-time control of SVS, the current off-line implementation already offers a direct and easy way of controlling synthetic singing. Some audio examples are also available online².

6. REFERENCES

- [1] X. Rodet, "Synthesis and processing of the singing voice," in *1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA 2002)*, Leuven, Belgium, 2002.
- [2] J. Bonada and A. Loscos, "Sample-based singing voice synthesizer by spectral concatenation," in *Proc. Stockholm Music Acoustics Conf. (SMAC'03)* Stockholm, Sweden, 2003, pp. 439–442.
- [3] Y. Meron, "High quality singing synthesis using the selection-based synthesis scheme," Ph.D. dissertation, University of Tokyo, 1999.
- [4] P. Cano, A. Loscos, J. Bonada, M. de Boer, and X. Serra, "Voice morphing system for impersonating in karaoke applications," in *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, 2000, pp. 109–12.
- [5] G. Widmer and W. Goebel, "Computational models of expressive music performance: The state of the art," *J. New Music Research*, vol. 33, no. 3, pp. 203–216, 2004.
- [6] S. Canazza, G. De Poli, C. Drioli, A. Roda, and A. Vidolin, "Modeling and control of expressiveness in music performance," *Proc. IEEE*, vol. 92, no. 4, pp. 286–701, 2004.
- [7] E. Métois, "Musical sound information: Musical gesture and embedding synthesis," Ph.D. dissertation, Massachusetts Institute of Technology, 1996.
- [8] T. Jehan and B. Schoner, "An audio-driven, spectral analysis-based, perceptually meaningful timbre synthesizer," in *110th Conv. Audio Eng. Soc.*, Amsterdam, the Netherlands, Amsterdam, Netherland, 2001. [Online]. Available: citeseer.nj.nec.com/448390.html
- [9] J. Janer, "Voice-controlled plucked bass guitar through two synthesis techniques," in *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, 2005, pp. 132–135.
- [10] P. Cano, "Fundamental frequency estimation in the SMS analysis," in *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, 1998, pp. 99–102.
- [11] A. Lee, T. Kawahara, and K. Shikano, "Julius — an open source real-time large vocabulary recognition engine," in *Proc. European Conf. on Speech Communication and Technology*, 2001, pp. 1691–1694.
- [12] A. Loscos, P. Cano, and J. Bonada, "Low-delay singing voice alignment to text," in *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, 1999, pp. 437–440.

²<http://www.iua.upf.edu/~jjaner/dafx06>

PARAMETERIZED MORPHING AS A MAPPING TECHNIQUE FOR SOUND SYNTHESIS

Chinmay Pendharkar, Michael Gurevich, Lonce Wyse

Mixed Media Modeling Lab

Institute of Infocomm Research, Singapore

ponce@i2r.a-star.edu.sg

ABSTRACT

We present a novel mapping technique for sound synthesis. The technique extends the familiar concept of morphing to the domain of synthesis parameters. A morph between defined points in the parameter space representing desirable sounds is itself parameterized with high-level controls. The choice of end points of the morph and the extent of the morph are used as input handles to map arbitrary control signals to the synthesis parameters. Additional off-line methods control the interpolation functions and selection of parameter points. We discuss a tool to allow creation, manipulation and usage of such mappings.

1. INTRODUCTION

Real-time interactive digital sound synthesis is becoming increasingly important in many applications such as music, games and personal electronics. Improvements in synthesis techniques and processing power, along with increased potential for expressivity make sound synthesis an attractive alternative to sample-based systems. A generic real-time interactive digital sound synthesis system can be modeled as a bank of synthesizers that expose certain parameters to be controlled. A hardware or software controller changes the parameters to obtain the intended sounds. The controller generates a number of digital control signals as input to the system, which must be mapped to the parametric control handles of the synthesis models.

This mapping is an important part of the synthesis system. Inadequate mappings can cause even rich models capable of dynamic, responsive sounds, to sound dull and “mechanical”. There remains a great deal of research of mapping techniques that allow synthesis models to give more expressive- or natural-sounding output, or to more substantially define the behavior of the system.

1.1. Related Works

Approaches to control in the form of mapping have existed in fields such as control systems for some time [1]. Only since the relatively recent advent of practical, parametric real-time digital synthesis has it become applicable to sound synthesis.

Over the years, many strategies have been proposed to approach mapping. A popular mapping strategy is to define formal deterministic relationships between control and synthesis parameters. These strategies are mainly based on techniques predominant in control theory, like linear algebra and matrix transformations [2].

Some authors choose to take non-linear and heuristic-driven approaches that lead to practical and interesting mapping strategies. These include using spatial layout of mapping or weighing functions over a representation of the input space [3] and the uses

of geometric shapes to define input parameter spaces and mapping such shapes to shapes of higher dimensionality [4].

Some of these systems introduce one or more intermediate or “middle” mapping layers between control and synthesis parameters. In such schemes, control parameters are mapped onto intermediate parameters, which are in turn mapped to synthesis parameters. These intermediate parameters may be arbitrarily defined, may describe some higher-level (*i.e.* perceptual) features of the desired sound [5, 6], or may represent some virtual system whose features are mapped onto the synthesis parameters. For example, Schatter et al. [7] use a gestural controller to manipulate virtual graphics objects, whose features are mapped to synthesis controls.

Multiple layered mapping can allow the intermediate mappings to be reused. The mapping from the middle layer to the synthesis parameters may remain unchanged, while another controller is mapped to the middle layer [8]. Multi-layered systems are also one way of dealing with the problem of dimensional asymmetry. Dimensional asymmetry occurs when the dimension of the input control parameters and the output synthesis parameters is not equal. As such, mapping can be considered as a dimension reduction problem, and certain dimension reduction schemes, like simplicial interpolation [9], can be used as a mapping strategy.

Though an m -input n -output configurable mapping technique is very generic and widely useful, a mapping technique with restricted input dimensions can be practical. Gestural control systems often have a limited number of controls with respect to the number of synthesis parameters. Such low input dimension mapping techniques allow users to comprehend and explore the mappings in much greater extent. Low-dimensional gestural controls can be more intuitive for the user and thus can lead to more expressive mappings. One example of this is a mapping strategy proposed by Bencina [10]. In the strategy the inputs are restricted to a 2-dimensional space. However no restriction is placed on the number of dimensions of the output.

2. EVALUATING MAPPINGS

We define three important criteria for analyzing mappings: complexity, expressivity, and intuitiveness. There is certainly a strong relationship between a mapping technique and the degrees to which these properties are reflected in the mappings that are generated. One could conceive of quantitative metrics for assessing these, but this is beyond scope of this paper. We leave these as subjective assessments in order to create a vocabulary for discussing and informing the design of mapping techniques.

The complexity, especially the computational complexity of mappings, is an important factor when designing mapping for real-time audio synthesis engine. Complexity can be conceived as related to the number of computational or logical steps required to

generate a synthesis parameter from an input control change. Expressivity refers to the ability of the user to generate a rich variety of sonic output from the synthesizer, under deliberate control. Mappings resulting in output that, on one hand “always sounds the same” or on the other hand are “random” or “uncontrollable” from the user’s perspective are not expressive. Expressivity is not a measurable property of the mapping, but it’s an affordance the mapping can provide. Exceptionally talented performers can make poor mappings expressive. However, by expressive mappings we mean those that relatively easily enable a broad range of expression. Intuitiveness in a mapping can be seen as related to the user’s mental model of the mapping [11]. If the user requires only a simple conception of how the mapping works, then it would be considered intuitive.

The goal in creating a successful mapping strategy is to achieve an optimal combination of the three criteria. A good mapping strategy should be intuitive and produce expressive mappings, with the least amount of complexity possible in the mapping, but no less.

3. PARAMETERIZED MORPHING

Morphing is a process used widely in image processing. It is used in animations and motion pictures to change from one image to another through a seamless transition. This is generally achieved by interpolating between certain common features in the initial and final images.

Audio morphing normally describes the generation of a transition between two segments of recorded or synthesized audio. As in image morphing, it is normally achieved by interpolating between sets of features that are determined after the sound has been rendered, based on analysis of the audio signal in the time, frequency, or perceptual domains. For example, Slaney [12] describes a technique for morphing between sounds by interpolating between extracted spectral shapes and pitch.

3.1. Morphing in Parameter Space

With parametric, real-time synthesis, one can appropriate the term morphing and extend its application to interpolation in the domain of the synthesis parameters themselves, since they are available at the time of sound generation [13]. If for a particular synthesis model, sound A can be specified by a vector of parameters \mathbf{P}_A , and sound B can be specified by a vector of parameters \mathbf{P}_B , then we can generate a morph between A and B by interpolating between \mathbf{P}_A and \mathbf{P}_B . Of course, the extent to which this creates a perceptual morph depends on the nature of the model and on the interpolation functions. For many classes of models, with appropriate interpolating functions, this can be an effective way of navigating sound spaces.

3.2. Mapping by Parameterized Morphing

We propose to use this concept of parameter-space morphing in the context of mapping. The basic mapping scheme is as follows. For any synthesis model, sets of parameters are defined in advance, corresponding to particular desirable or interesting points in the model’s synthesis space. A pair of parameter sets can be chosen and set to be the end points for the morph. The morphing control would then choose the extent to which each of the parameter sets contribute to the output of the mapping.

This is achieved with some interpolating function used over each common parameter pair in the parameter sets. The user can also set the parameters at the end point dynamically, thus exploring the synthesis space.

This scheme leads to a $3-n$ mapping, where the mapping takes in the 2 discrete control inputs, and a continuous control inputs, and yields any number of continuous synthesis parameter outputs. The 2 discrete inputs are used to choose which parameter sets should be the end points of the morph, by indexing them from a list. The continuous input, a slider, could then determine the extent of the morph. See Figure 1 for a conceptual representation of parameterised morphing.

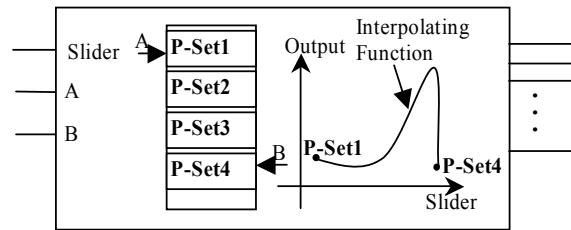


Figure 1: *Conceptual representation of parameterized morphing.*

3.3. Interpolation

A simple linear interpolation scheme will reduce the mapping to a simple range modification function and thus limit the utility and expressivity of the mapping. For example, while synthesizing the sound of a helicopter, if the desired control parameter of the mapping is the distance of the listener from the helicopter, the relationship between the input and the volume parameter of the model would be following the inverse square law. If the interpolation is limited to simple linear schemes, such common scenarios would not be addressable. Thus, a need for non-linear interpolation schemes arises. Piece-wise interpolation functions should also be available for dealing with discontinuous synthesis parameter spaces.

An interpolation scheme in morphing decides the output value of a certain feature, or parameter in sound synthesis, according to the two values being interpolated between. In the general case, the mathematical expression for the interpolation would be.

$$\mathbf{P}_O = (1 - \eta(s)) \cdot \mathbf{P}_A + \eta(s) \cdot \mathbf{P}_B \quad (1)$$

where $\eta(s)$ is the non-linear weighting function, \mathbf{P}_A & \mathbf{P}_B are the two parameter sets to be morphed between and s is the normalised value of the slider and \mathbf{P}_O the output of the morph.

4. ADDITIONAL FEATURES

With this morphing scheme as a basis, we propose a series of extensions in order to design a usable mapping software system which attempts to satisfy our criteria of complexity, intuitiveness and expressivity. The objective is to create a mapping application that can operate with a variety of synthesis systems and controllers (physical or virtual). However, the large numbers of available synthesis systems and controllers present a challenge to design a single application that would offer universal connectivity.

We therefore designed a general-purpose mapping application, around which different interfaces can be defined, both for incoming control signals and outgoing synthesis parameters. The front-end interface can translate the control signals to the appropriate internal representation of the mapping system; we can call this a premapping. Similarly, the back-end interface translates the internal representation of the generated synthesis parameters into a protocol and format supported by the receiving application. See Section 5 for further details of the implementation.

4.1. Partitioning and Maplets

An extension to the morphing scheme arose from the need to have more control over the mapping. If all the synthesis parameters are considered to span an n -dimensional space, then each parameter set of length m leaves $(n - m)$ parameters undefined. Morphing between two such parameter sets, the mapping does not have any control over the remaining $(n - m)$ parameters. Furthermore, this scheme cannot produce divergent mappings as defined by Rovan *et al.* [14].

To address both these limitations, we introduce the concept of maplets. Maplets are objects, which contain the ability to do a single parameterized morph as discussed above. A complete map can contain one or more maplets.

Maplets are based on the idea of partitioning the set of all output parameter into groups of parameters that are related, especially in their dynamics with respect to an input parameter. For example, all parameters which should vary exponentially with respect to a single control may be grouped together. Such groups can be assigned to individual maplets, thus allowing each group to have independent morphing functionality and inputs. Maplets serve as the atomic mapping entity.

Furthermore, maplets allow for multiple pairs of parameter sets to be morphed simultaneously. This is useful when controlling complex sound models with multiple sets of parameters closely related in their behaviour. This feature is also needed when different interpolation functions are needed to map different groups of parameters. For example, when controlling a sound model of a car, it would be effective to control all the parameter related to the tyres separately from all the parameters related to the engine.

This scheme represents a $(2 + x) \rightarrow n$ mapping, where x is any integer. The scheme need at least 1 continuous and 1 discrete control parameter, since Indices A and B could practically be premapped to 1 discrete input, for the lowest dimension case. Furthermore, multiple maplets can be manipulated by the same control parameter, thus allowing multiple sets of synthesis parameters to vary differently over a given input. Divergent mappings can now be generated by this scheme [14].

5. IMPLEMENTATION

We developed a tool to build and use mappings using parameterized morphing. The aim was to have a user friendly and intuitive means for designing mappings, and to test the practicality and verify the concept of the mapping scheme. Java was chosen as the development language for this project. Its mobility and portability have been incentives for its use in the development of a wide variety of applications for servers, PCs and mobile devices.

Modularity is important in the design of the system, since many of the subsystems are individual components that could be used on their own in a variety of mapping-related applications.

The core mapping functionalities are separated from the communication subsystem and the GUI. See Figure 2.

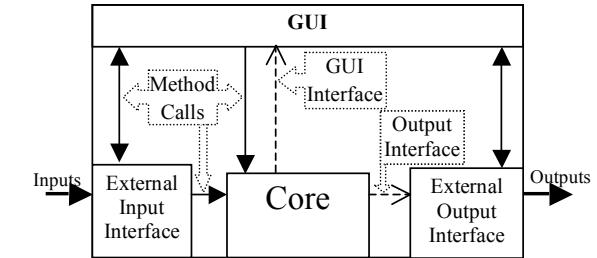


Figure 2: Structure of the mapping tool.

The core mapping functionality contains the interpolating and morphing logic of the mapping. It contains the maplets, the pre-defined parameter sets and the interpolation functions accessed through method calls.

The communication subsystem is based on Java Interfaces, which are provided in the core. Relevant communication systems can be designed according to the type of communication required by the application. Such a system can then be attached to the core using the interfaces provided. A default communication system has been implemented using the OpenSound Control (OSC) [15] protocol.

The GUI allows the tool to be used interactively. It provides functionality to audition the mappings that are created and attach interfaces to the mapping. It also gives the user a view of the current state of the system and the ability to store and load maps dynamically.

6. DISCUSSION

This implementation represents a multi-layered strategy, in that the notion of parameter sets represents an intermediate construct between the input and output parameters. One feature of this, as Wanderley [8] highlights, is that the mappings from parameter sets to synthesis parameters can persist regardless of the control parameters. Parameter sets can be reused in different maps and maplets. From the user's perspective, the concept of parameter sets also simplifies the mental representation of a potentially large space of synthesis parameters. Once parameter sets representing desirable points in the synthesis space have been defined in an off-line process, the user can ignore the target parameters themselves, and rather conceive of two sounds and a morph between them.

Mappings can be classified as one-to-one, convergent and divergent [14]. A good mapping scheme allows for all three types of mappings to be defined. The mapping scheme discussed allows one-to-one as well as divergent mapping. However, it is unable to produce a convergent mapping since, such a mapping does not fit easily into this morphing framework. If there is a need for simple convergent mappings to be coupled with this mapping scheme, they can easily be added as the pre-mapping layer to this mapping scheme.

7. FUTURE IMPROVEMENTS

There are a few areas for improvement that will be addressed in further work. These include further refinements to the specifica-

tion of the interpolation function. A tool that would allow users to graphically specify the shape of the interpolator in real-time is currently under development. Currently, the user must manually specify the OSC interface, the default communication system. This will be extended with a tool that can automatically explore the OSC address spaces of both the controller and synthesizer. This would be a key application for the implementation of a proposed OSC query scheme [16].

Interfaces for protocols other than OSC, including MIDI, and a native interface to our own Java-based synthesis environment [17] also are under development. The source code of this project will be made available shortly, pending internal approval.

8. REFERENCES

- [1] K. Ogata, *Modern Control Engineering*, 4th ed. Prentice Hall, 2001.
- [2] F. Bevilacqua, R. Müller, and N. Schnell, “MnM: a Max/MSP mapping toolbox,” in *Int. Conf. New Interf. for Musical Expr. (NIME’05)*, Vancouver, Canada, 2005, pp. 85–88.
- [3] A. Momeni and D. Wessel, “Characterizing and controlling musical material intuitively with geometric models,” in *Int. Conf. New Interf. for Musical Expr. (NIME’03)*, Montréal, Canada, 2003, pp. 54–62.
- [4] D. van Nort, M. M. Wanderley, and P. Depalle, “On the choice of mappings based on geometric properties,” in *Int. Conf. New Interf. for Musical Expr. (NIME’04)*, Hamamatsu, Japan, 2004, pp. 87–91.
- [5] D. Arfib, J. Couturier, L. Kessous, and V. Verfaille, “Strategies of mapping between gesture data and synthesis model,” *Organised Sound*, vol. 7, no. 2, pp. 122–144, 2002.
- [6] A. Hunt and M. M. Wanderley, “Mapping performer parameters to synthesis engines,” *Organised Sound*, vol. 7, no. 2, pp. 97–108, 2002.
- [7] G. Schatter, E. Züger, and C. Nitschke, “Synaesthetic approach for a synthesizer interface based on genetic algorithms and fuzzy sets,” in *Proc. Int. Comp. Music Conf. (ICMC’05)*, Barcelona, Spain, 2005, pp. 664–667.
- [8] M. M. Wanderley and P. Depalle, “Gestural control of sound synthesis,” *Proc. IEEE, Spec. Issue on Eng. and Music – Supervisory Control and Auditory Communication*, vol. 92, no. 4, pp. 632–644, 2004.
- [9] C. Goudeseune, “Interpolated mappings for musical instruments,” *Organised Sound*, vol. 7, no. 2, pp. 85–96, 2002.
- [10] R. Bencina, “The Metasurface – applying natural neighbour interpolation to two-to-many mapping,” in *Int. Conf. New Interf. for Musical Expr. (NIME’05)*, Vancouver, Canada, 2005, pp. 101–104.
- [11] D. A. Norman, *Some Observations on Mental Models*. D. Gentner and A. L. Stevens (eds.), Lawrence Erlbaum Associates, 1983.
- [12] M. Slaney, M. Covell, and B. Lassiter, “Automatic audio morphing,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, 1996, pp. 1001–1004.
- [13] E. Altman and L. Wyse, *Emergent Semantics from Media Blending*. Srinivasan and Nepal (Eds.), The Idea Group Inc., 2004.
- [14] J. Rovan, M. M. Wanderley, S. Dubnov, and P. Depalle, “Instrumental gestural mapping strategies as expressivity determinants in computer music performance,” Synthesis Team/Real-Time Systems Group, Ircam, Paris, France, Tech. Rep., 1997.
- [15] M. Wrig and A. Freed, “Open SoundControl: A new protocol for communicating with sound synthesizers,” in *Proc. Int. Comp. Music Conf. (ICMC’97)*, Thessaloniki, Greece, 1997, pp. 101–104.
- [16] A. Schmeder and M. Wright, “A query system for Open Sound Control,” in *Proc. 2004 OSC Conf.*, Accessed Mar. 21, 2006, [Online] <http://www.opensoundcontrol.org/proceedings>
- [17] L. Wyse, “A sound modeling and synthesis system designed for maximum usability,” in *Proc. Int. Comp. Music Conf. (ICMC’03)*, Singapore, 2003, pp. 447–451.

TOWARDS A NEW CONCEPTUAL FRAMEWORK FOR DIGITAL MUSICAL INSTRUMENTS

Joseph Malloch, David Birnbaum, Elliot Sinyor, Marcelo M. Wanderley

Input Devices and Music Interaction Laboratory

McGill University, Montreal

{joseph.malloch|marcelo.wanderley}@mcgill.ca

{elliot.sinyor|david.birnbaum}@mail.mcgill.ca

ABSTRACT

This paper describes the adaptation of an existing model of human information processing for the categorization of digital musical instruments in terms of performance context and behavior. It further presents a visualization intended to aid the analysis of existing DMIs and the design of new devices. Three new interfaces constructed by the authors are examined within this framework to illustrate its utility.

1. INTRODUCTION

When considering and categorizing devices that produce sound, it is common to become entangled in the differentiation of instruments, musical toys, and installations. Even within categories, confusion arises: conceptual models of musical instruments vary according to historical, cultural, and personal biases. New musical devices have varying degrees of success in penetrating the conceptual boundary between instrument and non-instrument, and frequently their path into the instrument domain is unexpected from the perspective of design intentionality. The issue is further confused by a layer of artistic interpretation, exploding the possible definitions of "instrument" to virtually any conceivable artifact that can involve sound (including the absence of sound). "Instrument" can thus refer to a traditional acoustic device, a controller with no specific mapping, a software program that maps control input to musical output, or can be synonymous with a musical piece itself, in which the interface (including its physical component) is integrated with musical sound output in the composer's expressive intent [1]. However, a systematic investigation of the design space of a musical device (such as dimension space analysis [2]) promotes an understanding of musical devices that considers both design goals and constraints arising from human capability and environmental conditions.

For the purposes of this investigation, the definition of "musical instrument" will be restricted to refer to a sound-producing device that can be controlled by a variety of physical gestures and is reactive to user actions [3]. A digital musical instrument (DMI) implies a musical instrument with a sound generator that is separable (but not necessarily separate) from its control interface, and with musical and control parameters related by a mapping strategy [4]. While computers are an essential part of a system such as this, the representation of the computer as a symbolic, metaphorical machine generating function-relationships to which we interface sensor and feedback systems does not adequately articulate its role in problem-posing task domains such as music composition and performance.

It has been said that, with respect to computer music, the differentiation between computer and musical instrument is a misconception, and this is a problem that has its solution in interface design [5]. Indeed, a computer may be used to *contain* structural components of an instrument, or many instruments, whose limits are only defined in terms of the computer's ability to implement known sound synthesis, signal processing, and interfacing methods. But if the "computer = instrument" paradigm is used, it is likely to leave the impression that digital instruments are also general purpose tools, and that the freedom to change mapping and feedback parameters arbitrarily provides the player with a better musical tool. Instead, the computer can be more aptly viewed as a semiotic, connotative machine that hypothesizes design criteria rather than exclusively representing *a priori* interaction metaphors based on the cultural and personal experience of the user [6]. Understanding the computer in this way endows it a constitutive role in performance behaviors that are not guided by explicit intention and evaluation of feedback, and directs scrutiny toward a variety other factors.

Fields of research that have been applied to instrument analysis and development range from human-computer interaction [7], theories of design [8], music cognition and perception [9], organology [10], and artistic/musicological approaches [11], to name only a few. We propose another possible approach, tying together ideas from human-machine interaction and music performance practice by emphasizing the *context* of a musical performance.

2. A HUMAN-MACHINE INTERACTION APPROACH

We have developed a paradigm of interaction and musical context based on Jens Rasmussen's model of human information processing [12], previously used to aid DMI design in [13]. Rasmussen examines the functions of "man-made systems" and human interaction in terms of the user's perception and the *reasons* (rather than causes) behind system design and human behavior. He describes interaction behaviors as being *skill-, rule-, or knowledge-based*. Rasmussen himself suggests that *knowledge-based* might be more appropriately called *model-based*, and we believe this term more clearly denotes this mode of behavior, particularly during performance of music, as "musical knowledge" can have various conflicting definitions.

Briefly, skill-based behavior is defined as a real-time, continuous response to a continuous signal, whereas rule-based behavior consists of the selection and execution of stored procedures in response to cues extracted from the system. Model-based behavior refers to a level yet more abstract, in which performance is directed towards a conceptual goal, and active reasoning must be

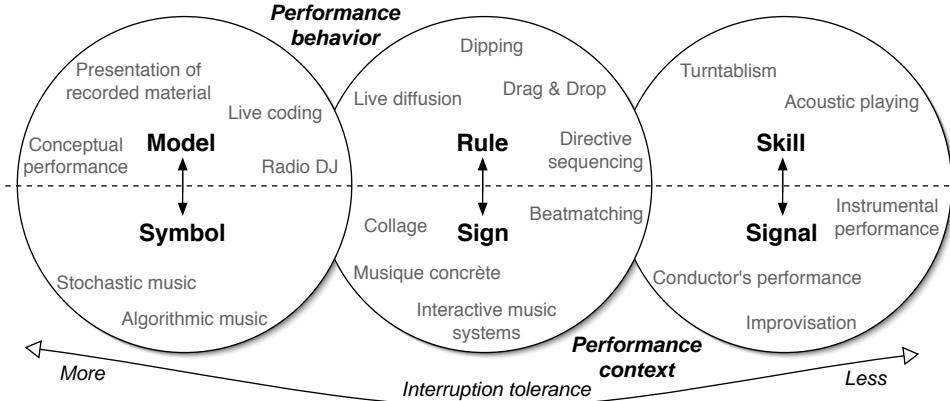


Figure 1: A visualization of the framework.

used before an appropriate action (rule- or skill-based) is taken. Each of these modes is linked to a category of human information processing, distinguished by their human interpretation; that is to say, during various modes of behavior, environmental conditions are perceived as playing distinct roles, which can be categorized as *signals*, *signs*, and *symbols*. Figure 1 demonstrates our adaptation of Rasmussen's framework, in which both performance behaviors and context are characterized as belonging to model/symbol, rule/sign, or skill/signal domains.

2.1. Skill-, Rule-, and Model-based Musical Performance

Skill-based behavior is identified by [8] as the mode most descriptive of musical interaction, in that it is typified by rapid, coordinated movements in response to continuous signals. Rasmussen's own definition and usage is somewhat broader, noting that in many situations a person depends on the experience of previous attempts rather than real-time signal input, and that human behavior is very seldom restricted to the skill-based category. Usually an activity mixes rule- and skill-based behavior, and performance thus becomes a sequence of automated (skill-based) sensorimotor patterns. Instruments that belong to this mode of interaction have been compared more closely in several ways. The "entry-fee" of the device [5], allowance of continuous excitation of sound after an onset [9], and the number of musical parameters available for expressive nuance [14] may all be considered.

It is important to note that comparing these qualities does not determine the "expressivity" of an instrument. "Expressivity" is commonly used to discuss the virtue of an interaction design in absolute terms, yet expressive interfaces rely on the goals of the user and the context of output perception to generate information. Expression, a concept that is unquantifiable and dynamically subjective, cannot be viewed as an aspectual property of an interaction. Clarke, for example, is careful not to state that musical expressivity depends on the possession of a maximum or minimum number of expressive parameters; instead, he states that the range of choices available to a performer will affect performance practice [14]. A musician can perform expressively regardless of the choices presented, but must transfer her expressive nuance into different structural parameters and performance behavior modes. This relates to the HCI principle that an interface is not improved by simply adding more degrees of freedom (DOF); rather, at is-

sue is the tight matching of the device's control structure with the perceptual structure of the task [15].

During rule-based performance the musician's attention is focused on controlling a process rather than a signal, responding to extracted cues and internal or external instructions. Behaviors that are considered to be quintessentially rule-based are typified by the control of higher-level processes and by situations in which the performer acts by selecting and ordering previously determined procedures, such as live sequencing, or using "dipping" or "drag and drop" metaphors [5]. Rasmussen describes rule-based behavior as *goal-oriented*, but observes that the performer may not be explicitly aware of the goal. Similar to the skill-based domain, interactions and interfaces in the rule-based area can be further distinguished by the rate at which a performer can effect change and by the number of task parameters available as control variables.

The model domain occupies the left side of the visualization, where the amount of control available to the performer (and its rate) is determined to be low. It differs from the rule-based domain in its reliance on an internal representation of the task, thus making it not only goal-oriented but goal-controlled. Rather than performing with selections among previously stored routines, a musician exhibiting model-based behavior possesses only goals and a conceptual model of how to proceed. He must rationally formulate a useful plan to reach that goal, using active problem-solving to determine an effectual course of action. This approach is thus often used in unfamiliar situations, when a repertoire of rule-based responses does not already exist.

2.2. Signals, Signs and Symbols

By considering their relationship with the types of information described by Rasmussen, performance context can also be distributed among the interaction domains. The signal domain relates to most traditional instrumental performance, whether improvised or pre-composed, since its output is used at the signal-level for performance feedback. The sign domain relates to sequenced music, in which pre-recorded or pre-determined sections are selected and ordered. Lastly, the symbol domain relates to conceptual music, which is not characterized by its literal presentation but rather the musical context in which it is experienced. In this case, problem-solving and planning are required — such as in the case of conceptual scores, which may lack specific "micro-level" musical

instructions but instead consist of a series of broader directives or concepts that must be actively interpreted by the performer [16].

3. USING THE VISUALIZATION

Consider the drum machine, for instance the Roland TR-808. To create a rhythm, the user selects a drum type using a knob, and then places the drum in a 16-note sequence by pressing the corresponding button(s). When the “start” button is pressed, the sequence is played automatically at the selected tempo. Using the diagram, this is clearly a rule-based way to perform a rhythm. A skill-based example in a similar vein would be using a drum machine controlled by trigger pads that require the performer to strike the pads in real-time. Of course, a drum kit would be another obvious skill-based example. Using the same musical idiom but on the opposite end of the diagram we can consider using the live coding tool Chuck [17] to create the same rhythm. Here the performer would take a model-based approach: playing a beat would require breaking the task into sub-tasks, namely creating a loop and deciding on an appropriate rest interval based on the desired tempo.

4. APPLICATIONS AND IMPLICATIONS

4.1. The Rulers

The *Rulers*, an interface developed by one of the authors¹, was designed to evoke the gesture of plucking or striking a ruler (or “tine”) that is fixed at one end. Utilizing infrared reflect sensors, the tines play one of seven percussive samples, slices extracted from the Amen breakbeat [18]. Each sample consists of either a single drum or cymbal, or a sequence of drums comprising a short rhythm. Because the samples contain sub-rhythms, the instrument must be played in the context of a global tempo set in the Max/MSP patch that remains fixed during the course of the performance. When plucked, each tine oscillates for a different amount of time; the sample it plays back has been assigned strategically, so that the length of sound output and physical oscillation are correlated. This provides an element of visual and passive haptic feedback to the player, as perceptual characteristics of the sound are tightly coupled to the physical construction of the interface. Output amplitude is determined by the amplitude of the tine’s oscillation, leading to control over the amplitude of initial excitation and damping — characteristics that classify it as an instrument that outputs musical events with a *non-excited middle* [9].

Playing the *Rulers* is principally a skill-based behavior, requiring constant performer input to sustain musical output. While it does not allow for continuous excitation, it does allow continuous modification after an onset, as the tines may be damped to affect the decay rate of musical events. Yet because the musical output contains fixed elements of rhythm over which the performer has no real-time control, the interaction is also directing short-time musical processes that do not originate from the player but are hard-wired into the instrument/system; it therefore incorporates elements of both the signal and sign domains.

4.2. The Celloboard

Another new interface, the *Celloboard*, was designed to tie sound output with continuous energy input from the performer. Using



Figure 2: *The Rulers*, by David Birnbaum.

contact microphones and accelerometers to sense the amplitude, direction and pressure of bowing gestures, this controller allows the continuous excitation, as well as modification, of its sound. Pitch and timbral sound elements, created using scanned synthesis [19], are controlled by sensors on the controller’s neck, sensing position and pressure of touch on two channels, and also strain of the neck itself on one axis.

With its many continuously-controlled parameters and integral mapping, the *Celloboard* controller easily fits into the skill/signal domain. Any interruption in performance will immediately be audible since sound output requires constant bowing of the interface. It possesses a high “entry-fee” for both sound excitation and modification, and does not easily allow high-level control of musical processes. Adaptations suggested by the framework might be to map the physical controls to a synthesis technique even less process-based than the present scanned synthesis implementation, or to allow the selection of discrete pitches (effectively lowering the modification entry-fee), in order to make the instrument more quickly mastered if a larger user-base is desired.



Figure 3: *The Celloboard*, by Joseph Malloch.

4.3. The Gyrotyre

The *Gyrotyre* [20] is a handheld bicycle wheel-based controller that uses a gyroscope sensor and a two-axis accelerometer to provide information about the rotation and orientation of the wheel. It was designed as a controller around a small group of mappings that would make use of the continuous motion data. We will look at two *Gyrotyre* mappings in order to place them in the framework.



Figure 4: *The Gyrotyre*, by Elliot Sinyor [20].

¹Developed at the Center for Computer Research in Music and Acoustics 2004 Summer Workshop.

In the first mapping, the interface controls playback of a sound file scrubbed backwards and forwards by spinning the wheel, evoking a turntable interface. The wheel may be spun very fast and then damped to achieve a descending glissando effect, or it may be kept spinning at a constant speed. This DMI (i.e. this particular mapping of the Gyrotyre controller) fits in the skill-based domain of the framework.

In an arpeggiator mapping, spinning the wheel while pressing one of the keys on the handle repeatedly cycles through a three-note arpeggio whose playback speed is directly correlated to the speed of the wheel. The performer changes the root note and the octave by tilting the Gyrotyre. In this case, performance behavior is predominantly rule-based. The musician reacts to signs, such as the current root note and the speed of the playback. The skill-based aspect of performance is the sustaining of a constant speed of rotation while holding a steady root-note position. Ostensibly, a performer could practice to develop these skills, but this would offer little advantage as the instrument outputs discrete, predetermined pitches. Considering musical context could lead to two changes: the mapping could be altered to reflect the required skill in the musical output, and/or the root-note selection method could be mapped to a gesture more appropriate for a rule-based behavior.

5. CONCLUSIONS

Our framework is intended to clarify some of the issues surrounding music interface design in several ways. Firstly, it may be used to analyze, compare, and contrast interfaces and instruments that have already been built, in order to facilitate an understanding of their relationships to each other. Additionally, the design of new instruments can benefit from this description, whether the designer intends to start with a particular interface concept or wishes to work within a specific musical context. Finally, it may be useful for adapting existing DMIs to different musics, or to increase their potential for performance within a specific musical context.

6. ACKNOWLEDGMENTS

The authors would like to thank Ian Knopke, Doug Van Nort, and Bill Verplank. The first author received research funds from the Center for Interdisciplinary Research in Music Media and Technology, and the last author received funding from an NSERC discovery grant.

7. REFERENCES

- [1] N. Schnell and M. Battier, "Introducing composed instruments, technical and musicological implications," in *Int. Conf. New Interf. for Musical Expr. (NIME'02)*, Dublin, Ireland, 2002, pp. 138–142.
- [2] D. Birnbaum, R. Fiebrink, J. Malloch, and M. Wanderley, "Towards a dimension space for musical devices," in *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, 2005, pp. 192–195.
- [3] B. Bongers, *Trends in Gestural Control in Music*. France: Ircam, 2000, ch. Physical interaction in the electronic arts: Interaction theory and interfacing techniques for real-time performance.
- [4] M. M. Wanderley and P. Depalle, "Gestural control of sound synthesis," *Proc. IEEE, Spec. Issue on Eng. and Music – Supervisory Control and Auditory Communication*, vol. 92, no. 4, pp. 632–644, 2004.
- [5] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music J.*, vol. 26, no. 3, pp. 11–22, 2002.
- [6] M. Hamman, "From symbol to semiotic: Representation, signification, and the composition of music interaction," *J. New Music Research*, vol. 28, pp. 90–104, 1999.
- [7] N. Orio, N. Schnell, and M. M. Wanderley, "Evaluation of input devices for musical expression: Borrowing tools from HCI," *Computer Music J.*, vol. 26, no. 3, pp. 62–76, 2001.
- [8] B. Cariou, "Design of an alternative controller from an industrial design perspective," in *Proc. Int. Comp. Music Conf. (ICMC'92)*, San Francisco, USA, 1992, pp. 366–367.
- [9] D. Levitin, S. McAdams, and R. L. Adams, "Control parameters for musical instruments: a foundation for new mappings of gesture to sound," *Org. Sound*, vol. 7, no. 2, pp. 171–189, 2002.
- [10] T. Kvifte and A. R. Jensenius, "Towards a coherent terminology and model of instrument description and design," in *Int. Conf. New Interf. for Musical Expr. (NIME'06)*, Paris, France, 2006, pp. 220–225.
- [11] S. Jordà, "Digital lutherie: Crafting musical computers for new musics' performance and improvisation," Ph.D. dissertation, Universitat Pompeu Fabra, 2005.
- [12] J. Rasmussen, *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. New York, NY, USA: Elsevier Science Inc., 1986.
- [13] B. Cariou, "The aXiO midi controller," in *Proc. Int. Comp. Music Conf. (ICMC'94)*, Århus, Denmark, 1994, pp. 163–166.
- [14] E. F. Clarke, *Generative Processes in Music*. Oxford: J. A. Sloboda (Ed), Clarendon Press, 1988, ch. Generative principles in music performance, pp. 1–26.
- [15] R. J. K. Jacob, L. E. Sibert, D. C. McFarlane, and M. P. Mullen Jr., "Integrality and separability of input devices," *ACM Trans. Human Computer Interaction*, vol. 1, no. 1, pp. 3–26, 1994.
- [16] J. Cage, *Silence: Lectures and Writings*. Middletown: Wesleyan University Press, 1961.
- [17] G. Wang and P. Cook, "On-the-fly programming: Using code as an expressive musical instrument," in *Int. Conf. New Interf. for Musical Expr. (NIME'04)*, Hamamatsu, Japan, 2004, pp. 138–143.
- [18] R. Fink, "The story of ORCH5, or, the classical ghost in the hip-hop machine," *Popular Music*, vol. 24, no. 3, pp. 339–356, 2005.
- [19] M. Mathews and B. Verplank, "Scanned synthesis," in *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, 2000, pp. 368–371.
- [20] E. Sinyor and M. M. Wanderley, "Gyrotyre: A dynamic hand-held computer-music controller based on a spinning wheel," in *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, 2005, pp. 42–45.

THE MODIFIED CHAMBERLIN AND ZÖLZER FILTER STRUCTURES

Duane K. Wise

Consultant

Boulder, Colorado, USA
dwise@wholegrain-ds.com

ABSTRACT

The Chamberlin and Zölzer filter structures run a risk, albeit small, of requiring a large value for one of their tuning coefficients, which may lead to performance issues. A simple modification leads to alternative structures that place an absolute bound on that coefficient while retaining the signal flow topology. The modified structures also affect the pole distribution. In the case of the Chamberlin structure, the changes upon modification can be interpreted as favorable.

1. BACKGROUND

1.1. The Chamberlin and Zölzer Filter Structures

Chamberlin, in [1], outlines a versatile filter structure based on embedded digital integrators. Zölzer, in [2], attributes this structure in a slightly modified form to a prior publication by N. G. Kingsbury. Also in [2], Zölzer makes an intuitive leap from the Chamberlin and Kingsbury works that results in a filter structure that bears his name.

Figure 1 presents the signal flow graph of the Chamberlin filter structure. The Kingsbury structure is virtually identical save that the input node is at a different location. In this paper, the filter structure will herein be called the Chamberlin structure.

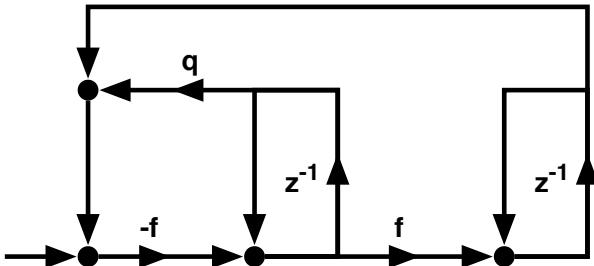


Figure 1: The Chamberlin filter structure.

The transfer function that the Chamberlin structure generates is

$$H_C(z) = \frac{N(z)}{1 - (2 - f^2 - f \cdot q) \cdot z^{-1} + (1 - f \cdot q) \cdot z^{-2}} \quad (1)$$

where $N(z)$ is a linear combination of node-specific transfer function numerators within the filter structure, which is beyond the scope of this paper.

Figure 2 presents the signal flow graph of the Zölzer filter structure. The structure depicted in the figure is slightly different from that presented in Zölzer's publication but results in an identical transfer function denominator.

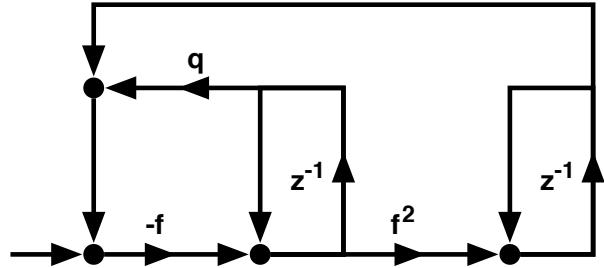


Figure 2: The Zölzer filter structure.

The transfer function that the Zölzer structure generates is

$$H_Z(z) = \frac{N(z)}{1 - (2 - f^3 - f \cdot q) \cdot z^{-1} + (1 - f \cdot q) \cdot z^{-2}} \quad (2)$$

where $N(z)$ is a linear combination of nodes as in (1).

1.2. Advantages

The Chamberlin filter structure can conveniently render several useful transfer functions simultaneously, such as highpass, low-pass, bandpass, and allpole [1], a trait that is also true of the Zölzer structure. In addition, the tuning coefficient f maps approximately to the tuning frequency parameter and the coefficient q maps approximately to the reciprocal of the Q parameter of the resulting poles.

Both filter structures offer a favorable distribution of poles when their coefficients are quantized. Figure 3 shows the pole distribution of the Chamberlin filter when f and q are quantized in linear steps. Figure 4 shows the pole distribution of the Zölzer filter with equivalent coefficient quantization. The pole distribution plots are more amenable to tuning with logarithmically scaled frequency and Q parameters [2] [3].

2. LIMITATIONS

In order to tune these filter structures to any stable transfer function, a formula to determine valid tuning ranges, based on reflection coefficient calculation, is given in [4]. Applying this formula to the Chamberlin structure, the following relations for f and q must apply for stability.

$$0 < f < 2 \quad \wedge \quad 0 < q < \frac{4 - f^2}{2f} \quad (3)$$

The relation (3) implies that, as coefficient f approaches zero, the value of coefficient q can become quite large, with no absolute

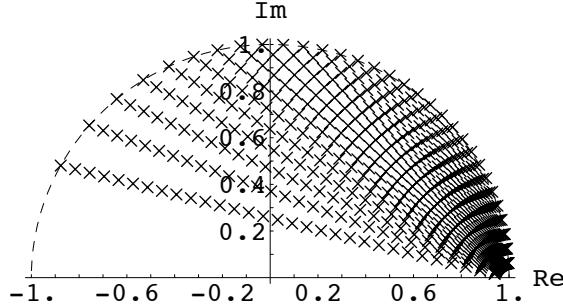


Figure 3: Linearly quantized pole distribution of the Chamberlin structure.

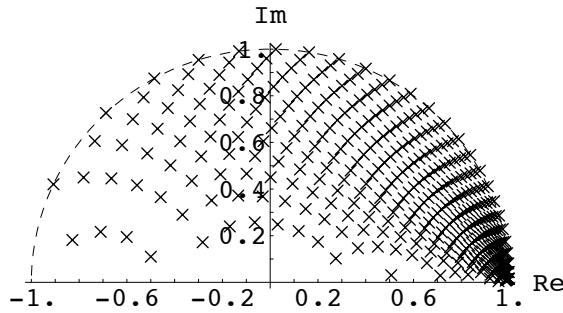


Figure 4: Linearly quantized pole distribution of the Zölzer structure.

bound. To illustrate, Figure 5 shows the value of q for a 10-octave tuning range ending at Nyquist $\times 20.48 \text{ kHz} / 24 \text{ kHz}$ and a range of Q from $1/32$ to 32 .

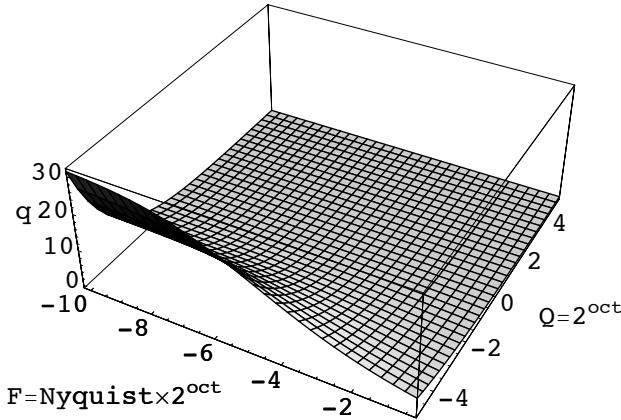


Figure 5: Value of coefficient q for the Chamberlin structure.

An inspection of Figure 3 and Figure 4 above shows that the Chamberlin structure has significantly lower pole density near Nyquist (-1 on the complex plane), whereas the Zölzer structure is more able to fill that void.

The Zölzer structure has the following relations for f and q to insure stability.

$$0 < f < 2^{2/3} \quad \wedge \quad 0 < q < \frac{4 - f^3}{2f} \quad (4)$$

Similar to (3), the relation (4) implies that the upper range of q is essentially boundless. However, Figure 6 shows that the Zölzer structure is less susceptible to q growth over the same tuning range.

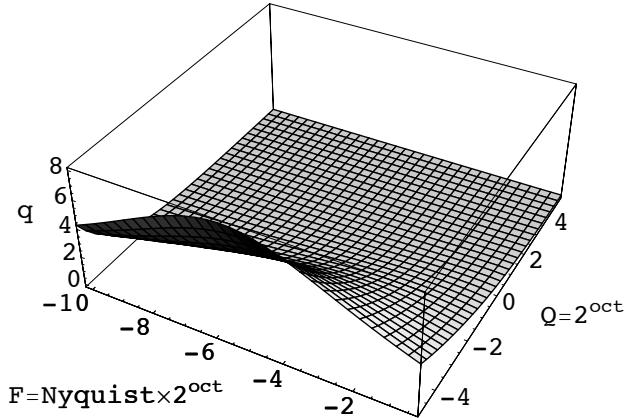


Figure 6: Value of coefficient q for the Zölzer structure.

While the wide range of the q parameter is clearly problematic for a fixed point implementation of these filter structures, a high level of q raises the signal level at the input node relative to the other summation nodes, which can raise the noise level at the input node regardless of whether fixed point or floating point math is utilized. Note that quantization noise generated at the summation nodes feeding the delays will eventually be multiplied by the q coefficient in the recursive network.

3. MODIFICATION

A simple modification to the above filter structures can help alleviate the above limitations. As illustrated in Figure 7, the $-f$ multiplier can be moved against the signal flow within the recursive loop to yield a $-f$ multiplier from the rightmost delay and a $-fq$ multiplier from the other delay. Figure 7 depicts what is herein called the Modified Chamberlin filter structure.

The transfer function that the Modified Chamberlin structure generates is

$$H_{MC}(z) = \frac{N(z)}{1 - (2 - f^2 - fq) \cdot z^{-1} + (1 - fq) \cdot z^{-2}} \quad (5)$$

where $N(z)$ is a linear combination of node-specific transfer function numerators within the filter structure. Applying the modification depicted in Figure 7 to the Zölzer filter structure results in what will be called the Modified Zölzer filter structure. The transfer function that it generates is

$$H_{MZ}(z) = \frac{N(z)}{1 - (2 - f^3 - fq) \cdot z^{-1} + (1 - fq) \cdot z^{-2}} \quad (6)$$

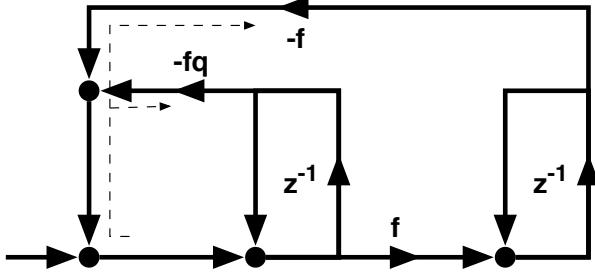


Figure 7: Multiplier modifications resulting in the Modified Chamberlin filter structure.

3.1. The Modified Chamberlin Filter Structure

The modified filter is still tuned with two coefficients, which are called f and fq in this paper, but the role of fq as opposed to q is slightly different. To illustrate this difference, Figure 8 shows the pole distribution of the Modified Chamberlin structure using the same quantization step size as that for Figure 3. Comparing the two figures indicates that the Modified Chamberlin structure retains the advantage of higher pole density towards DC while compensating for the Chamberlin structure's low pole density at Nyquist.

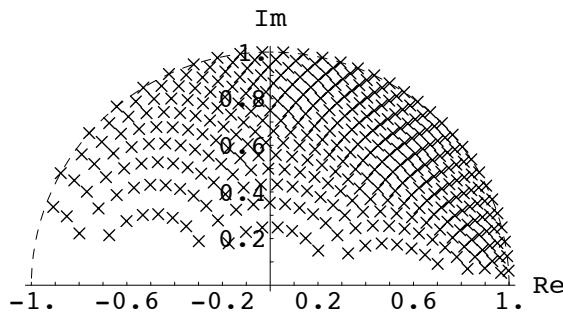


Figure 8: Linearly quantized pole distribution of the Modified Chamberlin structure.

The following relation must apply for the Modified Chamberlin structure to render a stable transfer function.

$$0 < f < 2 \quad \wedge \quad 0 < fq < \frac{4 - f^2}{2} \quad (7)$$

Plugging in the valid range of f into the range of fq shows that the value of fq cannot exceed 2 (also note that the z^{-2} terms of the modified transfer function denominators (5) and (6) cannot have their absolute values exceed unity in order for the denominator to remain in the stability triangle). Figure 9 shows the value of fq for the same tuning parameters as in Figure 5 and Figure 6. As compared to the Chamberlin structure in Figure 5, the range of the fq coefficient is considerably smaller than that of q . This aids in fixed point coefficient storage (note that both f and fq can fit in the range of a traditional unsigned fixed point data word) and in lowering the relative level at the input node, which can improve the overall noise performance.

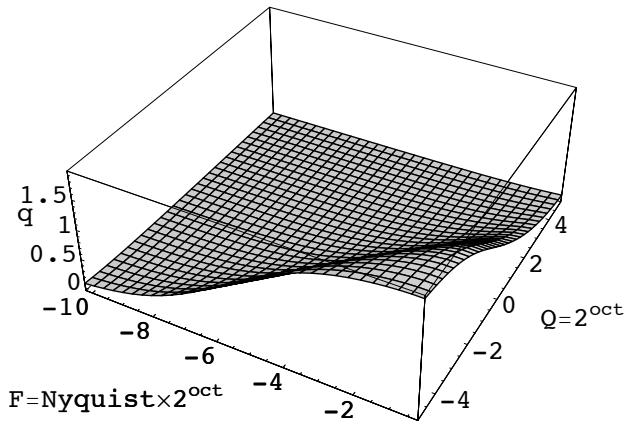


Figure 9: Value of coefficient fq for the Modified Chamberlin structure.

3.2. The Modified Zölzer Filter Structure

Applying the modification to the Zölzer filter structure yields less dramatic changes in performance. Figure 10 shows the pole distribution of the Modified Zölzer structure using the same quantization step size as that for Figure 4 and Figure 8. A comparison of Figure 10 with Figure 4 indicates that the two distributions are similar with the Modified Zölzer distribution having lower overall density than the Zölzer structure. Comparing Figure 10 with Figure 8 can make a case that the Modified Chamberlin structure outperforms the Modified Zölzer structure in pole distribution under coefficient quantization, which cannot be made when comparing the nonmodified versions of the two structures.

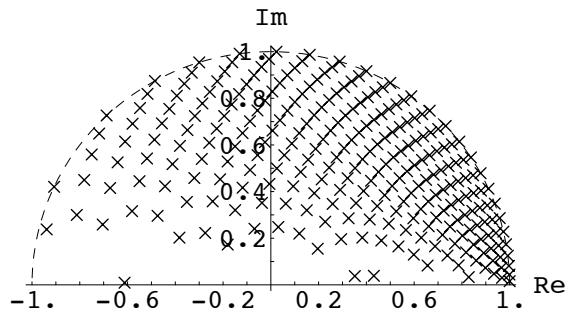


Figure 10: Linearly quantized pole distribution of the Modified Zölzer structure.

The following relation must apply for the Modified Zölzer structure to render a stable transfer function.

$$0 < f < 2^{2/3} \quad \wedge \quad 0 < fq < \frac{4 - f^3}{2} \quad (8)$$

The value of the fq tuning coefficient for the Modified Zölzer structure is identical to the fq coefficient for the Modified Chamberlin structure, given in Figure 9 (note that the z^{-2} terms of the modified transfer function denominators (5) and (6) are equivalent

and depend solely on fq). As the nominal range of q in the Zölzer structure is of lower magnitude than that of its Chamberlin counterpart, the noise improvement of the Modified Zölzer structure over the Zölzer structure is not as significant.

4. CONCLUSIONS

This paper offers for consideration a simple modification to the Chamberlin and Zölzer filter structures. This modification is offered not as a replacement to the two structures but as an alternative design that may aid in issues that involve a large value of the coefficient q . The q coefficient in the established structures has no absolute bound, though in most tuning circumstances the coefficient value is not large in magnitude. In the case where the value of q is large, performance issues may arise regardless of whether fixed point or floating point math is employed in the filter implementation.

The Modified Chamberlin and Modified Zölzer filter structures replace the coefficient q with a coefficient called fq , which represents the product of f and q in the established structures. The absolute upper bound of fq is 2 in all stable tuning configurations. The pole distribution of the modified structures is different from that of their canonical counterparts. The pole distribution of the Modified Chamberlin structure can compare favorably to that of the Chamberlin structure. It is unlikely that the pole distribution of the Modified Zölzer structure can be considered preferable.

The modified filters do not affect the signal flow topology of the structures, which retains most, if not all, of the advantages of the established structures outlined in [1] and [2].

5. ACKNOWLEDGEMENTS

The author wishes to thank Mic Chorn for his assistance in the preparation of this paper, and to the anonymous reviewers for their invaluable time, attention, and feedback. The filter analysis is performed with Mathematica version 5.2 (Wolfram Research, Inc.; 2005).

6. REFERENCES

- [1] H. Chamberlin, *Musical Applications of Microprocessors, Second Edition*. Hayden Books, 1985.
- [2] U. Zölzer, *Digital Audio Signal Processing*. Chichester, UK: J. Wiley & Sons, 1997.
- [3] D. Wise, "A survey of biquad filter structures for application to digital parametric equalization," 1998, *Presented at the 105th AES convention*, preprint #4820.
- [4] J. Markel and A. Gray Jr., *Linear Prediction of Speech*. New York: Springer-Verlag Berlin Heidelberg, 1976.

SMOOTH AND SAFE PARAMETER INTERPOLATION OF BIQUADRATIC FILTERS IN AUDIO APPLICATIONS

Victor Kalinichenko

ASK Industries GmbH
Hauptstrasse 73, D-94559
Niederwinkling, Germany
KalinichenkoV@askgroup.de
Private: homekvn@yandex.ru

ABSTRACT

Parameter interpolation (PI) of biquadratic IIR filters (biquads) is widely used in audio applications to avoid very undesirable audible effects like plops, clicks etc. It is shown that although the simplest linear PI of numerator and denominator coefficients of the transfer function of a biquad guarantees the stability it, however, may sometimes lead to other audible side effects caused, for example, by the possible growth of the magnitude response at some frequencies in some intermediate states of PI procedure. To avoid audible transients and to make the PI easily portable into digital signal processor (DSP) a special technique is applied called here “sliding edges”. The detailed description of this technique and their comparison is the goal of the present paper.

1. INTRODUCTION

Biquadratic IIR filters (biquads) are used in many audio applications; for example, for individual equalization of audio channels, or in some measurement algorithms to extract signal components within a certain bandwidth, etc. Sometimes it is necessary to change the parameters of one or several biquads together during a certain time with no undesirable audible transient effects that may occur in case of simple switching. The smooth PI is needed, for example, in case of changing the equalization filter-set in the cabin of a car from driver optimized setting to front seats.

A perfect PI must be smooth, safe and rather easy from the viewpoint of computations. The smoothness of PI may be well described at consideration of the behavior of magnitude to frequency response (MR) of the biquad during the PI procedure. It is naturally to require that MR had no sudden changes within the duration of PI. The term “safe” in this context means that during PI no instability or being very close to the stability boundaries may occur. The last demand is also of great importance as some audio systems may involve 100 and more biquads, 20-30 percents of which are to be interpolated in real-time. PI algorithm during rather short working time (usually 2-3 sec) requires additional computation power for all those filters. Using the complicated PI algorithm may be critical for the DSP installed in the audio system.

Different approaches of PI are described in [1, 2, 3, 4, 5, 6, 7, 8, 9]. The general classification of PI techniques is done in [2]. According to this classification the following types of PI are known: a) *Cross-fading method*; b) *Gradual variation of coefficients* [3]-[5]; c) *Intermediate coefficient matrix* [6]; d) *Input switching method* [7] and e) *Updating of the state vector* [8], [9]. Theoretically, all above methods are DSP-implementable. However, all of them have the restricted range of use: methods a) and d) require duplicating the processing power during PI; method b)

may cause audible transients especially at harmonic signals; methods c) and e) imply the representation of filters in state-space form; method e) requires a lot of computations. We emphasize on the design of easy-to-implement approach that would operate for the case, when the filter, whose parameters are to be changed, is represented as a cascade of biquads. An attention to DSP-portability will be given.

2. PROBLEM STATEMENT

2.1. Notations and assumptions

Biquad may be described by its transfer function (TF) in z-domain:

$$W(z) = k \cdot \frac{1 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (1)$$

where a_i, b_i — coefficients being normally constants, k — gain, $z \in \mathbb{C}$. Using the form (1) of TF is usual in DSP applications. Let $W_{\text{src}}(z)$ and $W_{\text{tar}}(z)$ are the TFs of initial and final state of the biquad whose appearance corresponds to (1). Let T_{PI} is the time of *duration of PI*. In case the duration of PI is long enough so that it corresponds to rather large number of input signal samples PI may be done with the lower discreteness than the discreteness of the input signal, *i.e.* updating of coefficients of (1) may be done not every sample but every 2, 3, 4, ... sample. We denote the number of samples after which the next update is done as L_{block} . Then $T_{\text{PI}} = N_{\text{PS}} \cdot L_{\text{block}} \cdot T_s$, where T_s is the sampling period; the integer number N_{PI} is called the *length of PI*, *i.e.* the total number of updates to complete PI. The notations like $W_{[m]}(z)$, $a_{1[m]}$, etc $m = 0, 1, 2, \dots, N_{\text{PI}}$ provided with the index within square brackets relates to the stage of PI, *i.e.* to the number of update. Note that $W_{[0]} = W_{\text{src}}(z)$, $W_{[N_{\text{PI}}]}(z) = W_{\text{tar}}(z)$.

2.2. Criteria for the parameter interpolation algorithm

The practical criteria for PI can be described in the following way:

1. No instability must occur while PI is done. In other words, for every m , $W_{[m]}(z)$ must not have any instable pole.
2. No remarkable audible side effects (like clicks, plops, ringing etc.) must occur during PI.
3. The complexity of PI algorithms must not be very high due to PI is done in real-time.

To illustrate the importance of the above criteria consider the simplest linear PI results drawn in Figure 2a. PI is done for $N_{\text{PI}} = 10$. It is clear that there are substantial drawbacks as PI procedure is not smooth because of great difference of MRs at two neighbor PI stages $m = 0$ and $m = 1$.

3. PROBLEM SOLUTION

The most important requirement while PI is in progress is to ensure the stability at all stages. As the stability condition does not depend on the numerator of (1) it is reasonable to make PI for numerator and denominator of the biquad's TF separately.

3.1. Parameter interpolation of the denominator

3.1.1. Linear PI

Fortunately, according to the investigations made in [10], the stability area of biquad is defined by the following condition:

$$|a_1| < a_2 + 1, \quad \text{and} \quad |a_2| < 1. \quad (2)$$

It shows that all boundaries of the stability region are straight lines and that in theory the linear PI

$$a_{i[m]} = a_{i[m-1]} + \Delta a_i, \quad \Delta a_i = (a_{i \text{ dst}} - a_{i \text{ src}}) / N_{\text{PI}}, \quad (3)$$

(where $i = 1, 2$) will result in the stability at every m if and only if $W_{\text{src}}(z)$ and $W_{\text{dst}}(z)$ are stable. In practice the stability may be lost in case of inaccurate computations due to accumulated error at the summation in (3). To avoid this, the non-recurrent formula must be used:

$$a_{i[m]} = a_{i \text{ src}} + m \cdot \Delta a_i, \quad \Delta a_i = (a_{i \text{ dst}} - a_{i \text{ src}}) / N_{\text{PI}} \quad (4)$$

It is easy to show that the summation error which is most important due to different order of the values a_i and Δa_i will not be accumulated in case (4). Note also that linear PI is applied to each coefficient independently and it requires minimal computations. Additional tricks must be made to ensure the stability at fixed-point computations.

3.1.2. Pole PI

Although the linear PI of denominator is very simple and provides the system stability it may result in undesirable behavior of poles which are responsible also for resonances in MR. The resonance may cause some audible side effect like “ringing” and in practice may even lead to “clipping” due to extremely high output values. Experiments have shown that in most practical cases it does not take place. However, if the source or the destination filters has the Q -factor more than 5–6, then in some cases high resonances may occur. If such cases a more sophisticated PI law should be applied.

The idea of the PI of poles is in step-by-step shifting of the poles and then recomputing new values of the coefficients a_i at each stage. Let the poles (the roots of denominator) of (1) are z_1 and z_2 . The association between a_i and z_i is expressed in the following formulae:

$$a_1 = -(z_1 + z_2), \quad a_2 = z_1 z_2, \quad (5)$$

$$z_{1,2} = 0.5 \cdot \left(-a \pm \sqrt{a_1^2 - 4a_2} \right). \quad (6)$$

Basing on (6) it is possible to calculate $z_{i \text{ src}}$ and $z_{i \text{ dst}}$. Depending on $a_{i \text{ src}}$ and $a_{i \text{ dst}}$ three cases of PI trajectories are possible:

1. Both couples of poles $z_{i \text{ src}}$ and $z_{i \text{ dst}}$ are real numbers;
2. Both couples of poles are complex¹ values;
3. One pole is complex and another one is real.

¹In principle, $\mathbb{R} \subset \mathbb{C}$. However, in this context saying that a value x is complex, we imply that $x \in \mathbb{C} \setminus \mathbb{R}$.

In case of complex poles z_1 and z_2 they must be conjugant, i. e. $z_1 = z_2^*$ $\iff \Re(z_1) = \Re(z_2), \Im(z_1) = -\Im(z_2)$. Depending on the type of poles there may be different but linear PI trajectories; see Figure 1, a-c. In case 3 the trajectory of poles consists of two straight-line pieces. The shift of poles at each straight piece is done according to linear law:

$$z_{i[m]} = z_{i[0]} + m \cdot \Delta z_i. \quad (7)$$

With a small adaptation the formula (7) is valid even if PI trajectories are of type 3. Note that formulae (5) and (7) are also very simple from the viewpoint of computation and thus satisfies the criterion 3.

3.2. Parameter interpolation of the numerator. “Sliding edges” algorithm

The algorithm for the readjustment of the numerator coefficients of TF is based on the idea which can be called “sliding edges”. This term relates to two specific points of frequency response: namely, $\omega_1 = 0$ and $\omega_2 = \pi$ (*i.e.* Nyquist frequency). Let the source biquad has the edge points equal to

$$\begin{aligned} g_{1[0]} &= A_{[0]}(\omega_1) = A_{\text{src}}(\omega_1), \\ g_{2[0]} &= A_{[0]}(\omega_2) = A_{\text{src}}(\omega_2), \end{aligned} \quad (8)$$

where

$$A_{[m]}(\omega) = |W_{[m]}(z)|_{z=\exp(j\omega)}. \quad (9)$$

Thus, g_1 and g_2 are non-negative. Let the destination biquad has the edge points

$$\begin{aligned} g_{1[N_{\text{PI}}]} &= A_{[N_{\text{PI}}]}(\omega_1) = A_{\text{dst}}(\omega_1), \\ g_{2[N_{\text{PI}}]} &= A_{[N_{\text{PI}}]}(\omega_2) = A_{\text{dst}}(\omega_2). \end{aligned} \quad (10)$$

Then, moving the edges:

$$g_{1[m]} := A_{[m]}(\omega_1), \quad g_{2[m]} := A_{[m]}(\omega_2) \quad (11)$$

use of some smooth law (for example, linear law) will be assumed to result in the appropriate intermediate MRs through all transitions. The above idea exploits the assumption based on the analysis of experimental data that the biggest “jump” between two neighbor stages of MR may occur namely because of big jump at the edge values. Then, the natural way to “impel” MRs not to “jump” very strongly is to move their edges along vertical axis $\omega_1 = 0$ and $\omega_2 = \pi$ using some smooth law. As hinted before the linear law

$$\begin{aligned} g_{i[m]} &= g_{i[m-1]} + \Delta g_i = g_{i \text{ src}} + m \cdot \Delta g_i, \\ \Delta g_i &= (g_{i \text{ dst}} - g_{i \text{ src}}) / N_{\text{PI}}, \quad i = \{1, 2\} \end{aligned} \quad (12)$$

is chosen. Let us establish the correspondence between edge values g_i and the coefficients a_i , b_i and k . Note that as it follows from (9) the value of argument at the edges $\omega = 0$ and $\omega = \pi$ corresponds to $z = 1$ and $z = -1$ respectively. Taking into account this fact and also (11) we may conclude that:

$$|W_{[m]}(1)| = g_{1[m]}, \quad |W_{[m]}(-1)| = g_{2[m]}. \quad (13)$$

Unwrapping (13) and taking into account (2) will result in:

$$\begin{aligned} \frac{|1+b_{1[m]}+b_{2[m]}|}{(1+a_{1[m]}+a_{2[m]})} \cdot |k_{[m]}| &= g_{1[m]}, \\ \frac{|1-b_{1[m]}+b_{2[m]}|}{(1-a_{1[m]}+a_{2[m]})} \cdot |k_{[m]}| &= g_{2[m]}, \end{aligned} \quad (14)$$

Assuming that for most practical applications k and the values in the module brackets are non-negative, we can express the numerator values:

$$\begin{aligned} b_1[m] &= \frac{(g_1[m] - g_2[m])(1 + a_2[m]) + (g_1[m] + g_2[m])a_1[m]}{2k[m]}, \\ b_2[m] &= \frac{(g_1[m] + g_2[m])(1 + a_2[m]) + (g_1[m] - g_2[m])a_1[m]}{2k[m] - 1}. \end{aligned} \quad (15)$$

Thus, applying the linear or pole PI for denominator, linear PI for $k[m]$ and the formula (12), one can obtain from (15) the numerator coefficients b_i automatically.

3.3. Aspects of parameter interpolation for cascaded biquads

If it is necessary to do PI in a cascade of biquads all the above formulae are valid. The only difficulty may occur due to the circumstance that the gains k of each cascade are normally not stored in the memory due to storing only the entire overall gain being a multiplicative product of gains of all cascades. Unfortunately direct PI of overall gain may generally lead to extremely big or small values in intermediate MRs. It is easy to see from the following fact:

$$k_1[m] \times k_2[m] \times \dots \times k_L[m] \neq k_{\text{overall}}[m] \quad (16)$$

where $k_l[m] = k_{l \text{ src}} + m \cdot \Delta k_l$ is the gain of cascade number $l = \{1, 2, \dots, L\}$ at the PI stage number m ; L is the number of cascades; overall $k_{\text{overall}}[m] = k_{\text{overall src}} + m \cdot \Delta k_{\text{overall}}$ is the overall gain at the PI stage number m . Thus, two variants are possible: either to store and then interpolate the gain of each cascade individually, or to use the multiplicative PI for overall gain, i.e.

$$k_{\text{overall}}[m] = k_{\text{overall src}} \cdot \delta^m, \quad \delta = (k_{\text{overall dst}}/k_{\text{overall src}})^{1/m}. \quad (17)$$

The inequality (16) in the last case would turn to the equality. This, however, may cause different problems if $k_{\text{overall src}} \leq 0$.

3.4. Capability to start new parameter interpolation while previous interpolation is still in progress

Practical usage of filters with PI may require the ability to start new PI while the previous PI is still in progress. This feature must be implemented in case when it is not 100% priority defined that the next service command used for starting PI will not interrupt the previous one. To start new PI it is necessary to store current values of $a_i[m]$, $b_i[m]$, $k[m]$ and to consider them as source values. All the above mentioned PI methods allow us to do this procedure.

4. RESULTS OF MODELLING

To show the quality of each PI algorithm the modeling is done at the number of PI stages $N_{\text{PI}} = 10$. For the synthesis of biquads the formulae from [10] are used. The computation of all types of biquads is based on four parameters: a) filter type: LPF1, LPF2, HPF1, HPF2, Notch, Peak, BP, APF1, APF2; b) f_0 — cut-off frequency; c) L — pass band level and d) Q — Q -factor. First two letters in the filter type abbreviation denote: LP — low-pass, HP — high-pass, BP — band-pass, AP — all-pass; the third letter “F” denotes “filter”; the number denotes the order of filter. Three variants of PI are compared: 1) linear denominator + linear numerator (LDLN) PI; 2) linear denominator + sliding-edge numerator (LDSEN) PI; 3) pole denominator + sliding-edge numerator (PDSN) PI. The results of modeling are represented in Figures 2-3, a-c.

5. ANALYSIS OF THE RESULTS AND CONCLUSIONS

Let us analyze the suitability of the modelling results to the criteria 1-3 introduced in the section 2.2. LDLN PI satisfies fully the criteria 1 and 3. However, in some cases the criterion 2 is failed. LDSEN PI also satisfies the criteria 1 and 3; a set of potential cases leading to the failure of the criterion 2 is reduced in comparison with LDLN. PDSN PI satisfy the criteria 1-3 (the criterion may probably fail only in some “exotic” cases). Its drawback is the relative complexity of program implementation due to separate consideration of three types of trajectories described in the subsection 3.1.2. But, it does not mean that PDSN will require a lot of processing power in real-time mode: only the program code will grow.

Although the proposed LDSEN and PDSN PI techniques are not the ideal solutions they are the good alternatives to the intuitive linear PI LDLN. Anyway the solution on what type of PI to use follows from the prior information about possible type of the filter to be interpolated.

6. REFERENCES

- [1] R. J. Clark, “Investigation into digital audio equalizer systems and the effects of arithmetic and transform errors on performance,” Ph.D. dissertation, Univ. of Plymouth, 2001, [Online] <http://www.tech.plym.ac.uk/spmc/pdf/audio/RobClarkPhD.pdf>.
- [2] V. Välimäki, “Discrete-time modeling of acoustic tubes using fractional delay filters,” Ph.D. dissertation, Helsinki University of Technology, 1995, [Online] <http://www.acoustics.hut.fi/~vpv/publications/vesan.vaitos>.
- [3] J. N. Mourjopoulos, E. D. Kyriakis-Bitzaros, and C. E. Goutis, “Theory and real-time implementation of time-varying digital audio filters,” *J. Audio Eng. Soc.*, vol. 38, no. 7/8, pp. 523–536, Jul./Aug. 1990.
- [4] U. Zölzer, B. Redmer, and J. Bucholtz, “Strategies for switching digital audio filters,” in *95th Conv. Audio Eng. Soc.*, New York, USA, Oct. 1993, preprint 3714.
- [5] S. S. Nikolaidis, J. N. Mourjopoulos, and C. E. Goutis, “A dedicated processor for time-varying digital audio filters,” *IEEE Trans. Circuits and Systems—II: Analog and Digital Sig. Proc.*, vol. 40, no. 7, pp. 452–455, July 1993.
- [6] R. Rabenstein, “Minimization of transient signals in recursive time-varying digital filters,” *Circuits, Systems, and Sig. Proc.*, vol. 7, no. 3, pp. 345–359, 1988.
- [7] W. Verhelst and P. Nilens, “A modified-superposition speech synthesizer and its applications,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'86)*, Tokyo, Japan, vol. 3, 1986, pp. 2007–2010.
- [8] 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, Dec. 1988.
- [9] V. Välimäki and T. I. Laakso, “Suppression of transients in time-varying recursive filters for audio signals,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'98)*, Seattle, USA, vol. 6, May 1998, pp. 3569–3572, [Online] <http://www.acoustics.hut.fi/~vpv/publications/icassp98-trel.pdf>.
- [10] K.-D. Kammerer and K. Kroschel, *Digitale Signalverarbeitung. Filterung und Spektralanalyse mit MATLAB-Übungen*, 5th ed. B. G. Teubner, 2002.

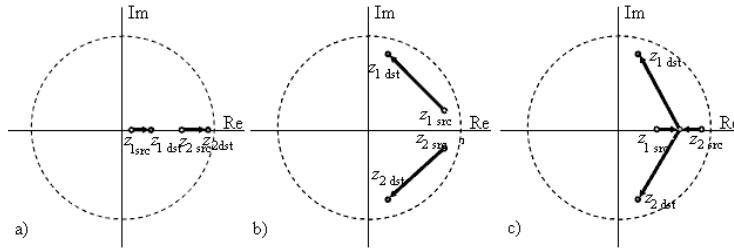
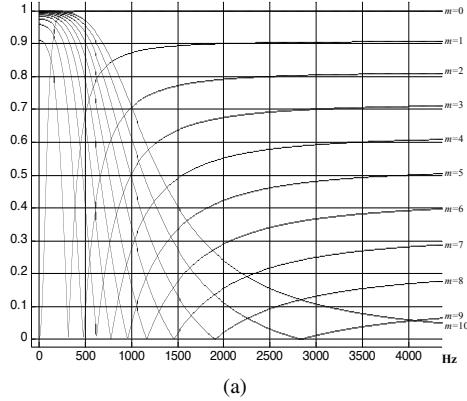
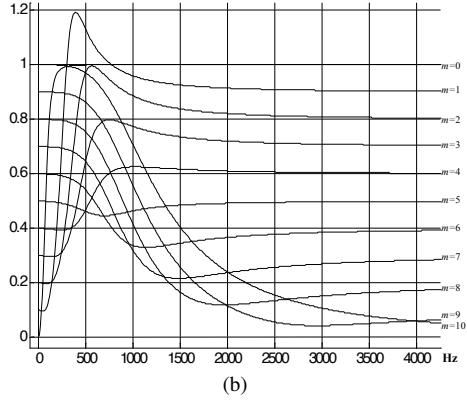


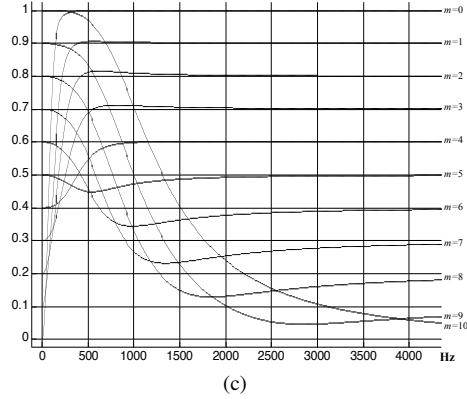
Figure 1: Three variants of pole moving for source and destination biquad.



(a)

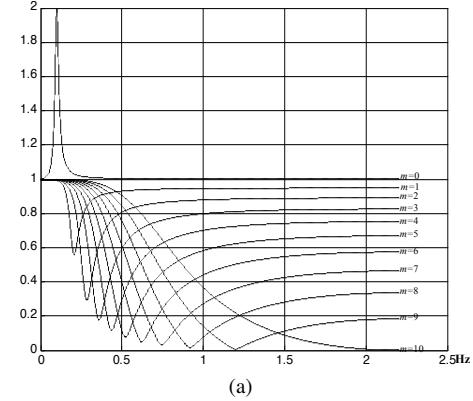


(b)

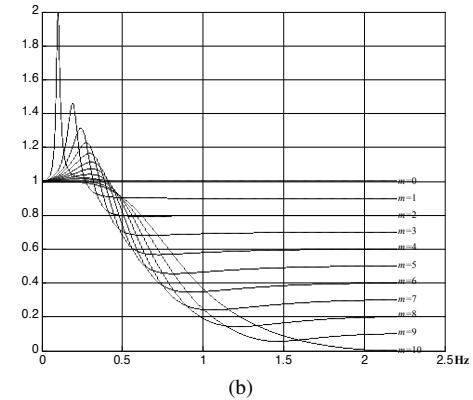


(c)

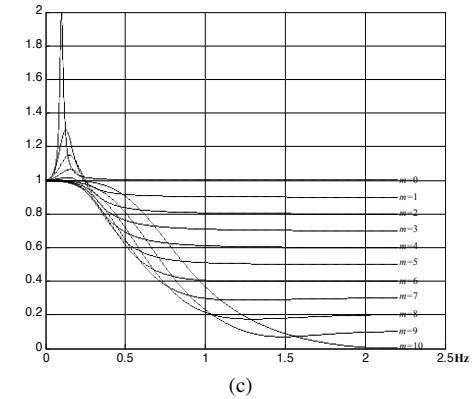
Figure 2: PI examples of a filter with the parameters: source: type: HPF2, $f_0 = 100$ Hz, $L = 0$ dB, $Q = 0.707$; destination: type: LPF2, $f_0 = 1$ kHz, $L = 0$ dB, $Q = 0.707$; a) LDLN, b) LDSEN, c) PDSEN.



(a)



(b)



(c)

Figure 3: PI examples of a filter with the parameters: source: type: PEAK, $f_0 = 1$ kHz, $L = 6$ dB, $Q = 3.0$; destination: type: LPF2, $f_0 = 7$ kHz, $L = 0$ dB, $Q = 0.707$; a) LDLN, b) LDSEN, c) PDSEN.

IDENTIFYING AND ANALYZING RELEVANT CHARACTERISTICS OF DYNAMIC RANGE COMPRESSION

Andrés Cabrera

Departamento de Música

Pontificia Universidad Javeriana, Bogotá, Colombia

andres@geminiflux.com

ABSTRACT

This document presents a method for measuring, identifying and analyzing the fundamental characteristics of dynamic range compressors. Emphasis is placed on making the analysis and testing method as automated as possible. Several characteristics of dynamic range compression have been identified by comparing the behavior of six different plug-in compressors.

1. INTRODUCTION

Dynamic range compression has always been an important aspect of audio processing both for aesthetic and technical reasons. Compression is basically an algorithm to derive a gain factor from an envelope follower, and there are many hardware devices that can achieve dynamic compression. With the advent of DAWs, many software plug-ins have been developed for this purpose. Most compressors have four parameters in common: *Threshold*, *Ratio*, *Attack* and *Decay time*. Others may offer control of *Knee* and *Auto Release* control, and others offer different modes of operation (*Opto* or *Electro* in the case of the Waves RCompressor [1], *Vintage* or *Normal* in the case of the Sonitus Compressor [2] and selection between *Peak* and *RMS* tracking in the case of the SC4 [3]). Some have fixed Attack and Decay times, and some have a fixed threshold but have input and output gain control to adjust compression amount. Other parameters which exist in compressors but which act independently from the actual compression process like makeup gain, and side chain EQ will not be dealt with here.

The *Threshold* of a compressor is the amplitude above which gain reduction begins to be applied. Everything below this threshold should pass unaffected, but as will be seen, the *Knee* and the way the amplitude of a signal is tracked by the compressor affects dramatically the exact location of the threshold.

The *Ratio* of a compressor determines how much gain reduction should be applied to signals crossing the threshold. It is a ratio between the amount of original signal above the threshold and the amount of signal above the threshold after gain reduction.

The *Attack* and *Decay times* determine how long it takes for the compressor's gain reduction to fully kick in after a signal has crossed the *threshold*, and how long it takes for the compressor to stop reducing gain after a signal is once again below the *threshold*. These stages of attack and decay are referred to here as gain reduction and gain recovery.

A compressor's *Knee* affects the behavior of gain reduction around the threshold. A soft knee or knee with high radius results in compression being introduced mildly for amplitudes below the threshold. As the signal level goes beyond the threshold, the ratio of

compression is increased until the target ratio is reached somewhere above the threshold. A hard knee introduces compression only on the threshold and only at the set ratio.

Even though all compressors process the signal in similar manner, the perceived and technical quality of a compressor's output vary dramatically.

This article will present an analysis method and some preliminary results for 6 different compressors: SC1 [4], SC4 [3], Tap dynamics [5], mda dynamics [6], Sonitus fx:compressor [2] and Waves Renaissance Compressor [1] attempting to identify the relevant characteristics of a compressor which determine the quality of its output. These compressors span most of the spectrum available in plug-in compression, including high-end commercial, freeware and open source plug-ins.

Although dynamic range compression of audio signals is almost as old as audio recording, no references could be found documenting similar testing or analysis methods, although certainly some must have been developed by the private industry for the purpose of emulation and design.

2. ANALYSIS METHOD

The principal effect of compression on an audio signal is that of gain reduction. Previous efforts to measure and model dynamic range compressors [7] have focused mainly on the processor's frequency response and the total gain reduction applied after a steady gain reduction has been reached. The primary goal of this particular project was to model vintage analog compressors whose frequency response at different gain reductions is particularly characteristic of the processor to be modeled. However, relevant characteristics of a compressor like exact treatment of transients and peaks, the actual gain curves, the effect of the input signal's frequency spectrum on the gain curves and the effects of the time variables are not considered. Other research has produced companding models designed as hardware replacement which though suitable for technical applications, do not seek to emulate desirable auditory characteristics of particular compressors [8]. There exist commercial plug-ins that model actual hardware, like the URS plugins [9], but these models are as closed and unknown as their analog originals and there seems to exist no documentation about the technical processes involved. Additionally the method presented here seeks to automate as much of the procedure as possible to optimize evaluation.

The analysis method proposed here seeks to analyze a compressor's characteristics in detail to cover all aspects of its operation. The method makes the following assumptions:

1. The gain curve and reduction depend only on the amount

- of signal above the threshold, and not on the actual threshold level. This means that a signal going 2dB above the threshold will be compressed always the same amount, independently of the actual threshold level.
2. The processor only affects the amplitude of a signal. All phase characteristics of a signal are retained.
 3. A compressor can be fully described by its effect on single frequencies.

The first assumption will probably hold true for digital compressors since making the behavior of the compressor be dependent of the threshold level would increase the algorithm's complexity without much gain other than irregular behavior of the process at different levels. For analog hardware compressors this assumption may not be so correct since anomalies in analog circuitry might lead to irregular behavior. However the analysis method proposed here can be repeated for different threshold levels to build one complex algorithm or several different models.

The second assumption, as the first one, probably holds true for most digital compressors. The method does a crude check for zero crossing incoherences (which would occur if the process introduced phase irregularities), and none of the results, even in the case of white noise reported any discrepancies.

The validity of the last assumption will have to be evaluated once the method has yielded further results. If it is seen this assumption leads to inconsistent results, the composition of the test sample (see section 1) will have to be adjusted to include more complex spectra than pure sine waves.

2.1. Characteristics of a compressor to be measured

This method proposes analyzing the following characteristics of a compressor:

1. The final gain reduction applied to a steady signal (constant sine wave or noise) after a constant gain reduction is reached. This gain reduction shall be measured for values well above and below the threshold, with particular detail around the threshold.
2. The effect of frequency on gain reduction applied, i.e. measuring differences in gain reduction for steady signals with equal peaks but different spectral content.
3. The effect of the signal's frequency on the shape of the gain reduction/recover curve.
4. The time required to achieve a steady gain reduction. This time will be compared to the actual attack time setting, and will be measured with different attack time values.
5. The shape, latency (see section 2.4) and duration of the gain reduction curve. This will reveal the exact treatment of peaks and transients in the compressor. This will provide additional clues as to the process the compressor uses to track amplitude.
6. The time required by the compressor to remove gain reduction when a signal's level decreases. This will be compared to the actual release time and will be measured with different release values.
7. The shape and time of the compressor's gain recover curve, i.e. the section where gain is restored after a signal goes below or comes down closer to the threshold.

8. The effect of gain reduction on the spectrum of broadband noise. This will reveal any coloring (EQ) applied by the compressor when reducing gain.
9. The shape of the compressor's knee, i.e. the way the gain reduction ratio varies around the threshold (going from 1:1 to the final ratio).
10. The characteristics and presence of non-linear distortion applied by the compressor and the effect of a signal's frequency on this distortion.

It is estimated that these characteristics accurately describe the behavior of a compressor, and will provide all information necessary for later modeling and analysis.

2.2. Test sample

The first step of the method consists in passing a test sample (in this test the sample rate used was 44100 kHz with 24-bit integer resolution) through the compressor. The compressor is used as a plug-in in any suitable host, and the resulting audio is exported either at 24-bit integer (for the Windows plug-ins) or 32-bit floating-point (on Linux, since the Linux host used permits this).

The current test sample (which is certain to continue evolving and was prepared using a Python [10] script to generate a Csound [11] score) contains the following parts:

Parts	Amp. (dB _{FS})	Freq. (Hz)
0-7	0	50, 200, 800, 1500, 3000, 8000, WN, DC
8-151	-5 to -30	50, 200, 800, 1500, 3000, 8000, WN, DC
152-226	-5 ramp to -10 to -35	200, 1500, 8000, WN, DC
227-301	-10 to -30	200, 1500, 8000, WN, DC
302-376	-5 to -28	200, 1500, 8000, WN, DC
377-451	jump to -30	WN, DC
452-471	-5 to -28	200, 1500, 8000, WN, DC
472-491	jump to -40	WN, DC
492-511	0, -10, -15, -20, -25	Square wave @ 50, 200, 1500, 8000
		Saw wave @ 50, 200, 1500, 8000
		Triangle wave @ 50, 200, 1500, 8000

Table 1: *The parts of the test sample.*

Each of the parts in the test sample is a short sound followed by silence. The duration of the sound and the silence are long enough for the compressor to apply full gain reduction and achieve a steady state, and then to recover completely from gain reduction. The parts are referenced by a number which will later identify the segment.

The test sample was designed to be used on a compressor with the threshold set to -20dBFS, and has the best resolution around that amplitude. It consists of the following parts:

- Parts 0 to 151 are designed to measure the compression gain reduction curve and the final gain applied to signals at different frequencies (Characteristics 1,2,3,4 and 5). Many

amplitudes and frequencies are used to judge both the effect of level and frequency on gain reduction. White Noise and a DC offset are also passed to obtain additional information.

- Parts 152 to 301 (Characteristic 10) are designed to measure the knee shape both for gain reduction and gain recovering. Perhaps the latter will be seen redundant after further study if is found to be consistently similar.
- Parts 302 to 451 will reveal the compressor's gain recover curve (Characteristics 6,7 and 8) by making the compressor apply gain reduction and then suddenly lowering the volume of the signal to evaluate the gain recovery stage. These parts and the previous ones could be useful in determining the release characteristics for compressors with options like Auto Release Control.
- Parts 452 to 511 are designed to test the compressor for non-linear distortion (Characteristic 11). These known -waves exhibit well known characteristics than can easily reveal any non-linear distortion applied. Non-linear distortion may be part of a compressor's process as shown in [12]
- All parts containing white noise will be used to evaluate the compressor's frequency response (Characteristic 9).

A single sample (of 20+ minutes length) was produced instead of many smaller ones, because it makes the processing through the compressor a one pass process, and it facilitates the redesign and modification of the test sample. This single sample will produce a new processed sample which will provide information on the behavior of the compressor at a particular setting. Several processed samples must be obtained at different settings to fully evaluate a compressor's behavior. Additionally the test sample has a pulse on its first sample to test and later compensate for any delay introduced by the compressor or the host.

For the current tests, all compressors were evaluated at their different modes when available (Opto-Electro; RMS-Peak; Vintage-Normal) and at all permutations of the following:

- Attack and release set equally to: 20ms, 40ms, 80ms and 400ms
- Ratios of 2:1 and 4:1

If a compressor had a knee setting, relevant settings were chosen, and all the measurements were done for them as well.

This is one the most time consuming processes of this method, since currently hosts cannot be scripted and ordered to change a plug-in's parameters between passes (on Windows), so this and the generation of the processed file must be done manually. This is not very problematic for simpler processes like the SC1 which might take half a minute to process (on a P4 1.6GHz machine), but it is definitely time consuming for CPU hungry plug-ins like the Sonitus which takes around 15 minutes to go through the sample on the same machine.

It is important to maintain consistent naming conventions throughout, especially since this analysis yields large numbers of files. In this case, the test file names included version number and threshold target, and the processed files included in their names information about the time and ratio settings as well as the name of the plug-in, the version and threshold target of the original test file and any other important settings used.

Notice that the amplitudes are calculated in absolute peak (dBFS) values. Since compressors usually track amplitude using some

type of RMS or integration technique, discrepancies are expected in the gain curve results according to frequency. It may be necessary in future versions of the test sample to provide equal RMS amplitude values instead of equal peak values. However since the tracking method varies so dramatically between compressor implementations, using the absolute peak values is a valid compromise, and provides useful data.

2.3. Comparing the samples

To obtain the gain reduction curve, the resulting sample must be compared to the original test sample. This can be done mainly in two ways:

- Comparing amplitude values directly for each sample.
- Comparing RMS values, or integration.

The most straight forward way is the first one, which involves simply dividing the resultant sample by the original sample. It is important to have absolute frame accuracy otherwise this process will yield unpredictable results. It is also important to ignore the results produced by samples which are 0. In this method, if the samples are 0, the result for the previous sample is used, though interpolation might be a more accurate method. This is the way gain reduction has been evaluated so far using this method, though some tests comparing RMS values have also been done. The process has been realized using a Csound program (running at kr=1) which on a single pass computes the gain reduction curves and generates a result file for each segment of data (corresponding to the parts listed on section 1).

2.4. Characteristics identified

From the analysis performed on 6 compressors, the following characteristics which seem to have impact on a compressor's quality have been identified:

1. *Latency*: This is the time required for a compressor to start acting. This characteristic is possibly related to the way a plug-in measures level. For instance to calculate the RMS level of a signal, one point is not sufficient, several points must be evaluated. It is possible that some plug-ins do not compensate for this delay and generate *Latency* when applying gain reduction. This latency was observed on the free and open source plug-ins, but it was absent (even to the point that gain reduction was applied to the first sample in the case of the Waves Rcomp). Where observed, latency depends on the amount of signal above the threshold. The higher the signal above the threshold, the lower the latency (See figures 1 and 2).
2. *Smoothness*: The smoothness of the gain reduction curve seems to be another important factor in the way a compressor sounds. Rugged gain reduction curves might be the result of rounding up or irregularities in an algorithm. These irregularities seem to be present only on the SC1, SC4 and Tap plug-ins (Figure 3).
3. *Single frequency handling*: Refers to the way the compressor reacts to different frequencies. All frequencies (more noticeably lower ones), depending on the way the compressor measures level, might trick the compressor into thinking they are actually a fluctuating signal, when they are not, since the window for measuring the level might not cover

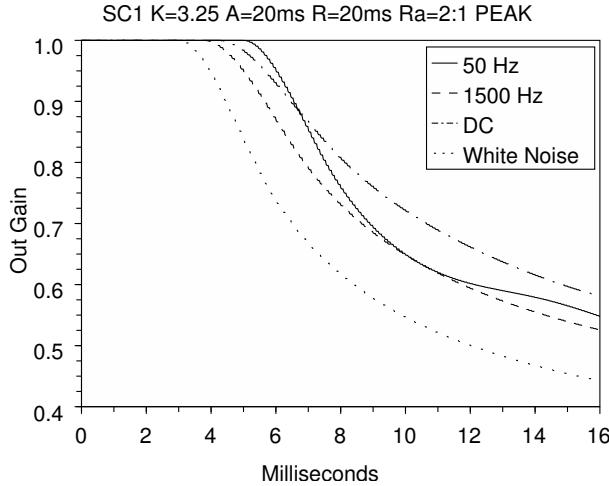


Figure 1: *Gain reduction (PEAK) for the SC1 plug-in at different frequencies.*

the whole wavelength of the signal. This artifact will certainly cause some degree of pumping or other undesired artifacts, and has been identified in all plug-ins tested (see the 50Hz curve in figures 1 and 4).

4. *Steady state frequency deviation:* When a steady gain reduction state has been achieved, plug-ins produce different amount of gain reduction for different frequencies. However, this doesn't amount to changes in frequency response, since the processed white noise doesn't exhibit any kind of coloring, but it will reflect how different kind of material (and especially with regards to spectral content), will be processed by the compressor. Some plug-ins exhibit very even response like the SC1, unlike the Sonitus plug-in which tends to apply less gain reduction to lower frequencies (Figure 4). All plug-ins tested exhibit very different characteristics, but it was observed that lower frequencies tended to be less attenuated, while DC offset tended to be the most attenuated. This seems to point that all plug-ins tested use some form of RMS or integration method in their envelope follower.

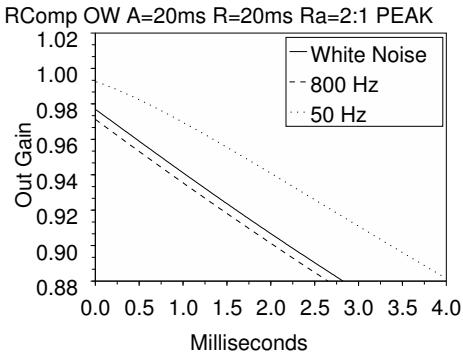


Figure 2: *Gain reduction detail (PEAK) for the RCompressor plug-in.*

2.5. Analysis of the gain reduction curves

A large number of files (1000+ for each result sample) are generated by the previous process. Using Scilab [13], a numerical computation program, it is possible to easily and efficiently extract information from the data, as well as provide graphical feedback which can guide the evaluation process.

It might also be desirable to calculate the derivative of the gain reduction curve. This has been done within Scilab using spline interpolation between each sample. This derivative will help identify when a steady gain has been reached, and may provide additional information about the gain reduction or recover curves.

A Scilab script has been written which can easily load and compare different sets of data from the files generated above. Information that can be extracted in this procedure include:

- Determination of the amplitude tracking technique. Figure 5 shows the RMS comparison value for the SC4 plug-in. The near coincidence of the derivate of the curve for all frequencies shows that this plug-in uses in fact an RMS algorithm. It is not shown here, but the deviations shown appear to be artifacts of interpolation and rounding.
- Localizing the point where gain has stabilized. This can be done again using the derivative of the function.

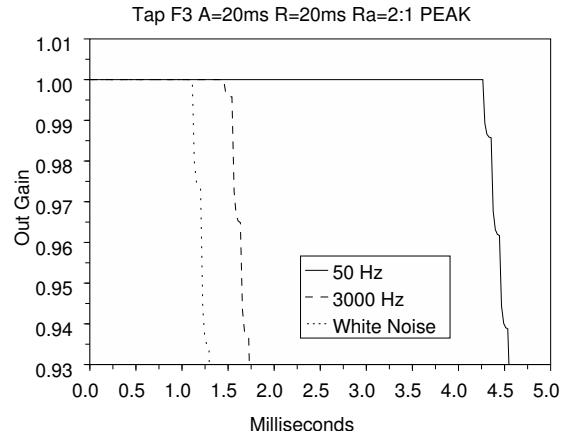


Figure 3: *Gain reduction (PEAK) for the TAP dynamics plug-in at different frequencies.*

- Determining the plug-in's *Latency*. In Figure 1, the SC1 exhibits a latency of around 4.5 ms. In Figure 6 the mda plug-in exhibits very different latencies for each frequency. The RCompressor has kicked in with gain reduction from the first sample (Figure 2).
- Determine the gain applied to a constant signal. Figure 4 shows how the Sonitus compressor behaves for signals 15 dB above the threshold. It can be compared to the behavior of the SC1, which exhibits a very different behavior (Figure 1). Observe particularly the difference in treatment of white noise, DC Offset and the 50Hz line.
- Compare the effect of amplitude above threshold and of the ratio on the gain reduction. See figure 7, which shows how the TAP dynamics plug-in applies gain reduction to signals of the same frequency but different amplitude. Notice the effect of signal level on latency.

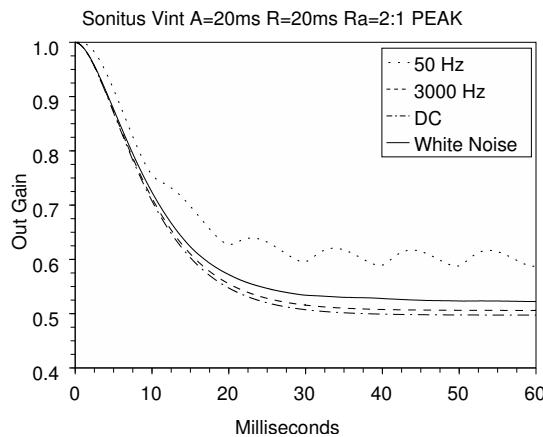


Figure 4: Gain reduction (PEAK) for the Sonitus plug-in at different frequencies.

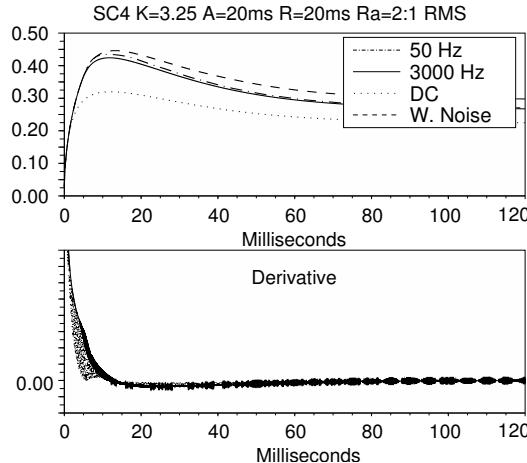


Figure 5: RMS Gain reduction and its derivative for the SC4.

- Find problems or anomalies in an algorithm. For example observe figure 3, where the curve is not straight but moves in undulating patterns. The frequency of the patterns is not dependent of the frequency of the signal, which points to an irregularity somewhere in the algorithm (or maybe even the host, though the same behavior was observed in three hosts).
- Information about the compressor's knee can be obtained by comparing the gain reduction levels for signals with level close to the threshold and by analyzing data from the parts with rising signal level.
- Other interesting effects (or side-effects) can be studied, like the relationship between the phase of the input signal and the gain reduction applied. This is particularly relevant for lower frequencies, and can be seen on Figure 4, where the oscillation in the gain occurs twice per wave period (at 50 Hz, the period is 20ms).
- The curvature and slope of the gain reduction and gain recover curves determine the sound of the compressor. In

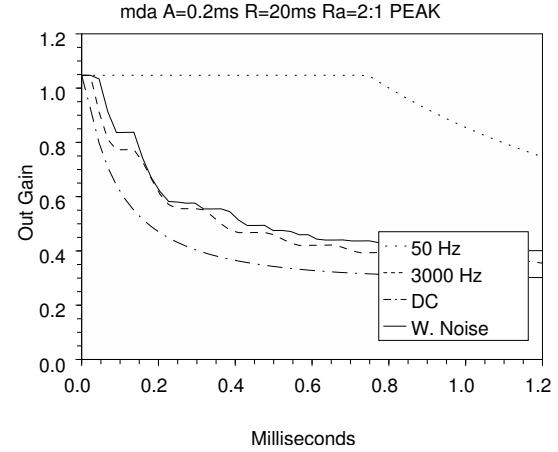


Figure 6: Gain reduction (PEAK) for the mda plug-in at different frequencies.

figure 8, we can see the gain recovery shape for the SC1 plug-in.

- The comparison of actual characteristics with subjective evaluation might yield information to dictate desirable characteristics of a compressor. For example, the (subjective) higher quality output of compressors like the Waves or R-compressor suggest that Latency and irregularities in the gain curve are undesirable artifacts that should be avoided.
- As can be seen in figure 9, compressors behave very differently even at the same settings. The amount of steady gain reduction for the same threshold and ratio varies dramatically. The attack time is not exact in any case, but merely a time constant which determines behavior. The actual time required to reach a steady state is between 3 to 6 times the specified setting in this case.

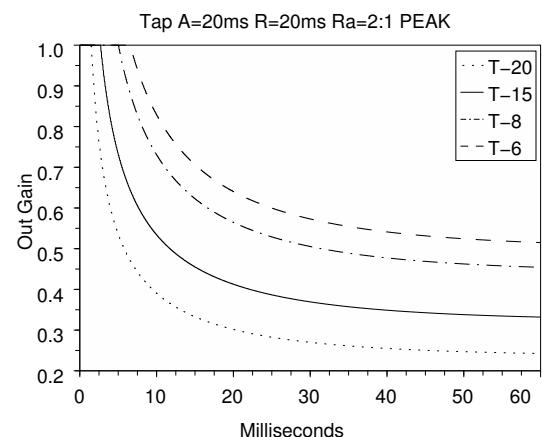


Figure 7: Gain reduction (PEAK) for the TAP dynamics plug-in at different amplitudes in relation to a threshold (for 1500Hz).

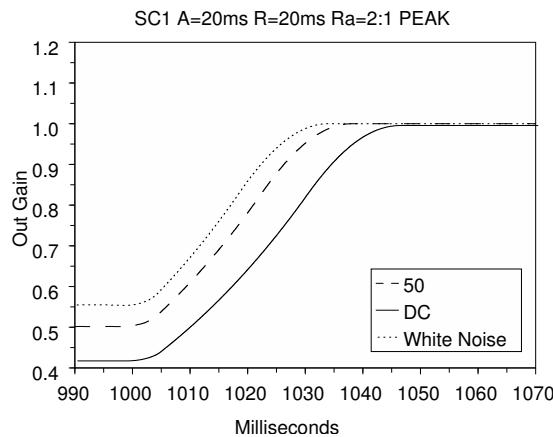


Figure 8: Recovery curve (PEAK) for the SC1 plug-in at different frequencies.

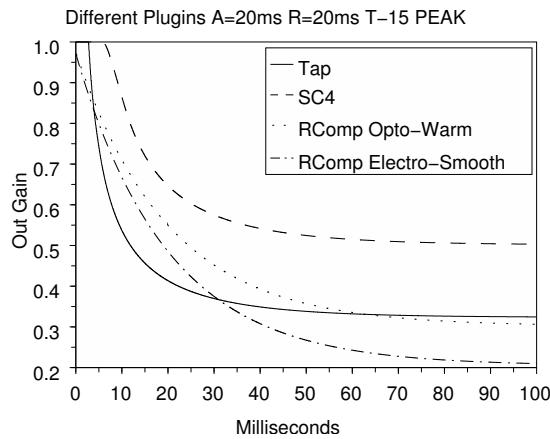


Figure 9: Comparison of gain reduction for different plugins at the same settings (for 1500Hz).

3. FUTURE DIRECTIONS

The method will be applied systematically to as much plug-in compressors as possible. In this process, data about the compressors will be gathered and the method will be further refined, leaving out redundant or unnecessary analysis. The method's goal is to provide a deep understanding of compression and a framework for efficient analysis of dynamic processing, to be able to design, emulate and improve dynamic range compressors. With some adjustments, this methodology can also be eventually applied to other dynamic processes like limiting, gate and expansion. The model will also serve to compare a model plug-in with the original by using a different test sample, probably involving musical material, to compare the treatment between the two models.

The perceived quality of the compressor can be compared with the measured results to determine which physical characteristics relate to desirable auditory output. By carefully analyzing a compressor's output, its method of operation can eventually be derived or closely approximated. The analysis should serve as the basis for compressor modeling of both digital and analog processors in

the digital domain. It should also serve as a basis for developing novel models that exhibit desirable behavior not present in extant compressors. Part of what defines a compressor is the way it tracks amplitude. Using data provided by this analysis can provide mathematical models for emulation. In addition, the slope and curvature of the gain reduction curves can be measured, to be modeled as well.

Using STFT analysis (or measuring the impulse response) to determine if the compressor applies some sort of EQ on the signal has not yet been implemented in Scilab, but tests using other software have shown flat spectrum on the output, indicating no coloring at all. However the process described here can be also be applied to analog compressors, where this can be significant (see [7]).

4. CONCLUSIONS

Presented here is an efficient method which permits detailed analysis of the effects of plug-in compressors on a carefully prepared test sample. The method at its initial stages has already provided insight into the characteristics of compression.

5. REFERENCES

- [1] Waves, "Renaissance compressor," Retrieved June 29th, 2006, [Online] <http://www.waves.com>.
- [2] Cakewalk, "Sonitus fx:Compressor," Retrieved June 29th, 2006, [Online] <http://www.cakewalk.com/Products/Sonitus/sonitus.asp>.
- [3] S. Harris, "SC4 Compressor," Retrieved June 29th, 2006, [Online] <http://www.plugin.org.uk/ladspa-swh/docs/ladspa-swh.html>.
- [4] —, "SC1 Compressor," Retrieved June 29th, 2006, [Online] <http://www.plugin.org.uk/ladspa-swh/docs/ladspa-swh.html>.
- [5] T. Szilagyi, "Tap Dynamics," Retrieved June 29th, 2006, [Online] <http://tap-plugins.sourceforge.net/ladspa/dynamics.html>.
- [6] Maxim digital audio, "mda dynamics," Retrieved June 29th, 2006, [Online] <http://www.mda-vst.com/>.
- [7] M. J. Kemp, "Analysis and Simulation of Analogue Dynamic Compressors and Limiters in the Digital Domain," in *Proc. 109th AES Convention*, Los Angeles, 2000, preprint 5185.
- [8] J. Peissig, J. Remmer ter Hasborg, F. Keiler, and U. Zoelzer, "Digital Emulation of Analog Companding Algorithms for FM Radio Transmission," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 285–290.
- [9] Unique Recording Software, "URS Plugins," Retrieved June 29th, 2006, [Online] <http://www.ursplugins.com/>.
- [10] Python, "Python programming language," Retrieved June 29th, 2006, [Online] <http://www.python.org/>.
- [11] R. Boulanger, *The Csound Book*. MIT Press, 2000.
- [12] J. Schimmel, "Using nonlinear amplifier simulation in dynamic range controllers," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003, pp. 49–52.
- [13] INRIA, "Scilab," Retrieved June 29th, 2006, [Online] <http://www.scilab.org/>.

X-MICKS – INTERACTIVE CONTENT BASED REAL-TIME AUDIO PROCESSING

Norbert Schnell, Diemo Schwarz, Remy Müller

Real Time Applications

IRCAM, Paris, France

{Norbert.Schnell|Diemo.Schwarz|Remy.Muller}@ircam.fr

ABSTRACT

In this article we present the real-time audio plug-in *X-Micks*, an audio processing application allowing for remixing and hybridization of two beat-synchronized audio streams, which provides user interaction based on the extraction and visual rendering of information from the two real-time audio streams. In the current version, the plug-in uses the beat grid information provided by the plug-in host and a real-time estimation of energy in chosen frequency bands to construct an interactive matrix representation allowing for intuitive and efficient user interaction based on familiar representations such as the sonogram and the step sequencer. After trying to formulate the constitutional qualities of a rising new generation of audio processing tools of which we claim *X-Micks* being an exemplary specimen, the article gives an overview over the application's interface, functionalities and implementation.

1. INTRODUCTION

With the availability of robust real-time analysis/resynthesis methods and powerful music information retrieval methods a new generation of interactive real-time processing tools becomes possible. Two additional encouraging factors for these development have to be mentioned, the availability of exchange file formats for temporal music information retrieval data and – most important for the development of all successive generations of interactive real-time processing tools – the availability of fast machines with efficient real-time graphic display providing reactive user interaction.

This new generation of tools goes beyond the friendly cohabitation of real-time display of audio parameters with graphical user interface elements for the manipulation of parameters of real-time audio processing algorithms, inherited from conventional mixing consoles and other audio devices. We permit ourselves to baptize this new generation of tools “Interactive Real-Time Content Based Audio Processing”.

The *X-Micks* application presented in this article is a modest implementation of such a tool satisfying the criteria one could list to define its genre:

- real-time rendering of the interaction interface according to the audio content
- robustness and intuitiveness of the chosen representation in terms of its pertinence for the given content
- integration of offline analysis with real-time analysis and (re-)synthesis

The first point can be seen as the constitutional kernel of this list, while the second defines the criteria of “pertinence” inviting for creativity as well as for further refinement of the definition and discussions concerning the pertinent qualities of different classes

of audio content such as speech, environmental sounds and music as well as different styles of music as known from other research topics in music technology and information retrieval as an interdisciplinary field.

Even if the third point can be considered as optional in terms of supporting our definition, it represents an important challenge for the development of future applications fully taking advantage of the increasing availability of musically pertinent, although inherently non real-time, music information retrieval algorithms within the context of real-time audio processing tools.

2. THE MA-TRICKS

The *X-Micks* interface mainly consists a $12 \times B$ matrix representing the distribution of energy in twelve frequency bands and B beats (or sub-beats – typically 16th notes) of one bar of music. For music based on a 4/4 meter, a 12×16 matrix is displayed.

The representation can be seen as a reduced sonogram obtained by coarse discretisation of the time axis to (sub-)beats and the frequency axis to perceptive frequency bands and was first developed for a simple application called *Ma-Tricks*.

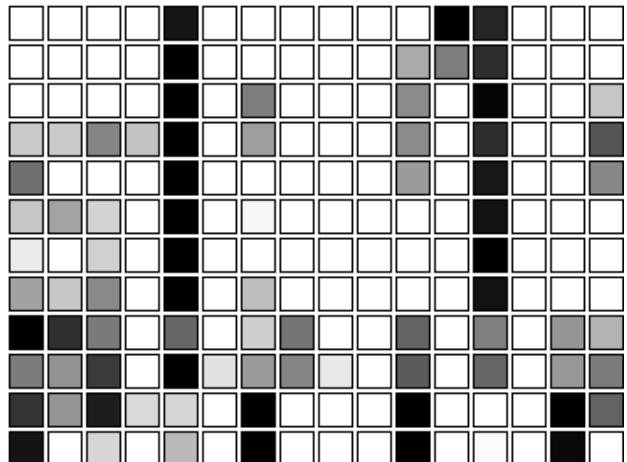


Figure 1: Representation of a reduced sonogram within the *Ma-Tricks* application.

Figure 1 shows a screenshot of the *Ma-Tricks* sonogram of one bar – sixteen sixteenth beats – of the chorus of the song “Die Another Day” by Madonna. One can easily distinguish the bass drum and snare drum beats, without listening to the music. Watching the representation while listening to the music permits to easily associate further perceived events in the auditory stream such as further

percussion instruments, but also sung words, or specific elements of the orchestration to regions in the time-frequency plane represented by the matrix squares. For example on the 11th and 12th sub-beat – just before the second snare drum off-beat on the 13th – one can clearly distinguish in the upper frequencies a column of 4 squares followed by 2, belonging to the word “yes” pronounced by the singer.

The *Ma-Tricks* display inherits properties from two familiar representations: the sonogram and the drum step sequencer. Both representations are united to a novel intuitive representation easy to calculate from an audio stream under the condition that tempo and/or beats are known or extracted from the audio.

2.1. Perceptive frequency bands

Since the works of Eberhard Zwicker [1] various approaches have been proposed to partition the bandwidth of human hearing into perceptually equal frequency bands from different points of view. Recently real-time audio processing applications of filter banks modeling the human auditory system have been proposed to the electronic music community by Pressnitzer and Gnansia [2].

The *Bark scale* [3] proposes a scale of 24 *critical bands of hearing* up to 15.5 kHz. Using the Bark scale is was easy to partition the frequency axis of the matrix display to 12 perceptually pertinent frequency bands each two Bark large. The number 12 seemed to be a good choice in terms of providing a good visual divisibility for the matrix display. An efficient implementation has been obtained by summing the corresponding frequency bins of the energy spectrum obtained by a short-time Fourier transform (SFFT). Here the FFT size has to be adjusted to permit the representation of the lowest frequency band by at least one bin and also a good approximation of the other frequency bands by the bin frequencies.

2.2. Beat-aligned analysis

Evidently the *Ma-Tricks* representation most pertinently applies to music with a rhythmic structure characterized by the recurrence of a limited number of beat patterns such as today’s dance music. To obtain the representation, the SFFT of the 12 frequency bands has to be well synchronised to the beat of the audio stream. The energy regarding each frequency band has to be integrated over one sub-beat. If taking into account tempos between 60 and 200 bpm with a subdivision of 4 sub-beats per beat, a sub-beat has a period between 75 and 250 milliseconds corresponding to 3307.5 and 11025 samples at a sampling rate of 44.1kHz.

Two possibilities have been considered to obtain the given time-frequency matrix using SFFT computation:

- adapting FFT size and hop size to the beat
- using a constant FFT and hop size adding successive FFT frames belonging to the same beat

The advantage of the first is the perfect adaptation of the processing to the perceived rhythm and onsets. For the current implementation of the application, the second possibility has been adopted for its simplicity with success. The hop size has been chosen rather low (128 samples) in order to provide a sufficient temporal resolution.

A bar by bar display of the beat pattern represented by the matrix synchronised to the audio stream in real-time requires either a prior (non real-time) analysis of the reduced sonogram or the delaying of the audio stream by the duration of one bar. The real-time

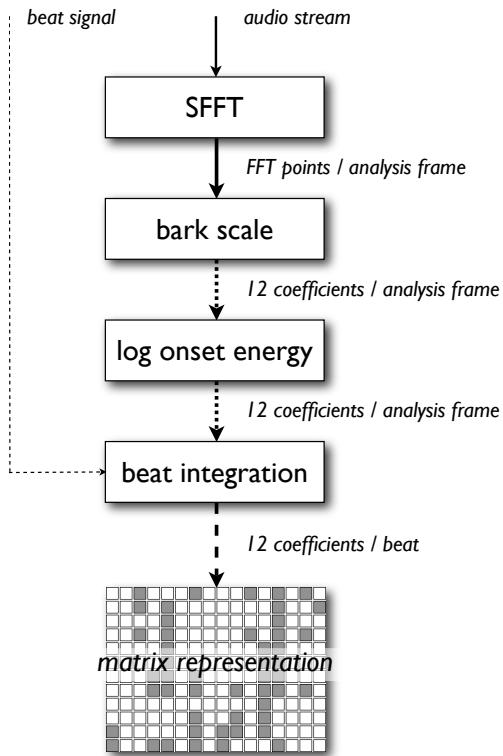


Figure 2: Data-flow diagram of the X-Micks/Ma-Tricks analysis stage.

display chosen for the *X-Micks* application continuously shows the analysis result of the last (sub-)beat in a column of the matrix, while the column representing the current (sub-)beat is hidden by a cursor advancing synchronously with the timing of the music. A memory effect can be added to the display reinforcing the visual presence of recurrent elements of the beat patterns of successive bars.

In the context of the described work the beat analysis and tracking of the audio stream is considered calculated outside of framework of the described application. The processing requires a simple control signal (or event) giving a count for each sub-beat of a bar. Generally, audio plug-in standards such as VST¹, AudioUnits² or RTAS³, provide information concerning the signature, tempo and the exact onset time of beats from which the required sub-beat synchronous signal can be easily derived.

Originally the meter and beat information of plug-ins was used in the context of composition environments mixing music representations with a metric structure and audio processing. Here it allows audio effects such as a multi-delay or a chorus to be synchronised to the meter of a synthesised accompaniment. A newer generation of tools allows beat synchronous processing by extracting the tempo and onsets in real-time from an audio stream and generating the beat related plug-in information on the fly or by a previous off-line analysis of the processed audio file. The actual availability of beat information within a particular audio applica-

¹Steinberg’s Virtual Studio Technology plug-in standard.

²Apple’s audio plug-in standard for Mac OS X.

³Digidesign’s plug-in format developed for ProTools.

tion depends on the implementation of the plug-in host.

Figure 2 shows an overview over the described analysis stage.

3. THE X-MICKS APPLICATION

The *X-Micks* application uses the described matrix representation to interact with a real-time audio processing algorithm filtering and mixing to beat synchronised (stereo) audio streams to a hybrid audio stream. Also the beat-synchronisation of the two incoming audio streams is considered to be external to the application. In the case of a plug-in implementation the beat signal and the audio streams have to be handled synchronised to the *X-Micks* plug-in. This is the case for the envisaged integration of an *X-Micks* VST plug-in into the host application *Traktor*⁴.

3.1. The graphical user interface

Each of the two matrices of the *X-Micks* interface is associated to one of the input audio streams. The matrices display each the reduced sonogram of one bar of the beat synchronised audio streams in real-time. Figure 3 shows the prototype interface of the *X-Micks* applications.

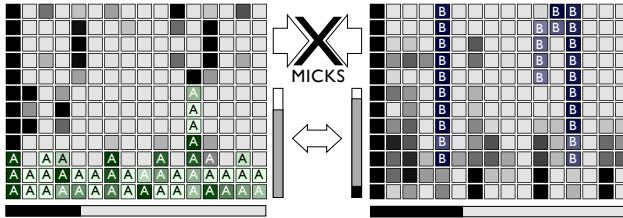


Figure 3: The prototype interface of the *X-Micks* application (version 3) showing the real-time analysis of two beat-synchronized songs, the selected time-frequency regions with the beat cursor (on the first beat) and the other user interface elements.

The user can interact with both matrix displays by clicking to the squares of the matrix toggling the state between unselected (grey) and selected (coloured and additionally marked A and B in figure 3). With the default settings, the selected matrix regions are audible, while the unselected are not. This way the user can reduce each of audio streams to certain beats and frequency bands. Short cuts are provided to control entire lines – corresponding to particular frequency bands – and entire columns – corresponding to particular beats. The matrices can be tied so the interaction with one matrix automatically also modifies the other in a way that one matrix always represents the inverse selection of the other.

An additional short cut permits to select all matrix squares belonging to the same energy range in the column around a selected square and to the same frequency bands in other beats. The tolerance of the energy range around the value of the selected square can be dynamically adjusted by dragging the mouse after clicking, so that the user can select easily the time-frequency regions of distinguishable recurrent components of the audio stream such as bass drum, snare drum or hi-hat. This interaction depends directly on the analysed energy so that one could speak here of a simple case of *content based interaction*.

⁴*Traktor DJ Studio* is a trademark of Native Instruments Software Synthesis GmbH – see <http://www.nativeinstruments.de/>

The range sliders between the matrix displays can be used to adjust the actual level for the selected (upper part of the slider) and unselected (lower part of the slider) time-frequency regions. The slider in the middle left of figure 3 shows that the selected regions of the audio stream represented by the left matrix is slightly lowered while the unselected regions are completely suppressed. In the contrary, the unselected regions of the audio stream represented by the right matrix are not completely suppressed as shown by the slider on the right.

3.2. Beat-synchronous filtering

The actual sound processing to create the final hybrid audio stream out of the two incoming audio streams is sufficiently described as *time-variant beat-synchronous filtering*. As well the filtering can be relatively easily implemented based on SFFT. Figure 4 shows an overview over the involved data-flow including the user interaction.

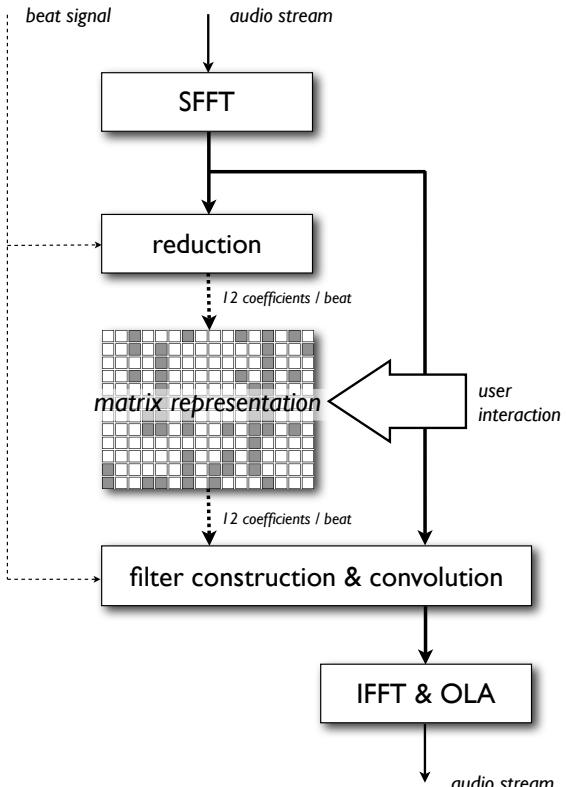


Figure 4: Data-flow diagram of the beat-synchronous processing of the *X-Micks* application.

A straight forward implementation derives the reduced sonogram representation from the same SFFT spectra of the input streams which are used for the convolution and re-synthesis (by IFFT and overlap-add) after the superposition of the spectra from both streams to a single output steam. One can see the overall algorithm as a simple real-time analysis/synthesis process which provides an intermediate representation allowing for intuitive user interaction. In the case of using beat synchronous SFFT the algorithm implements a beat-synchronous overlap-add (*BSOLA*)

algorithm as proposed by Peeters for a different context [4]. In analogy to *frequency-domain PSOLA*, here one could speak of *FD BSOLA*.

3.3. Prototyping and implementation environment

The implementation of the developed *X-Micks* application is entirely based on the *Gabor* [5] library developed at IRCAM with the *FTM* [6] extension for the Max/MSP [7] environment⁵. The matrix representations as well as the SFFT processing are implemented using the *FTM fmat* class and the related functionalities allowing very rapid and efficient prototyping within Max/MSP. The plug-in version has been developed using the *pluggo*⁶ extension for Max/MSP.

4. CONCLUSIONS AND FUTURE WORK

X-Micks is a prototype application of interactive real-time content based audio processing. Even if *X-Micks* features rather basic signal processing it perfectly embodies a novel paradigm of audio processing. We made a tentative to define this novel genre of interactive audio processing applications in a way that is meant to be inspiring for further developments.

For the *X-Micks* application itself we imagine multiple future enhancements. One is related to one of the ideas being at the origin of this work, which can be called *interactive source separation*. The act of choosing time frequency regions – beats and frequency bands – belonging to a certain instrument or sound source while listening to the filtered result can be seen as an interactive approach to source separation. One can easily imagine to enhance this approach by additional features and more sophisticated audio processing for example introducing source separation [8] and segmentation algorithms. A preliminary off-line analysis (or even decomposition) of the audio stream is acceptable for many applications.

Further efforts have to go in the further simplification and refinement of the graphical user interface and interaction to enhance the performability of the application.

We see a potential in the described matrix representation for the description and indexing of rhythmic patterns and segmented audio loops. When coding the energy of the matrix with 2-bit (four steps) each line of a 12×16 matrix can be represented by a 32-bit word. The 12 frequency bands maybe can be further reduced to 3 or 4 for this application.

5. ACKNOWLEDGEMENTS

This work was carried out in the framework of the European project *Semantic HIFI* and is the result of a fruitful collaboration with the company *Native Instruments*. Thanks to Florian Plenge and Egbert Juergens from NI for their encouraging support and helpful comments.

The ideas for the *X-Micks* application are inspired by the work of Andrea Cera and initially came up in a discussion with Andrea, Nicola Donin and Samuel Goldszmidt, with whom the mentioned *Ma-Tricks* application has been realized.

Further inspirations for this project are coming from the work and discussions with Geoffroy Peeters. Finally we'd like to thank our colleagues Riccardo Borghesi and Frederic Bevilacqua for courageously having endured unbearable hours and weeks of testing the *X-Micks* plug-in and its preceding prototypes with the same short loops extracted from pop songs.

6. REFERENCES

- [1] E. Zwicker, G. Flottorp, and S. S. Stevens, "Critical bandwidth in loudness summation," *J. Acoust. Soc. Am.*, vol. 29, pp. 548–557, 1957.
- [2] D. Pressnitzer and D. Gnansia, "Real-time auditory models," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 295–298.
- [3] J. O. Smith III and J. Abel, "Bark and ERB bilinear transform," *IEEE Trans. Speech and Audio Proc.*, vol. 7, no. 6, pp. 697–708, Nov. 1999.
- [4] A. La Burthe and G. Peeters, "Résumé sonore," in *Internship in the framework of the master ATIAM – final report IRCAM/INPG Grenoble*, Paris, France, 2002.
- [5] N. Schnell and D. Schwarz, "GABOR, multi-representation real-time analysis/synthesis," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005, pp. 122–126.
- [6] N. Schnell, R. Borghesi, D. Schwarz, F. Bevilacqua, and R. Müller, "FTM – complex data structures for Max," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 9–12.
- [7] M. Puckette, "Combining event and signal processing in the MAX graphical programming environment," *Computer Music J.*, vol. 15, no. 3, pp. 68–77, 1991.
- [8] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech and Language Proc.*, vol. 14, no. 4, pp. 1462–1469, July 2006.

⁵<http://www.cycling74.com/products/maxmsp>

⁶<http://www.cycling74.com/products/pluggo>

DISPERSION MODELING IN WAVEGUIDE PIANO SYNTHESIS USING TUNABLE ALLPASS FILTERS

Jukka Rauhala, Vesa Välimäki

Laboratory of Acoustics and Audio Signal Processing

Helsinki University of Technology, Espoo, Finland

jukka.rauhala@acoustics.hut.fi, vesa.valimaki@tkk.fi

ABSTRACT

This paper extends a previously proposed method for designing filters simulating the dispersion phenomenon occurring in string instruments. In digital waveguide synthesis, the phenomenon is traditionally modeled by inserting an allpass filter to the string model feedback loop. In this paper, the concept of tunable dispersion filter design, which provides a closed-form formula to design a dispersion filter, is applied to a cascade of first-order allpass filters. Moreover, the method is extended to design a filter cascade including an arbitrary number of first-order filters. In addition, it is shown how the designed dispersion filter can be used in a waveguide piano synthesis model.

1. INTRODUCTION

Dispersion is an important phenomenon present in string instruments making produced tones inharmonic. The audibility of the phenomenon depends on the instrument, for example, in the piano it is a perceptually important effect that needs to be taken into account in sound synthesis [1].

In digital waveguide modeling technique [2, 3], dispersion is modeled by inserting an allpass filter simulating the phase delay response of the dispersion phenomenon in the string model [2, 3, 4, 5, 6, 7, 8]. A common way to implement the dispersion filter is to use either a high-order allpass filter [4] or a cascade of low-order filters [5, 6, 7]. An excellent overview of dispersion filter design is given in [8].

Van Duyne and Smith [5] proposed the use of a cascade of first-order filters. In this paper, the idea is extended by introducing a closed-form formula to determine the filter coefficients based on the tunable dispersion filter design method [7].

This paper is organized in the following way. In the beginning, the dispersion phenomenon is introduced in Section 2. Then, the previously proposed tunable dispersion filter design method for a cascade of second-order filters is presented in Section 3, followed by the adaptation for first-order filters in Section 4. The results from the filter design are shown in Section 5 and the conclusions are presented in Section 6.

2. DISPERSION PHENOMENON

Dispersion is the property of a string, due to its stiffness, that causes the frequencies of the partials to be higher than the harmonic partials. The frequencies of inharmonic partials can be calculated [1]:

$$f_k = k f_0 \sqrt{1 + B k^2}, \quad (1)$$

where k is the partial number, f_0 is the nominal fundamental frequency of the ideal string (non-dispersive), and B is the inharmonicity coefficient. The value B can be calculated using string parameters [1]

$$B = \frac{\pi^3 Q d^4}{64 l^2 T}, \quad (2)$$

where Q is Young's modulus, d is the diameter of the string, l is the length, and T is the string tension. Hence, there is a strong relation between the B value and the tension of the string.

Fletcher *et al.* suggested in 1962 that inharmonicity is a significant factor in producing the sound characteristic of the piano [1]. Even though Galembó *et al.* recently proposed that the spectral density might be even a more important factor [9], it is still clear that inharmonicity is an essential property, which needs to be taken into account in piano synthesis.

Dispersion occurs in all string instruments, especially the piano is known to have strong inharmonicity [1]. The amount of inharmonicity depends on the type of the piano; grand pianos tend to have less inharmonicity than upright pianos. Figure 1 depicts estimated B values from a Steinway grand piano. The B values were obtained by an automatic estimation algorithm, which first uses the fast Fourier transform (FFT) to calculate the frequencies of the partials, and then estimates the B value by fitting (1) to the measured data. Some values are missing in Figure 1, because the estimated values for some keys were unreliable (the most common reasons for excluding a B estimate were that the nominal fundamental frequency was estimated incorrectly, or that our peak picking algorithm got confused). The solid lines in the figure show the expected B range for pianos based on our results. Our estimated range is in line with the results by Askenfelt and Galembó [10].

Although the inharmonicity is stronger at the higher end of the range, it has been shown by Järveläinen *et al.* that the inharmonicity is perceived mostly at the lower end of the range [11] (recent work by Järveläinen and Karjalainen suggest, however, that there may be variation in the perception depending on the instrument [12]). The estimated B values, displayed in Figure 1, are all above the threshold of audibility, which, along with the confidence limits, are published in [11]. Thus, it can be expected that inharmonicity is heard through-out the range of the piano.

3. TUNABLE DISPERSION FILTER DESIGN METHOD

The tunable dispersion filter design method provided a way to use closed-form formulas to design a dispersion filter [7]. The original design was made for a cascade of second-order filters, but the method can be used to design an arbitrary-order filter cascade.

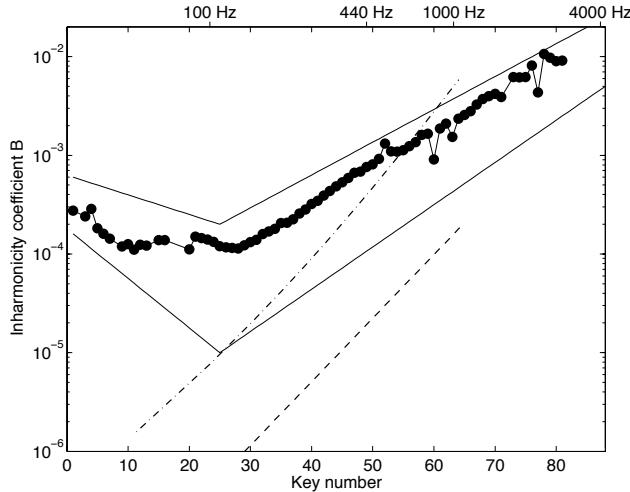


Figure 1: Measured B values (dots) from a Steinway grand piano¹. The solid lines show the expected B range for the piano, the dashed line shows the threshold of audibility, and the dash-dotted line its confidence limit [11].

The tunable dispersion filter design method uses the Thiran all-pass filter design to determine the filter coefficients [13, 14, 15]. The Thiran allpass design method is commonly used to design fractional delay filters, but with large delay values the phase delay response becomes similar to the phase delay required to simulate the dispersion.

In the Thiran design method, the filter coefficients are determined by using the desired delay value D at dc. The tunable dispersion filter design method offers a formula to compute D according to the desired f_0 and B [7]:

$$D(I_{key}, B) = e^{(C_d(B) - I_{key}k_d(B))}, \quad (3)$$

where

$$I_{key}(f_0) = \log \frac{f_0}{\sqrt[12]{2}} \frac{12\sqrt{2}}{27.5}, \quad (4)$$

$$k_d(B) = e^{(k_1(\ln B)^2 + k_2 \ln B + k_3)}, \quad (5)$$

$$C_d(B) = e^{(C_1 \ln B + C_2)}, \quad (6)$$

and k_1 , k_2 , k_3 , C_1 , and C_2 are predefined constants presented in Table 1. It has been noticed that, when the inharmonicity coefficient value is fixed, the relation between required delay values (on a logarithmic scale) and key indices can be approximated with a straight line. Hence, parameter k_d can be interpreted as the slope coefficient of the line and parameter C_d as the remainder of the line formula. The line parameters k_d and C_d are then parameterized, depending on the inharmonicity coefficient, with parameters k_1 , k_2 , k_3 , C_1 , and C_2 .

The dispersion filter, consisting of a cascade of four second-order allpass filters, produces an extra delay that must be taken into account in the string model by modifying the delay line length and the tuning filter coefficients [7]. The extra delay can be estimated by using D value and multiplying it with M , the number of filters in cascade.

¹Steinway grand piano samples from University of Iowa Electronic Music Studios, <http://theremin.music.uiowa.edu>

Parameter	Filter 1 ($M = 4$)	Filter 2 ($M = 1$)
k_1	-0.00050469	-0.0026580
k_2	-0.0064264	-0.014811
k_3	-2.8743	-2.9018
C_1	0.069618	0.071089
C_2	2.0427	2.1074

Table 1: Predefined parameters for the tunable dispersion filter using second-order filters [7]. Filter 1 (four filters in a cascade) is used for key numbers 1-44 and filter 2 (a single filter) for key numbers 45-88.

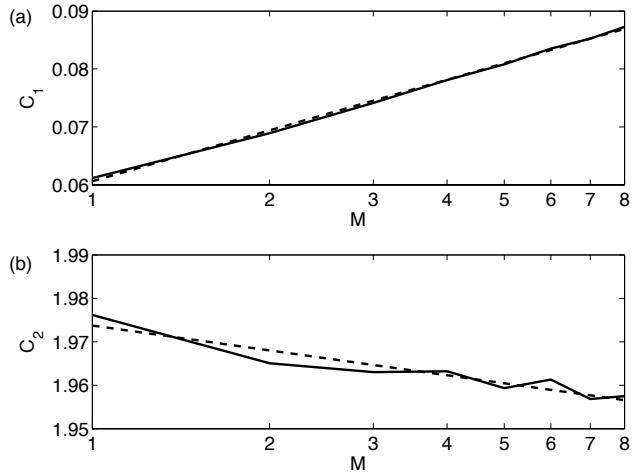


Figure 2: Searched (a) C_1 and (b) C_2 values for first-order filters with M values 1 to 8 (solid line). The dashed line illustrates the parameterized values using the straight line fit.

4. NEW FIRST-ORDER DISPERSION FILTER DESIGN

In this paper, a cascade of first-order filters is used for designing a dispersion filter originally proposed by Van Duyne and Smith [5]. A cascade of identical filters is not as effective from the performance point of view as a cascade of filters with different coefficient values. However, the advantage of using a cascade of identical filters is that it is easy to design and, hence, it is used in this work.

The problem with this idea has been the lack of closed-form design formulas, as Van Duyne and Smith did not indicate how the filter cascade can be designed, other than by trial and error or by using a search algorithm [5]. This problem can be solved by extending the tunable dispersion filter method [7]. In this section, the predefined parameters (see equations 3-6) are determined for the tunable dispersion filter method using a cascade of an arbitrary number of first-order allpass filters. Hence, the formula should take the fundamental frequency, the B value, and the number of filters in cascade M as input parameters.

The five predefined constant parameters in equation (5) and (6) were determined for eight cases with M varying from 1 to 8 in a way similar to [7]. It was noticed that parameters k_1 , k_2 , and k_3 lacked any kind of general trend. However, it seems that the importance of these parameters is minor compared to parameters C_1 and C_2 , which had a clear linear trend on a logarithmic scale, as seen in Figure 2. The explanation for this is that the slope coef-

Parameter	Value
k_1	-0.00179
k_2	-0.0233
k_3	-2.93
m_1	0.0126
m_2	0.0606
m_3	-0.00825
m_4	1.97

Table 2: Calculated filter design parameters for a cascade of first-order Thiran allpass filters.

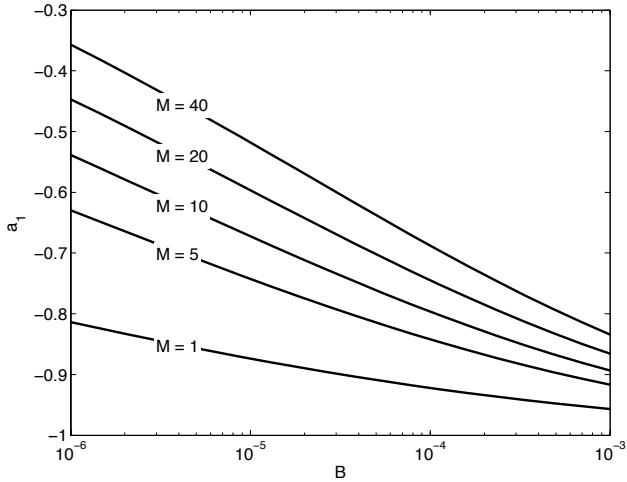


Figure 3: The dispersion filter coefficient a_1 values with B values from 10^{-6} to 10^{-3} when the number of first-order allpass filters in cascade M is 1, 5, 10, 20, and 40. The fundamental frequency is 65.406 Hz (note C_2 in all cases).

ficient k_d does not depend on the number of filters in cascade M , whereas the remainder term C_d is strongly linked to M . Hence, by using polynomial approximation, equation (6) can be extended as

$$C_d(B, M) = e^{((m_1 \ln M + m_2) \ln M + m_3 \ln M + m_4)} \quad (7)$$

where m_1 , m_2 , m_3 , and m_4 are the polynomial coefficients defined in Table 2.

The parameters required by the design formulas (3), (5), and (7) were optimized using polynomial approximation. Parameter values k_1 , k_2 , and k_3 were determined by using $M = 8$. The resulting values are shown in Table 2.

In summary, the whole design process goes as follows. First, the desired B , M , and f_0 values should be decided. Then, the required D values can be determined by using equations (3), (4), (5), and (7). Finally, the first-order allpass filter coefficient a_1 can be computed by using the Thiran allpass filter method [15]:

$$a_1 = \frac{1 - D}{D + 1} \quad (8)$$

Two examples on how a_1 behaves when the inharmonicity coefficient value is changed are shown in Figures 3 and 4.

The delay line length and the tuning filter coefficient should be modified according to the additional delay produced by the dispersion filter. It has to be accounted for that the product $D \cdot M$ may

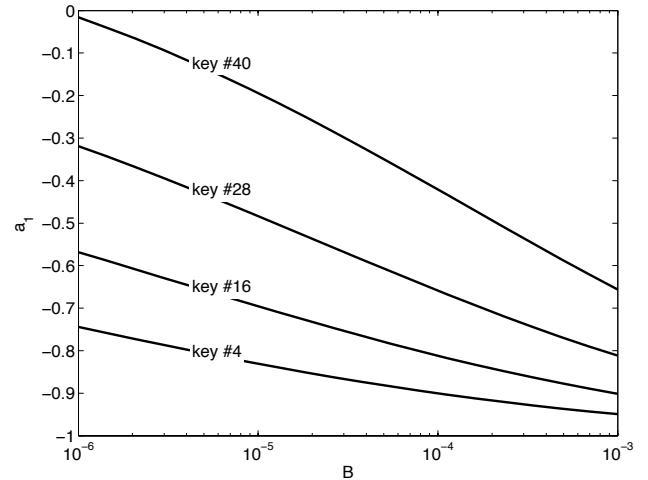


Figure 4: The dispersion filter coefficient a_1 values with B values from 10^{-6} to 10^{-3} at key indices 4, 16, 28, and 40, which correspond to keys C_1 , C_2 , C_3 , and C_4 , respectively. The number of filters in cascade M is 8 in all cases.

not be a sufficient estimate for the delay with large M values, as suggested in [7] for a cascade of four second-order filters, because the estimation error is approximately M times larger compared to a single allpass filter.

5. RESULTS

Figure 5 shows the deviation of the partial frequencies of the example filter proposed in this work at three fundamental frequencies ($\text{keys } C_1$, C_2 , and C_3) with reasonable inharmonicity coefficient values, with the values for the number of filters in cascade M set to 1, 5, 10, 20, 30, and 40. The partial frequencies are compared to the target partial frequencies obtained from equation (1). Hence, an ideal dispersion filter would correspond to zero deviation at all frequencies. The figure shows that the quality of the filter improves when M is increased. On the other hand, when the number of filters in cascade M reaches approximately 40, the phase delay response does not fit within the defined error limits. In practical cases, M values below 20 are preferred for computational reasons. Hence, the parameter selection for the filter design is a trade-off between computational load and quality, but the quality does not improve endlessly. Furthermore, the fixing of parameters k_1 , k_2 , and k_3 does not add too much bias to the results when $M < 40$, since the responses stay within the defined error limits.

The proposed filter, as well as the original tunable dispersion filter method [7], can be used for providing real-time control over the inharmonicity coefficient. This can be done by recomputing equations (3), (5), (7), and (8), and by updating the filter coefficient each time the inharmonicity coefficient value is changed. The updating process requires eight additions, six multiplications, and one division from the computational point of view. Additionally, the logarithmic function is called once and the exponential function three times during the process.

Finally, the filter was included in a simple piano synthesis model. The number of filters in cascade M was set to 8 in this example. The piano model included a basic string model, as shown

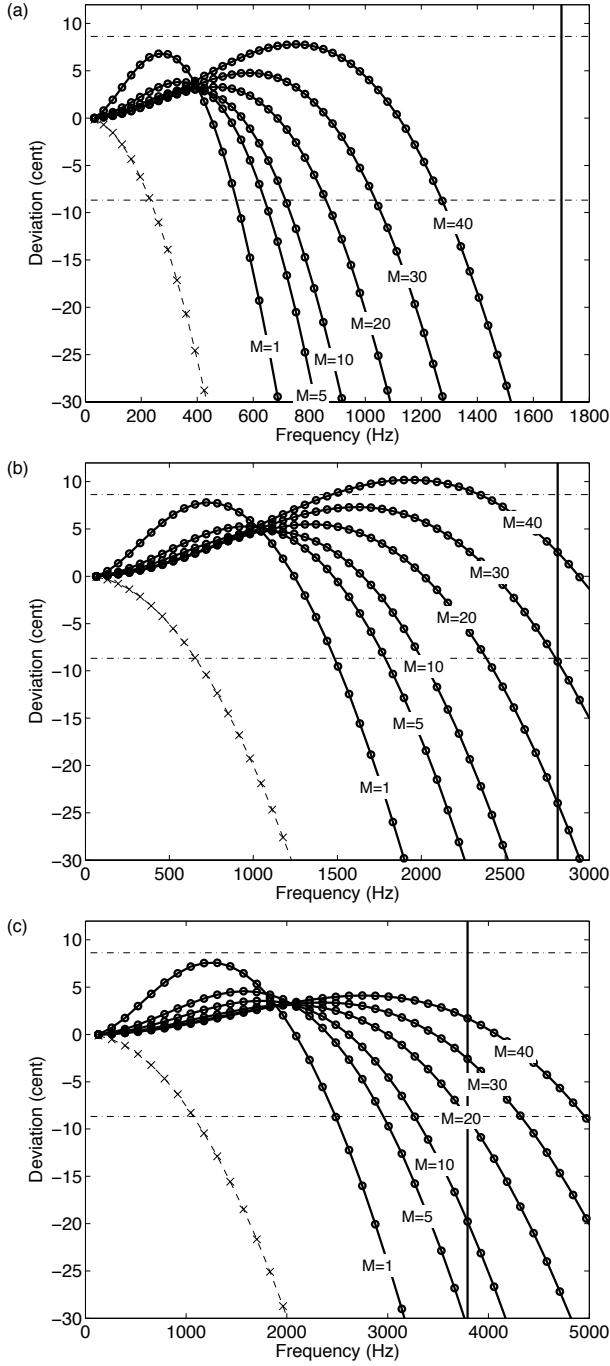


Figure 5: Deviation of the partial frequencies for the proposed filter (solid line with circles) with the number of filters in cascade M values 1, 5, 10, 20, 30, and 40, when (a) $B = 0.00020$, $f_0 = 32.703$ Hz (key C₁), (b) $B = 0.00010$, $f_0 = 65.406$ Hz (key C₂), and (c) $B = 0.00015$, $f_0 = 130.82$ Hz (key C₃). The deviation for a harmonic tone is denoted as a dashed line and the maximum number of perceived inharmonic partials [16] is denoted as a vertical line. The dash-dotted lines are 0.5% error limits corresponding to ± 8.39 cents.

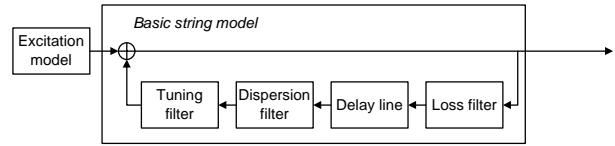


Figure 6: An illustration of a basic piano string model.

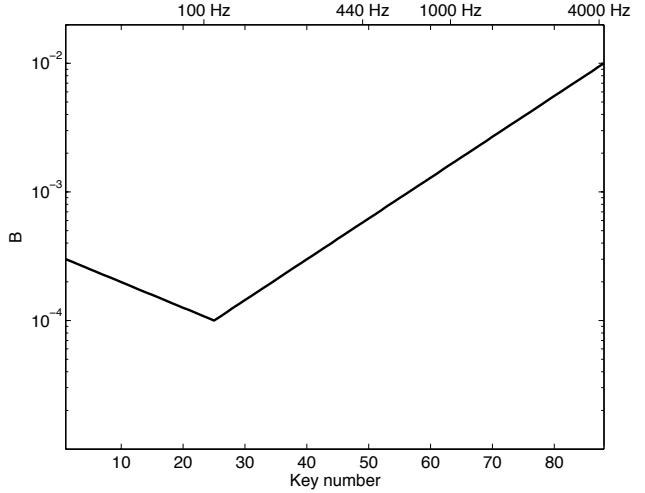


Figure 7: The inharmonicity coefficient B values used in the example piano synthesis model.

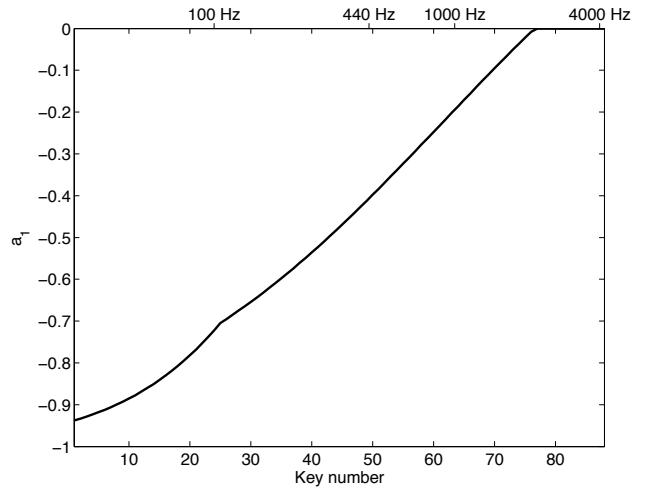


Figure 8: The dispersion filter parameter a_1 values for the piano model when a cascade of $M = 8$ first-order allpass filters is used.

in Figure 6, and the input signals were extracted from real piano samples [17]. The defined inharmonicity coefficient values used in the model are shown in Figure 7. In Figure 8, the corresponding dispersion filter coefficient values are presented. Moreover, the determined delay line length and the delay produced by the dispersion filter are illustrated in Figure 9.

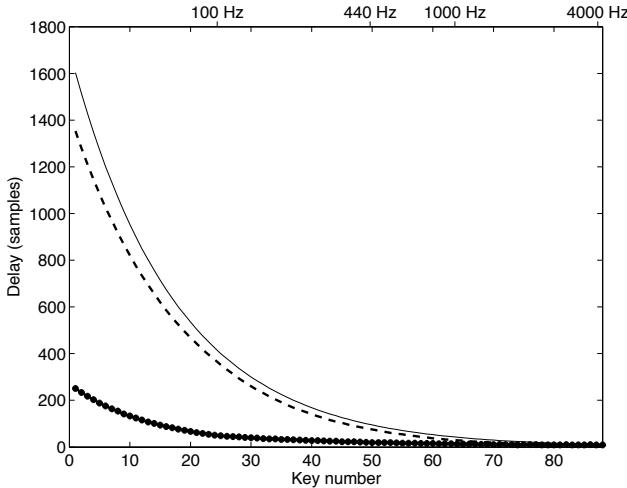


Figure 9: The delay parameters for the piano model: the length of the delay line (dashed line), the length without the dispersion filter (solid line) and the difference between these two which is the delay produced by the dispersion filter (solid line with dots).

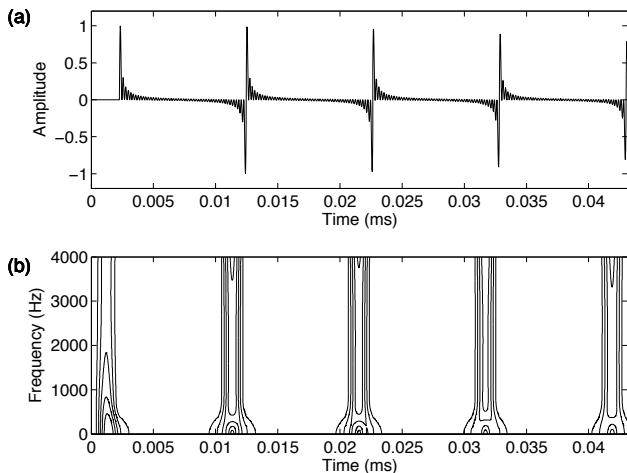


Figure 10: (a) The waveform and (b) the time-frequency plot of a harmonic tone ($f_0 = 98.0$ Hz, key G₂) produced with a string model with no dispersion filter. The excitation signal is simplified in order to emphasize the effect.

When equation 3 resulted in a D value less than 1 (for instance, key numbers 75-88 in Figure 9), D was set to be 1, which corresponds to replacing the allpass filters with the transfer function $A(z) = 1$. The tuning filter was implemented with a first-order Thiran allpass filter, and a delay of one sample was moved from the delay line to the tuning filter in order to have the fractional delay parameter in the range from 1 to 2. It can be noticed that the tunable dispersion filter reduces the need for a delay line memory up to 200 samples in this case (see Figure 9).

Figure 10 and Figure 11 illustrate how the dispersion filter affects the waveform and the spectral properties of the produced tone in time domain. This way of visualizing the effect of dispersion by using a short time window was suggested by Woodhouse [18]. The

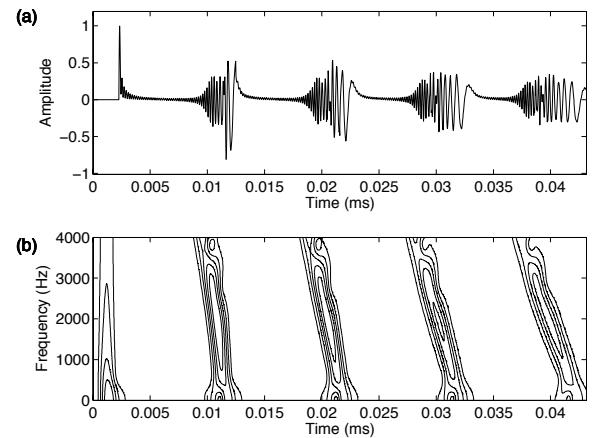


Figure 11: (a) The waveform and (b) the time-frequency plot of an inharmonic tone ($f_0 = 98.0$ Hz, key G₂, $B = 0.0001$) produced with a string model with eight first-order allpass filters in cascade. The excitation signal is simplified in order to emphasize the effect.

excitation of the synthetic tones contains 50 partials with sine starting phases produced by using additive synthesis. Two effects can be seen: the waveform of the tone becomes more spread out and the high frequencies travel faster in than the low frequencies in the feedback loop of the string model. Sound examples as well as Matlab code to calculate the dispersion filter coefficient are available at <http://www.acoustics.hut.fi/demos/ext-disp/>.

6. CONCLUSIONS

In this paper, a closed-form formula is proposed to be used in designing first-order allpass filters for dispersion modeling. The tunable dispersion filter design method that we introduced recently provides a technique which is extended in this work to determine the closed-form formula for an arbitrary number of first-order filters in cascade. It enables an easy design process for first-order dispersion filters, and, also, it provides real-time control over the dispersion phenomenon.

7. ACKNOWLEDGMENTS

This work was supported by the Nokia Foundation and the Academy of Finland (project no. 104934).

8. REFERENCES

- [1] H. Fletcher, E. D. Blackham, and R. Stratton, "Quality of piano tones," *J. Acoust. Soc. Am.*, vol. 34, no. 6, pp. 749–761, 1962.
- [2] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music J.*, vol. 16, no. 4, pp. 74–91, 1992.
- [3] D. A. Jaffe and J. O. Smith III, "Extensions of the Karplus-Strong plucked-string algorithm," *Computer Music J.*, vol. 7, no. 2, pp. 56–69, 1983.
- [4] D. Rocchesso and F. Scalcon, "Accurate dispersion simulation for piano strings," in *Proc. Nordic Acoust. Meeting (NAM'96)*, Helsinki, Finland, 1996, pp. 407–414.

- [5] S. A. Van Duyne and J. O. Smith III, "A simplified approach to modeling dispersion caused by stiffness in strings and plates," in *Proc. Int. Comp. Music Conf. (ICMC'94)*, Århus, Denmark, 1994, pp. 407–410.
- [6] J. Bensa, "Analysis and synthesis of piano sounds using physical and signal models," Ph.D. dissertation, Université de la Méditerranée, France, 2003.
- [7] J. Rauhala and V. Välimäki, "Tunable dispersion filter design for piano synthesis," *IEEE Sig. Proc. Letters*, vol. 13, no. 5, pp. 253–256, 2006.
- [8] B. Bank, F. Avanzini, G. Borin, G. D. Poli, F. Fontana, and D. Rocchesso, "Physically informed signal processing methods for piano sound synthesis: A research overview," *EURASIP J. on Applied Sig. Proc.*, vol. 2003, no. 10, pp. 941–952, 2003.
- [9] A. Galembro, A. Askenfelt, L. L. Cuddy, and F. A. Russo, "Perceptual relevance of inharmonicity and spectral envelope in the piano bass range," *Acta Acustica united with Acustica*, vol. 90, no. 3, pp. 528–536, 2004.
- [10] A. Askenfelt and A. S. Galembro, "Study of the spectral inharmonicity of musical sound by the algorithms of pitch extraction," *Acoust. Physics*, vol. 46, no. 2, pp. 121–132, 2000.
- [11] H. Järveläinen, V. Välimäki, and M. Karjalainen, "Audibility of the timbral effects of inharmonicity in stringed instrument tones," *Acoust. Research Letters Online*, vol. 2, no. 3, pp. 79–84, 2001.
- [12] H. Järveläinen and M. Karjalainen, "Importance of inharmonicity in the acoustic guitar," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 363–366.
- [13] J.-P. Thiran, "Recursive digital filters with maximally flat group delay," *IEEE Trans. Circuit Theory*, vol. 18, no. 6, pp. 659–664, 1971.
- [14] A. Fettweis, "A simple design of maximally flat delay digital filters," *IEEE Trans. Audio and Electroacoustics*, vol. 20, no. 2, pp. 112–114, 1972.
- [15] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay: Tools for fractional delay filter design," *IEEE Sig. Proc. Magazine*, vol. 13, no. 1, pp. 30–60, 1996.
- [16] D. Rocchesso and F. Scalcon, "Bandwidth of perceived inharmonicity for physical modeling of dispersive strings," *IEEE Trans. Speech and Audio Proc.*, vol. 7, no. 5, pp. 597–601, 1999.
- [17] J. Rauhala and V. Välimäki, "Parametric excitation model for waveguide piano synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'06)*, Toulouse, France, 2006, pp. 157–160.
- [18] J. Woodhouse, "Plucked guitar transients: comparison of measurements and synthesis," *Acta Acustica united with Acustica*, vol. 90, no. 5, pp. 945–965, 2004.

PREPARED PIANO SOUND SYNTHESIS

Stefan Bilbao

Music

University of Edinburgh

United Kingdom

sbilbao@staffmail.ed.ac.uk

John ffitch

Department of Computer Science

University of Bath

United Kingdom

jpff@codemist.co.uk

ABSTRACT

A sound synthesis algorithm which simulates and extends the behaviour of the acoustic prepared piano is presented. The algorithm is based on a finite difference approximation to multiple stiff string vibration, including an excitation method (a hammer) as well as several connected preparation elements, modeled as lumped nonlinearities. Numerical issues and implementation details are discussed, and sound examples are presented.

1. INTRODUCTION

The use of physical modeling principles in sound synthesis leads, eventually, to certain thorny questions. By their very nature, such methods allow very high-fidelity modeling of instrument timbres—and yet, by increasingly close attention to detail in the physics of a musical instrument, one can obtain a result which is, for musical purposes, useless: in the limiting case, an exact reproduction of the sound of an existing musical instrument. Though such sound synthesis methods are undeniably extremely important for purposes of checking the validity of a given musical instrument model, and for commercial sound synthesis applications, the focus here is on the creation of sounds which retain some of the natural texture of those produced by acoustic instruments, but which are, in some sense, new. In this regard, the use of physical models, though admirably suited to the task, is something of a balancing act.

One approach is to begin from a well-defined musical instrument, and to perform variations on its structure. This is essentially what was done early on, in the mechanical setting, in the case of prepared piano (probably best known through the work of John Cage [1], but still in extensive use up to the present day). Objects such as screws, rubber erasers, and washers (among others) are placed in proximity to the strings of a piano, yielding timbres sometimes quite foreign to the familiar piano tone—strongly percussive and bell-like tones may be generated in this way. Such an approach can be easily translated to physical modeling sound synthesis approaches, and leaves open the door to various types of preparation which are unrealizable in an actual piano (at least, not realizable without causing permanent damage to the instrument). In this study, string vibration will be simulated using finite difference schemes [2, 3, 4, 5], and various connecting or exciting elements, such as a standard hammer [2], and abstractions of preparing objects, are dealt with as lumped objects (though this is not strictly necessary). This approach is reminiscent of the mass-spring network type models used in the past [6], but the use of a fully distributed model of the string is far less unwieldy. Digital waveguides [7], [8] could also potentially be employed, but their efficiency advantage is substantially reduced under stiff string conditions, and when a string is broken up by various elements in con-

tact with it (i.e., with a scattering junction required at each such connecting points, as well as the accompanying fractional delay machinery [9]). Another more important reason for making use of a direct simulation approach such as finite differences is that any results can easily be extended to deal with systems far more complex than simply strings.

In Section 2, a model of the stiff string, suitable for piano sound synthesis is presented, followed in Section 3 by definitions of the hammer interaction and various preparing elements. In Section 4, a finite difference scheme for multiple strings is given, including a discussion of boundary termination and numerical stability. Implementation details, particularly with regard to the csound music programming language [10] follows in Section 5. Numerical results appear in Section 6.

2. A STIFF STRING MODEL

A useful general purpose model of string vibration may be written as [3]:

$$\ddot{u} = c^2 u'' - \kappa^2 u'''' - \sigma \dot{u} + b \ddot{u} \quad (1)$$

Here, $u(x, t)$ represents transverse string displacement in a single polarization, as a function of time $t \geq 0$ and $x \in [0, 1]$. Dots and primes indicate differentiation with respect to time and space, respectively. The first term on the right-hand side is the usual wave equation term, and the parameter c is related to the fundamental frequency f_0 of the string by $f_0 = c/2$. (Note that in (1), for simplicity, the spatial variable has been non-dimensionalized, so that c does not have dimensions of velocity.) The second term represents a contribution due to string stiffness, leading to inharmonicity of partials, parameterized by the constant κ ; in the limit as κ becomes large, Eq. (1) represents the behaviour of a vibrating bar. The final two terms model loss: the first, parameterized by $\sigma \geq 0$, describes the global decay rate of the string, and the second, parameterized by $b \geq 0$, approximates frequency-dependent loss characteristic of string vibration.

There are various boundary conditions which can be used to terminate the string equation (1) at either end point $x = 0$ or $x = 1$; two are required at any terminating point. In the case of strings, perhaps most useful are fixed end conditions of the clamped, i.e.,

$$u = u' = 0 \quad (2)$$

or simply supported type, i.e.,

$$u = u'' = 0 \quad (3)$$

Under low-stiffness conditions, the audible difference between the two types of conditions is negligible. If the equation (1) is intended

to represent the behaviour of a bar, then free boundary conditions may be implemented as well.

Eq. (1), though arguably suitable for piano sound synthesis [11], is by no means the most general musically useful string model; various nonlinear models, of varying degrees of complexity, allow rendering of more interesting effects such as pitch glides among others [12], [13], [14]. Any such model could easily be substituted here, with very little effect on the resulting development in this article (particularly with regard to the construction of finite difference schemes, which handle nonlinear distributed systems quite easily).

2.1. Multiple Strings and Excitation

The character of a single piano note is determined by the vibration of several strings, usually very similar in fundamental frequency, and identical in other respects. In the implementation described here, N_s such strings are coupled, each described by an equation of the form

$$\ddot{u}_{(q)} = c_{(q)}^2 u''_{(q)} - \kappa^2 u'''_{(q)} - \sigma \dot{u}_{(q)} + b \ddot{u}_{(q)} + \delta(x_0) \phi_{(q)} \quad (4)$$

where here q is an index which runs from $q = 1$ to $q = N_s$; notice that of the defining parameters, only $c_{(q)}$ has any dependence on q . Also, a set of excitations $\phi_{(q)}$, acting at $x = x_0$ on each string (through the Dirac delta function $\delta(x_0)$) has been introduced. The excitations represent the behaviour of a coupling element, such as a piano hammer, or one of various types of string preparation, which will be discussed shortly. This description of excitation or coupling is a lumped one—it is by no means necessary to make this assumption, but for many musically meaningful cases, the spatial extent of the excitation is smaller than the shortest audible wavelength in the string, for physically reasonable material sound speeds. See Section 4 for more discussion of this point.

3. EXCITATION AND PREPARATION

In this section, the hammer interaction, as well as various interactions with preparing elements are described; in each case, the element interacts with the entire set of strings. Though single elements are described here, it is simple to generalize to the case of connections to many of these objects simultaneously.

3.1. Hammers

A simple, but effective model of a hammer strike upon multiple strings from below [2] is easily described in terms of the displacement $u_H(t)$ of the hammer, and that of an individual string q at the strike location x_0 , $u_{(q)}(x_0, t)$, or rather the difference $w_{(q)}(t) = u_H(t) - u_{(q)}(x_0, t)$:

$$\phi_{(q)}(w_{(q)}) = \begin{cases} \mu \gamma^2 w_{(q)}^\alpha, & w_{(q)} > 0 \\ 0, & w_{(q)} \leq 0 \end{cases}$$

In other words, a nonlinear force is exerted when the hammer and string are in contact, with the nonlinearity controlled by the exponent α (a value between 1 and 3 is generally chosen). The parameter $\mu \geq 0$ represents the relative mass density of the hammer relative to the string, and γ is a parameter which scales the stiffness of the hammer. The hammer position itself is governed by

Newton's laws, i.e.,

$$\ddot{u}_H = -\frac{1}{\mu} \sum_{q=1}^{N_s} \phi_{(q)} \quad (5)$$

The hammer itself must be initialized, typically with an initial position $u_H(0)$ and an initial velocity $\dot{u}_H(0)$.

3.2. Traps

One type of string preparation is to vertically “trap” it, at a given point x_0 , using nonlinear springs. In this case, the functions $\phi_{(q)}$ are given by

$$\phi_{(q)}(u_{(q)}(x_0, t)) = -\gamma^2 (u_{(q)}(x_0, t))^\alpha \quad (6)$$

where here, γ controls the spring stiffness and α is the nonlinearity exponent.

3.3. Rubber Stoppers

In order to model a rubber stopper (such as an eraser) wedged underneath a set of strings, the following model is a useful first approximation. If $u_E(t)$ is the position of the top of the rubber element, then in terms of $w_{(q)}(t) = u_E(t) - u_{(q)}(x_0, t)$, one has, similarly to the case of the hammer,

$$\phi_{(q)}(w_{(q)}) = \begin{cases} \mu \gamma^2 w_{(q)}, & w_{(q)} > 0 \\ 0, & w_{(q)} \leq 0 \end{cases}$$

In contrast to the hammer, however, the rubber stopper is anchored, and its dynamics may be written as

$$\ddot{u}_E = -\sigma_E \dot{u}_E - \gamma^2 \left(u_E + \frac{1}{\mu \gamma^2} \sum_{q=1}^{N_s} \phi_{(q)} \right) \quad (7)$$

where σ_E is a damping constant (typically quite large), and γ , in this case, can be viewed as the fundamental frequency of the rubber element. μ , in (4), is again the mass density ratio of the rubber element to the string.

An even simpler model of such a stopper involves the use of a simple damping term, i.e.,

$$\phi_{(q)}(u_{(q)}(x_0, t)) = -\sigma_E \dot{u}_{(q)}(x_0, t) \quad (8)$$

which may be used in conjunction with the trap discussed above.

3.4. Rattling Elements

A rattling element may be modeled, in the simplest case, as a two point-mass “dumbbell,” of length ϵ , on which a force acts only when one of the two masses is in contact with the string. In this case, the coupling may be framed in terms of the displacement $w_{(q)}(t) = u_R(t) - u_{(q)}(x_0, t)$, where u_R is the vertical midpoint position of the rattling element. The force relationship is given by

$$\phi_{(q)}(w_{(q)}) = \begin{cases} \mu \gamma^2 (w_{(q)} - \epsilon/2), & w_{(q)} > \epsilon/2 \\ 0, & -\epsilon/2 \leq w_{(q)} \leq \epsilon/2 \\ \mu \gamma^2 (w_{(q)} + \epsilon/2), & w_{(q)} < -\epsilon/2 \end{cases}$$

The contact interaction here is piecewise linear, but could easily be made more strongly nonlinear through the introduction of an

exponent, as in the case of the hammer. The rattle position is again governed by Newton's laws, i.e.,

$$\ddot{u}_R = -\frac{1}{\mu} \sum_{q=1}^{N_s} \phi_{(q)} - g \quad (9)$$

where here, gravitational acceleration g has been introduced, as it will have particular importance in determining the collision dynamics of the rattle. Again, γ represents the fundamental frequency of the rattling elements, and μ in (4) the relative linear mass density of the rattle to the string.

4. A FINITE DIFFERENCE MODEL

For one-dimensional systems such as strings, finite difference schemes are a good choice of numerical method; though not as efficient as digital waveguides [7], they are simple to program, much more general, and allow a complete transparency of the entire state of the simulated system, which is especially useful if one is interested in, say, dynamically changing the position of an excitation or preparing element. As mentioned earlier, they extend easily to the case of fully nonlinear strings.

Considering the q th string, Eq. (4) may be discretized, at an interior point (to be defined presently) as

$$u_{(q),i}^{n+1} = \sum_{l=-2}^2 \alpha_{(q),|l|} u_{(q),i+l}^n + \sum_{l=-1}^1 \beta_{(q),|l|} u_{(q),i+l}^{n-1} + \frac{k^2}{h_{(q)}} \delta_{i_0} \mu \gamma^2 \phi_{(q)}^n \quad (10)$$

where here, $u_{(q),i}^n$ is a grid function representing an approximation to $u_{(q)}(x, t)$ at $x = ih_{(q)}$, and $t = nk$, and $\phi_{(q)}^n$ approximates $\phi_{(q)}(t)$ at $t = nk$, and at grid location $i = i_0$ corresponding to $x_0 = ih_{(q)}$ (hence the use of the Kronecker delta). $h_{(q)}$ is the spacing between adjacent grid points, and k is the time step ($1/k$ is the sample rate, generally set a priori). The scheme coefficients $\alpha_{(q),l}$ and $\beta_{(q),l}$ are given by

$$\begin{aligned} \alpha_{(q),0} &= \frac{2 - \frac{2c_{(q)}^2 k^2}{h_{(q)}^2} - \frac{6\kappa^2 k^2}{h_{(q)}^4} - \frac{2bk}{h_{(q)}^2}}{1 + \sigma k / 2} \\ \alpha_{(q),1} &= \frac{\frac{c_{(q)}^2 k^2}{h_{(q)}^2} + \frac{4\kappa^2 k^2}{h_{(q)}^4} + \frac{bk}{h_{(q)}^2}}{1 + \sigma k / 2} \\ \alpha_{(q),2} &= -\frac{\kappa^2 k^2}{h_{(q)}^4 (1 + \sigma k / 2)} \\ \beta_{(q),0} &= \frac{\sigma k / 2 - 1 + \frac{2bk}{h_{(q)}^2}}{1 + \sigma k / 2} \\ \beta_{(q),1} &= \frac{-\frac{bk}{h_{(q)}^2}}{1 + \sigma k / 2} \end{aligned}$$

A necessary stability condition, which follows from von Neumann (Fourier) analysis of scheme (10) under zero input conditions [15], is

$$h_{(q)} \geq h_{(q),min}$$

with

$$h_{(q),min} = \sqrt{\frac{c_{(q)}^2 k^2 + 2bk + \sqrt{(c_{(q)}^2 k^2 + 2bk)^2 + 16\kappa^2 k^2}}{2}}$$

In other words, for a given sample rate and set of material parameters, a minimum spacing between adjacent points must be respected. This spacing relates to the highest audible wavelength in the string. A slight additional complication comes from the fact that the scheme for each string will possess its own separate stability condition, giving rise, potentially, to different grid spacings on each string, which is inconvenient from a programming standpoint; it is useful to make use of a global spacing h over all strings, bounded by

$$h \geq \max_q h_{(q),min} \quad (11)$$

Use of such a global spacing is advisable only when the fundamental frequencies of the strings are relatively close; otherwise, one may be far from the ideal bound in at least some of the strings, and numerical dispersion [15] will play a significant role, leading to a numerical inharmonicity in partials.

Eq. (10) is implemented as a two-step recursion; it may be initialized at zero. It is also clear that any given grid function $u_{(q),i}^n$, which is defined over a set of grid points indexed by $i = 0, \dots, N$, where $N = 1/h$, will require access to values outside the problem domain at points near the boundary, according to (10). For fixed termination, as is the case presented here, one may always set $u_{(q),0}^n = u_{(q),N}^n = 0$; in other words, the terminal grid points are permanently set to zero, and need not be computed, satisfying the first of conditions (2) or (3). To satisfy the second of conditions (2) at, say, the left end of the string, one may set $u_{-1}^n = 0$. To satisfy the second of conditions (3), one may set $u_{-1}^n = -u_1^n$. Many other methods of setting the boundary conditions are feasible, and details may be found elsewhere; these particular settings are simple, and provably numerically stable.

4.1. Updating for the Connecting Elements

The various connecting elements described in Section 3 are also, for the most part, described by differential equations, and require numerical integration. Though in the case of nonlinear elements, there are obviously many ways of performing this integration, simple methods are presented here. In the case of the hammer, (5) may be updated as

$$u_H^{n+1} = 2u_H^n - u_H^{n-1} - \frac{k^2}{\mu} \sum_{q=1}^{N_s} \phi_{(q)}^n$$

The rattling element, defined by (9) may be dealt with similarly as

$$u_R^{n+1} = 2u_R^n - u_R^{n-1} - \frac{k^2}{\mu} \sum_{q=1}^{N_s} \phi_{(q)}^n - k^2 g$$

For the rubber stopper, one may use

$$\begin{aligned} u_E^{n+1} &= 2u_E^n - u_E^{n-1} - \sigma_E k (u_E^n - u_E^{n-1}) \\ &\quad - \gamma^2 k^2 \left(u_E^n + \sum_{q=1}^{N_s} \phi_{(q)}^n \right) \end{aligned}$$

The trapping element requires no additional state, and to approximate (6), one may simply use

$$\phi_{(q)}(u_{(q),i_0}^n) = -\gamma^2(u_{(q),i_0}^n)^\alpha$$

If loss is introduced, one may use, as an approximation to (8),

$$\phi_{(q)}(u_{(q),i_0}^n) = -\frac{\sigma_E}{k}(u_{(q),i_0}^n - u_{(q),i_0}^{n-1})$$

5. IMPLEMENTATION DETAILS

5.1. Operation Count

The bulk of the calculation in this prepared piano simulator is carried out in performing the update of the string state; hammers and various connections are of negligible computational cost. For a single string, of fundamental frequency f_0 , and under low inharmonicity conditions, at sample rate f_s the number of multiplies per second is approximately $2f_s^2/f_0$; the computational cost obviously scales inversely with the fundamental frequency.

5.2. csound Implementation

This algorithm was prototyped in the Matlab programming language, but the aim is to make the sound available to composers. Ideally such an implementation should be playable in real time, or at least close to real time, and from the point of view of practicality there needs to be some attempt to minimise the memory use. For composers, the flexibility of the operation is also significant; csound [10] would appear to be a good choice of programming environment.

The internal operations naturally rely on three arrays to represent the values $u_{(q),i}^{n-1}$, $u_{(q),i}^n$ and $u_{(q),i}^{n+1}$ for each string. At each time step these are cycled by pointer assignment, taking care to correct for the boundary conditions. Similar arrangements are taken to maintain the state of the preparations.

In Csound, as in all MusicV descended systems, the operations are split between an initialisation method and then a perform method that creates a short section of audio. Much of the speed of csound comes from this batching of operations, with its attendant good cache behaviour and the less frequent adjustment of parameters. In the case of the prepared piano there are twelve fundamental parameters covering basics like fundamental frequency and the number of strings, as well as stiffness, loss parameters and the characteristics of the hammer, most of which cannot reasonably be adjusted after the initialisation (although there is a future action to investigate this more closely). The harder problem is how to specify the various preparations. Our implementation uses two f-tables, one to define the characteristics of rubber preparations and one to do likewise for the rattles. Either of these can of course be omitted.

The other design question is how to handle multiple hammer strokes. Our implementation does this by having one stroke for each call, but it is possible to skip the initialisation phase, and so keep the previous state of the system. There is also an issue on control of the amplitude of the output. As in many physical models the parameters determine the inputs rather than the resulting amplitude. The code performs some scaling based on experience.

```
<CsoundSynthesizer>
<CsInstruments>
nchnls = 1;
instr 1
;; fund NS detune stiffness decay loss (bndry)
;; (hammer) scan prep
aa prepiano 60, 3, 10, p4, 3, 0.002, 2, 2, 1,
5000, -0.01, p5, p6, 0, 1, 2
out aa
endin
</CsInstruments>
<CsScore>
f1 0 8 2 1 0.6 10 100 0.001 ;; 1 rattle
f2 0 8 2 1 0.7 50 500 1000 ;; and 1 rubber
i1 0.0 0.5 1 0.09 20
i1 0.5 . -1 0.09 40 ;; 2nd strike keeping state
i1 1.0 . -1 0.09 60
i1 1.5 . -1 0.09 80
i1 2.0 1.8 -1 0.09 100 ;; last strike until silence
</CsScore>
</CsoundSynthesizer>
```

Figure 1: Simple csound example.

5.3. Performance

The csound opcode has been tested *in situ* in two simple cases, with one rattle and one rubber stopper, and with no preparations. Clearly the processor speed is important when we are concerned with how close to real time the code runs. The tests were run on two widely differing systems both running Csound5.01; a 1.4GHz Centrino laptop with Linux with floating-point samples, and an AMD Athlon64 3400+ (2.4GHz) with Linux, and double-precision samples. The actual test program is shown in Figure 1, where a very low frequency string ($f_0 = 60$ Hz) has been chosen, as a “worst case.”

Machine	Time	% real	Time	% real
Laptop	8.143	47%	8.239	46%
Workstation	0.591	643%	0.595	639%

The preliminary timings suggest that the cost of the preparations is small. The workstation is clearly showing usable performance while on the laptop the speed is too slow for real time, but not too bad for some uses. These results are preliminary, and there may be opportunities for further optimisation, such as moving some calculations from the sample-rate loop to the control rate.

6. NUMERICAL RESULTS

In this section, several illustrative plots of numerical output from the string model, under various types of preparation, are presented; all were created at the audio sample rate of 44.1 kHz. The effect of brightening with strike velocity for a nonlinear hammer of the type presented here has been discussed elsewhere [2]. Considering the case of a rattling element, in Figure 2 are shown output waveforms generated, as well as spectrograms, in the case of a struck string at fundamental frequency 600 Hz, both with and without a rattling element present. Though it is difficult to show the precise time-dependent effects of the rattling element, at the crudest level, one sees a considerable disruption of the output waveform, as well as a substantial presence of high frequencies in the case of the rattle.

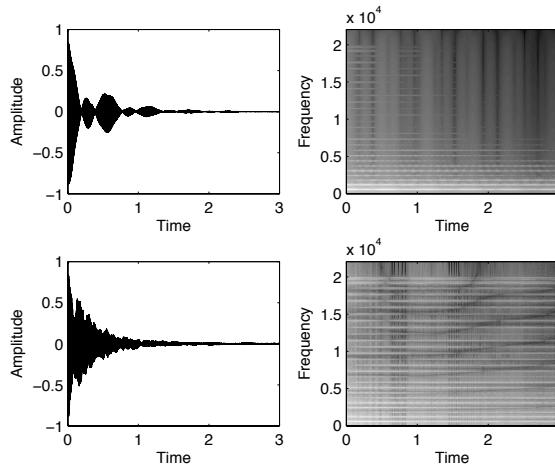


Figure 2: Top: output waveform and spectrogram, for a struck piano tone at 600 Hz. Bottom: output waveform and spectrogram for a struck piano tone at 600 Hz, including the effects of a rattling element placed 0.3 of the way along the string from the left end.

Consider the case of a trapping element, in the case of a string of fundamental frequency 200 Hz; a spectrogram of the output waveform is given in Figure 3. The string is struck successively with blows of initial velocity 5, 10, 50 and 100 m/s. Notice in particular the increase in high frequency energy with strike velocity (due to the effect of both the nonlinear hammer interaction, and the trapping element), as well as the rather complex effects of modulation visible in the upper harmonics of the tone under high amplitude striking velocity.

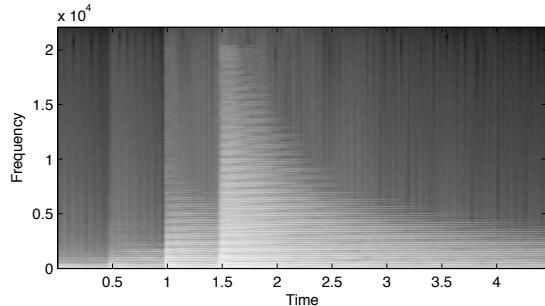


Figure 3: Top: output spectrogram, for a struck piano tone at 200 Hz, with a trap placed 0.3 of the way along the string from the left end, under strikes of increasing amplitude.

Under the same conditions as the example above, but with the addition of a rubber stopper (in this case at the same location as the trapping element), the output sounds become, predictably, shorter and more percussive (see Figure 4) with the additional feature that, under high amplitude conditions, a new, higher fundamental frequency emerges. The effect of a prepared piano sounding at a pitch different from that of the note struck is, of course, a well-known effect.

It is rather difficult to give the full flavor of the sound examples produced by presenting plots of waveforms and spectrograms; the

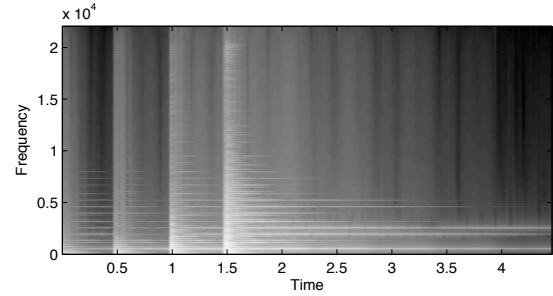


Figure 4: Top: output spectrogram, for a struck piano tone at 200 Hz, with a trap accompanied by a rubber stopper placed 0.3 of the way along the string from the left end, under strikes of increasing amplitude.

reader is directed to a series of sound examples, created in Matlab, which are available on the internet [16].

7. CONCLUSION

In this short article, a brute force numerical simulation of a prepared piano has been presented. As noted in Section 5, though more computationally demanding than techniques such as digital waveguides, these methods are already capable of real time performance.

One significant problem with the algorithm as presented here has to do with numerical stability concerns, with regard to the nonlinear interactions with the hammer and various preparing elements. The condition (11), which is derived using Fourier principles applied to the string scheme under linear conditions is merely necessary. One may develop energy-conserving or dissipating algorithms to deal with the various nonlinearities (indeed, in all cases, the model, including preparing elements, may be shown to be strictly dissipative) [17], but such algorithms may lead to implicit schemes, necessarily requiring the use of iterative methods, without a guarantee of convergence. Another approach would be to make use of so-called symplectic methods [18], but this approach is not strictly applicable to problems involving dissipation. The question is open as to how to construct robust (i.e. provably numerically stable) numerical methods for complex nonlinear systems such as that presented here.

It is worth noting that although the algorithm can be used to simulate the behaviour of a piano-like instrument, the model equation, as defined by (1) is quite general, and can be used, without modification to simulate the behaviour of various other percussion instruments, including bar-based instruments such as xylophones or marimbas, again under prepared conditions. One element which is missing here is a physical model of a soundboard, which can be easily modeled through the use of a finite difference plate model, connected to the string models mentioned here. Preparation of such a soundboard itself is, of course, a desirable option.

8. ACKNOWLEDGEMENTS

This work was supported in part by the Engineering and Physical Sciences Research Council UK, under grant number C007328/1.

9. REFERENCES

- [1] J. Cage and D. Charles, *For The Birds: John Cage in Conversation with Daniel Charles*. Marion Boyers, 1981.
- [2] 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. 1112–8, Feb. 1994.
- [3] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith III, “The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides,” *J. Acoust. Soc. Am.*, vol. 114, no. 2, pp. 1095–1107, 2003.
- [4] P. Ruiz, “A technique for simulating the vibrations of strings with a digital computer,” Master’s thesis, University of Illinois, 1969.
- [5] L. Hiller and P. Ruiz, “Synthesizing musical sounds by solving the wave equation for vibrating objects,” *J. Audio Eng. Soc.*, vol. 19, pp. 462–72, 542–51, 1971.
- [6] C. Cadoz, A. Luciani, and J.-L. Florens, “CORDIS-ANIMA: a modeling and simulation system for sound and image synthesis – The General formalism,” *Computer Music J.*, vol. 17, no. 1, pp. 19–29, 1993.
- [7] J. O. Smith III, *Physical Audio Signal Processing*. Stanford, CA: draft version, 2004, [Online] <http://ccrma.stanford.edu/jos/~pasp04/>.
- [8] T. Tolonen, V. Välimäki, and M. Karjalainen, “Modeling of tension modulation nonlinearity in plucked strings,” *IEEE Trans. Speech and Audio Proc.*, vol. 8, pp. 300–310, May 2000.
- [9] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, “Splitting the unit delay: Tools for fractional delay filter de-sign,” *IEEE Sig. Proc. Magazine*, vol. 13, no. 1, pp. 30–60, Jan. 1996.
- [10] R. Boulanger, *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming*. Cambridge, Massachusetts, USA: MIT Press, 2000.
- [11] 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. 2268–2278, Apr. 2005.
- [12] C. Vallette, “The mechanics of vibrating strings,” in *Mechanics of Musical Instruments*, A. Hirschberg, J. Kergomard, and G. Weinreich, Eds. New York: Springer, 1995, pp. 116–183.
- [13] S. Bilbao and J. O. Smith III, “Energy conserving finite difference schemes for nonlinear strings,” *Acta Acustica united with Acustica*, vol. 91, pp. 299–311, 2005.
- [14] S. Bilbao, “Conservative numerical methods for nonlinear strings,” *J. Acoust. Soc. Am.*, vol. 118, no. 5, pp. 3316–3327, 2005.
- [15] J. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Pacific Grove, Calif.: Wadsworth and Brooks/Cole Advanced Books and Software, 1989.
- [16] S. Bilbao, Retrieved June 29th, 2006, [Online] Plate sound synthesis examples at http://www.music.ed.ac.uk/Contacts/DrStefanBilbao_soundExamples.htm.
- [17] ——, “Robust physical modeling sound synthesis for nonlinear systems: Direct numerical simulation and the energy method,” 2006, under review, *IEEE Signal Processing Magazine*. Invited article.
- [18] J. Sanz-Serna, “Symplectic integrators for hamiltonian problems: An overview,” *Acta Numerica*, vol. 1, pp. 243–286, 1991.

DIGITAL SYNTHESIS MODELS OF CLARINET-LIKE INSTRUMENTS INCLUDING NONLINEAR LOSSES IN THE RESONATOR

Philippe Guillemain, Jonathan Terroir

CNRS – Laboratoire de Mécanique et d’Acoustique
31 ch. J. Aiguier, 13402 Marseille cedex 20, France
{guillem|terroir}@lma.cnrs-mrs.fr

ABSTRACT

This paper presents a real-time algorithm for the synthesis of reed instruments, taking into account nonlinear losses at the first open tonehole. The physical model on which the synthesis model relies on is based on the experimental works of Dalmont et al. who have shown that for high pressure levels within the bore, an air jet obeying the Bernoulli flow model, hence acting as a nonlinear resistance, is created at the open end of the bore. We study the effect of these additional losses on the response of the bore to an acoustic flow impulse at different levels and on the self oscillations. We show that at low frequencies, these nonlinear losses are of the same order of magnitude than the viscothermal linear losses and modify the functioning of the whole instrument. For real-time synthesis purposes, a simplified algorithm is proposed and compared to the more accurate model.

1. INTRODUCTION

Many works have been devoted to the physical study and the modeling of toneholes in woodwind instruments (see e.g. [1] and [2]) and digital resonator models including toneholes (see e.g. [3] and [4]) have been proposed for the real-time simulation of these instruments. These synthesis models describe toneholes, or more generally the “open termination” of an instrument as a passive linear element of the resonator which can then be fully characterized by its input impedance or by the linear relationship between the wave variables. Experimental studies [5] have shown that for high levels of acoustic pressure and flow within the bore of the instrument, nonlinear effects may appear, mainly due to the formation of a dissipative air jet at the open end, comparable to the one responsible of the birth of self oscillations at the mouthpiece level. While impedance measurements are done at low acoustic pressure and flow levels, pressure levels within the bore are very high under normal playing conditions and the nonlinear behavior of the termination can probably no longer be ignored.

In this paper, in order to quantify the relative weight of these nonlinear additional losses with respect to the classical, linear, viscothermal losses, we propose a real-time oriented synthesis model of clarinet-like instruments taking into account a nonlinear termination. The physical model is based on the experimental works of Dalmont [5] and the synthesis model is a modified version of the model presented in [6].

The paper is organized as follows. The physical model of a nonlinear dissipative termination is first presented. A time domain digital model, and a simplified version are then proposed. It is then shown that the nonlinear termination has a significant role on the response of the bore. Finally, the self oscillations of a full instrument model are studied and comparison is made between the clas-

sical linear model and the two nonlinear models proposed, showing that a very simple algorithm is able to reproduce the behaviors observed on a more complicated one. The nonlinear termination is shown to modify significantly the functioning and the timbre of the instrument.

2. BORE MODEL

The physical model on which is based the synthesis model is first summarized. Two digital formulations are presented and the effect of the nonlinear termination on the bore response is studied on the most accurate one.

2.1. Physical model

A perfectly cylindrical resonator is considered. One assumes linear propagation within the bore.

The wavenumber is denoted $k(\omega)$ and includes propagation delay, dispersion and dissipation corresponding to viscothermal effects. It is classically given by [7] :

$$k(\omega) = \frac{\omega}{c} - \frac{j^{3/2}}{2} \eta c \omega^{1/2}$$

$Z_c = \frac{\rho c}{S}$ is the characteristic impedance of the bore, $S = \pi R^2$ is the input surface of the bore and η is given by:

$$\eta = \frac{2}{R c^{3/2}} \left(\sqrt{l_v} + \left(\frac{c_p}{c_v} - 1 \right) \sqrt{l_t} \right)$$

R is the radius of the bore: $R = 7.10^{-3}$ in the clarinet case. Typical values of the physical constants, in mKs units, are: $c = 340$, $l_v = 4.10^{-8}$, $l_t = 5.6.10^{-8}$, $C_p/C_v = 1.4$.

2.1.1. Linear bore model

If the radiation impedance at the open end is taken into account only as a length correction, the acoustic pressure $p_s(t)$ at the end of the bore vanishes, and the input impedance of the bore, which is the ratio between the Fourier transforms (denoted with capital letters) of the acoustic pressure $p_e(t)$ and acoustic flow $u_e(t)$ at the mouthpiece level is classically expressed by:

$$\frac{P_e(\omega)}{U_e(\omega)} = j Z_c \tan(k(\omega)L) \quad (1)$$

where L is the total length of the bore, including the length correction corresponding to the radiation impedance.

2.1.2. Nonlinear bore model

When nonlinear losses at the open end are considered, the input impedance is not well defined. Indeed, according to Dalmont [5], the nonlinear termination of the open end of the bore is defined in the time domain by:

$$\frac{p_s(t)}{u_s(t)} = (0.6 \pm 0.1) \frac{|v_s(t)|}{c} Z_{ct} \quad (2)$$

where $v_s(t)$ is the mean acoustic velocity, averaged on the radiating surface S_t and $Z_{ct} = \rho c / S_t$ is the characteristic impedance corresponding to the radius r_t of the open end. The coefficient (0.6 ± 0.1) has been determined experimentally.

It is worth noting that equation (2) corresponds to a classical Bernoulli flow model since it can be rewritten as $p_s(t) = \frac{1}{2} \rho v_s^2(t)$ by considering the $(-)$ sign in (0.6 ± 0.1) and assuming that after the dissipation of the jet created by the hole, the pressure outside the resonator is zero. Hence, for the sake of simplicity and for synthesis purposes, we shall assume that the surface S_t at the open end corresponds to the surface of the first open tonehole and can be smaller than the surface S of the resonator. Such an approximation, which considers that the equivalent length of a resonator corresponds to the distance between the mouthpiece and the first open tonehole and ignore the resonator characteristics after the first open tonehole seems relevant in the context of this study.

From equation (2), one obtains in terms of pressure and flow variables:

$$p_s(t) = \alpha |u_s(t)| u_s(t) \quad (3)$$

where $\alpha = \frac{0.6 Z_{ct}}{c S_t}$. Equation (3) shows that the open end acts as a nonlinear resistance which is proportional to the absolute value of the acoustic flow.

In order to take into account these nonlinear losses in the bore model, we consider the classical transmission line equations between the input and the output of the bore:

$$\begin{cases} P_e - Z_c U_e = e^{-jk(\omega)L} (P_s - Z_c U_s) \\ P_s + Z_c U_s = e^{-jk(\omega)L} (P_e + Z_c U_e) \end{cases} \quad (4)$$

2.2. Digital model

In order to propose a time domain digital formulation of the system of equations (4), the propagation described by $e^{-jk(\omega)L}$ is first approximated by:

$$e^{-jk(\omega)L} \approx \frac{\beta_0}{1 - \alpha_1 z^{-1}} z^{-D} \quad (5)$$

where $z = e^{\frac{j\omega}{F_e}}$, F_e is the sampling frequency and $D = E \left(\frac{F_e L}{c} \right)$. The coefficients β_0 and α_1 are calculated according to [6] and correspond to the propagation over a length L .

In order to be able to collect versions of p_s and u_s delayed by D , that are required for the computation of p_e with the first equation of system (4), the second equation of system (4) is written as:

$$(P_s + Z_c U_s) e^{-jk(\omega)L} = e^{-2jk(\omega)L} (P_e + Z_c U_e) \quad (6)$$

which yields:

$$\frac{\beta_0}{1 - \alpha_1 z^{-1}} z^{-D} (P_s + Z_c U_s) = \left(\frac{\beta_0}{1 - \alpha_1 z^{-1}} z^{-D} \right)^2 (P_e + Z_c U_e) \quad (7)$$

thus:

$$(P_s + Z_c U_s) (1 - \alpha_1 z^{-1}) z^{-D} = \beta_0 (P_e + Z_c U_e) z^{-2D} \quad (8)$$

Using the same approximation of propagation and losses, the first equation of system (4) becomes:

$$P_e - Z_c U_e = \frac{\beta_0}{1 - \alpha_1 z^{-1}} z^{-D} (P_s - Z_c U_s) \quad (9)$$

yielding:

$$(P_e - Z_c U_e) (1 - \alpha_1 z^{-1}) = \beta_0 (P_s - Z_c U_s) z^{-D} \quad (10)$$

In the time domain, the digital equivalent of system (4) becomes:

$$\begin{cases} p_s(n - D) + Z_c u_s(n - D) = \alpha_1 [p_s(n - D - 1) \\ + Z_c u_s(n - D - 1)] + \beta_0 [p_e(n - 2D) + Z_c u_e(n - 2D)] \\ p_e(n) = Z_c u_e(n) + \alpha_1 [p_e(n - 1) - Z_c u_e(n - 1)] \\ + \beta_0 [p_s(n - D) - Z_c u_s(n - D)] \end{cases} \quad (11)$$

In order to integrate this resonator model in a full instrument model including a reed and a nonlinear coupling at the entrance of the resonator, we use the dimensionless variables defined by [8]: $\tilde{p}_{e,s} = \frac{p_{e,s}}{p_M}$ and $\tilde{u}_{e,s} = Z_c \frac{u_{e,s}}{p_M}$ where p_M is the static beating-reed pressure.

This change of variables leads to:

$$\begin{cases} \tilde{p}_s(n - D) + \tilde{u}_s(n - D) = \alpha_1 [\tilde{p}_s(n - D - 1) \\ + \tilde{u}_s(n - D - 1)] + \beta_0 [\tilde{p}_e(n - 2D) + \tilde{u}_e(n - 2D)] \\ \tilde{p}_e(n) = \tilde{u}_e(n) + \alpha_1 [\tilde{p}_e(n - 1) - \tilde{u}_e(n - 1)] \\ + \beta_0 [\tilde{p}_s(n - D) - \tilde{u}_s(n - D)] \end{cases} \quad (12)$$

In the same way, a dimensionless coefficient corresponding to the nonlinear losses α is defined: $\tilde{\alpha} = \frac{(0.6 \pm 0.1) p_M}{\rho c^2} \frac{S^2}{S_t^2}$.

With this change of variables, the instantaneous nonlinearity at the open end of the bore corresponding to equation (3) becomes:

$$\tilde{p}_s(n - D) = \tilde{\alpha} \operatorname{sign}(\tilde{u}_s(n - D)) \tilde{u}_s(n - D)^2 \quad (13)$$

From the first equation of system (12) and equation (13), the value of $\tilde{u}_s(n - D)$ can be obtained analytically by solving the equation:

$$\tilde{\alpha} \operatorname{sign}(\tilde{u}_s(n - D)) \tilde{u}_s^2(n - D) + \tilde{u}_s(n - D) - \tilde{V}_s = 0 \quad (14)$$

where:

$$\begin{aligned} \tilde{V}_s &= \alpha_1 (\tilde{p}_s(n - D - 1) + \tilde{u}_s(n - D - 1)) \\ &+ \beta_0 ((\tilde{p}_e(n - 2D) + \tilde{u}_e(n - 2D)) \end{aligned} \quad (15)$$

By considering successively the cases $\tilde{u}_s(n - D) \geq 0$ and $\tilde{u}_s(n - D) < 0$, since $\operatorname{sign}(\tilde{u}_s(n - D)) = \operatorname{sign}(\tilde{p}_s(n - D)) = \operatorname{sign}(\tilde{V}_s)$, $\tilde{u}_s(n - D)$ is finally given by:

$$\tilde{u}_s(n - D) = \operatorname{sign}(\tilde{V}_s) \frac{-1 + \sqrt{1 + 4 \operatorname{sign}(\tilde{V}_s) \tilde{\alpha} \tilde{V}_s}}{2 \tilde{\alpha}} \quad (16)$$

At any time sample n , the process leading to the calculation of the output $\tilde{p}_e(n)$ from any arbitrary input $\tilde{u}_e(n)$ consists in the sequential calculation of:

- \tilde{V}_s with equation (15)
- $\tilde{u}_s(n - D)$ with equation (16)
- $\tilde{p}_s(n - D)$ with equation (13)
- $\tilde{p}_e(n)$ with the second equation of system (12)

It is worth noting that the case of a linear termination corresponds to assume that $\forall n, \tilde{p}_s(n) = 0$ and $\tilde{u}_s(n - D) = \tilde{V}_s$

2.3. Simplified digital model

For the sake of simplicity and computational efficiency, we seek a digital resonator model that uses as few physical variables as possible and requires as less computation power as possible, while keeping the most important features of a more sophisticated model. As a first simplification, we seek an approximation that allows to remove the variables \tilde{p}_s and \tilde{u}_s of the whole scheme.

For that purpose, equations (7) and (9) modelling propagation in both directions are written in the following way:

$$\begin{cases} \frac{\beta_0}{1-\alpha_1 z^{-1}} z^{-D} (P_s + Z_c U_s) = \frac{b_0}{1-a_1 z^{-1}} z^{-2D} (P_e + Z_c U_e) \\ P_e - Z_c U_e = \frac{\beta_0}{1-\alpha_1 z^{-1}} z^{-D} (P_s - Z_c U_s) \end{cases} \quad (17)$$

where $\frac{b_0}{1-a_1 z^{-1}} z^{-2D}$ is a first order digital filter modelling the propagation over a length $2L$. It replaces the filter $\left(\frac{\beta_0 z^{-D}}{1-\alpha_1 z^{-1}}\right)^2$ appearing in equation (7) modelling the propagation over a length $2L$ as the product of two filters corresponding to the propagation over a length L .

In order to remove the variables p_s and u_s delayed of $(D+1)$ in the first equation of system (17), we assume $\alpha_1 = a_1$. This means that the rate of frequency variations of the losses are assumed to be similar for the propagation over a length L (determined by the value of α_1) and over a length $2L$ (determined by the value of a_1). By counterpart, the coefficient β_0 is changed into a coefficient β_{0ap} in order to keep the height of the first impedance peak the frequency of which is $\frac{c}{4L}$, which requires:

$$\left| \frac{\beta_{0ap}}{1-a_1 \tilde{z}^{-1}} \right|^2 = \left| \frac{b_0}{1-a_1 \tilde{z}^{-1}} \right| \quad (18)$$

where $\tilde{z} = e^{j\tilde{\omega}}$ and $\tilde{\omega} = \frac{\pi c}{2L}$, which yields:

$$\beta_{0ap} = \sqrt{b_0 \sqrt{1 - 2a_1 \cos(\tilde{\omega}) + a_1^2}} \quad (19)$$

With these modifications and dimensionless variables, the system of equations (17) becomes:

$$\begin{cases} b_0(\tilde{p}_e(n - 2D) + \tilde{u}_e(n - 2D)) = \\ \beta_{0ap}(\tilde{p}_s(n - D) + \tilde{u}_s(n - D)) \\ \tilde{p}_e(n) = \tilde{u}_e(n) + a_1(\tilde{p}_e(n - 1) - \tilde{u}_e(n - 1)) \\ + \beta_{0ap}(\tilde{p}_s(n - D) - \tilde{u}_s(n - D)) \end{cases} \quad (20)$$

Using the same calculation as the one of the previous subsection, the first equation of system (20) provides analytically the value of $\tilde{u}_s(n - D)$:

$$\tilde{u}_s(n - D) = \text{sign}(\tilde{V}_s) \frac{-\beta_{0ap} + \sqrt{\beta_{0ap}^2 + 4 \text{sign}(\tilde{V}_s) \beta_{0ap} \tilde{\alpha} \tilde{V}_s}}{2\beta_{0ap} \tilde{\alpha}} \quad (21)$$

where \tilde{V}_s is defined by:

$$\tilde{V}_s = b_0(\tilde{p}_e(n - 2D) + \tilde{u}_e(n - 2D)) \quad (22)$$

As a second approximation, the nonlinear losses coefficient $\tilde{\alpha}$ is assumed to remain small. This allows a second order series expansion of equation (21) which leads to:

$$\tilde{u}_s(n - D) \simeq \frac{\tilde{V}_s}{\beta_{0ap}} - \tilde{\alpha} \text{ sign}(\tilde{V}_s) \frac{\tilde{V}_s^2}{\beta_{0ap}^2} \quad (23)$$

This finally yields the following computation scheme expressing $\tilde{p}_e(n)$ with respect to $\tilde{u}_e(n)$, $\tilde{p}_e(n - 2D)$ and $\tilde{u}_e(n - 2D)$:

$$\begin{cases} \tilde{V}_s = b_0(\tilde{p}_e(n - 2D) + \tilde{u}_e(n - 2D)) \\ \tilde{V} = a_1[\tilde{p}_e(n - 1) - \tilde{u}_e(n - 1)] \\ - \tilde{V}_s + 2 \text{ sign}(\tilde{V}_s) \frac{\tilde{\alpha}}{\beta_{0ap}} \tilde{V}_s^2 \\ \tilde{p}_e(n) = \tilde{u}_e(n) + \tilde{V} \end{cases} \quad (24)$$

2.4. Role of the nonlinear termination on the bore response

In order to study the role played by the nonlinear termination, we consider a transient flow excitation of the form: $\tilde{u}_e(t) = \lambda \delta(t)$ and computes with the most accurate nonlinear model $\tilde{p}_e(n)$ for several values of λ , as well as the ratio of the Fourier transforms of \tilde{p}_e and \tilde{u}_e . For these examples, the output radius r_t is assumed to be smaller than the bore input radius R : $r_t = \frac{R}{3}$.

Figures (1) and (2) show respectively the first two reflections of the impulse response and the ratio of the Fourier transforms of the dimensionless pressure and flow: $Z_{eq}(\omega) = \frac{\tilde{P}_e(\omega)}{\tilde{U}_e(\omega)}$ in the linear case and in the nonlinear case, with $\lambda = 10^{-5}$. With such a small value of λ , the effects of the nonlinear termination are not significant, and $Z_{eq}(\omega)$ corresponds to the input impedance of the bore (linear case).

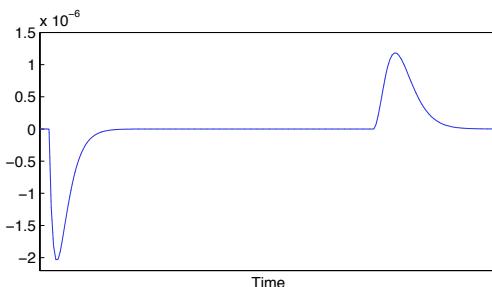


Figure 1: First two reflections of the impulse response. $\lambda = 10^{-5}$. Solid line : nonlinear case. Dotted line: linear case. Nonlinear and linear cases are exactly superimposed.

Figures (3) and (4) show the same computations for $\lambda = 0.5$. The effects of the nonlinear termination are clearly visible. In particular, the damping of the first reflection is much more important than in the linear case and the peaks of $Z_{eq}(\omega)$ are smaller than in the linear case. It is worth noting that opposite to the case of linear losses (taken into account by a radiation impedance) that become significant at high frequencies, the nonlinear losses are significant at low frequencies. This suggests that, depending on the flow level in the mouthpiece, the nonlinear termination may modify the functioning of the whole instrument.

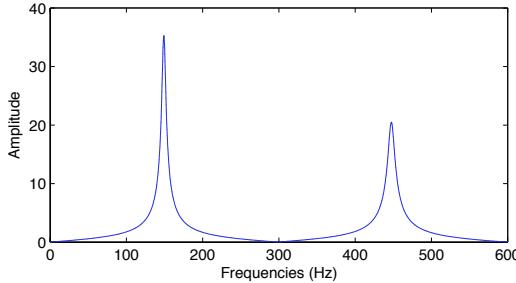


Figure 2: Fourier transform of the response to a Dirac excitation. $\lambda = 10^{-5}$. Solid line : nonlinear case. Dotted line: linear case. Nonlinear and linear cases are exactly superimposed.

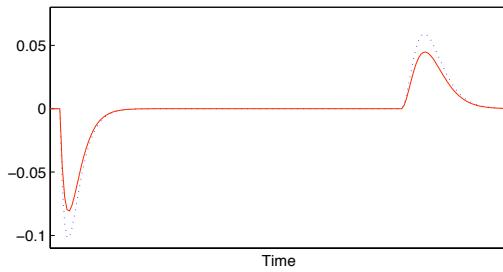


Figure 3: First two reflections of the impulse response. $\lambda = 0.5$. Solid line : nonlinear case. Dotted line: linear case.

3. FULL INSTRUMENT MODEL

In this section, the modifications of the functioning of the instrument induced by the presence of the nonlinear termination are studied. For that purpose, we first recall the classical physical model of the nonlinear coupling between the reed and the bore.

3.1. Physical model

The physical model describes first the dimensionless displacement $x(t)$ of the reed with respect to its equilibrium point by:

$$\frac{1}{\omega_r^2} \frac{d^2x(t)}{dt^2} + \frac{q_r}{\omega_r} \frac{dx(t)}{dt} + x(t) = \tilde{p}_e(t) \quad (25)$$

where $\omega_r = 2\pi f_r$ and q_r^{-1} are respectively the circular frequency and the quality factor of the reed.

The opening of the reed channel $S_r(t)$ is expressed from the reed displacement by:

$$S_r(t) = \Theta(1 - \gamma + x(t)) \times \zeta(1 - \gamma + x(t))$$

where Θ denotes the Heaviside function. Its role is to keep the opening of the reed channel positive by cancelling it when $1 - \gamma + x(t) < 0$. The parameter ζ characterizes the whole embouchure and is proportional to the square root of the reed position at equilibrium H . The dimensionless parameter γ is the ratio between the blowing pressure p_m and the static beating reed pressure p_M defined by: $p_M = \mu_r H \omega_r^2$, where μ_r is the mass per unit-surface of the reed.

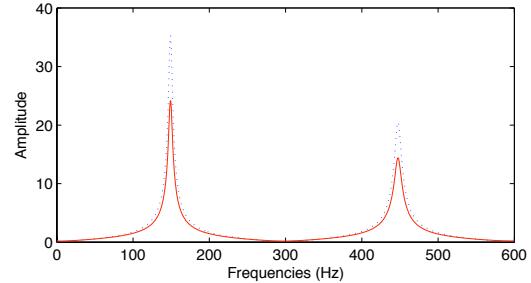


Figure 4: Fourier transform of the response to a Dirac excitation. $\lambda = 0.5$. Solid line : nonlinear case. Dotted line: linear case.

The nonlinear characteristics is based on the stationary Bernoulli equation and links the acoustic flow (the product between the opening of the reed channel and the acoustic velocity) to the pressure difference between the bore and the mouth of the player. It is given by:

$$\tilde{u}_e(t) = S_r(t) \text{sign}(\gamma - \tilde{p}_e(t)) \sqrt{|\gamma - \tilde{p}_e(t)|} \quad (26)$$

These equations are discretized according to the schemes presented in [6]. The third part of the full digital model is made of the bore, the two digital models of which have been presented in the previous section. The way the nonlinear bore models have been written makes their use straightforward in our synthesis scheme. The external pressure is computed by the time derivative of the output flow $\tilde{u}_s(t)$.

3.2. Results

In this part, both the role of the nonlinear termination and the relevance of the simplified nonlinear bore model are studied.

In the following examples, the output radius r_t is: $r_t = \frac{R}{2}$. The value of the parameter ζ is: $\zeta = 0.34$. The reed resonance frequency is 2200Hz and the reed quality factor is 0.4 . According to the values of the physical parameters, the weight $\tilde{\alpha}$ of the nonlinear termination is: $\tilde{\alpha} = 0.113$.

3.2.1. Steady-state oscillations

Figure 5 shows one period of the steady state oscillations of the pressure (top) and the flow (bottom) of a sound generated with a constant value of $\gamma = 0.42$. Though the blowing pressure is small, the effect of the nonlinear termination appears clearly.

Figure 6 shows the ratio of the Fourier transforms of the pressure and flow in the mouthpiece corresponding to $\gamma = 0.42$. The linear case, displayed in dotted line, corresponds to the input impedance of the bore. The nonlinear cases, displayed in solid line and dashed line, exhibit a strong lowering of the first peak (about two times smaller) and a smaller lowering of the second peak. Again, though the blowing pressure is small, the effect of the nonlinearity is clearly visible.

Both in the time and frequency domains, the difference between the accurate nonlinear model and the approximated nonlinear model is very small.

Figure 7 shows one period of the steady state oscillations of the pressure (top) and the flow (bottom) of a sound generated with a constant value of $\gamma = 0.56$. For this blowing pressure which is above the beating-reed pressure corresponding to $\gamma = 0.5$, the

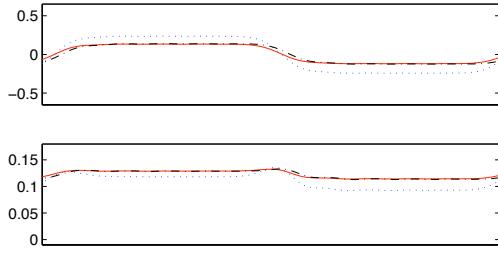


Figure 5: Top: mouthpiece pressure. Bottom: mouthpiece flow ($\gamma = 0.42$). Dotted line : linear case. Solid line : accurate model. Dashed line: approximated model.

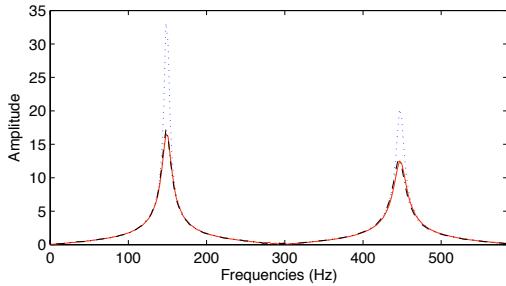


Figure 6: Ratio of the Fourier transforms of the mouthpiece pressure and flow ($\gamma = 0.42$). Dotted line : linear case. Solid line : accurate model. Dashed line: approximated model.

effects of the nonlinear termination produce an important modification of the acoustic flow.

Figure 8 shows the ratio of the Fourier transforms of the pressure and flow in the mouthpiece corresponding to $\gamma = 0.56$. Compared to the linear case (dotted line), the nonlinear cases (solid line and dashed line) exhibit a drastic lowering of the first peak (about three times smaller) and an important lowering of the second peak. For such a blowing pressure, the effect of the nonlinearity is very important.

As it was the case for a smaller blowing pressure, the difference between the accurate nonlinear model and the approximated nonlinear model is not noticeable, showing that for synthesis purposes, the second order series expansion of the flow proposed in equation (23) is sufficient.

3.2.2. Increasing blowing pressure

In this example, the sound duration is 1.5s. On this duration, the dimensionless blowing pressure γ increases linearly from 0.4 (around the self-oscillation threshold) to 0.95 (around the complete closure of the reed channel and stop of the self-oscillations).

Figure 9 compares the envelopes of the external pressure in the linear (left) case and the nonlinear (right) case. The nonlinear case is computed with the most accurate algorithm. The average sound level is smaller in the nonlinear case than in the linear case. While the sound level is continuously increasing in the linear case, one can observe that after an increasing phase, it decreases in the nonlinear case. Indeed, the synthesis shows that the output flow reaches a saturation level while the external pressure, correspond-

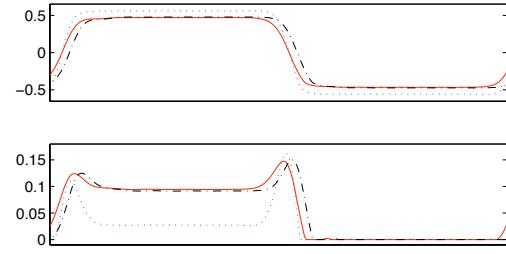


Figure 7: Top: mouthpiece pressure. Bottom: mouthpiece flow ($\gamma = 0.56$). Dotted line : linear case. Solid line : accurate model. Dashed line: approximated model.

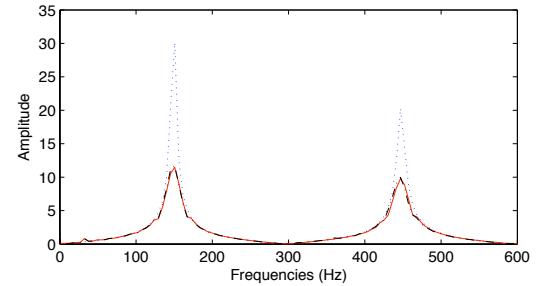


Figure 8: Ratio of the Fourier transforms of the mouthpiece pressure and flow ($\gamma = 0.56$). Dotted line : linear case. Solid line : accurate model. Dashed line: approximated model.

ing to the time derivative of the output flow, exhibits a decreasing spectral richness.

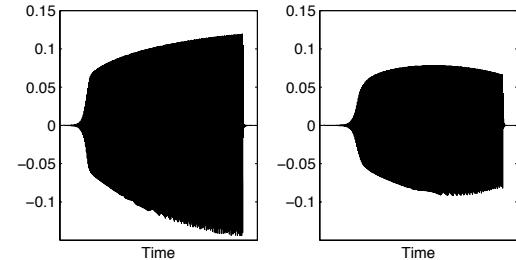


Figure 9: External pressures corresponding to a linear increase of the blowing pressure. Left: linear case. Right: nonlinear case with accurate model.

Pictures 10 and 11 show respectively the spectrogram of the external pressure in the linear case and in the nonlinear case. The vertical axis is frequency, from 0 to 4kHz and the horizontal axis is time, in seconds. It can be noticed that the birth of the fundamental frequency happens at the same time on both pictures. In the nonlinear case, the raising of the harmonics occurs later than in the linear case. The behavior of the even harmonics (coming from the mouthpiece flow, since the mouthpiece pressure contains very few even harmonics due to the impedance relationship) is significantly different: for small values of γ , the level of the amplitudes of the

even harmonics compared to that of the odd harmonics is higher in the nonlinear case than in the linear case. While the low order even harmonics keep a nearly constant level on the whole duration of the sound in the linear case, they first increase, then decrease until a minimum and increase again in the non linear case. It can be noticed that the time for which the even harmonics are minimum depends on their ranks; harmonic 2 has its minimum around $t = 1.1\text{s}$ while harmonic 12 has its minimum around $t = 0.8\text{s}$.

We point out that “usual” playing conditions correspond to the first 0.5s of the sound and that from a perceptual point of view, the most noticeable difference between the linear and nonlinear case is the balance between odd and even harmonics.

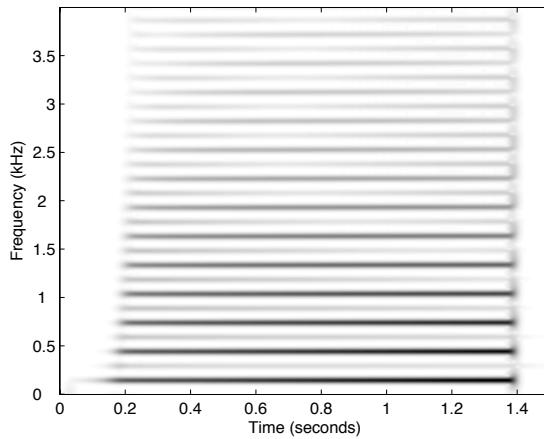


Figure 10: Spectrogram of the external pressure corresponding to the linear case and a linear increase of the blowing pressure.

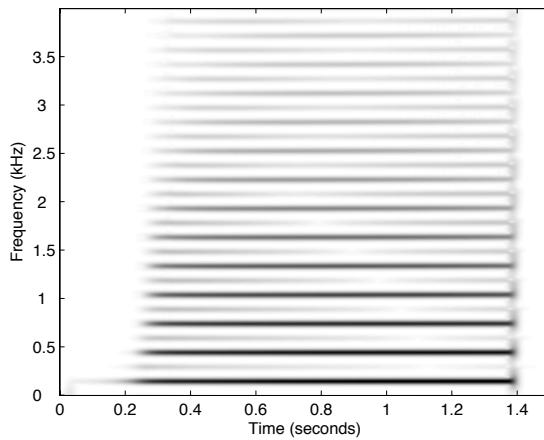


Figure 11: Spectrogram of the external pressure corresponding to the nonlinear case and a linear increase of the blowing pressure.

4. CONCLUSION

Confirming the results of Atig [2] and extending them to the case of frequency dependant bore losses and reed with mass, it has been shown that the introduction of nonlinear losses modifies significantly the role of the bore at low frequencies, hence affecting the functioning and the timbre of the instrument.

The work presented in this paper constitutes a first step towards a simple, dynamic nonlinear tonehole model extending the results presented in [9]. Moreover, during attacks consisting in a sudden burst of the blowing pressure, the effect of the nonlinear termination may modify the features of the transient.

Despite its simplicity and low computation cost, the simplified nonlinear model provides results fully similar to those of the more accurate model and seems applicable to the case of more complex bore geometries since it only adds to the resonator model a nonlinear function of the time-delayed variables at the mouthpiece level thanks to the removal of the variables at the open end level.

Sound examples are available at:
<http://www.lma.cnrs-mrs.fr/~guillemain/DAFX06/dafx06.htm>

5. REFERENCES

- [1] V. Dubos, J. Kergomard, A. Khettabi, J. Dalmont, D. Keefe, and C. Nederveen, “Theory of sound propagation in a duct with a branched tube using modal decomposition,” *Acustica - Acta Acustica*, vol. 85, pp. 153–169, 1999.
- [2] M. Atig, “Non-linéarité acoustique localisée à l’extrémité ouverte d’un tube. mesure, modélisation et application aux instruments à vent,” Ph.D. dissertation, Université du Maine, 2004.
- [3] G. Scavone and P. Cook, “Real-time computer modeling of woodwind instruments,” in *Proc. ISMA 1998*, Leavenworth, WA, USA, 1998.
- [4] M. van Walstijn and G. Scavone, “The wave digital tonehole model,” in *Proc. Int. Comp. Music Conf. (ICMC’00)*, Berlin, Germany, 2000.
- [5] J. Dalmont, C. Nederveen, V. Dubos, S. Ollivier, V. Mésérette, and E. te Sligte, “Experimental determination of the equivalent circuit of an open side hole: linear and non linear behaviour,” *Acustica - Acta Acustica*, vol. 88, pp. 567–575, 2002.
- [6] P. Guillemain, J. Kergomard, and T. Voinier, “Real-time synthesis of clarinet-like instruments using digital impedance models,” *J. Acoust. Soc. Am.*, vol. 118, no. 1, pp. 483–494, 2005.
- [7] A. Pierce, *Acoustics*. NY: McGraw-Hill, 1981.
- [8] T. A. Wilson and G. S. Beavers, “Operating modes of the clarinet,” *J. Acoust. Soc. Am.*, vol. 56, pp. 653–658, 1974.
- [9] J. Terroir and P. Guillemain, “A simple dynamic tone hole model for real-time synthesis of clarinet-like instruments,” in *Proc. Int. Comp. Music Conf. (ICMC’05)*, Barcelona, Spain, 2005.

A STABLE ACOUSTIC IMPEDANCE MODEL OF THE CLARINET USING DIGITAL WAVEGUIDES

Gary P. Scavone

CAML, Music Technology Area
Schulich School of Music
McGill University, Montreal, QC, Canada
gary@music.mcgill.ca

Julius O. Smith

CCRMA
Department of Music
Stanford University, Stanford, CA, USA
jos@ccrma.stanford.edu

ABSTRACT

Digital waveguide (DW) modeling techniques are typically associated with a traveling-wave decomposition of wave variables and a “reflection function” approach to simulating acoustic systems. As well, it is often assumed that inputs and outputs to/from these systems must be formulated in terms of traveling-wave variables. In this paper, we provide a tutorial review of DW modeling of acoustic structures to show that they can easily accommodate physical input and output variables. Under certain constraints, these formulations reduce to simple “Schroeder reverb-like” computational structures. We also present a stable single-reed filter model that allows an explicit solution at the reed / air column junction. A clarinet-like system is created by combining the reed filter with a DW impedance model of a cylindrical air column.

1. REFLECTION FUNCTION CALCULATIONS

The use of digital waveguides (DW) to model wave propagation within cylindrical air columns has been well documented [1, 2]. A DW structure like that diagrammed in Fig. 1 can be used to compute the time-domain pressure *reflection function*, $r_p(t)$, of a uniform pipe. The reflection function is defined as the pressure response at the input of an air column caused by the introduction there of a pressure impulse, assuming no reflections at the input end (an anechoic input termination). The digital filter \mathcal{R}_L models the frequency-dependent *reflectance* of the load impedance connected to the far end of the pipe. It can be expressed as

$$\mathcal{R}_L(f) = \left[\frac{Z_L(f) - Z_c}{Z_L(f) + Z_c} \right], \quad (1)$$

where Z_c is the real characteristic wave impedance of the pipe, $Z_L(f)$ is the frequency-dependent load impedance, and f is frequency in Hertz. A closed end is reasonably modeled by a load impedance $Z_L = \infty$, in which case $R_L = 1$. This indicates that pressure traveling-waves reflect from a rigid boundary in a cylindrical pipe with unity gain and no phase shift. For an open end condition, an analytic solution for $\mathcal{R}_L(f)$ has been reported by [3]. A second-order digital filter is sufficient to achieve a good fit to that analytic result for most pipe dimensions of musical interest [4].

The structure of Fig. 1 is a time-domain computational model of one-dimensional traveling-wave propagation along the length of the air column, together with wave reflection from the load impedance at the far end. The frequency-domain counterpart to

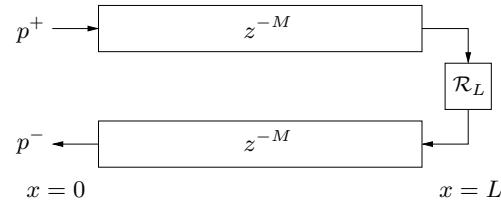


Figure 1: Digital waveguide model of a cylindrical air column.

the cylindrical air column reflection function is the plane-wave *reflectance*, which for lossless wave propagation can be written as

$$\mathcal{R}_p(f) = e^{-2jkL}\mathcal{R}_L(f), \quad (2)$$

where the load impedance is located a distance L from the pipe input. Neglecting losses along the air column walls, the phase term in Eq. (2) represents a time delay of $2L/c$ seconds in $r_p(t)$, so that the reflection function is given by a time shifted representation of the reflectance property of the load at the end of the pipe (c is the wave speed of propagation). Within a DW implementation, propagation losses along the air column length can be lumped and commuted with \mathcal{R}_L before the discrete-time filter is designed. As well, the delay-line lengths, M , represented in Fig. 1 can be fractional and implemented with fractional delay techniques [5].

2. IMPULSE RESPONSE CALCULATIONS

The *input impedance* of an air column is defined as its sinusoidal pressure response given the application of a unit sinusoidal volume velocity signal at its input, $Z_0(f) = P_0(f)/U_0(f)$. The input impedance of a cylindrical pipe of length 0.35 meters is plotted in Fig. 2, calculated using a frequency-domain technique [6]. The “peaks and valleys” of this response indicate the resonances

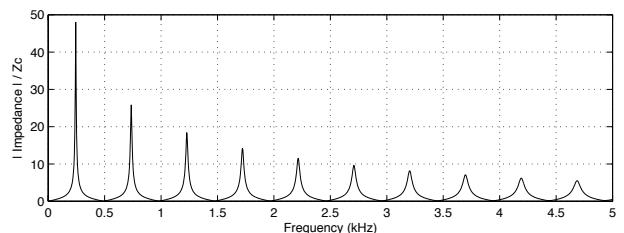


Figure 2: Theoretical input impedance of a cylindrical pipe of 0.35 meter length.

and anti-resonances of the acoustic structure, depending on the assumed boundary condition at the input. For example, pressure is ideally constrained to be zero (or equal to the ambient air pressure) at an open end, and thus the resonances of a pipe open at its input end are indicated by the minima of $Z_0(f)$. Likewise, volume velocity is ideally constrained to be zero at a closed end, so that the maxima of the input impedance specify the resonances of a cylindrical structure with a closed input end. The ideal “impedance head” has an infinite “source impedance” [7], which effectively closes the input end.

For a cylindrical pipe terminated by impedance R_L as modeled in Fig. 1, the input impedance can be written in terms of the plane-wave reflectance as

$$Z_0(f) = Z_c \left(\frac{1 + \mathcal{R}_p(f)}{1 - \mathcal{R}_p(f)} \right), \quad (3)$$

indicating that $Z_0(f)$ is completely defined by $\mathcal{R}_p(f)$. Assuming lossless propagation over the length of the air column, $Z_0(f)$ can also be written as

$$Z_0(f) = Z_c \left[\frac{Z_L(f) \cos(kL) + jZ_c \sin(kL)}{jZ_L \sin(kL) + Z_c \cos(kL)} \right], \quad (4)$$

where $k = 2\pi f/c$ is defined as the propagation wave number. This expression can be derived by substitution of Eq. (2) in Eq. (3).

The time-domain description of linear acoustic systems is traditionally given by the inverse Fourier transform of the input impedance, called the *impulse response* $h(t)$ or the Green’s function of the system. The impulse response describes the time-domain evolution of pressure at the input of an acoustic structure produced by the injection of a volume velocity unit impulse at the same position. Pressure and flow at the input are then related by means of the convolution product

$$p(t) = \int_0^t h(t-t')u(t')dt' = h(t) * u(t). \quad (5)$$

It is a simple process to extend the DW structure of Fig. 1 to compute the impulse response of an acoustic pipe, as was shown in [4]. The volume velocity impulse at the input is first scaled by Z_c to convert to a corresponding pressure value. The implicit closed-end condition at $x = 0$ is modeled with a reflection coefficient of +1 for pressure traveling-wave components. If we are concerned

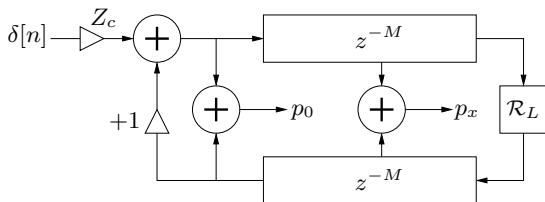


Figure 3: A digital waveguide cylindrical pipe impedance model (from [4]).

only with the physical pressure at the input of the acoustic system modeled in Fig. 3, it is possible to commute the two delay lines to form the structure of Fig. 4. The transfer function of the system modeled by Fig. 4 is given by

$$H(z) = Z_c \left(\frac{1 + z^{-2M} \mathcal{R}_L(z)}{1 - z^{-2M} \mathcal{R}_L(z)} \right), \quad (6)$$

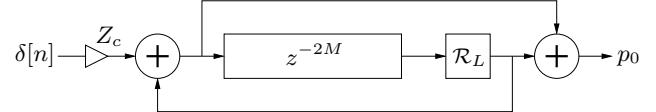


Figure 4: The commuted digital waveguide input impedance model of a closed-open cylindrical pipe.

which is equivalent to Eq. (3) with the substitution specified by Eq. (2). The DW structures shown in Figs. 3 and 4 provide efficient computational schemes that support the use of input and output physical acoustic variables, though the structure in Fig. 3 more clearly indicates how to extract physical output variables to be probed at arbitrary positions along the pipe. It is also possible to modify these structures to support the computation of volume flow rather than pressure.

The theoretical input impedance and impulse response of a cylindrical pipe terminated with a load impedance as determined by [3] is shown in Fig. 5, calculated using both a frequency-domain technique [6] and the DW structure of Fig. 4. Propagation losses along the length of the pipe were ignored in both cases.

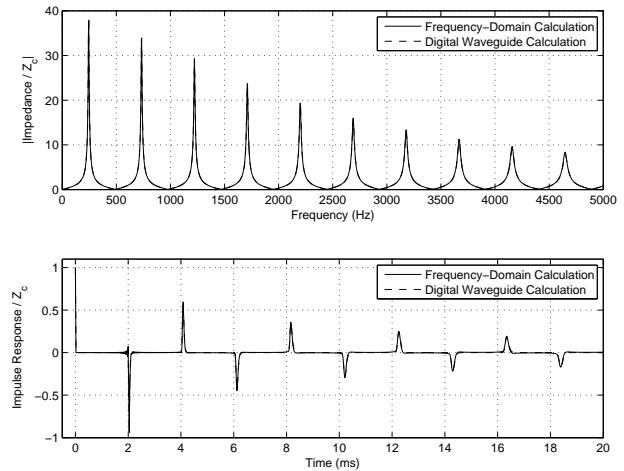


Figure 5: Calculated input impedance (top) and impulse response (bottom) of a cylindrical pipe of 0.35 meter length.

A recent paper [8] takes an acoustic modeling approach based on the input impedance transfer function with ideal boundary conditions. That is, by starting with Eq. (4) and assuming an ideal open-end load impedance $Z_L = 0$, $Z_0(f)$ can be written

$$\begin{aligned} Z_0(f) &= Z_c \left[\frac{1 - e^{-2jkL}}{1 + e^{-2jkL}} \right] \\ &= Z_c \left[\frac{1}{1 + e^{-2jkL}} - \frac{e^{-2jkL}}{1 + e^{-2jkL}} \right]. \end{aligned} \quad (7)$$

From the expression of Eq. (7), they propose a computational structure as shown in Fig. 6. As noted by [2, p. 52], this structure can be simplified to the one delay-line form shown in Fig. 7, which is reminiscent of the Schroeder allpass filter used for artificial reverberation [9]. We see that this is equivalent to the structure of Fig. 4, except that the DW approach allows an arbitrary load impedance characterization. In this sense, the DW approach makes it obvious how to achieve the same level

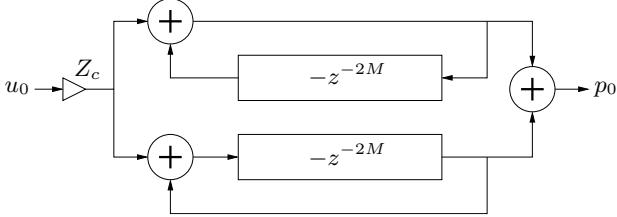


Figure 6: Ideal impedance transfer function model of cylindrical pipe assuming $Z_L = 0$.

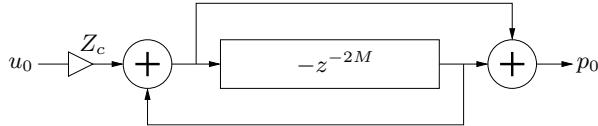


Figure 7: One-delay form of ideal impedance transfer function model of cylindrical pipe assuming $Z_L = 0$.

of efficiency while taking into account an acoustically accurate model of open-end reflection and radiation. As previously noted, propagation losses are easily accounted for in the DW structure and typically commuted with the load impedance reflection filter [4].

3. CONICAL IMPEDANCE MODELING

The analyses made for the cylindrical pipe can be repeated for a truncated conical acoustic structure, as diagrammed in Fig. 8. In general, the relationships are a bit more complicated given that the characteristic impedance for spherical waves is dependent on both frequency and position.

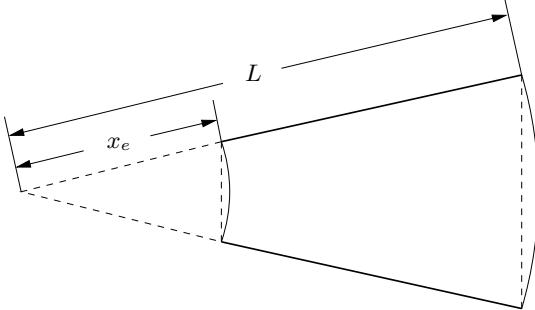


Figure 8: A conical acoustic structure, truncated at $x = x_e$.

For a conic section terminated by a load impedance $Z_L(f)$ at $x = L$, the spherical-wave pressure reflectance is given by

$$\mathcal{R}_s(f) = e^{-2jkL} \left[\frac{Z_L(f)Z_c^*(L, f) - Z_c(L, f)Z_c^*(L, f)}{Z_L(f)Z_c(L, f) + Z_c(L, f)Z_c^*(L, f)} \right], \quad (8)$$

where the spherical wave impedance $Z_c(x, f)$ is

$$Z_c(x, f) = \frac{\rho c}{S(x)} \left(\frac{jkx}{1 + jkx} \right) = \frac{\rho c}{S(x)} \left(\frac{1}{1 + \frac{1}{j k x}} \right), \quad (9)$$

ρ is the mass density of air, c is the speed of wave propagation, and $S(x)$ is the spherical wavefront surface area at position x . Note

that if we assume lossless propagation and an ideal open-end load impedance $Z_L = 0$, Eq. (8) reduces to $\mathcal{R}_s(f) = -e^{-2jkL}$, which is equivalent to the reflectance of an ideally terminated cylindrical pipe. Thus, the DW structure of Fig. 1 (with $R_L = -1$) can be used to simulate spherical wave propagation as well, though traveling-wave and physical output variables at points downstream from the input must be scaled by $1/x$ to account for the spread of pressure over increasing wavefront surface areas. Benade [10] provides an equivalent circuit for the conical waveguide in terms of a uniform transmission line, two acoustic inertances, and a transformer, as shown in Fig. 9. This representation suggests that a conical air

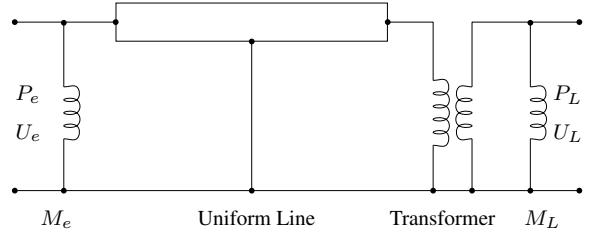


Figure 9: Equivalent circuit of a conical waveguide.

column model can be implemented using a cylindrical waveguide, a scalar “turns ratio” multiplier, and appropriately designed inertance components at each end of the waveguide. This interpretation is supported by the expression for the input admittance of a conical waveguide,

$$Y_e(f) = \frac{1}{Z_e(f)} = \frac{S(x_e)}{\rho c} \left\{ \left[\frac{1 - \mathcal{R}_s(f)}{1 + \mathcal{R}_s(f)} \right] + \frac{1}{jkx_e} \right\}, \quad (10)$$

where x_e is the length of the missing section of cone at the input. Eq. (10) makes clear the parallel combination of the spherical wave impedance and a term reminiscent of the impedance of a cylindrical waveguide. The inertance term near the output, M_L , can be combined with the output impedance representation and a single digital filter designed. Note, however, that the frequency dependence of this term is inversely proportional to distance from the conic section apex and thus can be ignored in most contexts without having noticeable affect on the resulting response. Further, an ideal load impedance $Z_L = 0$ will “short-circuit” the inertance term and we can expect that a more accurate open-end representation will have similar behavior, especially at low frequencies.

In developing a DW structure to calculate the input impedance of a conic section, the input end, being driven by an ideal velocity source, is terminated by an infinite source impedance, corresponding to a rigid termination. Thus, traveling-wave reflection at this point is completely defined by the input inertance term, M_e , which is represented by a first-order digital allpass reflectance filter of the form [5, 4, 11]

$$\mathcal{R}_e(z) = \frac{-a_1 - z^{-1}}{1 + a_1 z^{-1}}, \quad \text{where } a_1 = \frac{c - \alpha x_e}{c + \alpha x_e}, \quad (11)$$

and α is the bilinear transform constant that controls frequency warping. This first-order allpass filter accounts for the phase delay experienced by pressure traveling-wave components reflecting from a rigid input termination in a conical waveguide.

Impulse response calculations are based on the injection of a volume velocity unit impulse at the input of the system. Because

our DW structure will simulate pressure traveling waves, it is necessary to convert the incoming *flow* impulse to a variable of pressure via the characteristic wave impedance. For cylindrical ducts, Z_c is a real value and simply results in a scaling of the input variable. For conical structures, however, Z_c is a function of frequency and the transformation must make use of a digital filter. Using the bilinear transform, a discrete-time equivalent to Eq. (9) is given by

$$Z_c(x, z) = \frac{\rho c}{S(x)} \left(\frac{\alpha x}{c + \alpha x} \right) \frac{1 - z^{-1}}{1 + a_1 z^{-1}}, \quad (12)$$

where a_1 is equal to the allpass truncation filter coefficient given in Eq. (11). The resulting DW model is represented in Fig. 10. Note that the bilinear transform implicitly performs a continuous-

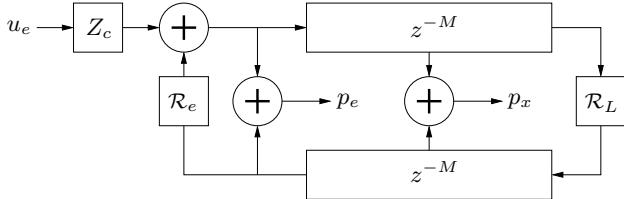


Figure 10: A digital waveguide conical air column impedance model (from [4]).

to discrete-time frequency “warping” and thus the results calculated with the DW model of Fig. 10 will be most accurate at frequencies well below half the sample rate.

If, as before, we are concerned only with the physical pressure at the input of the acoustic system modeled in Fig. 10, it is possible to commute the two delay lines to form the structure of Fig. 11.

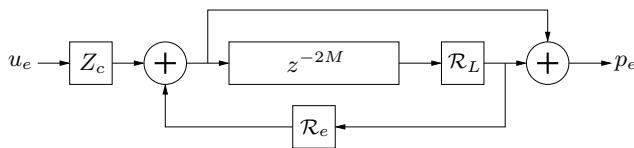


Figure 11: The commuted digital waveguide input impedance model of a conical air column.

The theoretical input impedance and impulse response of a conical section terminated with a load impedance as determined by [3] is shown in Fig. 12, calculated using both a frequency-domain [6] model and the DW structure of Fig. 11. Propagation losses along the length of the pipe were ignored in both cases.

The authors of [8] propose an alternate digital impedance model of a conical air column based again on an ideal open-end impedance approximation of $Z_L = 0$. Note that this approximation is less accurate with increasing open-end radius, which can be significant in musical instruments constructed from conical air columns. They “calibrate” their results by adding losses to the system such that the first two impedance magnitude peak values are matched to theoretical values. Their structure is a transfer function representation of the input impedance expression

$$Z_e(f) = \frac{\frac{x_e}{c} \mathcal{D}(f)}{1 + \frac{x_e}{c} \mathcal{D}(f) \mathcal{C}^{-1}(f)}, \quad (13)$$

where $\mathcal{D}(f) = jf$ is a differentiation operator and $\mathcal{C}(f)$ is the input impedance of a cylindrical pipe with an ideal open-end as-

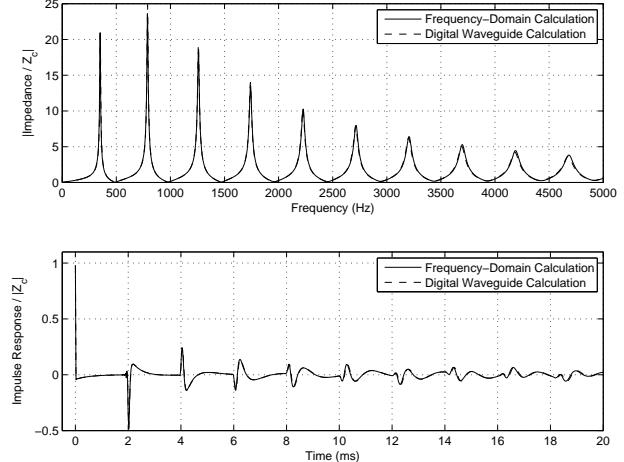


Figure 12: Theoretical input impedance magnitude (top) and impulse response (bottom) of a conical section, relative to $|Z_c(e)|$ at the input end.

sumption ($\mathcal{C}(f) = Z_0(f)$ as given by Eq. (7)). Figure 13 illustrates the computational structure used to implement this system, where $\mathcal{C}(f)$ is computed as in Fig. 6.

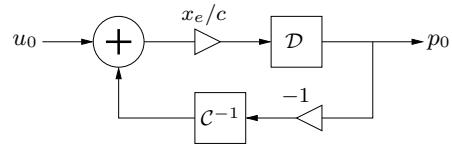


Figure 13: Ideal impedance transfer function model of a conic section assuming $Z_L = 0$.

The DW structures represented in Figs. 10 and 11 have a clear physical interpretation that supports the use of arbitrary filters to achieve as great a level of accuracy as needed for a given simulation. The structure of Fig. 13 is much more limited in this context.

4. THE REED INTERFACE

A pressure-controlled reed is traditionally modeled as a simple damped mechanical oscillator that is “displacement limited” by the mouthpiece facing. Assuming the reed motion is driven by the difference in pressures between the player’s mouth and at the air column entrance, $p_\Delta = p_m - p_0$, this system can be expressed as

$$\frac{d^2y}{dt^2} + g_r \frac{dy}{dt} + \omega_r^2 y = -\frac{p_\Delta(t)}{\mu_r}, \quad (14)$$

where ω_r is the reed resonance frequency, g_r is the reed damping coefficient, and μ_r is the reed’s dynamic mass per unit area. The flow through the reed channel is generally calculated using the Bernoulli equation and given by

$$u = w (y + H) \left(\frac{2|p_\Delta|}{\rho} \right)^{1/2} \operatorname{sgn}(p_\Delta), \quad (15)$$

where w is the reed channel width, y is the time-varying reed channel height, calculated from Eq. (14), and H is the equilibrium tip opening.

For single-reed geometries, the pressure and flow in the reed channel can be approximated as equivalent to the pressure and flow at the entrance to the instrument air column. This approximation is based on continuity and detachment of volume flow at the end of the reed channel such that pressure is not recovered in the mouth-piece. Thus, the acoustic interaction at the interface of the reed and air column can be solved using Eqs. (14) and (15), together with the DW input impedance computational structure of Fig. 3. From Fig. 3, it is clear that

$$p_0 = 2p_0^- + Z_c u_0, \quad (16)$$

where p_0^- is the traveling-wave pressure entering the reed junction from the downstream air column. This expression is also well known from [12]. Because of mutual dependencies, however, an explicit solution of these equations can be problematic. In a discrete-time computational context, these mutual dependencies can be understood to result in delay-free loops.

In [8], the reed system is discretized using a centered finite difference approximation that avoids a direct feedforward path through the reed transfer function. The resulting system equations can then be expressed in terms of a second-order polynomial equation and an explicit solution found.

The centered finite-difference approximation of Eq. (14) results in a digital filter structure of the form

$$\frac{Y(z)}{P_\Delta(z)} = \frac{-1/\mu_r}{(f_s^2 + \frac{g_r f_s}{2}) + (\omega_r^2 - 2f_s^2)z^{-1} + (f_s^2 - \frac{g_r f_s}{2})z^{-2}}, \quad (17)$$

where f_s is the computational sample rate. As noted in [13], however, this filter structure is only stable for $\omega_r < f_s \sqrt{4 - (g_r/\omega_r)^2}$, limiting its use at low sample rates and/or with high reed resonance frequencies.

5. A STABLE REED FILTER

A direct application of the bilinear transform to the system of Eq. (14) results in a digital filter structure given by

$$\frac{Y(z)}{P_\Delta(z)} = \frac{-1/\mu_r [1 + 2z^{-1} + z^{-2}]}{a_0 + 2(\omega_r^2 - \alpha^2)z^{-1} + (\alpha^2 - g_r\alpha + \omega_r^2)z^{-2}}, \quad (18)$$

where $a_0 = \alpha^2 + g_r\alpha + \omega_r^2$ and α is the bilinear transform constant that controls frequency warping. Note that we can achieve an exact continuous- to discrete-time frequency match at the resonance frequency of the reed by setting $\alpha = \omega_r / \tan(\omega_r/2f_s)$.

In this case, the use of the bilinear transform guarantees a stable digital filter at any sample rate. The presence of the direct feedforward path in Eq. (18), however, prohibits the explicit reed interface solution mentioned above. We therefore seek an alternative form of Eq. (18) that preserves stability and avoids an undelayed feedforward coefficient in the transfer function numerator.

By default, the bilinear transform substitution produces a system with “zeroes” at $z = \pm 1$ (or at frequencies of 0 and $f_s/2$ Hz). While this result is often desirable for digital resonators, we can modify the numerator terms without affecting the essential behavior and stability of the resonator. In fact, it is the numerator terms that control the phase offset of the decaying oscillation. Thus, we can modify and renormalize the numerator to produce a filter structure of the form

$$\frac{Y(z)}{P_\Delta(z)} = \frac{-4z^{-1}/\mu_r}{a_0 + 2(\omega_r^2 - \alpha^2)z^{-1} + (\alpha^2 - g\alpha + \omega_r^2)z^{-2}}. \quad (19)$$

The frequency- and time-domain responses of the centered finite-difference and “modified” bilinear transform filter structures are shown in Fig. 14 for a reed resonance frequency $f_r = 2500$ Hz and $f_s = 22050$ Hz.

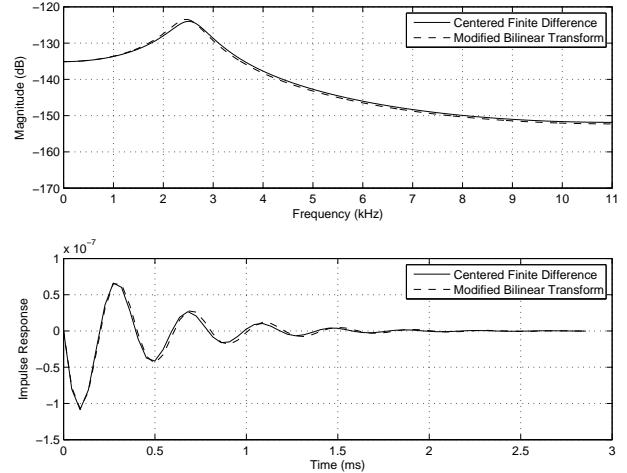


Figure 14: Reed filter frequency and impulse responses for centered finite-difference and modified bilinear transform structures with $f_r = 2500$ Hz and $f_s = 22050$ Hz: magnitude frequency response (top) and impulse response (bottom).

These plots are repeated in Fig. 15 for a reed resonance frequency $f_r = 8000$ Hz and $f_s = 22050$ Hz. It is clear that the centered finite-difference approximation is unstable in this case, while the modified bilinear transform solution remains accurate.

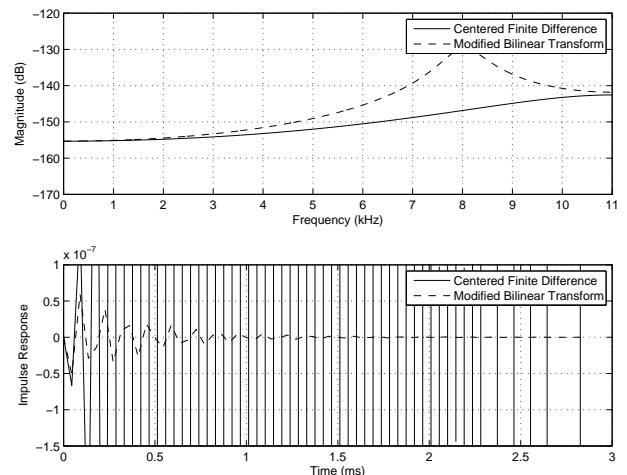


Figure 15: Reed filter frequency and impulse responses for centered finite-difference and modified bilinear transform structures with $f_r = 8000$ Hz and $f_s = 22050$ Hz: magnitude frequency response (top) and impulse response (bottom).

6. THE COMPLETE CLARINET MODEL

The complete clarinet model involves the calculation of the reed displacement using the stable reed model discussed in Section 5,

the volume flow through the reed channel as given by Eq. (15), and the relationship between flow and pressure at the entrance to the air column as given by Eq. (16). Because the reed displacement given by Eq. (19) does not have an immediate dependence on p_Δ , it is possible to explicitly solve Eqs. (16) and (15), as noted in [8], by an expression of the form

$$u_0 = 0.5 \left(B\sqrt{(Z_c B)^2 + 4A} - Z_c B^2 \right) \operatorname{sgn}(A), \quad (20)$$

where $A = p_m - 2p_0^-$ and $B = w(y + H)(2/\rho)^{1/2}$ can be determined at the beginning of each iteration from constant and past known values. Whenever the reed position $y + H < 0$, u_0 is set to zero and $p_0 = 2p_0^-$.

In Fig. 16, the normalized pressure response of the complete DW synthesis model is plotted using both reed models with $f_r = 2500$ Hz and $f_s = 22050$ Hz. The behaviors are indistinguishable for these system parameters, though as indicated above it is possible to run the modified bilinear transform model at significantly lower sample rates (and with higher reed resonance frequencies).

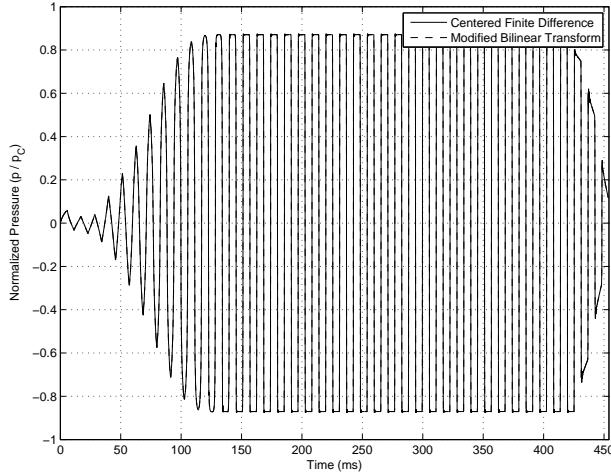


Figure 16: Normalized pressure response from complete DW synthesis model using the centered finite-difference and modified bilinear transform structures with $f_r = 2500$ Hz and $f_s = 22050$ Hz.

It should be noted that the explicit solution above is not possible if the reed is attached directly to a conical air column because the allpass truncation filter \mathcal{R}_e of Fig. 10 creates a delay-free loop at the junction. Likewise, the differentiation operator in Fig. 13 presents the same limitation. A common solution to this problem, as discussed in [11], is to insert a short cylindrical section between the reed and conical waveguide of equivalent volume to the missing, truncated conic section.

7. CONCLUSIONS

We have presented a tutorial review on the use of digital waveguide techniques for the modeling of acoustic impedance in cylindrical and conical air column structures. The resulting systems are compared with digital impedance models recently discussed in the literature [8]. Both approaches simulate traveling-wave propagation

using delay lines, though DW methods allow a more explicit and modular correlation between physical and computational components (for example, open-end filtering, toneholes, and radiation).

Further, we have presented a reed model that guarantees stability and an accurate continuous- to discrete-time resonance frequency mapping at any sample rate (assuming $f_r < f_s/2$). This model supports an explicit reed interface solution as proposed in [8].

8. REFERENCES

- [1] J. O. Smith, "Efficient simulation of the reed-bore and bow-string mechanisms," in *Proc. Int. Comp. Music Conf. (ICMC'86)*, Den Haag, Netherlands, 1986, pp. 275–280.
- [2] ———, *Physical Audio Signal Processing: for Virtual Music Instruments and Digital Audio Effects*. W3K Publishing, 2004.
- [3] H. Levine and J. Schwinger, "On the radiation of sound from an unflanged circular pipe," *Phys. Rev.*, vol. 73, no. 4, pp. 383–406, Feb. 1948.
- [4] G. P. Scavone, "An acoustic analysis of single-reed woodwind instruments with an emphasis on design and performance issues and digital waveguide modeling techniques," Ph.D. dissertation, Music Dept., Stanford University, Mar. 1997.
- [5] V. Välimäki, "Discrete-time modeling of acoustic tubes using fractional delay filters," Ph.D. dissertation, Helsinki University of Technology, Faculty of Electrical Engineering, Laboratory of Acoustic and Audio Signal Processing, Espoo, Finland, Report no. 37, Dec. 1995.
- [6] D. H. Keefe, "Acoustical wave propagation in cylindrical ducts: Transmission line parameter approximations for isothermal and nonisothermal boundary conditions," *J. Acoust. Soc. Am.*, vol. 75, no. 1, pp. 58–62, Jan. 1984.
- [7] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory*. New York: McGraw-Hill, 1969.
- [8] P. Guillemain, J. Kergomard, and T. Voinier, "Real-time synthesis of clarinet-like instruments using digital impedance models," *J. Acoust. Soc. Am.*, vol. 118, no. 1, pp. 483–494, July 2005.
- [9] M. R. Schroeder, "Natural-sounding artificial reverberation," *J. Audio Eng. Soc.*, vol. 10, no. 3, pp. 219–223, 1962.
- [10] A. H. Benade, "Equivalent circuits for conical waveguides," *J. Acoust. Soc. Am.*, vol. 83, no. 5, pp. 1764–1769, May 1988.
- [11] G. P. Scavone, "Time-domain synthesis of conical bore instrument sounds," in *Proc. Int. Comp. Music Conf. (ICMC'02)*, Gothenburg, Sweden, 2002, pp. 9–15.
- [12] M. E. McIntyre, R. T. Schumacher, and J. Woodhouse, "On the oscillations of musical instruments," *J. Acoust. Soc. Am.*, vol. 74, no. 5, pp. 1325–1345, nov 1983.
- [13] P. Guillemain, "A digital synthesis model of double-reed wind instruments," *EURASIP J. on Applied Sig. Proc.*, vol. 4, no. 7, pp. 990–1000, June 2004.

THE MELLIN PIZZICATOR

Antonio De Sena, Davide Rocchesso

Dipartimento di Informatica

Università di Verona, Italia

desena@sci.univr.it, davide.rocchesso@univr.it

ABSTRACT

In this paper an application of the Mellin transform to the digital audio effects will be presented. Namely, low-pass and band-pass like filtering in the Mellin domain will be described and used for obtaining some kind of *pizzicato* effect on audio samples (musical instruments, but not only). The pluck and damp effects will be obtained using filtering in Mellin domain only. The algorithm used for implementing the Mellin (scale) transform has been presented in DAFX'04 [1].

1. INTRODUCTION

The effect showed in this paper is entirely based on filtering in the Mellin (scale) domain. The Mellin transform is the mathematical tool that allows to pass from time domain to the so called Mellin domain. The scale transform, a restriction of the Mellin transform, introduced by Cohen [2], can represent a signal in terms of *scale*. The scale can be interpreted, similarly to frequency, as a physical attribute of signals [2]. Thus, we can conceive digital audio effects that work by handling the signal in the scale domain, with transformation of the magnitude and/or phase of the scale image. This is technically feasible as long as fast and accurate realizations of these transforms are available.

The effect presented aims at simulating a *pizzicato* on a generic audio sample. So, for example, a flute can be “pizzicated” for obtaining some kind of plucked string instrument with spectral characteristic of the flute sample. This effect can be realized using two filters in the scale domain. The first filter dynamically cuts the frequencies (from the higher to the lower) as time increases, and the second filter simulates a pluck at the beginning of the sample. Both the filtering processes are realized in the scale domain, the first is a low-pass filter and the second is a band-scale enhancer.

It's important to emphasize the fact that this is an experiment in exotic domains (Mellin, scale) and the first objective is the exploration of these domains. So this paper doesn't want to introduce the best way to do this kind of effects, but an alternative to classical Fourier approaches.

In section 2 an introduction to Mellin and scale domains will be given with a description of the Fast Mellin Transform (FMT). In section 3.1 the scale magnitude low-pass, high-pass and band-pass filtering will be discussed and their behavior will be described. In sections 3.2 a description on how to obtain damp and pluck effects will be given and a discussion about a more classic implementation of the effect will be introduced in section 3.3. Finally, in section 4 experiments and results will be described and shown, and a comparison with a real pizzicato will be described.

2. MELLIN AND SCALE TRANSFORMS

The Mellin transform of a function f is defined as:

$$M_f(p) = \int_0^\infty f(t) t^{p-1} dt , \quad (1)$$

where $p \in \mathbb{C}$ is the Mellin parameter. The scale transform [2] is a particular restriction of the Mellin transform on the vertical line $p = -jc + \frac{1}{2}$, with $c \in \mathbb{R}$. Thus, the scale transform is defined as:

$$D_f(c) = \frac{1}{\sqrt{2\pi}} \int_0^\infty f(t) e^{(-jc-\frac{1}{2}) \ln t} dt. \quad (2)$$

The scale inverse transform is given by

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^\infty D_f(c) e^{(jc-\frac{1}{2}) \ln t} dc. \quad (3)$$

The key property of the scale transform is the scale invariance. This means that if f is a function and g is a scaled version of f , the transform magnitude of both functions is the same. A scale modification is a compression or expansion of the time axis of the original function that preserves signal energy. Thus, a function $g(t)$ can be obtained with a scale modification from a function $f(t)$, if $g(t) = \sqrt{\alpha}f(\alpha t)$, with $\alpha \in \mathbb{R}^+$. When $\alpha < 1$ we get a scale expansion, when $\alpha > 1$ we get a scale compression. Given a scale modification with parameter α , the scale transforms of the original and scaled signals are related by

$$D_g(c) = \alpha^{jc} D_f(c). \quad (4)$$

This property derives from a similar property of the Mellin transform. In fact, if $h(t) = f(\alpha t)$, then

$$M_h(p) = \alpha^{-p} M_f(p). \quad (5)$$

In both (4) and (5), scaling is reflected by a multiplicative factor for the transforms, and for (4) such factor reduces to a phase difference. So, the scale transform magnitudes of the original signal and the scaled signal are the same.

$$|D_g(c)| = |D_f(c)|. \quad (6)$$

2.1. The Scale Interpretation

A parallel can be drawn between the properties of the Fourier and scale transforms. In particular, we can define a *scale periodicity* as follows: a function $f(t)$ is said to be scale periodic with period \mathcal{T} if it satisfies $f(t) = \sqrt{\mathcal{T}}f(t\mathcal{T})$, where $\mathcal{T} = b/a$, with a and b starting and ending point of the scale period. $C_0 = 2\pi/\ln \mathcal{T}$ is the “fundamental scale” associated with the periodic function. By

analogy with the Fourier theory, we can define a “scale series” and Parseval theorem [3].

For this work it is important to introduce an interpretation of the scale transform based on the scale decomposition of a signal. Like for the Fourier theory in which we can see signals like infinite sums of sines and cosines, for the scale theory we can interpret a signal as an infinite sum of time and frequency damped periodic functions (figure 1). So instead of sines and cosine we have the following functions¹:

$$dd\sin(t) = \frac{\sin(c \ln t)}{\sqrt{t}} \quad (7)$$

$$dd\cos(t) = \frac{\cos(c \ln t)}{\sqrt{t}}, \quad (8)$$

where c is the scale.

So, the components of a signal at low scales are functions (all components start from time values grater then zero, and are damped in time by the coefficient $t^{-\frac{1}{2}}$) in which their frequencies go rapidly near low values (heavy damped in frequency as time increases). This fact will be used for explaining the filtering in 3.2.

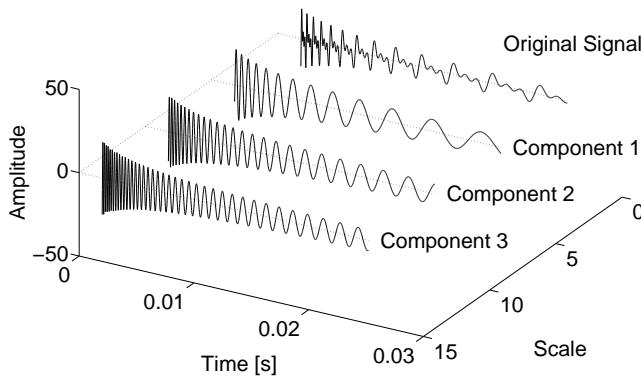


Figure 1: Example of scale decomposition (Phase, energy and other concepts are not taken into account. Energies are normalized.).

2.2. Relation with the Fourier transform

From its definition and interpretation, the Mellin transform provides a tight correspondence with the Fourier transform. More precisely, the Mellin transform with the parameter $p = -jc$ can be interpreted as a logarithmic-time Fourier transform. Similarly, we can define the scale transform of a function $f(t)$ using the Fourier transform of a function $g(t)$, with $g(t)$ obtained from $f(t)$ by time-warping f and multiplying the result by an exponential function. This result can be generalized for any p defined as $p = -jc + \beta$, with $\beta \in \mathbb{R}$. So, if $g(t) = e^{t\beta} f(e^t)$ with $\beta = \frac{1}{2}$ then:

$$D[f(t)] = F[g(t)], \quad (9)$$

¹Due to our purposes of magnitude filtering, the phase is not taken into account.

where $F[\cdot]$ and $D[\cdot]$ refer to the Fourier transform and scale transform, respectively.

2.3. A Fast Mellin Transform

Practical modifications of signals in the Mellin domain can be achieved only if an accurate and fast discrete realization of the Mellin transform is available. We have realized² a Fast Mellin Transform (FMT [1]) by exploiting the analogy between the Mellin and Fourier transforms (section 2.2), as a sequence of exponential time-warping, multiplication by an exponential, and Fast Fourier Transform, as represented in figure 2. So the algorithm performs

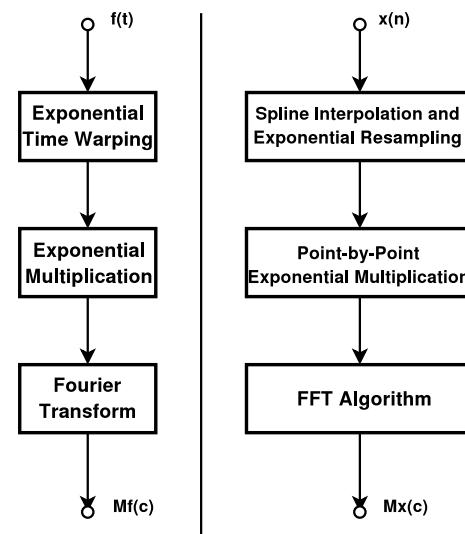


Figure 2: Implementation of the Fast Mellin Transform: theory (left) and practice (right).

an exponential resampling using a spline interpolator (cardinal or natural cubic [4]) for achieving the exponential time warping required, a point by point multiplication with an exponential function and a FFT (Fast Fourier Transform). The asymptotic complexity of the algorithm is $\mathcal{O}(n \ln^2 n)$, where n is the number of samples.

3. THE PIZZICATO EFFECT AND FILTERING IN SCALE DOMAIN

When a string (for example a guitar string [5]) is plucked we hear a sound. If analyzed from a time-frequency point of view (using a spectrogram for example) this signal presents a damping in time and in frequency. The signal loses energy (due to air resistance, losses at string termination, internal losses in the material due to viscoelastic losses, etc.), but the losses are not uniformly distributed in the spectrum. The attenuation works first (from a temporal point of view) on high frequencies and then on low frequencies.

This behavior can be simulated using filtering in Mellin (scale) domain. In the next sections we will explain how.

²The MATLAB code is available on <http://profs.sci.univr.it/~desena/FMT/>

3.1. Filtering in Scale Domain

In this section low-pass, high-pass and band-pass filters in scale domain will be introduced. Other types of filtering in scale domain can be found in [1] and a sample with water drops filtered using the scale transform has been submitted to the *Freesound Project*³.

The low-pass filter can be performed by multiplying the transform magnitude by a window (like filtering in Fourier domain). The simplest window could be a rectangular window, that sets to zero all magnitude components that are found between a cutoff scale and the signal maximum scale. Observing the results (the original signal spectrogram can be seen in figure 3, top), we can interpret this filter like a time-varying low-pass filter. The cutoff frequency approaches zero (hyperbolically, cf. [6]) as time increases and, at the same time, the amplitude of the signal is damped (see figure 4, top). The speed of frequencies cutting and the amplitude damping depend on the cutoff scale.

The high-pass filter behaves symmetrically, gradually moving the cutoff frequency toward zero (see figure 4, bottom).

The band-pass works in a midway, allowing to pass only certain frequencies at precise time instants. Viewing a band-pass in a spectrogram plot one could see only frequencies between two (time vs frequency) hyperbolic curves (see figure 3, bottom).

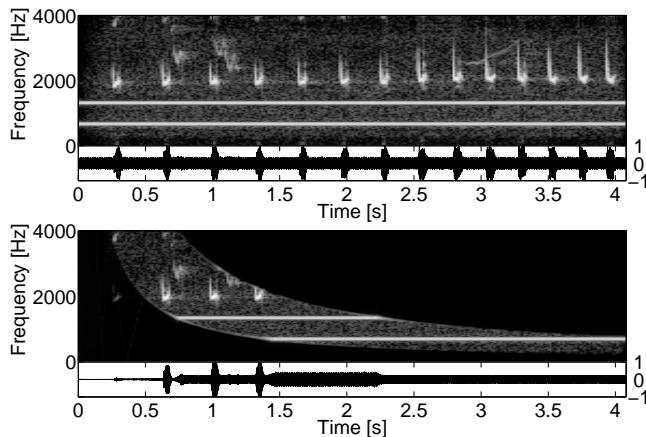


Figure 3: Original signal (top) and band-pass filtered signal (bottom) spectrograms.

3.2. The Damping Effect and the Plucking Effect

After the previous two sections we are now in the position to explain how the damping effect and the plucking effect have been obtained with the scale transform.

As already introduced, when plucked, a string starts to vibrate until the energy is dissipated. From a time and frequency viewpoint we can see the frequencies components approaches zero, with high frequencies converging faster. A similar effect can be obtained using a scale low-pass filter. So, the scale transform can be used to mimic the natural damping effect of a string. The effect can be driven using three parameters: the damping velocity (how fast frequencies are attenuated) that can be controlled modifying the bandwidth of the window (larger values meaning slower decays); the floor value of the window that affords an attenuation

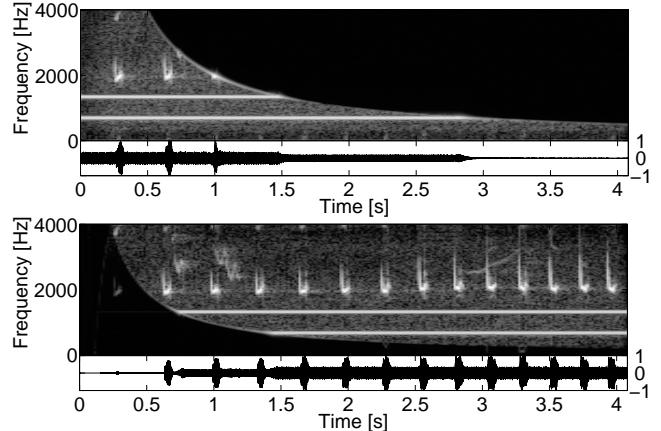


Figure 4: Low-pass (top) and high-pass filtered signal (bottom) spectrograms.

instead of a full cutting of out-of-band frequencies, and the window type (rectangular, Hann, etc.) that affects the way in which the frequencies are cut or attenuated over the time.

The plucking effect has been simulated using a band-enhancer that is like a band-pass but, instead of cutting, amplifies a range of scale components. This band-enhancer works at very low scale range with a very narrow band. So, only few scale components are enhanced and their frequencies go rapidly near zero (because they are ultra low scale components). Thus, an attack/pluck effect at the beginning of the signal will be produced. The parameters for this effect are similar to the previous one, but are more constrained. In particular the bandwidth of the window must be small and the central scale of the band-enhancer must be very small. There is also an enhancing factor that can be tuned to obtain heavier or softer attack/pluck effect.

3.3. A Comparison with STFT Approach

Even if the primary objective of the paper is the exploration of scale and Mellin domains, we can make a comparison of our framework with a classical Fourier approach. One way to obtain similar effects using a more classical approach is the use of the short time FFT/IFFT-based analysis-synthesis scheme [7, 8]. The STFT gives obvious advantages from a computational viewpoint, but the filtering framework is more complex. In fact using the STFT one should build a low-pass filter sequence with the cutoff frequency that varies with time (high in the first applied filters, and low in the last filters). Hop size and windows types that allow a correct re-synthesis must be taken into account. So, the entire structure becomes bigger than the proposed one. Another advantage of the STFT approach is that it is a more general approach that allows a control over the “shape” of the time-frequency damping that in the Mellin case can only be hyperbolic, but the latter can produce a continuous variation of the frequency cut over time. This is something difficult to achieve with the STFT.

4. EXPERIMENTS AND RESULTS

The experiments have been mainly conducted on samples of musical instruments. The samples come from the University of Iowa

³<http://freesound.iua.upf.edu/samplesViewSingle.php?id=17531>

(<http://theremin.music.uiowa.edu/MIS.html>) and from the “McGill University Master Samp” cds (<http://www.mcgill.ca/>).

Other experiments can be done on non instruments samples, like vowels for instance, to try to “pizzicate” the vowels.

All the samples are WAV files with a sampling frequency of 44.1 kHz and with a quantization of 16 bit. A normalization has been performed after processing.

Let us introduce some definitions for general readers. The term *pizzicato* means an audio event similar to the sound produced by a string plucked (sounded) by fingers or a plectrum. *Percussive* sound indicates a sound similar to the hit of a body on a surface. On the contrary of the previous definition this sound does not involve a string instrument. Finally, *vibrato* means a tremulous or pulsating effect produced by minute and rapid variations in pitch.

In figure 5 a bowed cello sample has been processed using a low-pass window with minimum value set to zero. This implies that the scale components after the scale-cutoff value are deleted and not simply attenuated. The spectrogram shows the hyperbolic decay in time of the frequencies, emulating the damp effect. The attack/pluck effect is not clearly visible in the graphical representation, but can be seen in the time domain. The obtained sample sounds different from a true pizzicato cello string, but, from a perceptual point of view, the effect resembles a true string like behavior.

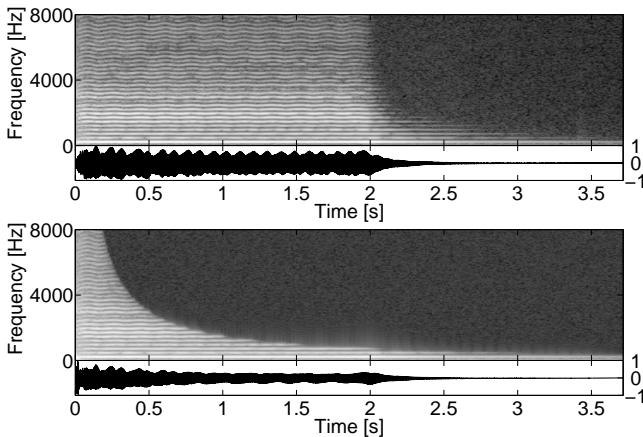


Figure 5: Cello sample. Original sample (top) and filtered (bottom).

In figure 6 a flute sample has been processed using a low-pass window with minimum value set to a value greater than zero. The spectrogram shows the hyperbolic decay in time of the frequencies, but, since the filter makes an heavy attenuation and not a full cut, the harmonics are still present and visible. This sound is perceived more as percussive rather than “pizzicato”.

The third example (figure 7) is again a flute sample, but played with a vibrato effect, processed with a low-pass window with minimum value set to a value greater than zero. The same considerations of the previous example can be done. In this case more harmonics can be seen along their “wave” motion due to the vibrato.

In figure 8 a comparison between a true pizzicato and a synthetic pizzicato is shown. The two signals sound different. The synthetic hold all the effects bound to the original sound, like the bow and string interaction that does not exist in the true pizzicato.

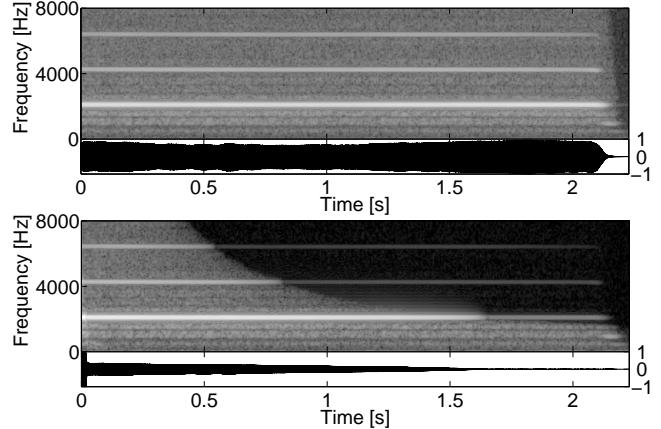


Figure 6: Flute sample. Original sample (top) and filtered (bottom).

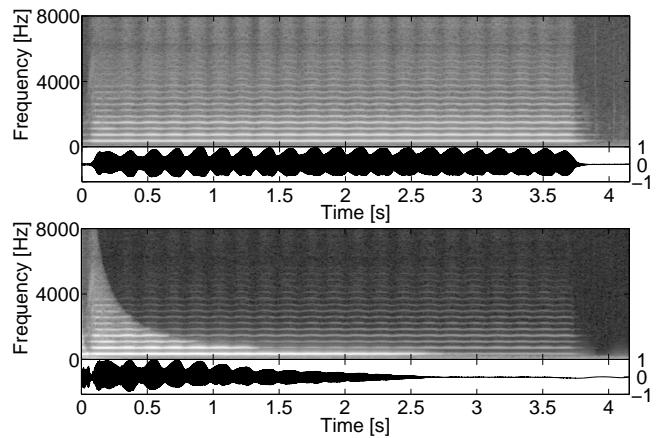


Figure 7: Flute sample played with vibrato effect. Original sample (top) and filtered (bottom).

The spectrogram shows that the actual pizzicato does not have a perfect hyperbolic decay in the time-frequency plane, even though a hyperbolic envelope can be traced over it.

All the examples shown that the artificial nature of the “Mellin pizzicato” can be immediately recognized if compared to a real pizzicato, especially with the help of time-frequency analysis. It is rather obvious that the real pizzicato decays in a more complex way. However, the artificial pizzicato gives the listener a strong sensation of pizzicato (or percussive sound). Other experiments have been done but are not presented in this paper, like an application of the pizzicato to vowels, thus producing an effect similar to plucking a string in the vocal tract.

5. CONCLUSIONS

The results obtained in this paper show how digital audio effects can be built in the Mellin/scale domain. In particular, this work shows how to simulate a “pizzicato” effect using only the scale domain. The results are satisfactory, in fact the artificial pizzicato gives to the listener an heavy sensation of pizzicato (or percussive

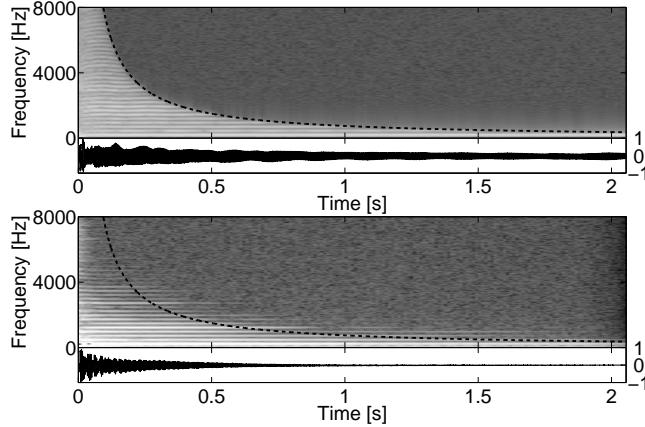


Figure 8: Synthetic pizzicato (top) compared to real pizzicato (bottom). The idealized hyperbolic envelope of cutoff frequencies is drawn in dashed line.

sound). A simple low-pass coupled with a band-enhancer in scale domain have been used to generate this effects. The entire process needs only few parameters that can be chosen with respect to the kind of damping and attack to simulate. Again, the first aim of this experiment is the exploration of the scale domain, in particular how to work using scale instead of more classical approaches, thinking at the scale like a physical attribute of the signal, interpreted like a joint time-frequency dimension.

6. REFERENCES

- [1] A. De Sena and D. Rocchesso, “A fast Mellin transform with applications in DAFX,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, Oct. 2004, pp. 65–69.
- [2] L. Cohen, “The scale representation,” *IEEE Trans. Sig. Proc.*, vol. 41, no. 12, pp. 3275–3291, Dec. 1993.
- [3] H. Sundaram, S. Joshi, and R. Bhatt, “Scale periodicity and its sampling theorem,” *IEEE Trans. Sig. Proc.*, vol. 45, no. 7, pp. 1862–1864, July 1997.
- [4] M. Unser, “Splines: A perfect fit for signal and image processing,” *Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, Nov. 1999.
- [5] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, “Discrete-time modelling of musical instruments,” *Report on Progress in Physics*, vol. 69, no. 1, pp. 1–78, Nov. 2006.
- [6] J. Bertrand, P. Bertrand, and J. Ovarlez, “The Mellin transform,” in *The Transforms and Applications Handbook*, A. Papoulis, Ed. CRC Press LLC, Boca Raton, Florida, U.S. in cooperation with IEEE Press, 2000, pp. (11–53)–(11–54).
- [7] D. Arfib, F. Keiler, and U. Zölzer, “Source-filter processing,” in *DAFX – Digital Audio Effects*, U. Zölzer, Ed. J. Wiley & Sons, 2002, pp. 361–362.
- [8] X. Amatrian, J. Bonada, A. Loscos, and X. Serra, “Spectral processing,” in *DAFX – Digital Audio Effects*, U. Zölzer, Ed. J. Wiley & Sons, 2002, pp. 429–431.

FRACTAL MODULATION EFFECTS

Gianpaolo Evangelista

Sound Technology, Digital Media Division
 Department of Science and Technology (ITN)
 Linköping Institute of Technology (LiTH), Norrköping, Sweden
 giae@itn.liu.se

ABSTRACT

Fractal modulation is obtained by forming a power weighted superposition of scaled and modulated versions of the signal. The resulting signal is self-similar with fractal characteristics. In this paper we explore fractal modulation as a powerful method to generate rich signals, useful both for the synthesis of complex sounds, like the sounds from natural events or ecological sounds, or as control functions of audio effects. The wavelet transform can be used as an efficient tool in order to generate a subset of fractal modulated signals that are power homogeneous. Any signal used as a seed for fractal modulation is transformed into a multiscale sound by means of a tree-structured multirate filter bank. Moreover, by superimposing a structured modulation scheme one can generate pseudo-periodic sounds whose partials have fractal behavior.

1. INTRODUCTION

Introduced in the context of communication, fractal modulation allows for redundant transmission of the same signal at different scales, useful for robust detection in a fading channel scenario or for information hiding, since the characteristics of the fractal modulated signal have a tendency to be more noise-like than the original signal. In this paper we explore fractal modulation as a powerful method to generate rich signals, useful both for the synthesis of complex sounds, like the sounds from natural events or ecological sounds, or as control functions of audio effects.

The fractal modulation scheme is somehow inspired by the construction of the self-similar function known as the Weierstrass cosine

$$w(t) = \sum_{n=0}^{+\infty} \gamma^n \cos 2\pi a^n t \quad (1)$$

in which infinite scaled versions of the same cosine function are weighted by a power of γ and added together, where $0 < \gamma < 1 < a$ and $a\gamma \geq 1$. The result is a nowhere differentiable sound whose partials are further and further spaced away, with their frequencies exponentially growing. The Weierstrass cosine is approximately self-similar in the sense that

$$w(at) = \frac{1}{\gamma} (w(t) - \cos 2\pi t) \quad (2)$$

so that, time-scaling the function by a factor a obtains the same function $w(t)$, except for a smooth $\cos 2\pi t$ additional term and an amplitude scaling factor γ . The graph of the Weierstrass cosine, shown in Figure 1, is a fractal with box counting dimension $D = 2 + \frac{\log \gamma}{\log a}$. Notice that γ controls the fractal dimension: D approaches 2 when γ approaches 1 and decreases with γ .

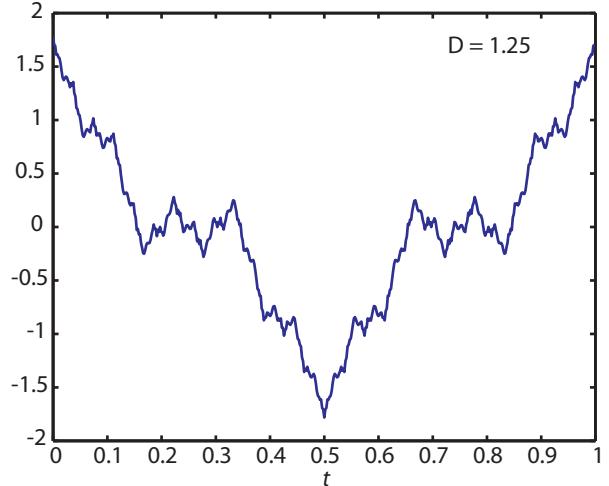


Figure 1: Weierstrass cosine function with box counting dimension $D = 1.25$.

2. FRACTAL MODULATION

Similar to the construction of the Weierstrass cosine, fractal modulation employs scaled versions of a signal in order to build a self-similar signal. Moreover, the scaled versions of the signal are modulated to proper band, which is implicit in scaling the cosine functions. The original signal can be considered as a “seed” for the generation of the fractal.

At least in a formal sense, one can construct self-similar signals by adding together a countable number of scaled and modulated versions of the seed. We assume that $a > 1$ and, usually, $1 < a \leq 2$. Suppose that our seed signal $x(t)$ is bandlimited to $[-(a-1)\pi, +(a-1)\pi]$, i.e., that its Fourier transform $X(\omega)$ is zero for ω outside this interval. We will use the notation $x(t) \in BL_{[A,B]}$ to specify bandlimited signals in the interval $[A, B]$.

By the Fourier scaling theorem, the scaled signal

$$x_n(t) = \frac{1}{a^n} x\left(\frac{t}{a^n}\right) \quad (3)$$

has Fourier transform

$$X_n(\omega) = X(a^n \omega) \quad (4)$$

and, therefore,

$$x_n(t) \in BL\left[-\frac{(a-1)\pi}{a^n}, +\frac{(a-1)\pi}{a^n}\right].$$

The peculiar modulation scheme we are going to employ is shown in Figure 2. The scaled signal is split into two subbands, one covering negative frequencies and the other one covering positive frequencies:

$$\begin{aligned} X_-(\omega) &= X(\omega)\chi_{[-(a-1)\pi, 0]}(\omega) \\ X_+(\omega) &= X(\omega)\chi_{[0, +(a-1)\pi]}(\omega) \end{aligned} \quad (5)$$

where $\chi_{[A, B]}(\omega)$ is the characteristic function of the semiclosed interval $[A, B]$, i.e.

$$\chi_{[A, B]}(\omega) = \begin{cases} 1 & A < \omega \leq B \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The negative frequency band $X_-(\omega)$ is modulated to positive frequencies while the positive frequency band $X_+(\omega)$ is modulated to negative frequencies. In this way one obtains, for each scaled and modulated version of the signal, a real signal whose effective bandwidth is the same as the baseband signal. The type of modulation is somehow arbitrary; the reason for this given choice will become apparent in the next Section. The choice of modulating frequencies

$$\omega_n^\mp = \pm \frac{\pi}{a^{n-1}} \quad (7)$$

guarantees that, ideally, the scaled and modulated signals do not overlap in the frequency domain. In fact, the scaled and modulated signals have disjoint Fourier transforms for any n :

$$\begin{aligned} X_-(a^n \omega - a\pi) &= X(a^n \omega - a\pi)\chi_{[\frac{\pi}{a^n}, +\frac{\pi}{a^{n-1}}]}(\omega) \\ X_+(a^n \omega + a\pi) &= X(a^n \omega + a\pi)\chi_{[-\frac{\pi}{a^{n-1}}, -\frac{\pi}{a^n}]}(\omega) \end{aligned} \quad (8)$$

In the frequency domain, the self-similar signal $s(t)$ is obtained as a superposition of the signals in (8). For $\omega > 0$ we can define the Fourier transform of the self-similar signal as follows:

$$S(\omega) = \sum_{n=-\infty}^{+\infty} \beta^n X(a^n \omega - a\pi)\chi_{[\frac{\pi}{a^n}, +\frac{\pi}{a^{n-1}}]}(\omega), \quad (9)$$

where $\beta > 1$ is an arbitrary parameter. Actually β can be expressed in terms of a as follows:

$$\beta = a^\gamma; \quad \gamma = \log_a \beta \quad (10)$$

For $\omega < 0$ the Fourier transform of the self-similar signal can be obtained from the relationship

$$S(-\omega) = S^*(\omega), \quad (11)$$

where the symbol $*$ denotes complex conjugation. Notice that since the terms of the sum in (9) do not overlap, for any given frequency there is only one nonzero term, hence the sum always converges, with a possible singularity at zero frequency if the original signal has a DC component.

The signal $s(t)$ obtained by taking the inverse Fourier transform of (9) is indeed self-similar with respect to scaling by powers of a . In fact, taking (10) into account and substituting $\frac{\omega}{a^K}$ for ω in (9), by performing a simple index change in the summation one obtains:

$$S\left(\frac{\omega}{a^K}\right) = a^{K\gamma} S(\omega). \quad (12)$$

for any $K \in \mathbb{Z}$. Therefore, in the time domain we have:

$$s\left(a^K t\right) = a^{K(\gamma-1)} s(t), \quad (13)$$

which is the required self-similarity property.

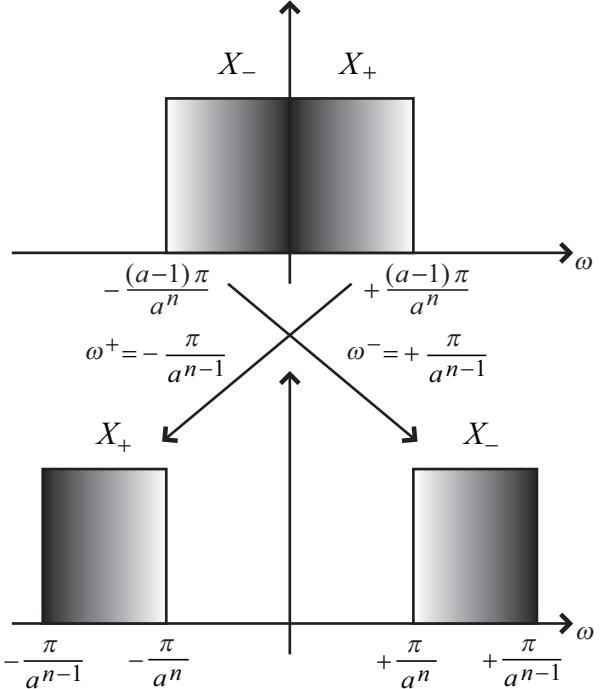


Figure 2: Modulation scheme employed for the scaled signals.

3. WAVELET BASED FRACTAL MODULATION

The frequency band subdivision of the scaling-modulation scheme to form self-similar signals illustrated in the previous Section is reminiscent of the structure of wavelet series. Indeed a wavelet based method for fractal modulation was introduced by Wornell and Oppenheim [1]. The simplest form of wavelet series [2, 3] is the expansion

$$s(t) = \sum_{n,m \in \mathbb{Z}} d_n(m) \psi_{n,m}(t) \quad (14)$$

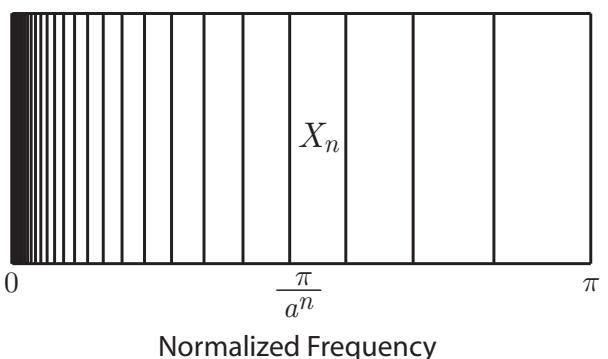


Figure 3: Frequency band subdivision of the scaled and modulated versions of the signal.

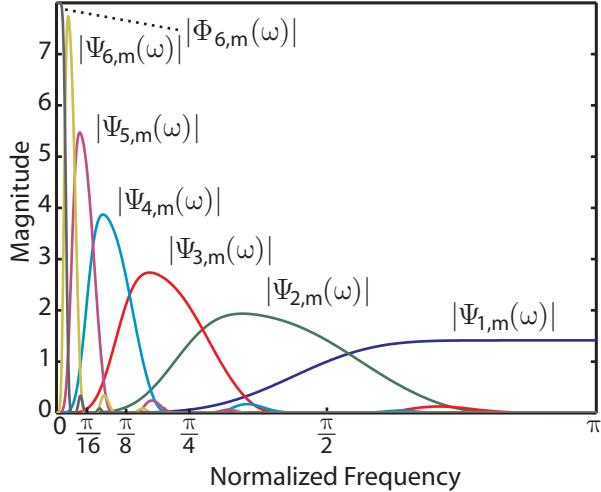


Figure 4: Magnitude Fourier transform of dyadic wavelets.

of a signal $s(t)$ in terms of a set of functions obtained by time-shifting and scaling a unique function, e.g., $\psi_{0,0}(t)$:

$$\psi_{n,m}(t) = \frac{1}{\sqrt{2^n}} \psi_{0,0} \left(\frac{t}{2^n} - m \right); \quad n, m \in \mathbf{Z}, \quad (15)$$

where the scaling factor is $a = 2$. As shown in Figure 4, the dyadic wavelets have bandwidth one octave, with bandwidth decreasing with the scale index n .

Theoretically the wavelets form the tessellation of the time-frequency plane shown in Figure 5, with higher frequency resolution at large scales and lower frequency resolution at small scales. In practice the cells overlap since the time-frequency uncertainty of the wavelets is larger than the time-frequency sample spacing.

For the orthogonal wavelet bases, the expansion coefficients are obtained by computing the scalar product of the signal with the basis functions $\psi_{n,m}(t)$:

$$d_n(m) = \int_{-\infty}^{+\infty} s(t) \psi_{n,m}^*(t) dt \quad (16)$$

Notice that, by performing the variable change $t = 2^n \tau$ in the integral in (16) and observing (15), one obtains:

$$d_n(m) = \sqrt{2^n} \int_{-\infty}^{+\infty} s(2^n \tau) \psi_{0,m}^*(\tau) d\tau \quad (17)$$

Suppose that the signal $s(t)$ is dyadically self-similar, i.e., that (13) is satisfied with $a = 2$, then:

$$d_n(m) = 2^{n(\gamma - \frac{1}{2})} d_0(m), \quad (18)$$

i.e., the expansion coefficients of a dyadic self-similar signal on a dyadic wavelet basis are obtained by scaling the amplitude of one and the same coefficient sequence $d_0(m)$. Vice versa, if the amplitude scaled versions

$$2^{n(\gamma - \frac{1}{2})} x(m)$$

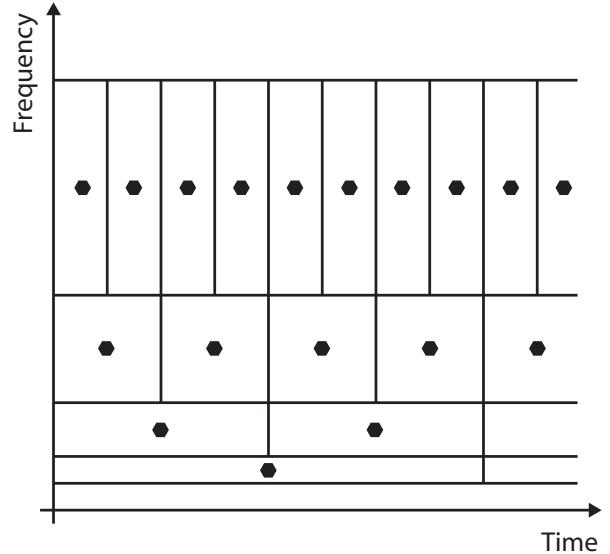


Figure 5: Time-Frequency cells corresponding to dyadic wavelets.

of a unique discrete-time ‘seed’ signal $x(m)$ are employed as wavelet coefficients, the self-similar signal

$$s(t) = \sum_{m \in \mathbf{Z}} x(m) \sum_{n \in \mathbf{Z}} 2^{n(\gamma - 1)} \psi_{0,0} \left(\frac{t}{2^n} - m \right) \quad (19)$$

is obtained. In fact, one can immediately verify that $s(t)$ defined by (19) satisfies (13) with $a = 2$. Indeed, the functions

$$\xi_m(t) = \sum_{n \in \mathbf{Z}} 2^{n(\gamma - 1)} \psi_{0,0} \left(\frac{t}{2^n} - m \right) = \sum_{n \in \mathbf{Z}} 2^{n(\gamma - \frac{1}{2})} \psi_{n,m}(t)$$

in the linear combination (19) are self-similar [1].

Intuitively, the wavelet way to fractal modulation is very efficient since, rather than producing scaled and modulated versions of a signal, one pre-scales the known basis functions, an operation that is implicit in the organization of wavelet bases. Moreover, the wavelet series synthesis of a signal can be achieved by means of an iterated two-channel filter bank, where, in each stage, the inputs are upsampled by a factor 2 and a lowpass filter $H(z)$ and a highpass filter $G(z)$ forming a quadrature mirror filter pair [4] are applied. The diagram in Figure 6 serves as a fast generator of self-similar signals.

The computational complexity of the wavelet based fractal modulation scheme depends on the algorithm employed to compute the wavelet transform. However, as we will see, the overall cost is linear with the number of generated samples.

One method to compute fractal modulation is to pre-compute the wavelets at each scale and multiply each of them by the samples of the seed signal. The output is formed by properly time-shifting and adding the pre-multiplied wavelet components. We will refer to this algorithm as the overlap-add method. In this case the complexity can be estimated at N operations per output sample, where N is the number of scales. In fact, the length of the scale n wavelet can be roughly estimated at $2^{n-1}L$ samples obtained as a cascade of upsampling and filtering operators where the order of the filters $L - 1$ is an odd integer for orthogonal wavelets.

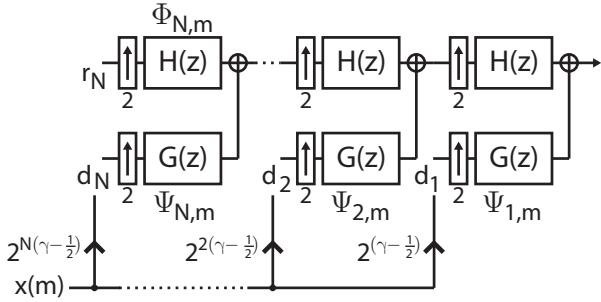


Figure 6: Wavelet scheme for fractal modulation: an iterated quadrature mirror synthesis filter bank is fed by the same sequence $x(m)$ at any branch. The residue coefficients $r_N(m)$ are arbitrary.

Thus, in order to generate an output signal of $2^{N-1}L$ samples one needs to multiply a single largest scale wavelet by one sample of the seed sequence; the two next to largest scale wavelets are each multiplied by one samples of the seed sequence and so forth. In general, computation at scale n requires multiplication of 2^{N-n} coefficients by the length $2^{N-1}L$ wavelets. This requires an order of $2^{N-1}L$ operations independently of scale. Therefore the total cost is $2^{N-1}LN$ operations in order to produce $2^{N-1}L$ output samples, which requires a rate of N operations per output sample.

A computational cost of the same order of magnitude is obtained when the scheme in Figure 6 is directly implemented. In this case one can show that the required rate of operations is proportional to the filter length L and does not depend on the number of scales N . It must be pointed out that, for sufficiently large filter orders, FFT based algorithms for the implementation of the filters can be employed to further reduce the overall computational cost.

Due to the finite number of scales obtained by a finite depth filter bank, the synthesized signal is only approximately self-similar. The residue coefficients $r_N(m)$, also known as the coefficients of the scaling function, are arbitrary. If a sufficient number of scales, corresponding to a sufficient number of stages of the filter bank, are employed then the scaling residue represents a very low-frequency signal and it can be disregarded in the synthesis of audio signals. However, in the next Section we will propose the use of self-similar signals also as low-frequency controlling functions of audio effects or as modulators of partials, in which case the scaling residue can be associated either to an amplitude scaled version of the seed signal or to an envelope function.

An important property of the wavelet based self-similar signal generator is found in the frequency domain:

$$S(\omega) = \sum_{n \in \mathbb{Z}} 2^{n\gamma} X(e^{j2^n\omega}) \Psi_{0,0}(2^n\omega), \quad (20)$$

where $X(e^{j\omega})$ denotes the DTFT of the seed signal $x(m)$. Equation (20) and Figure 7 show that the frequency spectrum of $x(m)$ is (periodically) contracted by a factor 2^n and directly weights the essential octave band of the corresponding wavelet at scale n , contributing to the frequency spectrum of the self-similar signal for the interval $[\frac{\pi}{2^n}, \frac{\pi}{2^{n-1}}]$ and its negative frequency mirror.

The type of crossed modulation illustrated in Section 2 emerges in wavelet based fractal modulation: the scaled negative frequency sideband of the seed signal contributes to the spectrum of the self-similar signal in the positive frequencies. Indeed, it is possible to

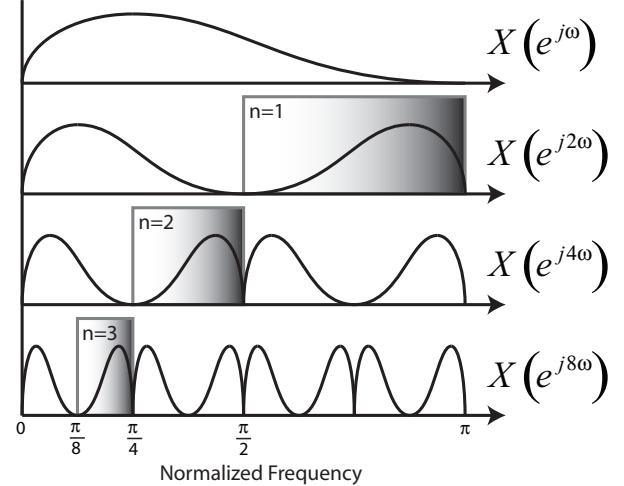


Figure 7: Frequency spectrum weighting of the wavelet bands (rectangles) by the DTFT of the seed signal.

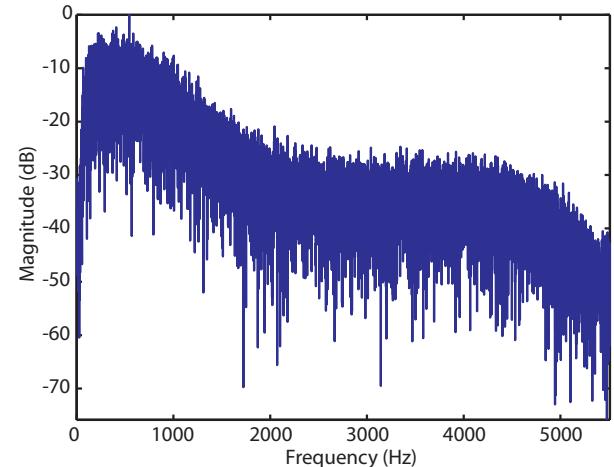


Figure 8: Frequency spectrum of ocean wave impact sound.

show that the *sinc* wavelets

$$\psi_{n,m}(t) = \frac{1}{\sqrt{2^n}} \cos\left(\frac{3\pi}{2}\left(\frac{t}{2^n} - m\right)\right) \text{sinc}\left(\frac{1}{2}\left(\frac{t}{2^n} - m\right)\right),$$

which are based on an ideal rectangular frequency subdivision, form an orthogonal set of wavelets. However, the time localization of the ideal wavelets is very poor and a wide class of regular orthogonal wavelets exist which have a better time-frequency behavior [3].

4. AUDIO EFFECTS

Interesting and natural sound textures can be produced by means of fractal modulation. As we showed in the previous Section, the wavelet way to fractal modulation has a low computational cost, which is linear in the number of output samples. The sounds and effects obtained are so different from those produced by conventional, e.g., AM and FM, modulation techniques that an attempt to

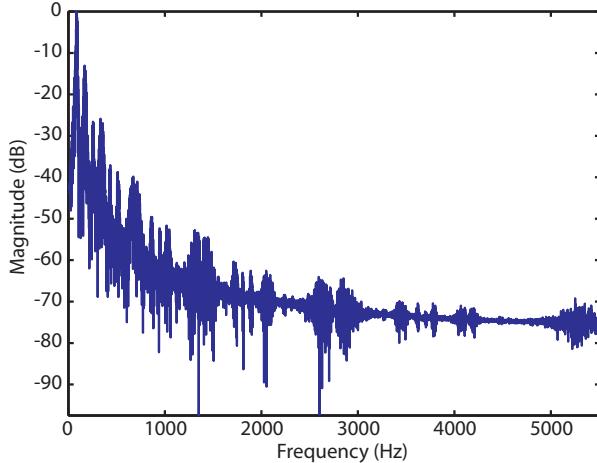


Figure 9: Frequency spectrum of fractal modulation of the ocean wave impact sound, with $\gamma = 3$ and $N = 7$ scales.

compare them with fractal modulation is totally meaningless both from a computational and qualitative point of view.

4.1. Dense Textures

In the example of Figure 9 we applied the wavelet based fractal modulation scheme to the impact sound of an ocean wave whose frequency spectrum is reported in Figure 8. For the computation we used $N = 7$ dyadic scales with Daubechies wavelets of degree 11 [3]. The frequency spectrum of the transformed sound shows a highly structured organization, together with an underlying $1/f$ decay. The modulated sound has a quite apocalyptic flavor, with multivoiced inserts originating from the small scale components and with rhythmic and reverberating patterns originating from the large scale components.

The parameter γ controls the overall spectral decay: large values ($\gamma \gg 1$) move the spectral centroid to low frequencies, while small values of ($\gamma \approx 1$) tend to produce brighter sounds with flatter spectra, closer to white noise. The parameter γ is indeed inversely related to brightness, as it can be seen from equation (20). For large values of γ a much higher weight is associated to the large scale (low frequency) wavelets while for $\gamma < 1$ the small scale (higher frequency) wavelets take the lead.

Sound examples using the ocean wave sound as seed can be found at the URL: <http://www.itn.liu.se/~giaev/soundexamples.html>, where fractal modulation is applied for various values of the γ parameter, showing a corresponding shifting of the spectral center of mass of the sound.

4.2. Sparse Textures

The ocean wave sound has the structure of several fine scale events with an overall swooshing evolution. This type of structure is interesting for the seed to generate complex sounds emulating a large number of concurrent sources. On the other extreme, the application of fractal modulation to sounds with sparser textures, such as the cracking noise of fire, produces an augmented rhythmic texture that lies halfway from the original sound to the sound of dripping water, as it can be appreciated from the corresponding example

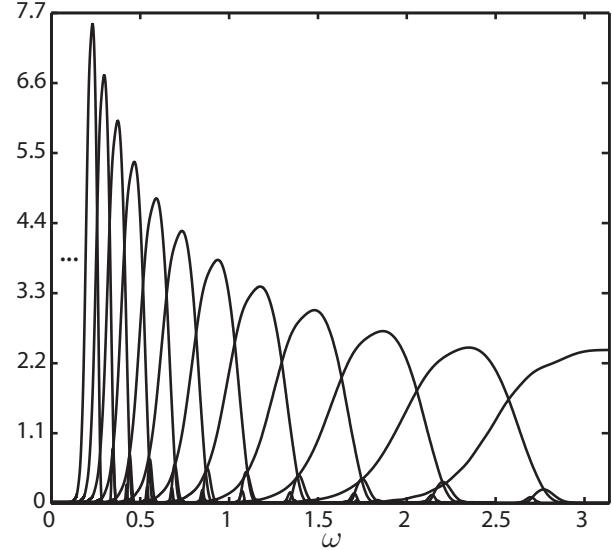


Figure 10: Magnitude Fourier transform of frequency warped wavelets with $1/3$ octave frequency resolution.

available at the URL <http://www.itn.liu.se/~giaev/soundexamples.html>.

In sparse textures it is important or even critical to control the number of scales associated with fractal modulation. Limiting the number of scales achieves sounds where the original rhythmic pattern can still be distinguished, while a multiscale structure is superimposed. At the given URL we show patterns generated using $N = 7$ and $N = 3$ scales, which have a different flavor in terms of sharpness of attack and reverberation.

4.3. Extensions

A limitation of the wavelet method for fractal modulation lies in the fact that only dyadic self-similarity with $a = 2$ can be implemented. As a result, the frequency spectrum is organized by octave bands, which can be too coarse for spectrum modeling. However, orthogonal wavelets having rational scale factor a have been devised [5] based on rational sampling rate filter banks. Alternately, arbitrary scale wavelets, shown in Figure 10, have been defined based on iterated frequency warping schemes [6, 7, 8], which are suitable for richer fractal modulation.

Frequency warping can also be used in order to modify the perceived pitch, if any, of the sounds generated by fractal modulation. In an example that can be found at our URL, frequency warping is employed in order to play a short melody with the fractal modulated ocean wave, which renders the acoustics of a stadium-like crowd singing in an unorganized or spontaneous choir. An efficient approximated frequency warping algorithm can be found in [9], which allows for real-time computation.

4.4. Dynamic Fractal Modulation

The fractal modulation parameter γ can be dynamically and interactively changed by suitably scaling the seed signal. Since the sampling rate is different for each wavelet component, suitably downsampled amplitude envelopes must be considered. However,

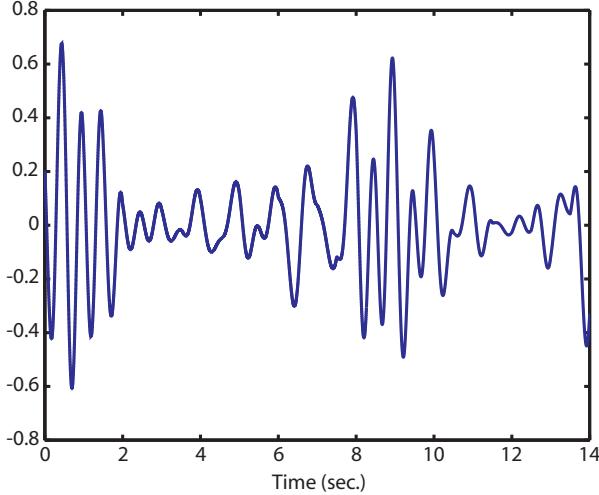


Figure 11: *Pseudo-sinusoid obtained by low-frequency cosine modulated fractal modulation.*

as the low-frequency components are generated by coefficient sequences at a very low sampling rate, excessive downsampling can result in audible distortion of the time envelopes. A more costly solution for higher fidelity time modulation of the γ parameter can be realized by amplitude scaling the synthesized wavelet components at the same sampling rate as the output signal. In the overlap-add method this can be achieved by the pre-computed shifted wavelets by the proper time-varying factor $2^n \gamma(k)$, where $\gamma(k)$ is the time-varying fractal modulation parameter. Interesting brightness transition effects can be realized in his fashion.

4.5. Fractal Modulated Low Frequency Oscillator

Modulated fractal noise has been employed in the Fractal Additive Synthesis technique [10, 11, 12] in order to model the $1/f$ behavior of the harmonic partials in the sounds of natural instruments. There, the seed signals are given by independent colored noise sources for each wavelet scale. Parallel wavelet synthesis structures are cascaded with a cosine modulated filter bank (MDCT) in order to synthesize the partials, where each wavelet section controls a single sideband of a partial. The same scheme but with deterministic signals replacing the colored noise sources can be employed in order to generate self-similar pseudo-periodic sounds.

Using only the lower frequency fraction of the MDCT, one can generate low-frequency pseudo-sinusoids with fractal fluctuations. These signals with a pseudo-random behavior can prove interesting as control functions of audio effects such as phaser, flanger, chorus, vibrato, e.g., by replacing the traditional LFO with a fractal LFO, whose oscillations are shown in Figure 11, where the main sinusoidal behavior, controlled by the scaling residue sequence $r_N(m)$ of Figure 6, is not included in the signal in order to better display the fluctuations. The introduction of a time-varying fractal modulation parameter allows us to dynamically control the depth of the effect, ranging from pure sinusoidal to extremely erratic behavior.

5. CONCLUSIONS

In this paper we explored the use of fractal modulation for the synthesis of complex sounds and of control functions to be employed in audio effects. The technique is very promising and efficient and it allows us to control complex behavior with a very small amount of parameters. Textured sounds with multiscale organization are typically generated by applying fractal modulation, where the small scale similar components produce voiced patterns and the large scale similar components produce rhythmic patterns.

While further experimentation is needed in order to assess the aesthetical value of the proposed modulation scheme in musical composition, simple examples serving the scope of appreciating the new sonority can be found at our URL.

6. REFERENCES

- [1] G. W. Wornell and A. V. Oppenheim, "Wavelet-based representations for a class of self-similar signals with applications to fractal modulation," *IEEE Trans. Information Theory*, vol. 38, no. 2, pp. 785–800, Mar. 1992.
- [2] S. Mallat, *A Wavelet Tour of Signal Processing*. Cambridge, MA: Academic Press, 1997.
- [3] I. Daubechies, *Ten Lectures on Wavelets*, ser. CBMS-NSF Reg. Conf. Series in Applied Math. Philadelphia: SIAM, 1992, vol. 61.
- [4] P. Vaidyanathan, *Multirate Systems and Filter Banks*. Upper Saddle River NJ: Prentice Hall Signal Processing Series, 1993.
- [5] T. Blu, "A new design algorithm for two-band orthonormal rational filter banks and orthonormal rational wavelets," *IEEE Trans. Sig. Proc.*, vol. 46, no. 6, pp. 1494–1503, June 1998.
- [6] G. Evangelista and S. Cavaliere, "Frequency warped filter banks and wavelet transform: A discrete-time approach via Laguerre expansions," *IEEE Trans. Sig. Proc.*, vol. 46, no. 10, pp. 2638–2650, Oct. 1998.
- [7] ———, "Discrete frequency warped wavelets: Theory and applications," *IEEE Trans. Sig. Proc.*, vol. 46, no. 4, pp. 874–885, Apr. 1998, special issue on Theory and Applications of Filter Banks and Wavelets.
- [8] G. Evangelista, "Dyadic warped wavelets," *Adv. in Imaging and Electron Physics*, vol. 117, pp. 73–171, Apr. 2001.
- [9] G. Evangelista, I. Testa, and S. Cavaliere, "The role of frequency warping in physical models of sound sources," in *Proc. Forum Acusticum*, Budapest, Hungary, Aug. 2005, pp. 715–720.
- [10] P. Polotti and G. Evangelista, "Analysis and synthesis of pseudo-periodic $1/f$ -like noise by means of wavelets with applications to digital audio," *EURASIP J. on Applied Sig. Proc.*, vol. 2001, no. 1, pp. 1–14, Mar. 2001.
- [11] ———, "Fractal additive synthesis by means of harmonic-band wavelets," *Computer Music J.*, vol. 25, no. 3, pp. 22–37, 2001.
- [12] P. Polotti, "Fractal additive synthesis," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2003. [Online]. Available: <http://library.epfl.ch/theses/?nr=2711>

AN INTERDISCIPLINARY APPROACH TO AUDIO EFFECT CLASSIFICATION

Vincent Verfaillie

SPCL / CIRMMT, McGill University
Montréal, QC, Canada
vincent@music.mcgill.ca

Catherine Guastavino

GSLIS / CIRMMT, McGill University
Montréal, QC, Canada
Catherine.Guastavino@mcgill.ca

Caroline Traube

LIAM, Université de Montréal
Montréal, QC, Canada
caroline.traube@umontreal.ca

ABSTRACT

The aim of this paper is to propose an interdisciplinary classification of digital audio effects to facilitate communication and collaborations between DSP programmers, sound engineers, composers, performers and musicologists. After reviewing classifications reflecting technological, technical and perceptual points of view, we introduce a transverse classification to link discipline-specific classifications into a single network containing various layers of descriptors, ranging from low-level features to high-level features. Simple tools using the interdisciplinary classification are introduced to facilitate the navigation between effects, underlying techniques, perceptual attributes and semantic descriptors. Finally, concluding remarks on implications for teaching purposes and for the development of audio effects user interfaces based on perceptual features rather than technical parameters are presented.

1. INTRODUCTION

More than 70 different digital audio effects have previously been identified¹ ([1] and [2]). Digital audio effects are tools used by composers, performers and sound engineers, but are generally described from the standpoint of DSP engineers who designed them. They are therefore documented and classified in terms of the underlying techniques and technologies in both software documentations and textbooks.

However, other classification schemes are commonly used by different communities. These include perceptual classification [3, 2], signal processing classification [4, 5, 6, 7, 1], control type classification [8], and sound and music computing classification [9].

The comparison of these classifications reveal strong differences. Specifically, each classification was introduced to best meet the needs of a specific audience, thus relying on features that are relevant for a given community, but may be meaningless or obscure for a different community. For example, signal processing techniques are rarely presented according to perceptual features but rather according to acoustical dimensions. Conversely, composers usually rely on perceptual or cognitive features rather than acoustical dimensions.

¹This value (70) is highly dependent on the degree of refinement as well as the experience and goals of the user, and therefore highly subjective!

The main motivation for an interdisciplinary approach to audio effect classification is to facilitate communication between researchers and creators working on or with audio effects (e.g. DSP programmers, sound engineers, sound designers, electroacoustic music composers, performers using augmented or extended acoustic instruments or digital instruments, musicologists). The concerned disciplines are acoustics, electrical engineering, psychoacoustics, music cognition and psycholinguistics.

In the next section, vocabulary is clarified regarding audio effects vs sound effects and sound transformations. In section 3, we present the various standpoints on digital audio effects through a description of the communication chain in music. In section 4 we describe three discipline-specific classifications²: based on underlying techniques, control signals and perceptual attributes. In section 5, we introduce an interdisciplinary classifications linking the different layers of domain-specific descriptors in section 5, before concluding with pedagogical and human computer interfaces remarks in section 6.

2. SOME CLARIFICATION ON VOCABULARY

Audio Effect or Sound Effect? The word ‘effect’ denotes an impression produced in the mind of person, a change in perception resulting from a cause.

Sound effects and audio effects denote two related but different concepts. Sounds effects are sounds that affect us, whereas audio effects are transformations applied to sounds in order to modify how they affect us. Sound effect databases provide natural (recorded) and processed sounds (resulting from audio effects) that produce specific effects on perception used to simulate actions, interaction or emotions in various contexts (e.g. music, movie soundtracks).

In the field of audio effects, the meaning of ‘effect’ has shifted from the perception of a change to the processing technique used to produce this change, reflecting a semantic confusion between what is perceived (the effect of processing) and the signal processing technique used to achieve this effect.

Audio Effect or Sound Transformation? Distinctions between audio effects and sound transformations originated from technical

²It should be noted that the presented classifications are not classifications in the strict sense of the term since they are neither mutually exclusive (one effect can be classified in more than one class), nor exhaustive.

and/or historical considerations that distinguishes simple vs. complex and surface vs. deep processing.

Audio effects originally denoted simple processing systems based on simple mathematical operations, *e.g.* chorus by random control of delay line modulation; echo by a delay line; distortion by non-linear processing. It was assumed that audio effects process sound at its surface, since sound is represented by wave form samples (not a high-level sound model) and simply processed by delay lines, filters, gains, etc. Sound transformations, on the other hand, denoted complex processing systems based on analysis-synthesis models (high-level sound models such as additive and subtractive models). They were considered to offer deeper modifications, such as high-quality pitch-shifting with formant preservation, timbre morphing, and time-scaling with attack, pitch and panning preservation.

Over time, practice blurred the boundaries between audio effects and sound transformations. Indeed, sound transformations can be used to produce the same changes as audio effects but often at a higher processing cost (*ie.* panning or comb filtering in the spectral domain). Moreover, some audio effects considered as simple processing actually require complex processing. For instance, reverberation systems are usually considered as simple audio effects because they were originally developed using simple operations, even though they apply complex sound transformations. The surface/depth distinction is also confusing because it can refer to either the technique used or to the effect it has on perception. For example, a distortion effect is a simple surface transformation from a technical point of view, but has a strong impact on perception and could therefore be described as a deep transformation from the perceptual standpoint. Conversely, a time-scaling with pitch, formants and attack preservation requires complex techniques but is perceived as a simple and clear transformation by the listener.

Therefore, we consider that the terms ‘audio effects’, ‘sound transformations’ and ‘musical sound processing’ all refer to the same process: applying signal processing techniques to sounds in order to modify how they will be perceived, or in other words, to transform a sound into another sound with a different quality [10]. The different terms are often used interchangeably. For consistency sake, we will use ‘audio effects’ throughout this paper.

3. MULTIPLE STANDPOINTS ON AUDIO EFFECTS

Despite the variety of needs and standpoints, the technological terminology is predominantly employed by the actual users of audio effects : composers and performers. This technological classification might be the most rigorous and systematic one but it unfortunately only refers to the techniques used while ignoring our perception of the resulting audio effects, which seems more relevant in a musical context.

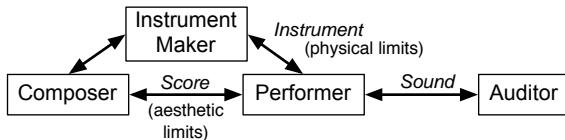


Figure 1: Communication chain in music. The composer, performer and instrument maker are also listeners, but in a different context than the auditor.

The communication chain in music [11, 12] essentially produces musical sounds. This concept has been adapted from

linguistics and semiology to music [13], based on [14], in a tripartite semiological scheme distinguishing three levels of musical communication between a composer (producer) and a listener (receiver) through a physical, neutral trace such as sounds. In order to investigate all possible standpoints on audio effects, we apply this scheme to a complete chain, as depicted on Fig. 1, including all actors intervening in the processes of the conception, creation and perception of music (instrument makers, composers, performers and listeners). The poietic level concerns the conception and creation of a *musical message* to which instrument makers, composers and performers participate in different ways and at different stages. The neutral level is that of the physical ‘trace’ (instruments, sounds or scores). The aesthetic level corresponds to the perception and reception of the *musical message* by a listener. In the case of audio effects, the instrument maker is the signal processing engineer who designs the effect and the performer is the user of the effect (musician, sound engineer). In the context of mixed music creation, composers, performers and instrument makers (music technologists) are usually distinct individuals who need to efficiently communicate with one another. But all actors in the chain are also listeners who can share descriptions of what they hear and how they interpret it. Therefore we will consider the perceptual and cognitive standpoints as the entrance point to the proposed interdisciplinary network of the various domain-specific classifications.

We also consider the specific case of electroacoustic music composers who often combine additional programming and performance skills. They conceive their own processing system, control and perform on their instruments. Although all production tasks are performed by a single multidisciplinary artist in this case, a transverse classification is still helpful to achieve a better awareness of the relations between the different levels of description of an audio effect (from technical to perceptual standpoints).

4. DISCIPLINE-SPECIFIC CLASSIFICATIONS

4.1. Classification Based on Underlying Techniques

The first classification we present is from the ‘instrument maker’ standpoint (DSP engineer or programmer). It is focused on the underlying techniques used to implement the audio effects. Many digitally implemented effects are in fact emulations of their analog ancestors³. We distinguish the following analog technologies [2]:

- mechanics/acoustics (*e.g.* musical instruments, effects due to room acoustics);
- electromechanics (*e.g.* vinyls: pitch-shifting by changing disk rotation speed);
- electromagnetics (*e.g.* magnetic tapes: flanging);
- electronics (*e.g.* filters, vocoder, ring modulators).

For example, flanging was originally obtained when pressing the thumb on the flange of a magnetophone wheel and is now emulated with digital comb filters with varying delays. Digital ring modulation (referring to the multiplication of two signals) borrows its name from the analog ring-shaped circuit of diodes originally used to implement this effect. Other digital effects emulating acoustical or perceptual properties of electromechanic, electric or electronic

³Similarly, some analog audio effects implemented with one technique were emulating audio effects already existing with another analog technique. At some point, analog and/or digital techniques were also creatively used so as to provide new effects.

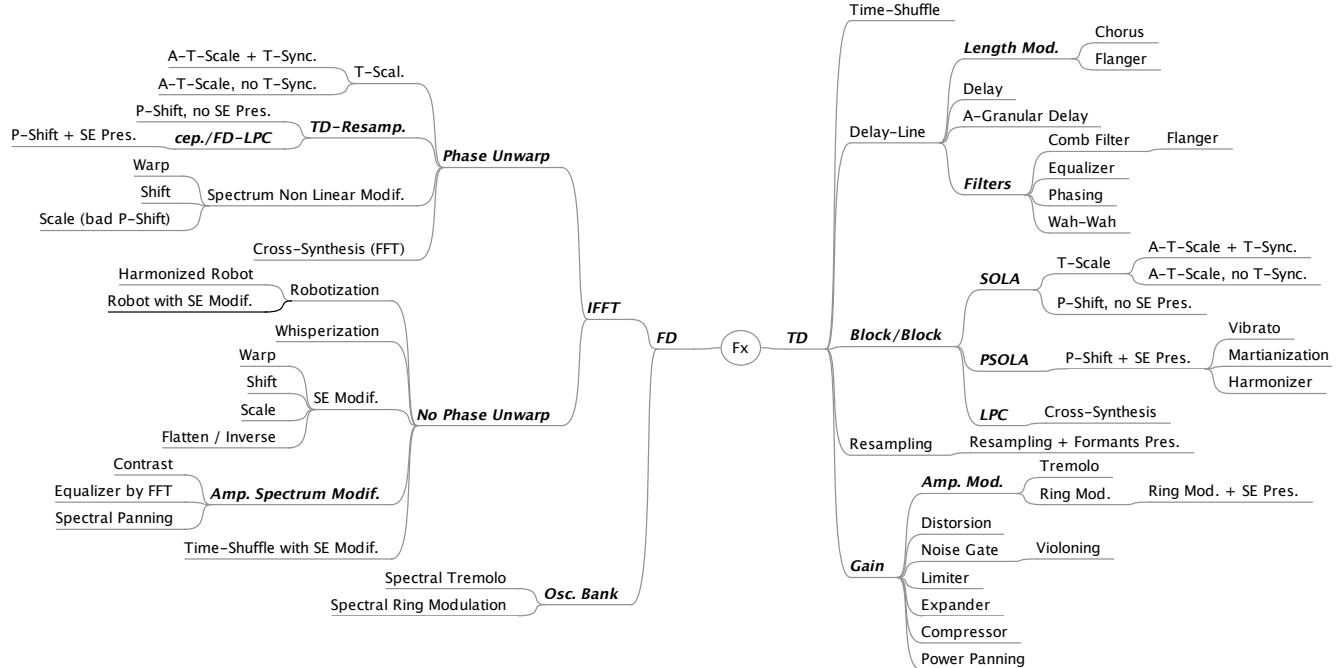


Figure 2: A technical classification of audio effects used to design a multi-effects system. 'TD' stands for 'time-domain', 'FD' for 'frequency domain', 't-scale' for 'time-scaling', 'p-shift' for 'pitch-shifting', '+' for 'with', 'A-' for adaptive control', 'SE' for 'spectral envelope', 'osc' for 'oscillator', 'mod.' for 'modulation' and 'modif.' for 'modification'. Bold italic font words denote technical aspects, whereas regular font words denote audio effects.

effects include filtering, the wah-wah effect, the vocoder effect, reverberation, echo and Leslie effect.

Digital audio effects can be organized on the basis of implementation techniques, as proposed in [1]:

- filters;
- delays (resampling);
- modulators and demodulators;
- nonlinear processing;
- spatial effects;
- time-segment processing e.g. SOLA, PSOLA [15, 16];
- time-frequency processing e.g. phase vocoder [17, 18];
- source-filter processing (e.g. LPC);
- spectral processing (e.g. SMS [19]);
- time and frequency warping.

As shown in this list, a sub-classification of digital audio effects can be based on the domain of application (time, frequency or time-frequency) and on whether the processing is performed sample-by-sample or block-by-block [2]:

- time domain: block processing, e.g. OLA, SOLA, PSOLA; sample processing: e.g. delay line, gain, non linear processing, resampling/interpolation;
- frequency domain (block processing): frequency domain synthesis (IFFT) e.g. phase vocoder with or without phase unwrapping; time domain synthesis (oscillator bank);
- time and frequency domain: e.g. phase vocoder and resampling, phase vocoder and LPC.

The advantage of a classification based on the underlying techniques is that the developer can easily see the technical and implementation similarities of various effects, thus simplifying understanding and implementation of multi-effect systems. However, some audio effects can be classified in more than one class.

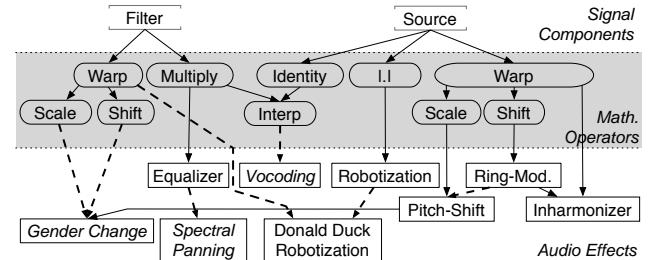


Figure 3: Technical classification of some source-filter effects based on signal components (source / filter) and applied mathematical operations (grey-shade background). Dashed bold lines and italic effect names indicate the use of adaptive control.

For instance, time-scaling can be performed with time-segment and time-frequency processing. Depending on the user expertise (DSP programmer, electroacoustic composer), this classification may not be the easiest to understand. In fact, this type of classification does not explicitly handle perceptual features.

Users may choose between several possible implementations of an effect depending on the artifacts of each effect. For instance, with time-scaling, resampling does not preserve pitch nor formants; OLA with circular buffer adds the window modulation and sounds rougher and filtered; phase vocoder sounds a bit reverberant, the 'sinusoidal + noise' additive model sounds good except for attacks, the 'sinusoidal + transients + noise' additive model preserves attacks, but not the spatial image of multi-channel sounds, etc. Therefore, in order to choose a technique, the user

must be aware of the audible artifact of each technique. The need to link implementation techniques to perceptual features (see 4.3) thus becomes clear.

As depicted in Fig. 2, some audio effects (*e.g.* adaptive time-scaling with time-synchronization [10]) can be performed with techniques from various domains (*e.g.* SOLA: block-by-block and time-domain; phase vocoder: block-by-block frequency domain with IFFT synthesis). This classification can be used to design a multi-effect system, as well as to provide an overview of technical domains and signal processing techniques involved in effects. However, it may not be relevant for the listener.

Another classification relies on the mathematical operation applied to the components of the sound representation used to process the sound. For example, with source-filter model based audio effects, the basic mathematical operations that can be applied to the filter, the source or both components [20] are scaling, shifting, warping, multiplying, identity and, interpolation (see Fig. 3).

4.2. Classification Based on the Type of Control

The second discipline-specific classification takes the standpoint of the composer and the performer, and is based on the type of control that audio effects offer or require [2, 8]: constant or variable. If variable, the control can be provided by a wave generator – periodic or low frequency oscillator (LFO): sinusoidal, triangular, exponential/logarithmic – or by arbitrary generators, such as gestural control (realtime user-defined control), automation (offline user-defined control) and adaptive (sound-defined control, using sound descriptors that represent musical gestures). This classification completes previous ones and appeals to performers, composers and developer. It can also help defining a general framework and designing new audio effects [2, 10].

4.3. Classification Based on Perceptual Attributes

Finally, audio effects can be classified according to the various perceptual attributes they modify:

- pitch: height and chroma; melody, intonation, contour, harmony, harmonicity; *glissando*;
- dynamics: nuances, phrasing (*legato*, and *pizzicato*), accents, *tremolo*;
- time: duration, tempo, rhythm *accelerando/decelerando*;
- space: localization (distance, azimuth, elevation) and sound motion; room effect (reverberation, echo); directivity;
- timbre: formants (color); brightness (or spectral height), quality; metamorphosis; texture, harmonicity; *vibrato*, *trill*, *flatterzunge*, *legato*, *pizzicato*.

This classification was introduced in the context of content-based transformations [3], and adaptive audio effects [2, 10].

For the main audio effects, Table 1 indicates the main and secondary perceptual attributes modified, along with complementary informations for programmers and users about real time implementation and control type. Another way to represent the various links between an effect and perceptual attributes uses a heuristic map [21] as depicted in Fig. 4, where audio effects are linked in the center to the main perceptual attribute modified. Other sub-attributes are introduced. For the sake of simplicity, audio effects are attached to the center only for the main modified perceptual attributes. When other attributes are slightly modified, they are indicated on the opposite side, *i.e.* at the figure bounds.

DAFx name	Main	Perceptual Attr. Other	RT	Control
compressor, limiter	L	T		A
expander, noise gate	L	T		A
gain/amplification	L		—	
normalisation	L		—	LFO
tremolo	L		—	A
violoning	L	T		
inversion	D	P,L,T	—	
time-scaling w/ formant pres.	D		—	
time-scaling w/ attack pres.	D		—	A
time-scaling w/ vibrato pres.	D		—	
rhythm/swing change	D	T	—	A
pitch-shifting w/ formant pres.	P			
pitch-shifting, no formant pres.	P	T		
pitch change	P			A
pitch discretization (auto-tune)	P	T		A
smart harmony	P			A
(in-)harmonizer	P			A
distance change	S	L,T		
directivity	S	P,T		
Doppler	S	L,P		
echo	S	L		
granular delay	S	L,D,P,T		A
panning	S	L,D,T		
reverberation	S	L,D,T		
spectral panning	S	L,T		
Rotary/Leslie	S	P,T		LFO
filter	T	L		
arbitrary resolution filter	T	L		
comb filtering	T	L,P		
equalizer	T	L		
resonant filter	T	L,P		
wha-wha	T	L,P		
auto-wha	T	L,D,P		LFO
envelope shifting	T	L		
envelope scaling	T	L		
envelope warping	T	L		
spectral centroid change	T	L		
chorus	T			random
flanger	T	P		LFO
phaser	T	P		
spectrum shifting	T	P		
adaptive ring modulation	T	P		A
texture change	T			
distortion	T	L,P		
fuzz	T	L,P		
overdrive	T	L,P		
mutation	T	L,P		cross-A
spectral interpolation	T	L,P		cross-A
vocoding	T	L,P		cross-A
cross-synthesis	T	L,P		cross-A
voice morphing	T	L,P		cross-A
timbral metamorphosis	T	L,P		cross-A
timbral morphing	T	L,P		cross-A
whispering/hoarseness	T	L	—	
de-esser	T	L		A
declicking	T	L		
denoising	T	L		
exciter	T	L		
enhancer	T	L		
spectral compressor	L,T			
gender change	PT	L		A
intonation change	L,P			A
martianisation	PT	L		A
prosody change	L,D,P		—	A
resampling	D,T	L,P	—	
ring modulation	P,T			
robotisation	P,T	L		
spectral tremolo	L,T	D		
spectral warping	T,P	L		
time shuffling	L,D,P,T		—	LFO
vibrato	L,P	T,D	—	LFO

Table 1: *Digital audio effects according to modified perceptual attributes (L: loudness, D: duration and rhythm, P: pitch and harmony, T: timbre and quality, S: space). We also indicate if realtime implementation (RT) is possible ('—' means 'not possible'), and built-in control type (A: adaptive, cross-A: cross-adaptive, LFO).*

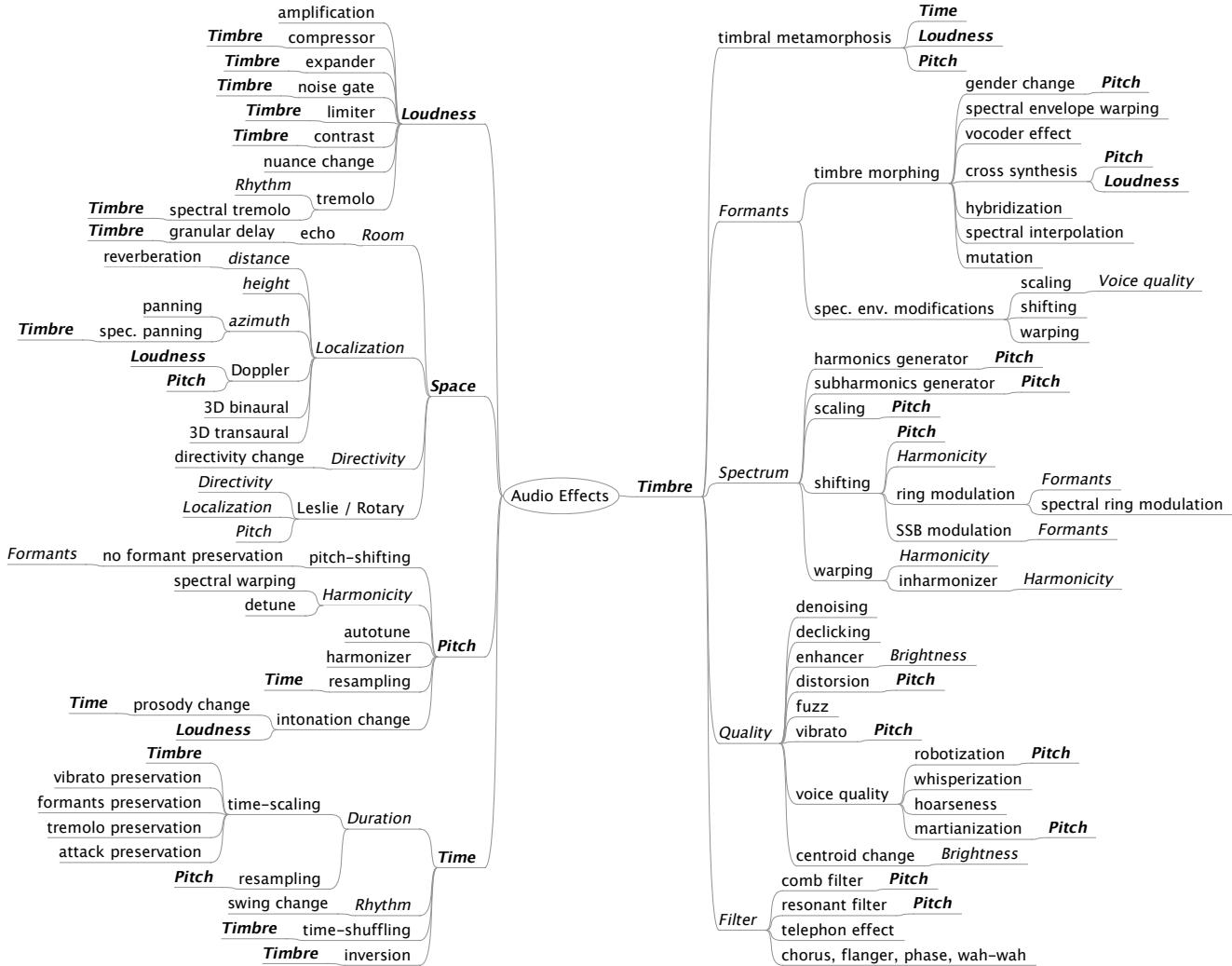


Figure 4: Perceptual classification of various audio effects. **Bold-italic words** are perceptual attributes (pitch, loudness, etc.). **Italic words** are perceptual sub-attributes (formants, harmonicity, etc.). Other words refer to the corresponding audio effects.

When other perceptual attributes are slightly modified by an audio effect, those links are not drawn to the center in order not to overload the heuristic map, but rather to the outer direction.

When the user chooses an effect to modify one attribute, the implementation technique used may introduce artifacts, implying modifications of other attributes. We differentiate the perceptual attributes that we primarily want to modify ('main' perceptual attributes, and the corresponding dominant modification perceived) and the 'secondary' perceptual attributes that are slightly modified (on purpose or as a by-product of the signal processing).

A perceptual classification has the advantage of presenting audio effects according to the way they are perceived, taking into account the audible artifacts of the implementation techniques. Of course, none of the proposed classifications is perfect, and depends on the goal we have in mind when using it. However, for sharing and spreading knowledge about audio effects between DSP programmers, composers and listeners, this classification offers a vocabulary dealing with our auditory perception of the sound produced by the audio effect, that we all share since we all are listeners.

in the communication chain (see section 1).

5. INTERDISCIPLINARY CLASSIFICATIONS

This section recalls sound effect classifications and then introduces a interdisciplinary classification linking the different layers of domain-specific descriptors.

5.1. Sound Effects Classifications

Sound effects have been thoroughly investigated in electroacoustic music. For instance, Schaeffer classified sounds according to matter (harmonicity and noisiness), form (dynamics) and variation (melodic profile) [22]. In the context of ecological acoustics, Schafer [23] introduced the idea that soundscapes reflect human activities. He proposed four main categories of environmental sounds: mechanical sounds (traffic and machines), human sounds (voices, footsteps), collective sounds (resulting from social activities) and sounds conveying information about the environment

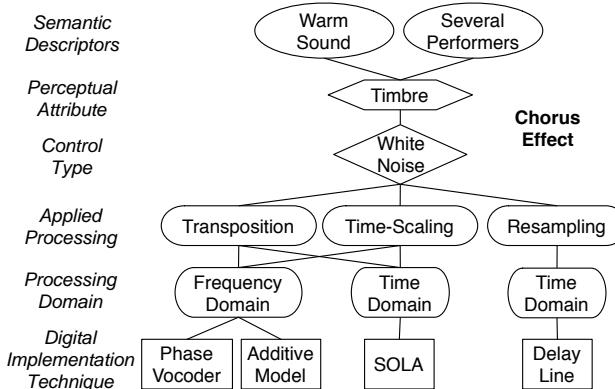


Figure 5: Transverse diagram for the chorus effect: it is applied by mixing modified versions of the signal (slight pitch-shifting, time-scaling and spectral envelope), either by using the usual white-noise modulated delay-line, or by using a model of sound (such as phase vocoder, SOLA or additive).

(warning signals or spatial effects).

Gaver [24] also introduced the distinction between musical listening and everyday listening. Musical listening focuses on perceptual attributes of the sound itself (*e.g.* pitch, loudness), whereas everyday listening focuses on events to gather relevant information about our environment (*e.g.* car approaching), that is, not about the sound itself but rather about sound sources and actions producing sound. Recent research on soundscape perception validated this view by showing that people organize familiar sounds on the basis of source identification. But there is also evidence that the same sound can give rise to different cognitive representations which integrate semantic features (*e.g.* meaning attributed to the sound) into physical characteristics of the acoustic signal (see [25] for a review). Therefore, semantic features must be taken into consideration when classifying sounds, but they cannot be matched with physical characteristics in a one-to-one relationship.

Similary, audio effects give rise to different semantic interpretations depending on how they are implemented or controlled. Semantic descriptors were investigated in the context of distortion [26] and different standpoints on reverberation were summarized in [27]. Our contribution is to propose a interdisciplinary classification of audio effects in an attempt to bridge the gaps between discipline-specific classifications by extending previous research on isolated audio effects.

5.2. Interdisciplinary Audio Effects Classification

The proposed interdisciplinary classification links the various layers of discipline-specific classifications presented in section 4, ranging from low-level to high-level features as follows: digital implementation technique, processing domain, applied processing, control type, perceptual attributes, and semantic descriptors. The first example concerns the chorus effect (Fig. 5). The usual implementation involves one or many delay lines, with modulated length and controlled by a white noise. An alternative and more realistic sounding implementation consists in using several slightly pitch-shifted and time-scaled versions of the same sound, and mixing them together. In this case, the resulting audio effect sounds more like a chorus of people or instruments playing the same harmonic and rhythmic patterns together. The second

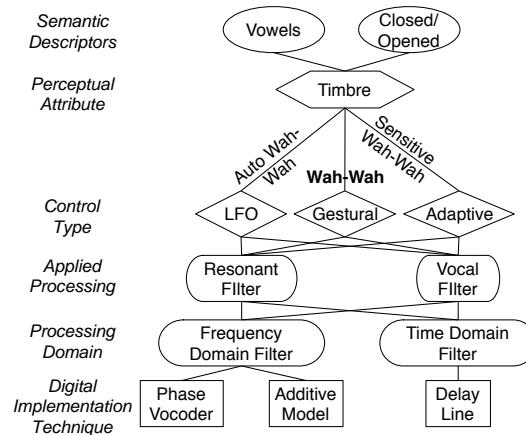


Figure 6: Transverse diagram for the wah-wah effect: the control type defines the effect's name, i.e. wah-wah, automatic wah-wah (with LFO) or sensitive wah-wah (adaptive control). The wah-wah effect itself can be applied using a delay line modulation technique, or a sound model (such as phase vocoder, additive).

example in Fig. 6 illustrates various control types for the wah-wah effect: gestural (with a foot pedal), periodic (with an LFO) and adaptive (controlled by the attack and a release time). We can see there the importance of specifying the control type as part of the effect definition. Another example is the delay line modulation, which results in a chorus when modulated by a white noise, and in a flanger when modulated by an LFO.

6. CONCLUSIONS

This articles summarizes various classifications of audio effects elaborated in different disciplinary fields. It then proposes an interdisciplinary classification linking the different layers of domain-specific features that aims to facilitate knowledge exchange between the fields of musical acoustics, signal processing, psychoacoustics and cognition.

Not only do we address the classification of audio effects, but we further explain the relationships between structural and control parameters of signal processing algorithms and the perceptual sub-attributes of timbre modified by audio effects. This interdisciplinary classification emerged from our experience using audio effects, and teaching them to engineers and musicians⁴. We are convinced that a wider range of audiences, including electronic music composers would benefit from this interdisciplinary approach [28]. Indeed, a generalization of this classification to all audio effects would have a strong impact on pedagogy, knowledge sharing across disciplinary fields and musical practice. For example, it is well known that DSP engineers conceive better tools when they know how it can be used in a musical context. Furthermore, linking perceptual features to signal processing techniques will enable the development of more intuitive user interfaces providing control over high-level perceptual and cognitive attributes rather than low-level signal parameters. An example of a system providing a

⁴C. Traube teaches sound synthesis and audio effects to electroacoustic composers at the *Université de Montréal*, Canada. V. Verfaillie teaches digital audio effects to computer and electrical engineers at the *ENSEIRB*, France. C. Guastavino teaches multimedia systems at *McGill University*.

mapping from perceptual attributes to audio effect control parameters is the Spatializer developed at IRCAM, where the user can manipulate perceptual attributes to control audio effects [29].

Similar systems are envisaged to generalize this approach to other audio effects. However, research on semantic descriptors, the higher-level layer in our interdisciplinary classification still remains at a very early stage. Further research is needed to investigate verbal descriptors and identify relationships between semantic features and relevant correlates of lower-level attributes.

Future directions also include the development of tools for communicating, visualizing, and navigating within the interdisciplinary classification of audio effects, as shown in the figures. Depending on users' needs and expertise, information for each effect can be represented at the level of one of the layers corresponding to domain-specific classifications, or using the interdisciplinary classification to navigate between layers. This information could be collaboratively collected using a Wiki, and organized as complex trees and networks, to allow for navigation and exploration. Users could easily locate features that are relevant to them for the task at hand. They could then accordingly identify groups of relevant audio effects, for instance on the basis of perceptual attribute(s) modified, underlying signal processing techniques, or type of control used, etc. This generalized classification system would best meet the needs of a wide range of users with differing strategies, goals and expertise.

7. REFERENCES

- [1] U. Zölzer, *DAFx – Digital Audio Effects*. J. Wiley & Sons, 2002.
- [2] V. Verfaille, "Effets audionumériques adaptatifs : Théorie, mise en œuvre et usage en création musicale numérique," Ph.D. dissertation, Université Aix-Marseille II, 2003.
- [3] X. Amatriain, J. Bonada, A. Loscos, J. L. Arcos, and V. Verfaille, "Content-based transformations," *J. New Music Research*, vol. 32, no. 1, pp. 95–114, 2003.
- [4] S. Orfanidis, *Introduction to Signal Processing*. Prentice Hall Int. Editions, 1996.
- [5] G. D. Poli, A. Picialli, S. T. Pop, and C. Roads, *Musical Signal Processing*. Eds. Swets & Zeitlinger, 1996.
- [6] C. Roads, *The Computer Music Tutorial*. Cambridge, Massachusetts: MIT Press, 1996.
- [7] F. R. Moore, *Elements of Computer Music*. Englewood Cliffs, N.J.: Prentice Hall, 1990.
- [8] V. Verfaille, M. M. Wanderley, and P. Depalle, "Mapping strategies for gestural control of adaptive digital audio effects," *J. New Music Research*, vol. 35, no. 1, pp. 71–93, 2006.
- [9] A. Camurri, G. D. Poli, and D. Rocchesso, "A taxonomy for sound and music computing," *Computer Music J.*, vol. 19, no. 2, pp. 4–5, 1995.
- [10] V. Verfaille, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (A-DAFx): A new class of sound transformations," *IEEE Trans. Audio, Speech and Language Proc.*, vol. 14, no. 5, pp. 1817–1831, 2006.
- [11] C. A. Rabassó, "L'improvisation: du langage musical au langage littéraire," *Intemporel: bulletin de la Société Nationale de Musique*, vol. 15, 1995, [Online] <http://catalogue.ircam.fr/hotes/snsm/itpr15rabatxt.html>.
- [12] D. Hargreaves, D. Miell, and R. MacDonald, "What do we mean by musical communication, and why is it important?", intro. of 'musical communication (part 1)', session, ICMPC CD-ROM," in *Proc. Int. Conf. on Music Perception and Cognition*, 2004.
- [13] J.-J. Nattiez, *Fondements d'une sémiologie de la musique*. Paris: U. G. E., Coll. 10/18, 1975.
- [14] J. Molino, "Fait musical et sémiologie de la musique," *Musique en jeu*, vol. 17, pp. 37–62, 1975.
- [15] E. Moulines and F. Charpentier, "Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Communication*, vol. 9, no. 5/6, pp. 453–67, 1990.
- [16] J. Laroche, *Applications of Digital Signal Processing to Audio & Acoustics*. M. Kahrs and K. Brandenburg (Eds.), Kluwer Academic Publishers, 1998, ch. Time and pitch scale modification of audio signals, pp. 279–309.
- [17] M. Portnoff, "Implementation of the digital phase vocoder using the fast Fourier transform," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 24, no. 3, pp. 243–8, 1976.
- [18] D. Arfib, F. Keiler, and U. Zölzer, "Time-frequency processing," in *DAFx – Digital Audio Effects*, U. Zölzer, Ed. J. Wiley & Sons, 2002, pp. 237–97.
- [19] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [20] V. Verfaille and P. Depalle, "Adaptive effects based on STFT, using a source-filter model," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 296–301.
- [21] T. Buzan and B. Buzan, *Mind Map Book*. Plume, 1996.
- [22] P. Schaeffer, *Traité des Objets Musicaux*. Seuil, 1966.
- [23] R. M. Schafer, *The Tuning of the World*. Knopf: New York, 1977.
- [24] W. W. Gaver, "What in the world do we hear? an ecological approach to auditory event perception," *Ecological Psychology*, vol. 5, no. 1, pp. 1–29, 1993.
- [25] C. Guastavino, B. F. Katz, J.-D. Polack, D. J. Levitin, and D. Dubois, "Ecological validity of soundscape reproduction," *Acta Acustica united with Acustica*, vol. 91, no. 2, pp. 333–341, 2005.
- [26] A. Marui and W. L. Martens, "Perceptual and semantic scaling for user-centered control over distortion-based guitar effects," in *110th Conv. Audio Eng. Soc.*, Amsterdam, the Netherlands, 2001, preprint 5387.
- [27] B. Blesser, "An interdisciplinary synthesis of reverberation viewpoints," *J. Audio Eng. Soc.*, vol. 49, no. 10, pp. 867–903, 2001.
- [28] L. Landy, "Reviewing the musicology of electroacoustic music: a plea for greater triangulation," *Org. Sound*, vol. 4, no. 1, pp. 61–70, 1999.
- [29] J.-P. Jullien, E. Kahle, M. Marin, O. Warusfel, G. Bloch, and J.-M. Jot, "Spatializer: a perceptual approach," in *96th Conv. Audio Eng. Soc.*, Amsterdam, the Netherlands, # 3465, 1993, pp. 1–13.

ON THE DYNAMICS OF THE HARPSICHORD AND ITS SYNTHESIS

Henri Penttinens

Laboratory of Acoustics and Audio Signal Processing
Helsinki University of Technology
Espoo, Finland
Henri.Penttinens@tkk.fi

ABSTRACT

It is common knowledge that the piano was developed to produce a keyboard instrument with a larger dynamic range and higher sound radiation level than the harpsichord possesses. Also, the harpsichord is a plucked string instrument with a very controlled mechanism to excite the string. For these reasons it is often falsely understood that the harpsichord does not exhibit any dynamic variation. On the contrary, the signal analysis and the listening test made in this study show that minor but audible differences in the dynamic levels exist. The signal analysis portrays that stronger playing forces produce higher levels in harmonics. The energy given by the player is not only distributed to the plucking mechanism but also carried on from the key to the body. This is evident from the increased level of body mode radiation. A synthesis model for approximating the dynamic behavior of the harpsichord is also proposed. It contains gain and timbre control, and a parallel filter structure to simulate the soundboard knock characteristic for high key velocity tones.

1. INTRODUCTION

The objective of this paper is two fold. Firstly the goal is to clarify the issue of harpsichord dynamics through signal analysis and a listening test. Secondly, the aim is to extend an existing model-based harpsichord sound synthesis algorithm [1] to include realistic modeling of dynamics.

The issue of harpsichord dynamics can easily be considered as an uninteresting topic and solved case, mainly since the piano was developed to obtain some dynamic expression. On the contrary, the theme has still some open issues and there are basically two camps with opposite opinions. One camp says that the mechanism of the instrument does not allow dynamic variation: "In a harpsichord the energy input is fixed by the nature of the plucking mechanism. The loudness of the note is therefore determined by the efficiency with which string energy is transferred to the soundboard," states [2]. According to [3] "The transient is very reproducible and allows no dynamic variation." Similar remarks can be found in, for example [4, 5]. The other half says that "the tone color and (to some extent) the loudness are both altered when a key is struck more or less hard" [6]. Hall [7] agrees with Benade [6], based on brief informal tests with an FFT (Fast Fourier Transform) analyzer and simulations of the interaction between the plectrum and string. Furthermore, Hall's [7] simulations predict that higher plectrum speeds produce a brighter spectrum. There are also some who are in between, but do not give any specific explanation [8].

The revival of the harpsichord has not only been musical, but also its acoustics has been investigated during the past few decades. General descriptions of the instrument can be found in [4] and [8],

and in [2] Fletcher discusses interesting design issues. The behavior of soundboards and air and structural modes have been discussed [9, 10, 11]. Weyer discusses temporal characteristics in the attack [5, 12]. Details of the attack, called pretransients, are discussed in [3], and sonological investigations are presented in [13].

This study tackles the issue firstly from an analysis point of view, by looking at measurement data, and secondly by creating a sound synthesis model for the process. Sound examples related to the work can be found at <http://www.acoustics.hut.fi/publications/papers/dafx2006-harpsichord-dynamics/>. The remaining parts of this paper are organized as follows. In Sec. 2 the construction of the harpsichord is discussed. Section 3 presents the signal analysis made and Sec. 4 proposes a synthesis model based on this. Section 5 discusses the conducted listening tests and its results. Sections 6 and 7 give a short discussion and conclusions, respectively.

2. HARPSICHORD CONSTRUCTION

The harpsichord is a keyboard instrument belonging to the family of plucked string instruments due to its string excitation mechanism. The harpsichord is played so that a set of keys that form a manual are pushed down and the machinery of the instrument causes a plectrum, also called a quill, to pluck the string. When the key is released, a spring mechanism prevents a repluck to occur. Moreover, a damper comes in contact with the string and dampens it. In contrast to the guitar, where the plucking event can be controlled very freely in many dimensions (force, plucking point, plucking direction, etc.), the excitation process in the harpsichord is bounded by the machinery executing it.

In a harpsichord two to four sets of strings are controlled with one or two manuals. The string sets are also called registers or string choirs, and two of the registers are usually tuned in unison and they are called the 8' (eight foot) registers. These two registers differ in the plucking point and are called the 8' back and 8' front registers, respectively. The plucking point in the 8' back register is further away from the player. Often, a third register is also included and it is tuned an octave higher. This register is called the 4' register.

One end of the strings is attached to the nut close to the tuning pins. This is the end where the keys are. The other end is attached to a long curved bridge. The string continues after the bridge to a hitch pin, which is on top of the soundboard. The nut is set on a very rigid wrest plank, and the bridge is on top of the soundboard. The soundboard is very thin, about 2 to 4 mm, supported by several ribs. The main function of the soundboard is to amplify the weak sound of the vibrating string. Sometimes, a rose opening is included in the soundboard. Consequently, a Helmholtz resonance is produced with a frequency usually below 100 Hz [2].

The harpsichord used in this study was built in 2000 by Jonte Knif and Arno Peltt. It has characteristics adapted from harpsichords built in Italy and Southern Germany. Moreover, it has three sets of strings, the typical 8' back and front registers and a 4' register. Additionally, the highest octave of the 4' register does not have dampers, causing the instrument to have a reverberant response. The registers are controlled through two manuals. The instrument was tuned to the Vallotti tuning [14] to an A₄ that has a fundamental frequency of 415 Hz. Old instruments from the baroque era are typically tuned lower than the current standard, which is 440 Hz or higher.

3. SIGNAL ANALYSIS

3.1. Recorded material

A sound database for a synthesis model [1] was recorded in the large anechoic chamber at Helsinki University of Technology. The recording was done with two pairs of studio microphones placed about 1 m above the soundboard. The data of importance to the dynamics is based on tones played with the 8' back register and all the C keys (C₂, C₃, C₄, C₅, and C₆), each with three different striking forces or velocities. For practical reasons, these three playing levels are referred to as piano pianissimo (*pp*), mezzo forte (*mf*) and forte fortissimo (*ff*). These dynamic levels were played successively without changing the setup and, therefore, comparing them is legitimate. The fundamental frequencies, f_0 , are as follows: C₂ = 62.9 Hz, C₃ = 124.4 Hz, C₄ = 247.6 Hz, C₅ = 495.6 Hz, and C₆ = 991.2 Hz.

3.2. Comparison of partial amplitudes

Next, we take a look at the amplitudes of harmonic partials and how they behave in relation to each other.

3.2.1. Absolute levels

One way to go about investigating the existence of harpsichord dynamics is to analyze the levels of the harmonics of the string vibrations. If only the spectra of string vibrations were compared, the differences in the time domain would remain hidden. Thus, the evolution of string harmonics is observed as a function of time.

Figure 1 illustrates the envelopes of harmonics one (solid line), two (dashed), and three (dotted) for the C₄ tone played with dynamic levels *ff* and *pp*. The first two envelopes of the *ff* tone appear at a higher level throughout, whereas the behavior of the third harmonic seems practically identical. At $t = 0.4$ s, the differences between the levels for the first, second, and third harmonic are 5 dB, 3 dB, and 0.5 dB, respectively.

Similarly Fig. 2 displays the envelopes of harmonics one (solid line), two (dashed), and three (dotted) for the C₆ tone played with dynamic levels *ff* and *pp*. At $t = 0.4$ s, the differences between the levels for the first, second, and third harmonic are 0 dB, 2 dB, and 1 dB, respectively. The differences in the levels are smaller for C₆ than for C₄.

The forms of the displayed envelopes are, again, strikingly similar. However, some differences can be noted, for example in the amplitude modulation depth of the envelope of the second harmonic (Fig. 2) is around 0.4 s or for the third harmonic after 1 s. Also, no significant differences or deviating trends in decay times were noted for tones played with different dynamic levels. In addition, it mainly seems that the form of the envelope for a

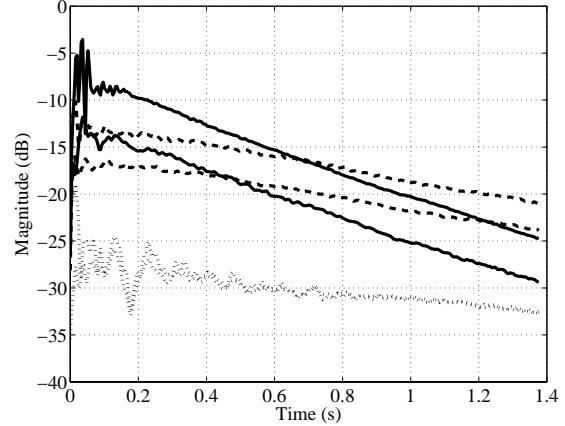


Figure 1: Envelopes of the first three harmonics for key C₄ tones played as forte fortissimo (*ff*) and piano pianissimo (*pp*). Harmonic number and respective line type: 1 - solid, 2 - dashed, 3 - dotted.

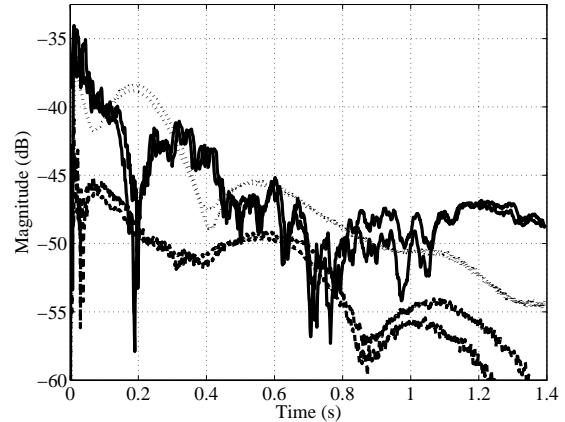


Figure 2: Envelopes of the first three harmonics for key C₆ tones played as forte fortissimo (*ff*) and piano pianissimo (*pp*). Harmonic number and respective line type: 1 - solid, 2 - dashed, 3 - dotted.

string harmonic is quite independent of the striking velocity. More suggestions for the causes behind the differences are given in the discussion section.

3.2.2. Relative levels of harmonics

Now we compare the levels of harmonics as a function of frequency. As above, the difference in harmonic levels are compared between *pp* and *ff* tones. The *pp* and *ff* tones have been scaled according to the harmonic with the largest level typically being the lowest harmonic. Figure 3 shows the relative difference in levels as a function of harmonic index for tones (a) C₃, (b) C₄, and (c) C₅. There exists a clear increase in level in total, but a relative level difference in harmonics is not evident. In one way, Fig. 3 gives an estimate of how much the higher harmonics of a *pp* tone should be amplified to simulate the characteristics of a *ff* tone. An increase in the relative level of harmonics as a function of excitation force

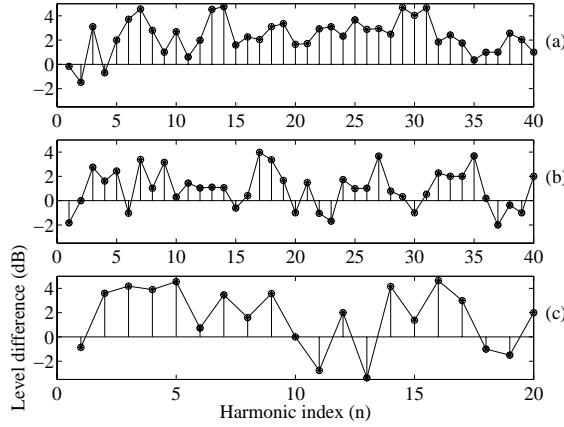


Figure 3: Relative difference between the level of harmonic envelopes measured at 0.4 s for *pp* and *ff* played tones for n harmonics: (a) C_3 , (b) C_4 , and (c) C_5 .

is typical for musical instruments. However, due to the excitation mechanism this does not seem to apply for the harpsichord, at least to the one measured in this study.

3.3. Comparison of body mode levels

Next, we examine how the soundboard and body modes behave for tones played with different dynamic levels. Now, all recorded levels are compared. Figure 4 shows the magnitude responses of tone C_6 played at *ff* (solid line), *mf* (dashed), and *pp* (dotted) dynamic levels. Due to the high f_0 (991.2 Hz) the lowest body modes are nicely separated from the string vibrations and clearly visible. Figure 4 indicates at least a general 10 dB level difference for the lowest body modes (< 300 Hz) from one dynamic level to another. The *ff* tone shows several distinct body resonances, whereas the *pp* tone perhaps only has an emphasis at 50 Hz and the rest of the response is very noise-like¹. The general level of body modes of the *mf* tone falls between the *pp* and *ff* tones. Similarly as the *ff* tone, the *mf* tone has distinct resonances, but not as prominent.

Figure 5 depicts the magnitude responses of tone C_3 played at *ff* (solid line), *mf* (dashed), *pp* (dotted) dynamic levels. Now the f_0 (124.4 Hz) is considerably lower, but body modes can still be analyzed. The body modes below the fundamental of the string behave for different dynamic levels similarly as for the C_6 case. Body modes can be distinguished also between the string harmonics with the corresponding order of levels, i.e., the *ff* tone has typically the highest level and the *pp* tone the lowest. The behavior of the body modes between harmonics in general is not as clear as for the C_6 tone, but the trend is still notable.

The changes in body mode amplitudes for different dynamic levels are considerably larger than for the level variations in string motion. Hence, the changes in body mode amplitudes cannot be induced by the string vibrations only. Therefore, it seems that the with a forceful push of a key the soundboard and its modes are excited through mechanical coupling, separate from the string plucking mechanism. This mechanical excitation path and the mechanism from the key to the soundboard explains also why the body

¹If the 50 Hz resonance were a mains-borne disturbance, the level would not increase with stronger playing levels.

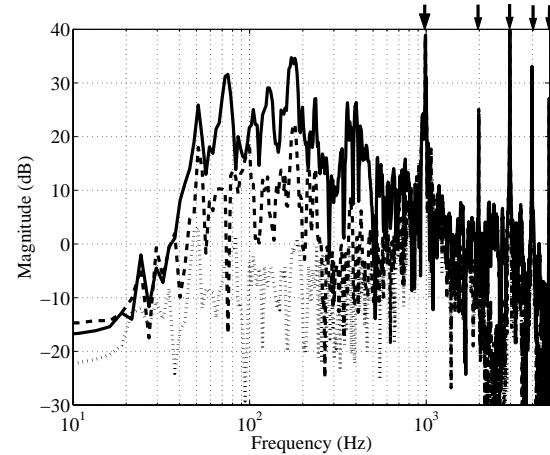


Figure 4: Visualization of body mode behaviour for tone C_6 at different playing levels: *ff* (solid), *mf* (dashed), and *pp* (dotted). The string modes are indicated with arrows.

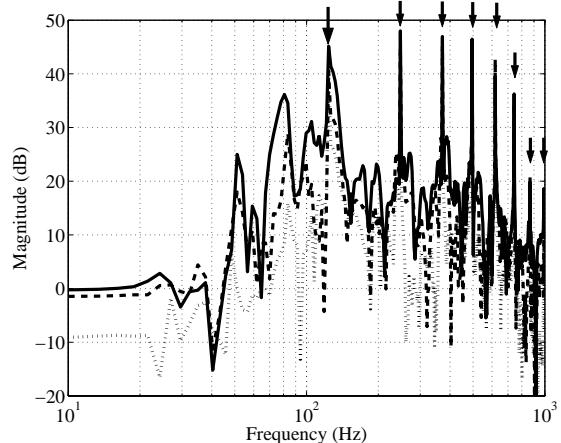


Figure 5: Visualization of body mode behaviour for tone C_3 at different playing levels: *ff* (solid), *mf* (dashed), and *pp* (dotted). The string modes are indicated with arrows.

modes have slightly different levels, Q -values, and frequencies in Fig. 4 and 5. This is because different excitation points along the keys excite the soundboard differently. In addition, coupling of body modes to the string modes can cause modes to be split or shifted in frequency.

4. LISTENING TEST

Computational models for the human auditory system exist, but they work the best for stationary sounds [15, 16]. So far, the reliability of auditory models is vague for complex tones, where the harmonic content and timbre change as a function of time. In addition, the loudest harpsichord tones exhibit the characteristic knock/thump of the soundboard in the attack as discussed in the previous section. This adds some ambiguousness to the *ff* and *mf* tones compared to the *pp* tones. Hence, a listening test is required to get to the bottom of the question on harpsichord dynamics.

A listening test was conducted where the perception of loudness of two tones was to be adjusted to be equal. Two recorded harpsichord tones were played consecutively, so that the gain of the latter sound could be adjusted in steps of 0.5 dB in a range of ± 4 dB. The adjusting was done with a slider and the tone pair could be listened to as many times as needed. The play list of the conducted test comprised of all pairwise comparisons between dynamic levels (*pp* to *ff*, *pp* to *mf*, and *mf* to *ff*) for all C tones (3x5). In addition, two blank pairs for each octave was added and the pairwise comparisons were also played in reverse order. All in all, the play list contained 40 pairs of tones.

The test was conducted in the listening room of the Acoustics Laboratory at Helsinki University of Technology. This high standard listening room has a very low noise level and it meets the ITU-R BS.1116 standard. The sounds were played through headphones to six participants with normal hearing. The sound level to the headphones was calibrated with the help of a dummy head and a pre-amplifier (Cortex electronic, Manikin MK1). The pre-amplifier had a sound pressure level (SPL) meter with A weighting and its values were matched with previously measured SPL values. The measured SPL values were obtained with an SPL meter (B&K 2238 Mediator with a 4188 capsule) placed near the ear of a player playing the same tones used in the analysis. These reference tones were measured with an Italian-type harpsichord in a rehearsal room at the Sibelius Academy. The *ff* C₄ tone was calibrated to 68.5 dBA.

Figure 6 shows the results of the listening test for each compared dynamic pair and all strings: (a) *pp* to *ff*, (b) *pp* to *mf*, and *mf* to *ff*. The x axis indicates the name of the tone and the y axis shows the perceived loudness difference in dB. In Fig. 6 each box has lines at the upper (75%) and lower (25%) quartile values. The median value is indicated between these values with a line at the center of the hourglass-shaped part of each box. The whiskers (---) show the extent of the rest of the data. Outliers, indicated with a star (*), are data points with values beyond the whiskers.

For the results shown in Fig. 6, the largest mean value, 2.41 dB, was obtained for the C₃ tone with a standard deviation of 0.76, naturally found for the *pp* to *ff* case displayed in pane (a). Respectively, the smallest mean value 0.42 dB with a 0.7 dB standard deviation was obtained for the C₆ tone with a *pp* to *mf* tone pair. It seems that the dynamic differences are slightly more notable for C₃ and C₄ tones than for the lower or higher tones. For easy comparison, the mean values over all strings for each tested pair are given in Table 1. This way the tone-wise comparison is lost, but based on this it can be proposed that each dynamic step (*pp* to *mf* and *mf* to *ff*) is about the size of 1 dB and the dynamic range, at the largest, is slightly larger than 2 dB.

Based on the written and verbal feedback of the listening test, the testees mainly concentrated on listening to the decaying part of the tones. The knock of the soundboard present in the *mf* and especially in *ff* tones was noted and described as a characteristic that made the listening of loudness more challenging. This characteristic directed the listeners to focus on the decaying part of the tones.

The findings from the signal analysis discussed above, at least to some degree, support the results of the listening test, in the sense that the differences in the dynamics is smaller for higher strings than lower ones. On the other hand, the signal analysis results discussed in Sec. 3 could suggest quite large differences in the perceived loudness of the instrument. In the overall result, the

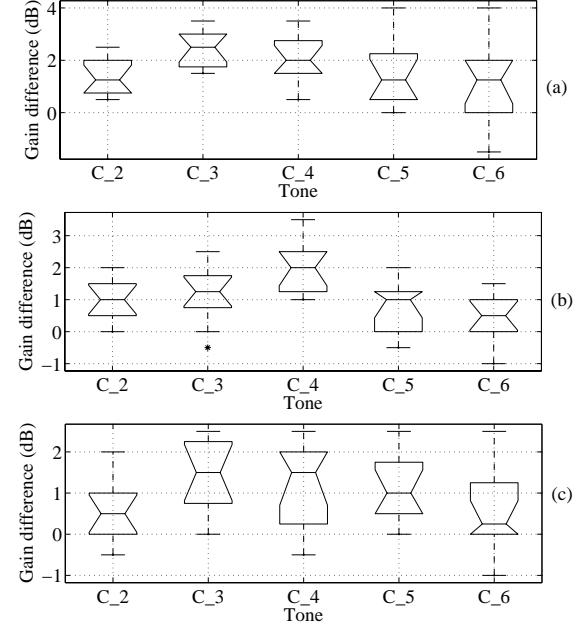


Figure 6: Listening test results for tested tone pairs (a) *pp* to *ff*, (b) *pp* to *mf*, and (c) *mf* to *ff*. The x axis indicates the name of the tone and the y axis shows the perceived loudness difference in dB.

Type	Mean	Std. deviation
<i>pp</i> to <i>ff</i>	1.68	0.37
<i>pp</i> to <i>mf</i>	1.08	0.09
<i>mf</i> to <i>ff</i>	0.98	0.18
blank	-0.45	0.24

Table 1: Results from the listening test as mean values with standard deviation over all strings and for all three different transition types and the blank trials.

ability of the soundboard to radiate efficiently and/or effects of the auditory system come into play. Consequently, the results from the listening test are quite understandable.

5. SYNTHESIS MODEL WITH DYNAMICS

A general framework for modeling the dynamics of a harpsichord should include control over gain of string vibrations, frequency envelope of string vibrations, and simulation of the mechanically excited soundboard vibrations. These assumptions are based on the signals analysis and listening test discussed in Secs. 3 and 4.

Suppose that a model-based (physics-based) sound synthesis algorithm contains models for the excitation mechanism, the string vibrations, and the soundboard and radiation. Then, adding the modeling of dynamics would consist of a gain and timbre control for the string vibrations and an additional path from the excitation mechanism to a soundboard model bypassing the string model.

This train of thought is the basis for the synthesis model proposed here. The starting point for the synthesis model is a previously proposed model-based harpsichord sound synthesis algorithm [1]. This model [1] is founded on digital waveguide mod-

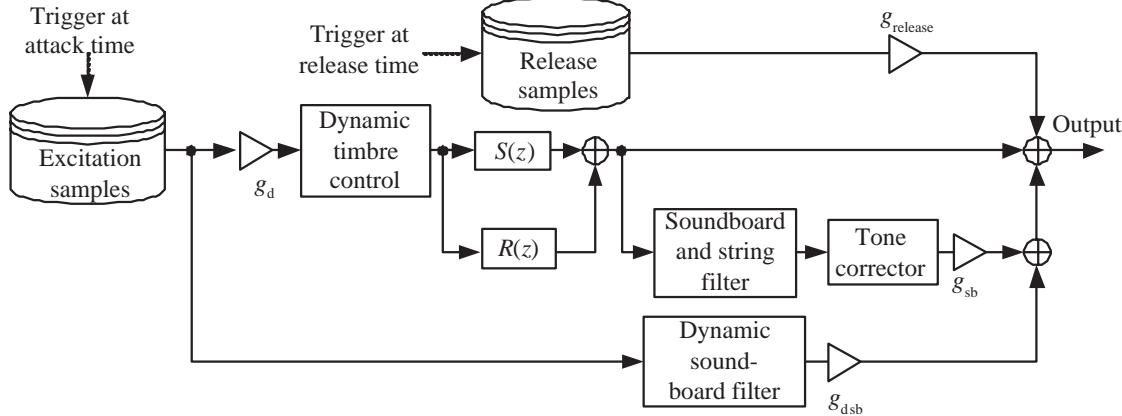


Figure 7: Block diagram of the waveguide synthesis algorithm for the harpsichord with dynamics modeling.

eling [17] and is a version of the commuted waveguide synthesis approach. The model already contains a so-called soundboard model/filter [1]. However, it is mainly intended for simulating the combination of soundboard modes and the slowly decaying string vibrations of the last octave of the 4' register (without dampers) and the part of the strings that lies behind the bridge. These slowly decaying string vibrations dominate at frequencies above 350 Hz and their decay time T_{60} can be as long as 4.5 s. These vibrations are completely different from those examined in Figs. 4 and 5, which decay fast ($T_{60} < 0.5$ s). Therefore, the proposed synthesis model, shown in Fig. 7, contains an additional dynamic soundboard filter intended for simulating the soundboard knock audible in *mf* and *ff* tones. Moreover, since the origin of the soundboard knock seems to be the mechanical coupling from the tangent to the soundboard, the proposed model also bypassed the string model.

The excitation database for the harpsichord synthesizer is based on *mf* tones. The excitation signals are short bursts that are obtained by canceling the partials of the string vibrations with a sinusoidal model as discussed in [18]. To crudely model the higher levels of string harmonics the excitation fed to the string model is multiplied with g_d (see Fig. 7). To compensate for errors in the initial levels produced by the crude raising of the string vibration amplitudes with g_d the timbre control filter can be used. This naturally requires a higher order filter, and this necessity can be evaluated separately and can be simulated with any suitable parametric EQ filter (dynamic timbre control in Fig. 7). Finally, the behavior of the soundboard knock audible in *mf* and *ff* tones is simulated with the dynamic soundboard filter in parallel with the string model. The adjective dynamic is added since the filter has to change according to the kind of a dynamic level played, i.e., the filter coefficients are changed accordingly. Again, any proper parametric EQ-filter can be used. For modeling the transition from a *mf* tone to a *ff* tone the soundboard filter is required to boost the proper resonances, whereas in the case of a *mf* to *pp* transition, the soundboard filter attenuates the body modes.

Figure 8 shows the magnitude responses of a synthesized *mf* tone (dotted line), synthesized *mf* tone with a soundboard model (solid), target spectrum of the *ff* tone (dashed), and the soundboard model (solid), all for the C₆ tone. The soundboard filter is a cascade of six second-order parametric peak filters. The peak filters are discussed in [19](pp. 117-125). The body response of the synthesized *mf* tone with the soundboard model (solid) coincides

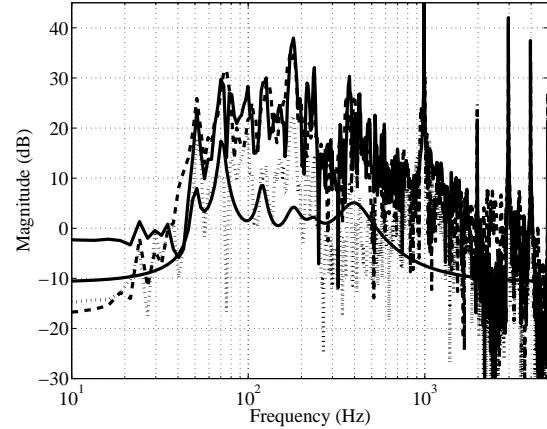


Figure 8: Magnitude responses of C₆ tones: target.*ff* (dashed), synthesized tone *mf* to *ff* (solid), synthesized *mf* (dotted), and soundboard filter (solid).

quite well with the target *ff* tone (dashed) already with the cascade of only six peak filters.

6. DISCUSSION

Some of the reasons behind why the harpsichord exhibits dynamics and timbral changes have been covered; however, the final truth behind the string behavior still remains to be revealed.

A high speed camera would help out tremendously to give a more accurate estimate on how long the plucking event and contact between the plectrum and string lasts. This kind of data could enable to continue further the steps taken by Hall [7]. In addition, the question remains, does the release of the plectrum from the string occur before the reflected wave arrives back to the plectrum or after. A natural continuum from there on would be to investigate the interaction between the plectrum and the string when a guitar player is in charge. Another future path would be to investigate what happens in the harpsichord and to its loudness when two or more registers are played.

Even if the high frequency partials have relatively higher val-

ues when the dynamic level rises they decay significantly faster than the lower modes. This and the fact that the listener concentrates on the attack part could partly explain why the loudness differences were not larger in the listening test results. Longitudinal string modes or phantom partials [20, 21] are also present in the string vibrations, and they are clearly more pronounced for the *ff* tones than for *pp* tones. Their effect on the perceived loudness was, however, interpreted not to be as significant as the raising of absolute harmonic levels. They could, however, have an effect on the timbre of a tone.

7. CONCLUSIONS

Contrary to common assumption, the harpsichord contains a limited amount of dynamics and some timbral changes occur when the key is pressed down with different forces/speeds.

The signal analysis made on recorded harpsichord tones revealed differences in the levels of string harmonics, indicating stronger playing forces produced higher levels. The differences for isolated harmonics were as high as 5 dB for some low tones. For higher tones the level differences were smaller, about 1-3 dB.

Based on the conducted listening test, it can be said that during each dynamic step (from *pp* to *mf* and from *mf* to *ff*) the loudness of the instrument increases by about 1 dB. Furthermore, for the investigated harpsichord the dynamic range from *pp* to *ff* is at its largest for the C₃ tone with a value of slightly above 2 dB.

A synthesis model for approximating the dynamic behavior of the harpsichord is proposed. A general framework for building a model-based synthesis algorithm for the phenomenon is also given. Based on this, a specific model is proposed with gain and timbre control, and a parallel filter structure to simulate the sound-board knock.

The harpsichord exhibits dynamics albeit in a limited range, both in a musical and acoustical sense. However, it is doubtful that dynamic expressions will start appearing in harpsichord scores. Nevertheless, dynamic expressions can and should be included in the scores controlling a harpsichord synthesizer. What is more, the synthesizer can naturally take the dynamic range further than a real instrument can, as suggested previously [1]. Sound samples are available at the URL: <http://www.acoustics.hut.fi/publications/papers/dafx2006-harpsichord-dynamics/>.

8. ACKNOWLEDGEMENTS

Warm thank you to the participants of the listening tests, Hanna Järveläinen for her help, and to Jonte Knif for the intense discussions on the subject. This work has been financially supported by the Pythagoras Graduate School of Music and Sound Research.

9. REFERENCES

- [1] V. Välimäki, H. Penttinen, J. Knif, M. Laurson, and C. Erkut, "Sound synthesis of the harpsichord using a computationally efficient physical model," *EURASIP J. on Applied Sig. Proc.*, vol. 2004, no. 7, pp. 934–948, 2004.
- [2] N. H. Fletcher, "Analysis of the design and performance of harpsichords," *Acustica*, vol. 37, pp. 139–147, 1977.
- [3] T. Gäumann, "The pretransient of the harpsichord sound. I. vertical movement of the plectrum," in *Proc. Stockholm Music Acoustics Conf. (SMAC'03)* Stockholm, Sweden, 2003, pp. 163–166.
- [4] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. New York: Springer-Verlag, 1991.
- [5] R. D. Weyer, "Time-frequency-structures in the attack transients of piano and harpsichord sounds - I," *Acustica*, vol. 35, pp. 232–252, 1976.
- [6] A. H. Benade, *Fundamentals of musical acoustics*, 2nd ed. New York, NY, USA: Dover Publications, Inc., 1990. [Online]. Available: <http://ccrma-www.stanford.edu/marl/Benade/>
- [7] D. E. Hall, "String excitation: piano, harpsichord and clavichord," in *Proc. Stockholm Music Acoustics Conf. (SMAC'93)* Stockholm, Sweden, 1993, pp. 309–314.
- [8] M. Campbell and C. Greated, *The Musician's Guide to Acoustics*. New York, NY, USA: Schirmer Books, 1988.
- [9] E. L. Kottick, "The acoustics of the harpsichord: Response curves and modes of vibration," *Galpin Soc. J.*, vol. 38, pp. 55–77, Apr. 1985.
- [10] E. L. Kottick, K. D. Marshall, and T. J. Hendrickson, "The acoustics of the harpsichord," *Scientific American*, vol. 264, no. 2, pp. 94–99, 1991.
- [11] W. R. Savage, E. L. Kottick, T. J. Hendrickson, and K. D. Marshall, "Air and structural modes of a harpsichord," *J. Acoust. Soc. Am.*, vol. 91, no. 4, pp. 2180–2189, 1992.
- [12] R. D. Weyer, "Time-varying time-frequency-structures in the attack transients of piano and harpsichord sounds - II," *Acustica*, vol. 36, no. 4, pp. 241–258, 1976.
- [13] A. Beurmann and A. Schneider, "Sonological analysis of harpsichord sounds," in *Proc. Stockholm Music Acoustics Conf. (SMAC'03)* Stockholm, Sweden, 2003, pp. 167–170.
- [14] J. Sankey and W. A. Sethares, "A consonance-based approach to the harpsichord tuning of Domenico Scarlatti," *J. Acoust. Soc. Am.*, vol. 101, no. 4, pp. 2332–2337, 1997.
- [15] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*. Berlin: Springer-Verlag, 1990.
- [16] 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, pp. 224–240, 1997.
- [17] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music J.*, vol. 16, no. 4, pp. 74–91, 1992, [Online] <http://www-ccrma.stanford.edu/~jos/pmudw/>.
- [18] V. Välimäki and T. Tolonen, "Development and calibration of a guitar synthesizer," *J. Audio Eng. Soc.*, vol. 46, pp. 766–778, 1998.
- [19] U. Zölzer, *Digital Audio Signal Processing*. Chichester, UK: J. Wiley & Sons, 1997.
- [20] H. A. Conklin, "Generation of partials due to nonlinear mixing in a stringed instrument," *J. Acoust. Soc. Am.*, vol. 105, no. 1, pp. 536–545, 1999.
- [21] 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. 2268–2278, 2005.

MUSICAL SOUND TIMBRE: VERBAL DESCRIPTION AND DIMENSIONS

Jan Štěpánek

MARC — Musical Acoustics Research Centre Prague
Music Faculty, Academy of Performing Arts in Prague, Czech Republic
jan.stepanek@hamu.cz <http://www.hamu.cz/sound>

ABSTRACT

Two approaches to the study of musical sound timbre are described and documented by psychoacoustic experiment examples.

The classical bottom-up approach is demonstrated on the study of contexts of violin sounds and pipe organ sounds. Verbal attributes collected during listening tests were used for the interpretation and comparison of resulted perceptual spaces of sounds.

The proposed top-down approach is based on the collection of musical experts experiences and opinions going from very common to more specific ones. Here the common perceptual space (perceptual space of verbal attributes) was constructed from non-listening test of dissimilarity of verbal attributes describing timbre (verbal or soundfree context of stimuli).

The verbal interpretation of perceptual spaces of sound contexts and perceptual space of verbal attributes are compared and the hypothesis of the four basic dimensions of timbre is formulated: 1. gloomy — clear, 2. harsh — delicate, 3. full — narrow, 4. noisy — ?.

1. INTRODUCTION

In the past the timbre research was focused on different aspects of this phenomena (overview see for example in [1]): timbre of steady state part of the sound, transient influence, different contexts — many kinds of natural sounds of musical instruments, synthetic sounds, respondent contexts (not frequently pointed out for example in relation to [2]), isolated tones and tone sequences, etc. Most frequently the goal was to explain used scales or to found dimensions using time and/or spectral properties of stimuli (sound signals) qualitatively or even quantitatively. Sometimes the effort to generalize the results in timbre research led to the changes of the sound context (used stimuli in [3] and [4]), or to the study more sound contexts in parallel [5].

Plomp in [6] recognizes two types of processing in hearing: bottom-up (audition) and top-down (cognition). Analogically, we will speak about bottom-up and top-down approach to the general study of timbre (unlike using the same terms for the description or classification of individual experiment, see [7]).

Bottom-up approach is based on the study of more contexts (of basic level) and then the selection of representative stimuli to create contexts of the second level, etc. Bottom-up approach is demonstrated here on the study of steady state timbre of five contexts of violin sounds (tones B3, F#4, C5, G5 and D6) [5, 8, 9, 10] and five contexts of pipe organ sounds (Principal 8' contexts C2, C3, C4, C5 and C6 tones) [13, 14] (thus only basic level of bottom-up approach was provided up to now but on many sound contexts). Dissimilarity in timbre in pairs of sounds was acquired in listening tests of each sound context. The test results (individual dis-

similarity matrices) were evaluated using latent class approach of multi-dimensional scaling (MDS) method (CLASCAL) [15, 16]. Perceptual space of appropriate dimensionality was obtained as an optimal MDS model for each sound context. The objects of each perceptual space represent test stimuli (sounds), dissimilarities between pairs of sounds were transformed into Euclidean distances. Verbal attributes (collected as spontaneous verbal description of timbre differences during listening tests) were used for the interpretation of resulted perceptual spaces (attributes in Czech for violin and in German for organ) and their comparison.

Top-down approach is based on the collection of musical experts experiences and opinions going from very common to more specific ones. Experts judged (without any listening) dissimilarity measure between pairs of verbal attributes describing timbre (attributes in the Czech language comes from questionnaire survey). Common perceptual space of verbal attributes was constructed using CLASCAL [17].

The results of above-mentioned studies were presented in several "conferences ([5, 8, 9, 10, 11, 12, 13, 14, 16, 17] and others) but never discussed together. The goal of the comparison of results of our studies presented here is the description of timbre via "common verbal" dimensions. They can be used not only for the interpretation of perceptual spaces, constructed from the results of listening test, but also for better communication among experts by the description of qualities of any kind of musical sounds and audio effects as well.

2. LISTENING EXPERIMENTS

Each stimuli context consisted of sounds of (nearly) the same pitch, level and duration.

2.1. Method

Listening experiments were focused on the description of timbre perception of quasistationary parts of sounds. Sound stimuli for tests were manipulated to decrease the influence of transients on the perception (uniform fade in and fade out).

The pairwise comparison listening tests were used for the collection of dissimilarities in timbre. Verbal attributes describing timbre dissimilarity in judged sound pairs were also collected (Spontaneous Verbal Description, SVD, [9]), in organs during each test or in violins for pairs of selected representative sounds.

External interpretation of perceptual spaces was made using collected verbal attributes. The number of use of verbal attribute in (optimal fitting of the external scale) [11, 18] was applied. Only successfully embedded verbal attributes (represented by significant reproduction of external scale values by embedding)

were selected for further analysis. The angles between the embedding vectors were calculated [18] and the relations between the embedded verbal attributes were observed and discussed. Additional experiment was provided with violin sounds — Verbal Attribute Ranking and Rating (VARR) [8]. VARR test consisted in two parts supported by PC program: 1. ranking stimuli according verbal attribute, 2. rating the magnitude of verbal attribute on ranked stimuli.

2.2. Violin sound

2.2.1. Contexts, stimuli and respondents

Violin sounds of tones B3, F#4, C5, G5 and D6 played the same technique (*détaché*, *naturale*, non-vibrato and *mezzoforte*) in anechoic room were recorded and prepared for tests as stimuli. The same loudness, pitch and tone duration was maintained during monophonic recording. Seventeen sounds for each tone were used in dissimilarity test (with twenty judges), after the selection based on their positions in constructed perceptual space eleven of them were used in verbal attributes collection SVD tests (ten judges). The test respondents were professors and students of violin play from Prague Academy. Verbal attributes were expressed in Czech language, but their approximate translation into English is also used here.

In VARR tests the same eleven sounds were used as stimuli, different eleven judges take part in tests. The selection of attributes was based on the results of dissimilarity tests (comparison of perceptual spaces and verbal attributes relations [5]). Verbal attributes *sharp*, *dark*, *clear*, *narrow* and also perceived sound quality were selected for the test and judged.

2.2.2. Results

Results related to verbal attributes in selected tones are summarized in Tables 1, 2.

Tone	B3	F#4	C5	G5	D6
No. of dimensions	3	3	2	2	2
No. of attributes	$f_{abs} \geq 10$	65	64	58	64
	$\alpha \leq 5\%$	45	41	32	31
	$\alpha \leq 1\%$	27	28	22	12
	$\alpha \leq 0.1\%$	13	13	12	4
					15

Table 1: Number of perceptual space dimensions of optimal MDS solution and number of verbal attributes which have number of use at least 10 and embedding correlation significance at least α .

The magnitudes of verbal attributes *sharp*, *dark*, *clear*, *narrow* and perceived sound quality received in VARR test were averaged over all judges' and evaluated using Factor Analysis (FA) in each tone. Two-dimensional factor space of attributes was optimal in all five tones [8]. The positions of attributes in factor space are shown in Figure 3. Successful prediction of perceived sound quality from these four attributes in each tone was also verified [8].

2.2.3. Discussion

The significant embedding of verbal attribute (number of it use on individual sounds — external scale) into perceptual space of sounds can be interpreted as follows: the feature represented by

Tone B3				
Verbal attribute		no. of use	rank	R_{embed}
Czech	English			
temný	gloomy	144	3	0.97
hladký	smooth	45	17.5	0.95
úzký	narrow	105	6	0.94
mečivý	bleaty	87	9	0.93
tmavý	dark	208	1	0.92
kovový	metallic	99	7	0.90
ostrý	sharp	176	2	0.87
jemný	delicate	45	17.5	0.87

Tone C5				
Verbal attribute		no. of use	rank	R_{embed}
Czech	English			
měkký	soft	121	6	0.98
jemný	delicate	108	7	0.97
kovový	metallic	76	9	0.96
ostrý	sharp	208	1	0.96
světlý	bright	143	3	0.94
kulatý	round	123	5	0.93
vysoký	high	45	16.5	0.92
tmavý	dark	152	2	0.92
pronikavý	penetrating	35	21	0.91

Tone D6				
Verbal attribute		no. of use	rank	R_{embed}
Czech	English			
měkký	soft	134	2	0.98
ostrý	sharp	249	1	0.95
kulatý	round	92	7	0.94
šustí	rustle	95	6	0.92
hladký	smooth	59	12	0.92
tmavý	dark	60	11	0.90
pronikavý	penetrating	57	14	0.90
temný	gloomy	71	8	0.88
přidušený	damped	40	18	0.87
jemný	delicate	62	10	0.87
pískový	sandy	53	15	0.87
vysoký	high	38	19	0.86

Table 2: Selected verbal attributes with number of use at least 30 and embedding significance $\alpha \leq 0.1\%$, sorted according to embedding correlation R_{embed} .

the attribute has significant influence on perception and gives important contribution to the timbre dissimilarities on judged sound context. Moreover the significant embedding defines the direction along which this feature grows in perceptual space. Thus the features with embeddings containing small angles (near 0°) have similar influence on perception (their use on sounds is similar or proportional, they have coincident occurrence on sounds), large angles (near 180°) imply perceptually opposite and nearly orthogonal embeddings (near 90°) perceptually independent features.

The embedding of selected verbal attributes into perceptual spaces of optimal dimensionality are in Figure 1. The presented attributes were selected using following criteria: a) high number of spontaneous use in listening test, b) perceptual space filling, c)

embedding success (significance level typically 0.1%). The attributes creating orthogonal system in sound-context free test (see section 3) were preferred. Since the sharpness was found as substantial feature of sound timbre [2], the presented comparison of perceptual features in individual violin tones will be related to embeddings of attribute sharp. Summary of angles contained between *sharp* and some other attributes is in Table 3. *Sharp* and *soft* are nearly opposite features in all five tones like *sharp* and *gloomy*, but *dark* changes the relative position according to *sharp* in tones G5 and D6. *Narrow* has similar influence with *sharp* in B3, F \sharp 4 and C5, but is nearly independent in G5 and D6.

Verbal attribute	B3	F \sharp 4	C5	G5	D6
soft	164	171	163	160	174
gloomy	160	163	167	171	159
dark	166	159	168	94	121
narrow	28	19	15	75	86

Table 3: Angles [$^{\circ}$] of embeddings of selected attributes contained with embedding of *sharp*.

Attributes *sharp* and *dark* in factor space of VARR test results (Figure 2) have opposite positions in all five tones, the change of position *narrow* is more pronounced then in SVD embedding.

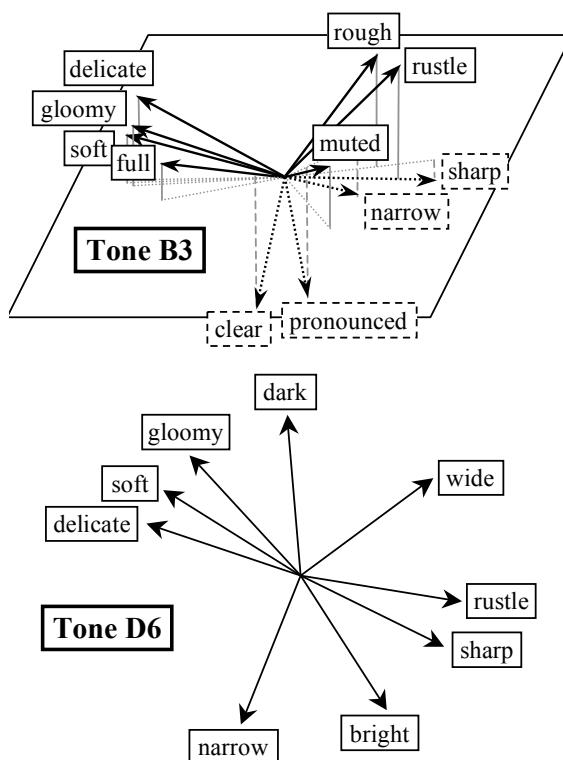


Figure 1: Directions of embeddings of selected verbal attributes into perceptual spaces (tones B3, D6).

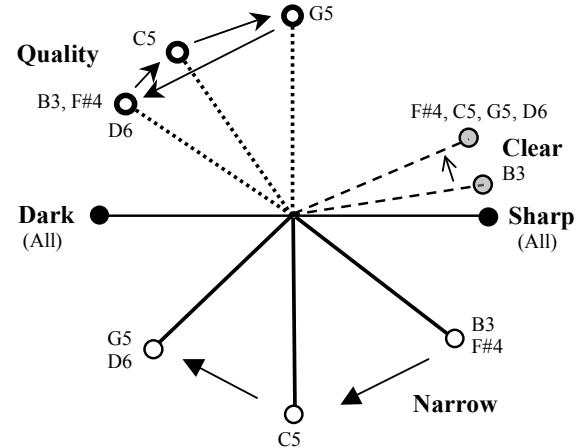


Figure 2: Violin timbre: schematic factor space of verbal attributes and perceived sound quality.

2.3. Pipe organ sound

2.3.1. Contexts, stimuli and respondents

Principal 8' sounds of tones C2, C3, C4, C5 and C6 of twelve organs from four European countries recorded *in situ* during the EU CRAFT project [13] were prepared for dissimilarity tests as stimuli. The tests were performed with twelve respondents — organ players and organ builders. Verbal attributes were expressed in German language, but their approximate translation into English is also used here.

2.3.2. Results

Results related to verbal attributes are summarized in Table 4. The three-dimensional perceptual space was identified as the optimal model for dissimilarity data for all studied C tones. Verbal attributes embeddings were calculated and representative attributes were selected based on following criteria: a) embedding significance at least 5 presence in many C tones. For better comparison with other experiment results the attributes specific only for organ sound description were omitted: *streichen* (stringed in English), *obertönig* (rich on high harmonics), *flötig* (flute like), *prinzipalig* (principal like), etc.

2.3.3. Discussion

Angles contained between embedding of selected verbal attributes in three-dimensional space do not fill up the whole space, their relative positions can be simplified with satisfactory precision in the plane, see Figure 3. The perceptual spaces were rotated to fix the position of *round* for further view simplification. The perceptual features represented by *round* and *soft* have similar influence in all C tones. The relative positions of *round* and *noisy* is stable. The attributes *sharp* and *narrow* have similar positions and position changes relative to *round*.

3. MUSICIANS OPINIONS

The collection of musicians opinions was based on nonlistening experiments: a) collection of verbal attributes describing timbre

Verbal attribute	German	eng	rauschig	rund	scharf	weich
	English	narrow	noisy	round	sharp	soft
C2	no. of use	15	6	10	6	15
	rank	1.5	12.5	5	12.5	1.5
	R_{embed}	0.86	—	0.85	0.61	0.78
C3	no. of use	9	4	6	4	9
	rank	3.5	14	6.5	14	3.5
	R_{embed}	0.62	0.63	0.76	0.67	0.94
C4	no. of use	18	12	10	5	8
	rank	2	4.5	6.5	14	8.5
	R_{embed}	0.95	0.64	0.81	0.66	0.70
C5	no. of use	19	6	7	19	19
	rank	3.5	12.5	10	3.5	3.5
	R_{embed}	0.65	0.66	0.58	0.86	0.72
C6	no. of use	9	1	4	38	12
	rank	5.5	58.5	14	1	4
	R_{embed}	0.81	—	0.71	0.98	0.58

Table 4: Selected verbal attributes describing Principal 8' sound timbre (embedding significance $\alpha \neq 5\%$, values of R_{embed} with $\alpha \leq 0.1\%$ are bold).

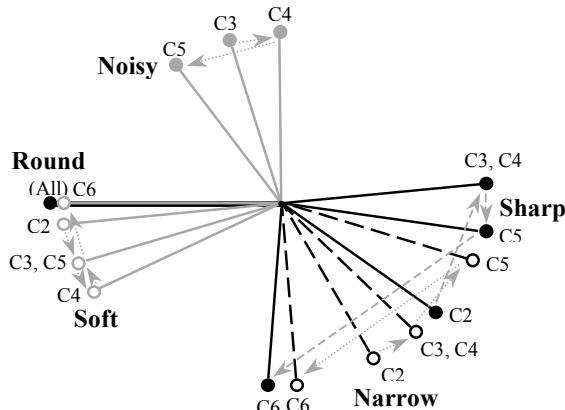


Figure 3: Organ Principal 8' timbre: schematic perceptual space of selected verbal attributes.

-questionnaire survey, b) test with selected verbal attributes in dissimilarity according to timbre understanding — attributes relations; then construction of common perceptual space of timbre using CLASCAL. The more detailed description is in [17].

3.1. Questionnaire survey

The questionnaire consisted of two parts: a) respondents personal profile (age, education, music profession and experience time, played instrument, etc.), b) words and expressions the respondent uses for the description of musical sound timbre. One hundred twenty respondents wrote 1964 different verbal attributes in total. The list of 25 most frequently used attributes is in Table 5.

3.2. Sound-context free test

3.2.1. Stimuli, respondents and methods

The 25 most frequently used verbal attributes from questionnaire (Table 5) were used as stimuli, 43 music professionals took part in

Verbal attribute		Frequency	
Czech	English	f_{abs}	f_{rel}
ostrý	sharp	94	78.3
temný	gloomy	79	65.8
měkký	soft	78	65.0
jasný	clear	75	62.5
sametový	velvety	61	50.8
jemný	delicate	58	48.3
kulatý	round	58	48.3
tupý	unpointed	55	45.8
drsný	harsh	54	45.0
světlý	bright	54	45.0
tvrdý	hard	54	45.0
sladký	sweet	53	44.2
plný	full	51	42.5
hrubý	rough	46	38.3
tmavý	dark	46	38.3
teplý	warm	43	35.8
zářivý	radiant	42	35.0
čistý	pure	40	33.3
vřelý	hearty	40	33.3
barevný	colored	38	31.7
zvonivý	ringing	38	31.7
chladný	cool	36	30.0
průzračný	lucid	36	30.0
široký	wide	36	30.0
úzký	narrow	36	30.0

Table 5: List of 25 most frequently used verbal attributes with their absolute(f_{abs}) and relative(f_{rel}) frequencies.

the test (five professional groups: players of string, wind and key instruments, composers & conductors and sound designers).

Dissimilarity pair test (dissimilarities from 0 to 5) was provided, the respondent quantified his/her opinion on the dissimilarity in each pair of verbal attributes going from his/her internal image of timbre. The test results were evaluated using CLASCAL [15] and common perceptual space of verbal attributes was constructed.

The angles between attribute positions in the coordinate system of common perceptual space were calculated and the relations between them were qualified as follows [18]:

- a) small angles ($\alpha \leq 20^\circ$) \Rightarrow similar,
- b) nearly orthogonal positions ($70^\circ \leq \alpha \leq 110^\circ$) \Rightarrow independent,
- c) nearly opposite positions ($\alpha \geq 160^\circ$) \Rightarrow opposite meaning according to timbre.

The aim of the interpretation was to search for a system of verbal attributes, which completely describes the perceptual space and constitutes nearly orthogonal system.

3.2.2. Results

The three-dimensional common perceptual space as the optimal model for dissimilarity data and following nearly orthogonal system of verbal attributes were identified (see also Table 6 and Figure 4):

1. temný / tmavý — jasný / světlý
2. drsný / hrubý — jemný
3. plný / široký — úzký
- gloomy / dark — clear / bright
harsh / rough — delicate
full / wide — narrow

Verbal attributes creating nearly orthogonal system in common perceptual space can be regarded as an approximation of the new coordinate system of this space with dimensions defined by these attributes.

Angle [°]	gloomy	clear	harsh	delicate	full	narrow
gloomy	—		84	91	71	109
clear	176	—	97	89	106	75
harsh			—		92	77
delicate			159	—	106	85
full					—	
narrow					169	—

Table 6: Angles between selected verbal attributes representing dimensions of common perceptual space.

3.2.3. Discussion

The position of the most frequently used attribute *sharp* in perceptual spaces of all respondents and of individual professional groups [17] is similar and approaches to the plain of the first two dimensions in quadrant defined by attributes harsh and clear.

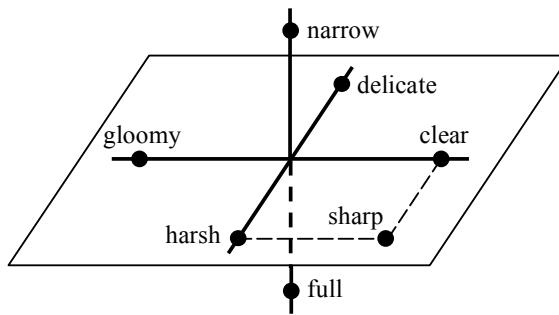


Figure 4: Nearly orthogonal system of verbal attributes in three-dimensional common perceptual space of timbre.

4. GENERAL DISCUSSION

The nearly orthogonal systems of verbal attributes fully explaining perceptual spaces constructed from results of listening tests of violin and organ sounds were not found. Let us try to look on these perceptual spaces (Figure 1–3) by monitoring positions of attributes of nearly orthogonal system describing common perceptual space of verbal attributes (Figure 4).

Attribute positions (when significant embeddings were found) in violin sound contexts are rarely opposite (1st dimension in violin tones F#4, C5 and D6, 3rd dimension only in B3) or orthogonal (only some attributes in 2nd and 3rd dimensions in F#4 and D6). The pair of attributes sharp — soft seems to be more prominent 'dimension' (see angles in Table 3), dimensional attributes of common perceptual space (Figure 4) surround this pair of attributes (Figure 1) in agreement with their positions according to sharp in common perceptual space: *clear, bright, harsh, rough to sharp*, and *gloomy, dark, delicate to soft*, but with small (acute) angles. The changing position of *narrow* is moreover supported in VAR test results (Figure 2). Attribute *rustle* with independent spectral source out of harmonic part of the sound in high violin

tones [10, 12] is another candidate for orthogonal (independent) timbre dimension.

In organ sound contexts the features of *sharp* and *narrow* attributes have similar influence and *round* with *soft* and also *noisy* have strong influence on the perception of organ sound.

Many spontaneously pronounced verbal attributes in described experiments of timbre assessment are the words used in the description of other senses perception and are "borrowed" by hearing. Only *noisy* and *rustle* are originated in hearing but were not a part of timbre imagination in nonlistening test, probably due to musicians effort to avoid undesired disturbing of noise from the ideal concept of timbre.

This provokes to the formulation of hypothesis of the four basic dimensions of timbre: 1. *gloomy* — *clear* (vision, sense of sight), 2. *harsh* — *delicate* (touch), 3. *full* — *narrow* (volume, filling), 4. *noisy* / *rustle* — ? (hearing).

5. CONCLUSIONS

Musicians internal imagination of timbre supports orthogonal dimensions but their saliency or relationship in real sounds is sound context dependent (for example depends on pitch or type of the instrument — violin, organ, etc.). The imagination of the timbral features orthogonality probably comes just from the large span of their relations (contained angles) in real situations.

The independence of musical sound timbre features defined by discovered verbal attributes must be proved by the future search for their sound-invariant spectral resources (like it was started in [10, 12, 13]) and followed by their quantitative description (like it was done for sharpness and for some other features of other sound contexts). The knowledge of these resources can be used for the intended control of musical instrument sound and audio effects as well.

6. ACKNOWLEDGEMENTS

Research was supported partly by the Ministry of Education and Youth of the Czech Republic, project No. 1M6138498401 and by the EU CRAFT project G1ST-CT-2002-50267-DEMORGPIPE.

7. REFERENCES

- [1] J. M. Hajda, R. A. Kendall, E. C. Carterette, and M. L. Harshberger, "Methodological issues in timbre research," in *Perception and Cognition of Music*, I. Deliège and J. E. Sloboda, Eds. Psychology Press, 1997, pp. 253–286.
- [2] G. von Bismarck, "Timbre of steady sounds: A factorial investigation of its verbal attributes," *Acustica*, vol. 30, pp. 146–159, 1974.
- [3] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *J. Acoust. Soc. Am.*, vol. 61, no. 5, pp. 1270–1277, 1977.
- [4] J. M. Grey and J. Gordon, "Perceptual effects of spectral modification on musical timbres," *J. Acoust. Soc. Am.*, vol. 63, no. 5, pp. 1493–1500, 1978.
- [5] J. Štěpánek, Z. Otčenášek, and A. Melka, "Comparison of five perceptual timbre spaces of violin tones of different pitches," in *CD-ROM of Joint Meeting 137th ASA, 2nd EAA Forum Acusticum 1999, 25th DAGA Berlin, 1999, 5aMUb5*.

- [6] R. Plomp, *The Intelligent Ear*, 2nd ed. London: Lawrence Erlbaum Associates, 2002.
- [7] M. Garnier, D. Dubois, J. Poitevineau, N. Henrich, and M. Castellengo, "Perception et description verbale de la qualité vocale dans le chant lyrique : une approche cognitive," in *Journées d'Étude sur la Parole (JEP04)*, Fes, Maroc, 2004.
- [8] J. Štěpánek, "Evaluation of timbre of violin tones according to selected verbal attributes," in *32nd Int. Acoustical Conf., European Acoustics Association (EAA) Symp. "Acoustics Banská Štiavnica"*, 2002, pp. 129–132.
- [9] ———, "The study of violin timbre using spontaneous verbal description and verbal attribute rating," in *Forum Acusticum*, Sevilla, Spain, 2002.
- [10] J. Štěpánek and Z. Otčenášek, "Interpretation of violin spectrum using psychoacoustic experiments," in *CD of Proc. Int. Symposium on Musical Acoustics (ISMA2004)*, vol. 4-S2-1, Nara, Japan, 2004.
- [11] J. Štěpánek, "Relations between perceptual space and verbal description in violin timbre," in *acústica 2004*, 2004, CD ROM, AFP 077-S.
- [12] J. Štěpánek and Z. Otčenášek, "Acoustical correlates of the main features of violin timbre perception," in *Proc. Colloq. Interdisciplinary Musicology (CIM'05)*, Montréal, Québec, Canada, Electronic proceedings, 2005, [Online] http://www.oicm.umontreal.ca/cim05/cim05_articles/STEPANEK_J_CIM05_01.pdf.
- [13] J. Štěpánek, Z. Otčenášek, V. Syrový, C. Täsch, and J. Angster, "Spectral features influencing perception of pipe organ sounds," in *Forum Acusticum 2005*, Budapest, 2005, pp. 465–469.
- [14] J. Štěpánek, V. Syrový, Z. Otčenášek, C. Täsch, and J. Angster, "Verbal description of organ principal 8' sound," in *32. DAGA, Braunschweig*, in print, 2006.
- [15] S. Winsberg and G. De Soete, "A latent class approach to fitting the weighted euclidean model, CLASCAL," in *Psychometrika*, vol. 58, 1993, pp. 315–330.
- [16] J. Štěpánek and Z. Otčenášek, "Listener common and group perceptual dimensions in violin timbre," in *Proc. Stockholm Music Acoustics Conf. (SMAC'03)* Stockholm, Sweden, 2003, pp. 663–666.
- [17] J. Štěpánek and O. Moravec, "Verbal description of musical sound timbre in Czech language and its relation to musicians profession and performance quality," in *Proc. Colloq. Interdisciplinary Musicology (CIM'05)*, Montréal, Québec, Canada, Electronic proceedings, 2005, [Online] http://www.oicm.umontreal.ca/cim05/cim05_articles/STEPANEK_J_CIM05_02.pdf.
- [18] J. Štěpánek, "Interpretation and comparison of perceptual spaces," in *Inter-Noise 2005*, Rio de Janeiro, 2005, CD-ROM, file: /papers/doc_1627.pdf.

MUSICAL KEY ESTIMATION OF AUDIO SIGNAL BASED ON HIDDEN MARKOV MODELING OF CHROMA VECTORS

Geoffroy Peeters

Ircam - Sound Analysis/Synthesis Team, CNRS - STMS

1, pl. Igor Stravinsky - 75004 Paris - France
peeters@ircam.fr

ABSTRACT

In this paper, we propose a system for the automatic estimation of the key of a music track using hidden Markov models. The front-end of the system performs transient/noise reduction, estimation of the tuning and then represents the track as a succession of chroma vectors over time. The characteristics of the Major and minor modes are learned by training two hidden Markov models on a labeled database. 24 hidden Markov models corresponding to the various keys are then derived from the two trained models. The estimation of the key of a music track is then obtained by computing the likelihood of its chroma sequence given each HMM. The system is evaluated positively using a database of European baroque, classical and romantic music. We compare the results with the ones obtained using a cognitive-based approach. We also compare the chroma-key profiles learned from the database to the cognitive-based ones.

1. INTRODUCTION

Considering its numerous applications (search/query music databases, playlists generation or automatic accompaniment), automatic estimation of musical key (key-note and mode) or of chord progression over time of a music track has received much attention in the recent years. Because symbolic transcriptions of music tracks are not always available, and because automatic transcription algorithms (audio to symbolic) are still limited and costly, many systems attempts to extract the key or chord progression directly from the audio signal. Most existing algorithms therefore start by a front-end which converts the signal frames to the frequency domain (FFT or CQT [1]) and then map it to the chroma domain [2] (or Pitch Class Profile [3]). Chroma/PCP vectors represent the intensities of the twelve semi-tones of the pitch classes over time. Algorithms then try to find the key or chord progression that best explains the succession of extracted chroma vectors. In order to estimate the key, Chew [4] proposes the Spiral Array Model/Center of Effect Generator. Most other authors [5], [6], [7], [8] use theoretical chroma/PCP profiles corresponding to the various keys. These profiles are derived from the probe tone experiment of Krumhansl & Schmukler [9] or from the modified version proposed by Temperley [10]. These experiments aimed at describing the perceptual importance of each semi-tone in a key. The result is a pitch distribution profile for each key. These profiles are then converted by the authors to key-chroma profiles.

Polyphonic audio signal: However, when trying to estimate the key from a music audio signal, a major difference with these experiments is the work with polyphonic signal (several notes played at the same time) and with audio signal (not only the frequencies of the pitch notes are observed but also all their

harmonics; therefore high values exist in the chroma vector at the fifth, third, ... intervals of the pitch notes). This problem has been addressed by Gomez [5]. She proposes to take into account the contribution of the harmonics of a note (by using a theoretical spectral envelope) and the polyphony (by considering the three main triads in each key) during the creation of the key-chroma profiles. This problem has also been addressed by Izmirli [6] who estimates directly the contribution of the harmonics of a note by measuring it with a database of piano notes.

Key estimation: The systems compute chroma/PCP vectors on a frame-based. The systems then try to find the most likely key that explains the overall set of frames. An assumption is often made about the existence of the key in the beginning of the track. Therefore, only the first part (first 20s of the first movement) of the track is considered. Several approaches are taken to go from the frame level to the global key. Gomez [5] finds the key by choosing the key-chroma profile which has the highest correlation coefficient with a global average chroma vector. Izmirli [6] computes at each time a cumulated chroma-vector (average from the beginning of the track) $\bar{c}(t)$; he then searches at each time the key-chroma profile which has the highest correlation with $\bar{c}(t)$, assigns to this key a score equal to the distance between the 1st and 2nd maximum correlation; and finally chose the key which has the maximum score over the first 20s.

When implementing such a key estimation system, one soon notes that the choice of the parameters (choice between Krumhansl or Temperley profile, number of considered harmonics, number of considered triads, ...) and the choice of the key estimation algorithm (type of distance, type of score, considered track duration) influence a lot the recognition rate of key. In fact many assumptions are made in these systems (fixed spectral envelope of a note, fixed polyphony, no modulation of the key) that do not necessarily correspond to the reality of the audio signal corresponding to a music track in a given key.

For these reasons, in this paper we study a system that models the keys using a set of hidden Markov models trained directly on the chroma representation. This choice allows us to avoid making the above-mentioned assumptions.

The paper is organized as follows: in section 2.1, we present the front-end of our system which extracts the chroma representation over time of a music track; in section 2.2, we present our key estimation system based on hidden Markov modeling; in section 3, we evaluate this system with a database of 300 tracks and compare the results with the ones obtained using a cognitive-based approach (combining [5] and [6] approaches). We then compare the key-chroma profiles learned by the HMMs to the cognitive-based ones.

2. PROPOSED METHOD

Our key estimation system is represented in Figure 1. In the following we detail the various steps of it.

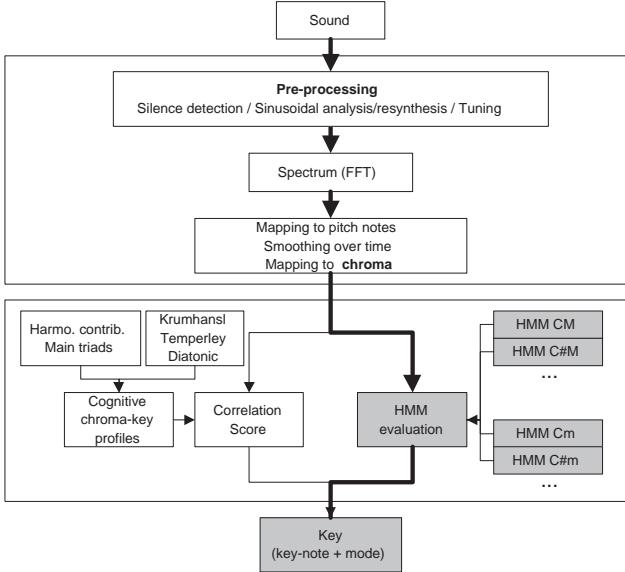


Figure 1: Key estimation systems: based on cognitive models (thin arrows) and on hidden Markov models (thick arrows).

2.1. Front-end: signal observation

2.1.1. Pre-processing

We first apply a set of pre-processing algorithms to the audio signal. In order to save computation time, the signal is first converted to mono and down-sampled to 11.025 Hz.

A *silence detectors* (based on loudness and spectral flatness measure) is applied in order to detect the actual beginning of the music in the audio signal.

A simple *sinusoidal analysis/re-synthesis* (spectrum peak-picking and short-term partial tracking) is applied in order to reduce transient and noise influence in the measures.

As in [11], the *tuning of the track* is then estimated. This is necessary because the instruments used during the recording may have used another tuning than 440 Hz and because possible transcoding of the audio media may have changed its tuning. We suppose the tuning constant over the track duration. We test a set of tunings between 427 Hz and 452 Hz (the quarter-tones below and above A4). For each tuning t and for each frame m , we compute a “modeling error” defined as the ratio between the energy of the spectrum explained by the current tuning (sum of the energy at the frequencies f_t corresponding to the semi-tones pitches based on the tuning t) and the total energy of the spectrum:

$$\epsilon(t, m) = 1 - \sum_n A(f_{t,n}, m) / \sum_f (A(f, m)) \quad (1)$$

where A denotes the amplitude of the Fourier transform and $f_{t,n}$ are the semi-tones pitches based on the tuning t :

$$f_{t,n} = t \cdot 2^{(n-69)/12} \quad n \in [43, 44, \dots, 95] \quad t \in [427, 452] \quad (2)$$

The tuning is then chosen as the value t which minimizes the modeling error over time. In Figure 2, we present the histogram of the tunings estimated on the 300 tracks database we will used in our experiments (see section 3.1). The signal is then re-sampled (using a polyphase filter implementation) in order to bring the tuning back to 440 Hz. The rest of the system can now be based on a tuning of 440 Hz.

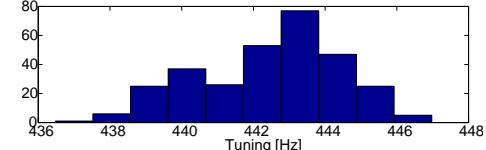


Figure 2: Histogram of the tunings estimated on the evaluation database (see section 3.1).

2.1.2. Chroma representation

Shepard [12] proposes to represent the pitch as a two dimensional structure: the tone height (octave number) and the chroma (pitch class). Based on that, the chroma spectrum or Pitch Class Profile (PCP) has been proposed in order to map the values of the Fourier transform (or Constant-Q transform) frequencies to the 12 semi-tones pitch classes C .

In our system, we first map the values of the Fourier transform to a *semi-tone pitch spectrum*, smooth the corresponding channels over time and then map the results to the *semi-tone pitch class spectrum* (chroma spectrum).

Semi-tone pitch spectrum: The mapping function between the frequencies f_k of the Fourier transform and the semi-tone pitch scale n (expressed in a midi-note scale) is defined as:

$$n(f_k) = 12 \log_2 \left(\frac{f_k}{440} \right) + 69 \quad n \in \mathbb{R}^+ \quad (3)$$

The computation of the semi-tone pitch spectrum is made using a set of filters $H_{n'}$ centered on the semi-tone pitch frequencies $n' \in [43, 44, \dots, 95]$ (corresponding to the notes G2 to B6 or the frequencies 98Hz to 1975 Hz). In order to increase the “pitch resolution”, we define a factor $R \in \mathbb{N}^+$ which fixes the number of filters used to represent one semi-tone. The center of the filters are now defined by $n' \in [43, 43 + \frac{1}{R}, 43 + \frac{2}{R}, \dots, 95]$. Each filter is defined by the function

$$H_{n'}(f_k) = \frac{1}{2} \tanh(\pi(1 - 2x)) + \frac{1}{2} \quad (4)$$

where x is the relative distance between the center of the filter n' and the frequencies of the Fourier transform: $x = R |n' - n(f_k)|$. The filters are equally spaced and symmetric in the logarithmic semi-tone pitch scale, extend from $n'-1$ to $n'+1$ with a maximum value at n' .

The values of the semi-tone pitch spectrum $N(n')$ are then obtained by multiplying the Fourier transform values $A(f_k)$ by the set of filters $H_{n'}$:

$$N(n') = \sum_{f_k} H_{n'}(f_k) A(f_k) \quad (5)$$

Smoothing: The semi-tone pitch spectrum $N(n')$ is computed for each frame m . The output signal of each filter $N(n', m)$ is

then smoothed over time using median filtering. This provides a reduction of transients and noise in these signals. Also, for the rest of the process, only the filters centered on the exact semi-tone pitches are considered (i.e. among the R filters representing one semi-tone, we only consider the middle one; for example if $R = 3$, we only keep $n=69$ but not $n=68.666$ and $n=69.333$). We can do this because the tuning is now guaranteed to be 440 Hz. This process also allows a reduction of the influence of noise in the computation of the chroma spectrum.

Semi-tone pitch class spectrum (chroma spectrum): The mapping function between the semi-tone pitches n and the semi-tone pitch classes (chroma) c is defined as $c(n) = \text{mod}(n, 12)$. The mapping to the 12-chroma scale vector $C(l)$ (pitch classes) is achieved by adding the equivalent pitch classes

$$C(l) = \sum_{n' \text{ so that } c(n')=l} N(n') \quad l \in [0, 12] \quad (6)$$

Parameters: The analysis is performed using Short Time Fourier Transform with a window of type blackman, length 371.5ms and 50% overlap. Because of frequency resolution limits (the frequency distance between adjacent semi-tone pitches becomes small in low frequency), we only consider frequencies above 100 Hz. The upper limit is set to 2000 Hz. The variable $A(f_k)$ in (5) can represent either amplitude, energy, log-amplitude or sone-converted values of the DFT. The results given in the following were obtained using the sone-converted values, which has given the best results in our case. The computation of the sone-converted values is similar to the one used in [13]. The value of R is set to 3.

2.2. Key estimation

In the following we will compare the estimation of key based on key-chroma profiles derived from Krumhansl/Temperley experiments with a system based on hidden Markov models trained directly on chroma representations. We first start by presenting the key-chroma profile method we will use in the experiment then we present our system.

2.2.1. Key estimation based on cognitive models

We have tested several systems based on cognitive models. The best results were obtained using a combination of [5] and [6].

Creation of key-chroma profiles: We use an approach similar to Gomez [5]: the key profiles are created by extending the Krumhansl & Schmukler (Temperley or Diatonic) pitch distribution profile to the polyphonic (several pitches) and audio (several harmonics for each pitch) cases. For each key, we consider the three main triads in this key: the tonic, dominant and sub-dominant triads (for example in C Major: C-E-G, G-B-D, F-A-C). The chroma vector corresponding to each single note of a specific triad is computed by adding the contribution of its harmonics h . The harmonic h is given a contribution of 0.6^{h-1} . Only the first 4 harmonics are considered. For a specific triad, the chroma vectors corresponding to the three notes are added. Finally for a specific key, the key-chroma vector is computed by adding the three triad-chroma vectors. Each triad-chroma vector is weighted by the value of the Krumhansl's (Temperley or Diatonic) profile at the position corresponding to the position of the root of the triad in the key (for example 6 for the F-A-C triad in C Major). The result is a 12 dimensions chroma profile vector for each of the 24 keys: $\underline{C}_i \ i \in [1, 24]$. In the following, we will

use the pitch distribution profile proposed in [6] (combination of Temperley and Diatonic profile) which has given the best results in our case.

Estimation of key: The most likely key of the track is estimated using an approach similar to Izmirli [6]. The chroma vectors $\underline{c}(t)$ are extracted on a frame basis. At each time t , we estimate the key \underline{C}_i that has the highest correlation with a cumulated-over-time chroma-vector¹. We attribute a score to this key proportional to the distance between its correlation value and the correlation value of the second most likely key. This score acts as a reliability coefficient. The final key decision is chosen as the key with the maximum score cumulated over time. Only the first 20 seconds of the tracks are considered.

2.2.2. Key estimation based on hidden Markov models

In the following we propose the use of hidden Markov models [14] trained and evaluated directly on the temporal sequences $\underline{c}(t)$ of chroma vectors.

In comparison with the above-mentioned approach, the advantages are

1. it does not necessitate to make assumptions about the presence of harmonics of the pitch notes in the chroma representation, or assumptions about specific polyphony since they will be inherently learned from the training set;
2. it does not necessitate the choice of a specific pitch distribution profile (Krumhansl, Temperley or Diatonic);
3. it allows to take into account possible modulation of key over time (through the transition probabilities).

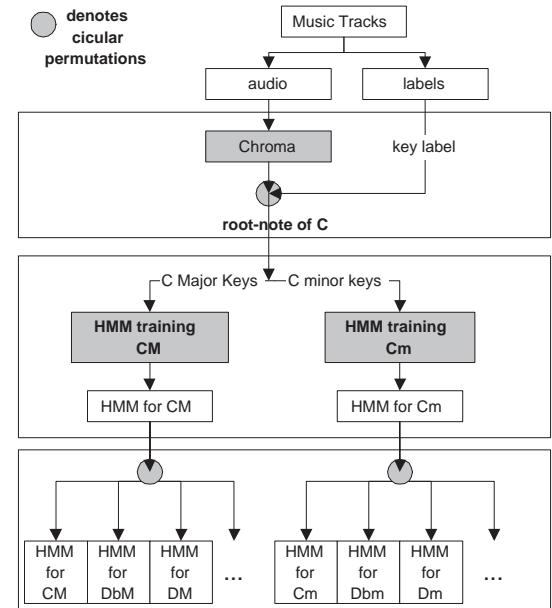


Figure 3: HMM training for key estimation.

Training of key-chroma HMMs: We want to create a specific HMM for each of the 24 possible keys (12 key-notes, 2 modes).

¹ At time t , the cumulated-over-time chroma-vector is computed by averaging the chroma vectors $\underline{c}(\tau)$ since the beginning of the track: $1/t \sum_{\tau=0}^t \underline{c}(\tau)$.

However, because the number of instances in our database strongly differs among the 24 keys, training directly the HMMs on the set of items belonging to a specific key could lead to over-fitting (learning the track characteristics instead of the key characteristics). We therefore start by training only two models, a Major and a minor *mode* model, and then map the two trained models to the various possible *key-notes*. The training process is depicted in Figure 3. It consists in the following steps:

1. map the chroma-vectors of all the tracks of the training set to a root-note of C (by using circular permutation of chroma vectors);
2. train an HMM for the C Major (C minor) key by using all the tracks in C Major mode (C minor mode);
3. construct the HMMs for the other Major (minor) keys by mapping the Major (minor) HMM to the various key-notes (Db, D, Eb, ...). This is done by circular permutation of the mean vectors and covariance matrices of the state observation probability. 24 HMMs are obtained in this way from the two trained HMMs.

The training of the HMMs is made using the Baum-Welsh algorithm. We have tested various HMM configurations (number of states, number of mixtures)^{2 3}.

Estimation of key: For a song with unknown key, we evaluate the log-likelihood of its chroma-vector sequence given each of the 24 HMMs. This is done using the forward algorithm. The model giving the maximum log-likelihood determines the key.

3. EVALUATION

3.1. Test set

The evaluation of our system is performed on a database of 302 European baroque, classical and romantic music extracts: Bach (48), Corelli (12), Handel (16), Telleman (17), Vivaldi(6), Beethoven (33), Haydn (23), Mozart (33), Brahms (32), Chopin (29), Dvorak (18), Schubert (23), Schuman (7). The pieces are for solo keyboard (piano, harpsichord), chamber and orchestra music. No opera or choir music has been considered in the present study. As in [6], the database was derived from the NAXOS web radio service. The ground-truth key (key-note and mode) was derived from the title of the piece. Only the first movement of each piece, supposed to correspond to the provided key, was used. Note that we had to manually correct the annotation of part of the baroque pieces since they were based on a tuning of A4=415Hz.

3.2. Evaluation method

For each track, we extract the chroma vectors of the first 20s as described in section 2.1. We then compare the estimation of the key

² Note that HMM has already been used in the context of chord progression estimation. The system proposed in [15] recognizes the chord progression by decoding a single HMM in which each state represents a specific chord. In our case, a specific key is represented by a specific HMM and the meaning of its states remains unspecified. Our system recognizes the key by finding the most likely HMM over 24 HMMs.

³ Also, the idea of chroma rotation has been used in the context of chord progression estimation. [15] uses it after the training of its single HMM in order to average the parameters of the state observation probabilities (the chord models). Our process is different since we use it before the training (in order to train only two HMMs) and after the training (in order to construct the 24 HMMs). We do not perform averaging.

	MIREX score	Correct key	Correct key-note	Correct mode	5th up	5th down	relative M/m	parallel M/m	Correct + neighboring
Cog.-base approach	88.9	85.1	88.4	92.7	0.6	5.0	1.0	3.3	95.0
HMM S=3, M=1 diag	84.6	80.4	86.2	86.8	0.9	2.7	3.9	5.8	93.7
HMM S=6, M=1 diag	81.3	76.1	85.3	83.2	1.5	3.3	3.0	9.1	93.0
HMM S=12, M=1 diag	84.2	79.8	88.0	85.6	2.4	1.5	2.7	8.2	94.6
HMM S=3, M=3 diag	85.5	81.0	87.4	88.0	2.1	2.4	3.0	6.4	94.9
HMM S=6, M=3 diag	84.2	79.5	87.7	85.3	1.8	2.1	3.6	8.2	95.2
HMM S=12, M=3 diag	85.5	80.7	86.2	87.4	2.1	2.7	3.9	5.5	94.9
HMM S=12, M=3 full	62.7	52.5	66.0	59.0	2.1	1.5	16.5	13.4	86.0

Table 1: Recognition rate of key for the cognitive-based and HMM-based approach.

using the cognitive-based (section 2.2.1) and HMM-based (section 2.2.2) approaches. For the system based on HMMs, we have performed a ten-folds cross-validation (each time the HMMs were trained on 9 folds and evaluated on the remaining one). We indicate the recognition rate of key, key-note alone and mode alone. We also indicate the score used for the MIREX-2005 key estimation contest⁴. This score uses the following weights: - 1 for correct key estimation (CM → CM), - 0.5 for perfect fifth relationship between estimated and ground-truth key (CM → GM), - 0.3 if detection of relative Major/minor key (CM → Am), - 0.2 if detection of parallel Major/minor key (CM → Cm).

3.3. Results

The results are indicated into Table 1. We have tested various configurations of the HMMs: number of emitting states (S=3, 6, 12), number of Gaussian distributions for each state (M=1, 3). Each Gaussian is described by its 12-dimensions mean vector $\mu_{s,m}$ and its covariance matrix $\Sigma_{s,m}$. We have considered independence between chroma values, so that $\Sigma_{s,m}$ are diagonal matrices.

Compared to the cognitive-based approach (88.9% MIREX score), the HMM-based approach leads in all cases to a lower recognition rate (maximum of 85.5% MIREX score). The confusion with the 5th up, relative and parallel Major/minor of the key are larger than the ones obtained with the cognitive-based approach. This can be explained partly by the fact that - the tracks used for the training of the models do not only contain the main key but also neighboring keys - part of the tracks (especially for the romantic period: Brahms, Schuman) start in a neighboring key. The last column of the table indicates the number of “not so bad recognition” (correct recognition + recognition of neighboring keys). This number is very similar for both cognitive and HMM-based approaches (95%). This means that the number of gross errors is similar in both cases. It is important to note that no a priori musical knowledge has been introduced in the HMM-based approach.

Comparing the various configurations of the HMMs, we see that increasing the number of Gaussian distributions for each state (M) slightly increases the recognition rate, but increasing the number of emitting states (S) does not influence significantly the results. In the last row, we indicate the results obtained when considering dependence between the chroma values (full covariance

⁴http://www.music-ir.org/mirex2005/index.php/Audio_and_Symbolic_Key-Finding

matrix) for the case S=12 / M=3. This decreases significantly the results.

Comparison between HMM-based and cognitive-based key-chroma profiles: In Figure 4, we indicate the 12-dimensions mean vector of the emitting states of the Major and minor mode HMMs. For this, we have trained the two HMMs on the whole database in a S=3 / M=1 configuration. We compare these vectors with the two cognitive key-chroma profiles as computed according to section 2.2.1. For the Major mode, state 1 (S1) and 3 (S3) of the HMM are pretty close to the cognitive key-chroma profile. For the minor mode, the closest state of the HMM is S2 but it does not agree with the cognitive profile on the importance of the 10th and 11th pitch classes (A and Bb in C minor). This could correspond to the presence in our database of other minor modes than the harmonic one which is the one used in the Temperley/Diatonic profile.

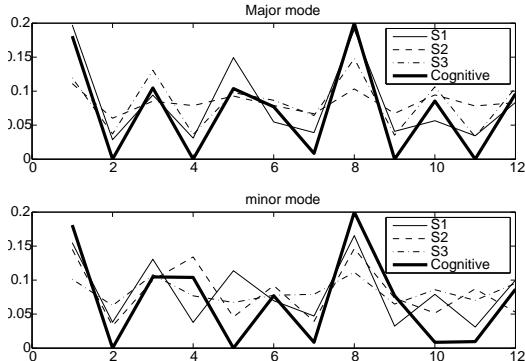


Figure 4: Comparison between learned and cognitive-based key-chroma profiles for the Major and minor mode.

4. CONCLUSIONS

In this paper, we have proposed a system for the automatic estimation of the key of a music track from the analysis of its audio signal. The system is based on a front-end that extracts chroma vectors over time and uses them as observations. The characteristics of the Major and minor modes are learned by training two hidden Markov models on a labeled database. 24 hidden Markov models corresponding to the various keys are then derived from the two trained models. The estimation of the key of a music track is then obtained by computing the likelihood of its chroma sequence given each HMM. The system is evaluated using a database of European baroque, classical and romantic music. The results are compared with the ones obtained using a cognitive-based approach based on extensions of Krumhansl/Temperley pitch distribution profiles to the audio/polyphonic case. The results obtained with the HMM approach (85.5%) remain lower than the ones obtained with the cognitive-based approach (88.9%) but the number of gross errors is similar in both cases. This indicates that a system without any a priori musical knowledge can learn the characteristics of the keys from a labeled database. Comparing the chroma-key profiles learned from the database to the cognitive-based ones gives fairly good agreement for the Major mode, but differs at the 10th and 11th pitch classes for the minor mode. Future works will concentrate on improving the chroma representation and on testing the training/evaluation on the whole track duration. We would also like to test this method for modeling the characteristics of composition styles.

5. ACKNOWLEDGEMENTS

Part of this work was conducted in the context of the European I.S.T. project Semantic HIFI⁵ [16]. To my father.

6. REFERENCES

- [1] J. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Am.*, vol. 89, no. 1, pp. 425–434, 1991.
- [2] G. Wakefield, "Mathematical representation of joint time-chroma distributions," in *SPIE Conf. Advanced Sig. Proc. Algorithms, Architecture and Implementations*, Denver, Colorado, USA, 1999, pp. 637–645.
- [3] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, 1999, pp. 464–467.
- [4] C.-H. Chuan and E. Chew, "Fuzzy analysis in pitch class determination for polyphonic audio key finding," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'05)*, London, UK, 2005, pp. 296–303.
- [5] E. Gomez, "Tonal description of polyphonic audio for music content processing," *INFORMS J. on Computing. Special cluster on music computing*, vol. 18, no. 3, to appear in 2006.
- [6] O. Izmirli, "Template based key finding from audio," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 211–214.
- [7] M. Cremer and C. Derboven, "A system for harmonic analysis of polyphonic music," in *AES 25th Int. Conf.*, London, UK, 2004, pp. 115–120.
- [8] S. Pauws, "Musical key extraction from audio," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, 2004, pp. 96–99.
- [9] C.-L. Krumhansl, *Cognitive foundations of musical pitch*. New-York: Oxford University Press, 1999.
- [10] D. Temperley, "What's key for key? the Krumhansl-Schmuckler key finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.
- [11] C. Harte and M. Sandler, "Automatic chord identification using a quantised chromagram," in *AES 118th Convention*, Barcelona, Spain, 2005, preprint 6412.
- [12] R. Shepard, "Circularity in judgements of relative pitch," *J. Acoust. Soc. Am.*, vol. 36, pp. 2346–2353, 1964.
- [13] E. Pampalk, S. Dixon, and G. Widmer, "Exploring music collections by browsing different views," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'03)*, Baltimore, Maryland, USA, 2003, pp. 201–208.
- [14] L. Rabiner, "A tutorial on Hidden Markov Model and selected applications in speech," *Proc. IEEE*, vol. 77, no. 2, pp. 257–285, 1989.
- [15] A. Sheh and D. Ellis, "Chord segmentation and recognition using EM-trained Hidden Markov Models," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'03)*, Baltimore, Maryland, USA, 2003, pp. 183–189.
- [16] H. Vinet, "The semantic Hifi project," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 503–506.

⁵<http://shf.ircam.fr>

ONSET DETECTION REVISITED

Simon Dixon

Austrian Research Institute for Artificial Intelligence
Freyung 6/6, Vienna 1010, Austria
simon.dixon@ofai.at

ABSTRACT

Various methods have been proposed for detecting the onset times of musical notes in audio signals. We examine recent work on onset detection using spectral features such as the magnitude, phase and complex domain representations, and propose improvements to these methods: a weighted phase deviation function and a half-wave rectified complex difference. These new algorithms are compared with several state-of-the-art algorithms from the literature, and these are tested using a standard data set of short excerpts from a range of instruments (1060 onsets), plus a much larger data set of piano music (106054 onsets). Some of the results contradict previously published results and suggest that a similarly high level of performance can be obtained with a magnitude-based (spectral flux), a phase-based (weighted phase deviation) or a complex domain (complex difference) onset detection function.

1. INTRODUCTION

Many music signal analysis applications require the accurate detection of onsets of musical tones, and it is not surprising that several different methods have been proposed for performing onset detection. At first sight, onset detection is a well-defined task: the aim is to find the starting time of each musical note (where a musical note is not restricted to those having a clear pitch or harmonic partials). However, in polyphonic music, where nominally simultaneous notes (chords) might be spread over tens of milliseconds, the definition of onsets starts to become blurred. Likewise, instruments with long attack times (e.g. flute) produce notes for which it is difficult to define an unambiguous and precise onset time.

The most natural way to approach this problem is to attempt to define onset times in line with human perception, for example using the work of Vos and Rasch [1], who distinguished between the physical and perceptual onset times of musical tones, and showed that the perceptual onset time occurs when the tone reaches a level of approximately 6 – 15 dB below its maximum value. However, their research did not deal with the type of situations faced in analysing audio recordings of complex musical works, where factors such as masking, temporal order thresholds and just noticeable differences lay to rest any hope of a crisp definition of onset for real-world data.

In this paper, we take a more pragmatic approach to onset detection, allowing the available data sets to guide the definition of onsets, sometimes corresponding to perceived onset times, and sometimes to physical onset times. The bulk of the first data set is hand-labelled, that is, listeners have marked the positions in the audio file at which they perceive onsets. If multiple listeners are used for each file, a reasonably robust data set can be developed, but this involves much work, so only small data sets can be produced in this way. The remaining data is collected from computer-monitored pianos, which are able to measure the physical onset times of notes with a high degree of accuracy. Since it is not fea-

sible for listeners to annotate the perceived onsets of all 106054 notes in this collection of piano sonatas, this is a method by which large data sets can be quickly collected. The only drawback is the disparity between physical and perceptual onset times (assuming that the goal is to find perceived onsets). In the case of percussive instruments, the difference is of the order of a few milliseconds, which is sufficiently precise for our purposes, and at least as accurate as human-labelled data.

In a recent tutorial article, Bello et al. [2] reviewed a number of onset detection algorithms, making a theoretical and empirical comparison of several state-of-the-art approaches. In this paper, we complement and extend their work by introducing new onset detection functions based on their work, and by testing the new methods alongside independent implementations of a subset of the published methods on the same data set and on a second data set which is two orders of magnitude larger. Other comparisons of onset detection methods can be found in [3, 4]. We restrict our comparison to methods based on short term spectral coefficients, which are the most widely used methods, and the most successful according to the 2005 MIREX audio onset detection evaluation [4].

In the next section, we introduce onset detection functions, review three state-of-the-art functions as presented in [2, 5], and describe three new functions which are extensions of the published algorithms. Section 3 addresses the evaluation of these onset detection functions, starting with methodological concerns, then describing the test data and finally presenting and discussing the results of the tests. The final section contains the conclusions and some ideas for further work.

2. ONSET DETECTION FUNCTIONS

An onset detection function is a function whose peaks are intended to coincide with the times of note onsets. Onset detection functions usually have a low sampling rate (e.g. 100Hz) compared to audio signals; thus they achieve a high level of data reduction whilst preserving the necessary information about onsets. Most onset detection functions are based on the idea of detecting changes in one or more properties of the audio signal. But audio signals, whether composed of natural or synthetic sounds, are in a continual state of change, so the task of onset detection also involves distinguishing between the various types of change, such as onsets, offsets, vibrato, amplitude modulation and noise.

If an audio signal is observed in the time-frequency plane, the onset of a new sound has noticeable energy in the frequency bands in which the sound is not masked by other simultaneous components. Thus an increase in energy (or amplitude) within some frequency band(s) is a simple indicator of an onset. Alternatively, if we consider the phase of the signal in various frequency bands, it is unlikely that the frequency components of the new sound are in phase with previous sounds, so irregularities in the phase of var-

ious frequency components can also indicate the presence of an onset. Further, the phase and energy (or magnitude) can be combined in various ways to produce more complex onset detection functions. This is the basis of the onset detection functions presented in this section.

The next 3 subsections briefly review existing approaches to onset detection using spectral flux, phase deviation and complex domain methods (for a more in depth review, see [2]). Then we present potential improvements to these methods, defining weighted phase deviation, normalised weighted phase deviation and half-wave rectified complex domain onset detection functions. All of these methods make use of a time-frequency representation of the signal based on a short time Fourier transform using a Hamming window $w(m)$, and calculated at a frame rate of 100 Hz. If $X(n, k)$ represents the k th frequency bin of the n th frame, then:

$$X(n, k) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(hn + m) w(m) e^{-\frac{2j\pi mk}{N}}$$

where the window size $N = 2048$ (46 ms at a sampling rate of $r = 44100$ Hz) and hop size $h = 441$ (10 ms, or 78.5% overlap).

2.1. Spectral Flux

Spectral flux measures the change in magnitude in each frequency bin, and if this is restricted to the positive changes and summed across all frequency bins, it gives the onset function SF [6]:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n, k)| - |X(n-1, k)|)$$

where $H(x) = \frac{x+|x|}{2}$ is the half-wave rectifier function. Empirical tests favoured the use of the L_1 -norm here over the L_2 -norm used in [7, 2], and the linear magnitude over the logarithmic (relative or normalised) function proposed by Klapuri [8].

2.2. Phase Deviation

The rate of change of phase in an STFT frequency bin is an estimate of the instantaneous frequency of that component. This can be calculated via the first difference of the phase of $X(n, k)$. Let $\psi(n, k)$ be the phase of $X(n, k)$, that is:

$$X(n, k) = |X(n, k)| e^{j\psi(n, k)}$$

where $-\pi < \psi(n, k) \leq \pi$. Then the instantaneous frequency is given by the first difference $\psi'(n, k)$:

$$\psi'(n, k) = \psi(n, k) - \psi(n-1, k)$$

mapped onto the range $(-\pi, \pi]$. The change in instantaneous frequency, which is an indicator of a possible onset, is given by the second difference of the phase:

$$\psi''(n, k) = \psi'(n, k) - \psi'(n-1, k)$$

which is also mapped onto the range $(-\pi, \pi]$. Large discontinuities in the unwrapped phase or its derivatives can wrap around to 0, but the onset detection function based on phase deviation, PD , takes the mean of the absolute changes in instantaneous frequency across all bins [9, 2], which reduces the chance of a missed detection:

$$PD(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |\psi''(n, k)|$$

2.3. Complex Domain

Amplitude and phase can be considered jointly to search for departures from steady-state behaviour by calculating the expected amplitude and phase of the current bin $X(n, k)$, based on the previous two bins $X(n-1, k)$ and $X(n-2, k)$. The target value $X_T(n, k)$ is estimated by assuming constant amplitude and rate of phase change:

$$X_T(n, k) = |X(n-1, k)| e^{\psi(n-1, k) + \psi'(n-1, k)}$$

and therefore a complex domain onset detection function CD can be defined as the sum of absolute deviations from the target values:

$$CD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n, k) - X_T(n, k)|$$

This formulation is simpler but equivalent to the complex domain detection function in [2, 5].

2.4. Weighted Phase Deviation

In the remainder of this section we propose improvements to the onset detection functions described in the literature. The first idea addresses the problem that the PD function “is susceptible ... to noise introduced by components with no significant energy” [2]. That is, the function considers all frequency bins k equally, although the energy of the signal is concentrated within the bins containing the partials of the currently sounding tones. We propose weighting the frequency bins by their magnitude, giving a new onset detection function which we call the *weighted phase deviation* (WPD):

$$WPD(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n, k) \psi''(n, k)|$$

This is similar to the CD function, in that the magnitude and phase are considered jointly, but with a different manner of combination. A further option is to define a *normalised weighted phase deviation* ($NWPD$) function, where the sum of the weights is factored out to give a weighted average phase deviation:

$$NWPD(n) = \frac{\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n, k) \psi''(n, k)|}{\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n, k)|}$$

2.5. Rectified Complex Domain

One problem with the CD method is that it does not distinguish between increases and decreases in amplitude of the signal. Since it is important to distinguish onsets from offsets, we propose using a similar idea to that used in the SF function, where half-wave rectification is used to preserve only the increases in energy in spectral bins. This idea can easily be incorporated into the CD method, giving a (half-wave) rectified complex domain (RCD) onset detection function as follows:

$$RCD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} RCD(n, k)$$

where

$$RCD(n, k) = \begin{cases} |X(n, k) - X_T(n, k)|, & \text{if } |X(n, k)| \geq \\ & |X(n-1, k)| \\ 0, & \text{otherwise} \end{cases}$$

2.6. Onset Selection

The onsets are selected from the detection function by a peak-picking algorithm which finds local maxima in the detection function, subject to various constraints. The thresholds and constraints used in peak-picking have a large impact on the results, specifically on the ratio of false positives to false negatives. For example, a higher threshold generally reduces the number of false positives and increases the number of false negatives. The best values for thresholds are dependent on the application and the relative undesirability of false positives and false negatives. It is difficult to generate threshold values automatically, so we follow Bello et al. [2] in reporting results for optimal parameter settings, which also allows a fair comparison with their published results.

Peak picking is performed as follows: each onset detection function $f(n)$ is normalised to have a mean of 0 and standard deviation of 1. Then a peak at time $t = \frac{n_h}{r}$ is selected as an onset if it fulfils the following three conditions:

$$\begin{aligned} f(n) &\geq f(k) \text{ for all } k \text{ such that } n-w \leq k \leq n+w \\ f(n) &\geq \frac{\sum_{k=n-mw}^{n+w} f(k)}{mw + w + 1} + \delta \\ f(n) &\geq g_\alpha(n-1) \end{aligned}$$

where $w = 3$ is the size of the window used to find a local maximum, $m = 3$ is a multiplier so that the mean is calculated over a larger range before the peak, δ is the threshold above the local mean which an onset must reach, and $g_\alpha(n)$ is a threshold function with parameter α given by:

$$g_\alpha(n) = \max(f(n), \alpha g_\alpha(n-1) + (1-\alpha)f(n))$$

Experiments were performed with various values of the two parameters δ and α , and it was found that best results were obtained using both parameters, but the improvement in results due to the use of the function $g_\alpha(n)$ was marginal, assuming a suitable value for δ is chosen.

3. EVALUATION OF ONSET DETECTION FUNCTIONS

In this section, we discuss our testing methodology, describe the data sets, and present the results from testing the above onset detection functions on each data set.

3.1. Methodology

The main difficulty with the evaluation of onset detection algorithms is that of obtaining a significantly large and balanced set of recordings for which the onset times are known (ground truth data). Precise measurements of onset times are only available for a small fraction of music, such as piano music recorded on computer-monitored pianos, and music generated with a MIDI synthesiser. Other data must be labelled by hand, which is a laborious and error-prone task. By balanced, we mean that the data set should be representative of the full range of data that the system is intended to be used for, including the proportions of pieces per instrument, per style, per level of complexity, etc. If the test set

is not representative of real-world data, then the results reported will be overly optimistic (or in some cases pessimistic) of the actual performance of the algorithm. Finally, the data set should be large enough that separate training and test sets can be established in order to avoid overfitting.

A second methodological problem is determining how to report and compare results. Each onset detection function has parameters which can be tuned to alter the proportion of false positives (reported detections where no onset exists) and false negatives (missed detections). These proportions are often expressed in terms of precision and recall (defined below), but it is not sufficient to report the precision and recall alone, that is, via one pair of values. The relationship of precision to recall is often shown graphically in a receiver operating characteristic (ROC) curve, and if a single scalar statistic is desired (to make comparisons simple), the precision and recall can be combined into a single value such as the area under the ROC curve or the F-measure, which represents the optimal point on the ROC curve.

A third problem is how to deal with situations where a number of onsets are very close together, for example when a chord is played on a guitar or piano. Depending on the time between the notes, one or more onsets might be perceived, but this is dependent on the instrument and presence of other simultaneous sounds. The MIREX 2005 [10] onset detection evaluation addressed this problem by counting the number of merged onsets (two onsets detected as a single onset) and double onsets (a single onset recognised as two) in addition to the standard counts of correct detections, false positives and false negatives¹.

In this work, we consider an onset to be correctly matched if a detected onset is reported within 50 ms of the ground truth onset time. We do not penalise merged onsets, since the data we have contains many simultaneous or almost-simultaneous notes, and we are not attempting to recognise the notes. The results are summarised by three statistics: the precision P , recall R and F-measure F (for the optimal parameter settings), which are given by:

$$\begin{aligned} P &= \frac{c}{c + f^+} \\ R &= \frac{c}{c + f^-} \\ F &= \frac{2PR}{P + R} = \frac{2c}{2c + f^+ + f^-} \end{aligned}$$

where c is the number of correct detections, f^+ is the number of false positives and f^- is the number of false negatives. Parameters were chosen to maximise F ; for certain applications where false positives and false negatives are not equally undesirable, different parameter values would be more suitable. Further discussion of onset detection evaluation can be found in [11].

3.2. Data

The first set of tests were performed on the data used by Bello et al. [2], consisting of 4 sets of short excerpts from a range of instruments, classed into the following groups: NP — non-pitched percussion, such as drums (119 onsets); PP — pitched percussion, such as piano and guitar (577 onsets); PN — pitched non-percussion, in this case solo violin (93 onsets); and CM — complex mixtures from popular and jazz music (271 onsets). Although there are only 1060 onsets in these sets of excerpts, they offer two important advantages: to test the algorithms on a range of different

¹http://www.music-ir.org/mirex2005/index.php/Audio_Onset_Detection

instruments, and second, to enable a direct comparison with other published work.

The second set of data contains about 4 hours of solo piano music played by a professional pianist on a Bösendorfer computer-monitored grand piano². This data consists of 106054 notes — two orders of magnitude more than that used in other evaluations — and includes complex passages such as trills, fast scale passages with pedal and arpeggiated chords. The level of complexity is such that a human annotator would have immense difficulty marking all the onsets precisely.

3.3. Results and Discussion

Table 1 shows the results for 8 different onset detection functions tested on the 4 data sets used in [2]. In each case, the results are shown for the point on the ROC curve which gives the maximum value of the F-measure. That is, the ground-truth data was used to select optimal values of δ and α . A similar approach was taken in [2], so the comparison with the published results, which are included in the table in the rows marked by asterisks (SF* and PD*), is fair.

The first point to note is that there are some large discrepancies between the published results and our own implementations of the same functions. For example, SF* performs particularly well with the data set PP in comparison with the PN and NP data sets, but our implementation (SF) shows much smaller performance differences across these 3 sets of excerpts. SF also achieves better performance across the entire range of data, presumably due to a better peak-picking function. Even greater differences are evident in the results of the phase deviation functions, where our PD function achieved much worse performance than the published PD* results. The closeness of the PD* results to the WPD and NWPD results raised the suspicion that perhaps some weighting scheme been used in the PD* algorithm, and this was later confirmed by one of the authors, who had mistakenly thought it was an unimportant detail. It is noteworthy that relatively small differences in implementation have a large impact on results, and that some of the differences are specific to particular data sets. However, we add two caveats: first, that parameter settings greatly influence the results, and this could be the source of some of the differences, and second, that the differences in performance are not necessarily significant considering the size of the test sets.

The second point that we note from Table 1 is that the WPD and NWPD are both very significant improvements on the PD function, but the normalisation is only an improvement on the WPD in two cases (PP and CM), while for the other two cases a slight degradation in performance results. Finally, the RCD method offers a small improvement on the CD on this data, but considering the small size of the data set, this difference might not be significant.

Overall, these results show that spectral flux, weighted phase deviation and complex domain methods can all achieve a similarly high level of performance on these data sets. Since the data sets are small and not sufficiently general, we are not willing to draw further conclusions about the differences between these methods, except to state that spectral flux has the advantage of being the simplest and fastest algorithm.

We now turn to the second set of results (shown in Table 2), which comes from a much larger but more homogeneous set of data, 13 complete piano sonatas by Mozart. Results were collected using a single parameter set for each function, and then

²Sonatas K. 279–284, 330–333, 457, 475 and 533, from Wolfgang Amadeus Mozart: *The Complete Piano Sonatas*, played by Roland Batik, Gramola 98701–705, 1990.

	P	R	F	E (ms)
SF	0.958	0.969	0.964±0.017	8.8
PD	0.555	0.868	0.677±0.044	19.5
WPD	0.903	0.921	0.912±0.028	9.6
NWPD	0.945	0.944	0.944±0.021	10.3
CD	0.970	0.962	0.966±0.015	12.8
RCD	0.952	0.958	0.955±0.018	9.3

Table 2: Results for a database of complex piano music consisting of 106054 onsets: the columns are precision (P), recall (R), F-measure with standard deviation of F-measures across sonatas (F) and average absolute error in ms (E).

summed across all sonatas, but the F-measure was also calculated separately for each piece, to give an indication of the variation in results. Since piano is a pitched percussion instrument, one would expect the results to be similar to those shown in the PP columns of Table 1, and this is basically true. The top 3 algorithms are the same in both cases (SF, CD and RCD), with a range of less than one standard deviation between them. The fact that their order has changed does not appear to be significant. The remaining algorithms preserved their ranking, with NWPD very close to the third placed algorithm, WPD somewhat further behind, and PD performing very poorly.

The results for the piano sonatas are overall lower than for the PP data set. This would be expected, since the PP data set contains relatively simple music, and most of the piano music does not even use the damper pedal, so the likelihood of sustained tones masking new onsets is greatly reduced. Considering the difference in complexity of the music, it is surprising that the drop in algorithm performance is not greater; on the contrary, the results are very encouraging.

Another factor that we can take into account with this data is the errors in correctly detected onsets, which are summarised in the right column of Table 2 as an average absolute value. If precision of onset detection is important, the SF function has a slight advantage over other methods. Such a comparison is not possible with the hand-labelled data, since the timing errors in hand-labelling are much greater than the errors we observe in Table 2.

One previous study involved the use of genetic algorithms to learn an optimal set of parameters for combining various simple onset detection functions [12]. Testing was performed on 10 of the same piano sonatas, and the results were equivalent to an average F-measure of 0.94 across different training sets, with an average error of 11 ms. This is slightly worse than the best results achieved here, and the detection window was greater (70 ms instead of 50 ms), which would give higher detection rates. Further, the system had to be trained to learn a larger set of parameters than the one or two parameters which were varied in this work.

4. CONCLUSIONS

We revisited a recent study on onset detection and proposed 3 new onset detection functions and a new peak-picking algorithm as improvements on the published methods. Tests on a common data set supported the claim that the new methods are better, but more extensive tests using a large set of piano music showed the spectral flux and complex domain functions to be marginally better than the weighted phase deviation functions. The test results contradicted some findings in the literature, which probably indicates an instability in the results with respect to small differences in imple-

	PN data			PP data			NP data			CM data		
	P	R	F	P	R	F	P	R	F	P	R	F
SF*	0.914	0.871	0.892	0.984	0.949	0.966	0.945	0.816	0.876	0.896	0.804	0.848
SF	0.938	0.968	0.952	0.981	0.988	0.984	0.959	0.975	0.967	0.882	0.882	0.882
PD*	0.957	0.957	0.957	0.997	0.955	0.976	0.945	0.807	0.871	0.753	0.801	0.776
PD	0.654	0.935	0.770	0.482	0.865	0.619	0.750	0.933	0.831	0.663	0.749	0.704
WPD	0.937	0.957	0.947	0.899	0.925	0.912	0.974	0.958	0.966	0.843	0.830	0.836
NWPD	0.909	0.968	0.938	0.961	0.981	0.971	0.950	0.966	0.958	0.916	0.845	0.879
CD	0.946	0.946	0.946	0.971	0.984	0.978	0.948	0.924	0.936	0.941	0.819	0.876
RCD	0.948	0.978	0.963	0.983	0.979	0.981	0.944	0.983	0.963	0.945	0.819	0.877

Table 1: Results of onset detection tests, showing precision (P), recall (R) and F-measure (F) for the data sets pitched non-percussive (PN), pitched percussive (PP), non-pitched percussive (NP) and complex mixture (CM), for 8 different onset detection functions (see section 2). The functions marked with asterisks are results in [2].

mentation details or parameter settings.

A large-scale evaluation was performed on a database of piano music, and it was found that the differences in F-measure between the best algorithms are not significant, implying that the choice of algorithm could be based on other factors such as simplicity of programming, speed of execution and accuracy of correct onsets (all of which speak for SF, the spectral flux onset detection function). The results for the large-scale test were worse than previously published results for pitched percussion instruments, which were somewhat optimistic as they were based on simple data. The present results are also optimistic, since the parameter values were generated using feedback from the ground truth data, and all recordings came from the same instrument and recording conditions.

In future work, we will address the issue of automatic parameter estimation, with the aim of producing a fully automatic onset detection algorithm. (The MIREX 2005 audio onset detection competition was won by a neural network which was trained to detect onsets from spectral data [4].) We also intend to compare these results with another recording of the same data on a different piano with different recording conditions. Finally, an analysis of errors will be performed to determine the extent to which the methods fail at different points, so that by combining the methods (e.g. by voting [13]) a more robust onset detection algorithm could be developed.

5. ACKNOWLEDGEMENTS

Many thanks to Juan Bello of Queen Mary University of London for providing the data for the first set of experiments. This work was supported by the Vienna Science and Technology Fund, project CI010 *Interfaces to Music*, and the EU project S2S². OFAI acknowledges the support of the ministries BMBWK and BMVIT.

6. REFERENCES

- [1] J. Vos and R. Rasch, "The perceptual onset of musical tones," *Perception and Psychophysics*, vol. 29, no. 4, pp. 323–335, 1981.
- [2] J. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in musical signals," *IEEE Trans. Speech and Audio Proc.*, vol. 13, no. 5, pp. 1035–1047, 2005.
- [3] N. Collins, "A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated
- [4] J. Downie, "2005 MIREX contest results – audio onset detection," [Online] <http://www.music-ir.org/evaluation/mirex-results/audio-onset>, 2005.
- [5] J. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Sig. Proc. Letters*, vol. 11, no. 6, pp. 553–556, 2004.
- [6] P. Masri, "Computer modelling of sound for transformation and synthesis of musical signal," Ph.D. dissertation, University of Bristol, UK, 1996.
- [7] C. Duxbury, M. Sandler, and M. Davies, "A hybrid approach to musical note onset detection," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 33–38.
- [8] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, Phoenix, Arizona, 1999.
- [9] C. Duxbury, J. Bello, M. Davies, and M. Sandler, "A combined phase and amplitude based approach to onset detection for audio segmentation," in *Proc. 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS-03)*, 2003, pp. 275–280.
- [10] J. Downie, K. West, A. Ehmann, and E. Vincent, "The 2005 music information retrieval evaluation exchange (MIREX 2005): Preliminary overview," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'05)*, London, UK, 2005, pp. 320–323.
- [11] P. Leveau, L. Daudet, and G. Richard, "Methodology and tools for the evaluation of automatic onset detection algorithms in music," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, 2004, pp. 72–75.
- [12] S. Dixon, "Learning to detect onsets of acoustic piano tones," in *Proc. MOSART Workshop on Current Dir. in Computer Music Res.*, IUA-UPF, Barcelona, 2001, pp. 147–151.
- [13] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, and C. Uhle, "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. Speech and Audio Proc.*, vol. 14, no. 5, 2006, to appear.

detection functions," in *118th Conv. Audio Eng. Soc.*, Barcelona, Spain, 2005.

- [1] J. Downie, "2005 MIREX contest results – audio onset detection," [Online] <http://www.music-ir.org/evaluation/mirex-results/audio-onset>, 2005.
- [2] J. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Sig. Proc. Letters*, vol. 11, no. 6, pp. 553–556, 2004.
- [3] P. Masri, "Computer modelling of sound for transformation and synthesis of musical signal," Ph.D. dissertation, University of Bristol, UK, 1996.
- [4] C. Duxbury, M. Sandler, and M. Davies, "A hybrid approach to musical note onset detection," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 33–38.
- [5] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, Phoenix, Arizona, 1999.
- [6] C. Duxbury, J. Bello, M. Davies, and M. Sandler, "A combined phase and amplitude based approach to onset detection for audio segmentation," in *Proc. 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS-03)*, 2003, pp. 275–280.
- [7] J. Downie, K. West, A. Ehmann, and E. Vincent, "The 2005 music information retrieval evaluation exchange (MIREX 2005): Preliminary overview," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'05)*, London, UK, 2005, pp. 320–323.
- [8] P. Leveau, L. Daudet, and G. Richard, "Methodology and tools for the evaluation of automatic onset detection algorithms in music," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, 2004, pp. 72–75.
- [9] S. Dixon, "Learning to detect onsets of acoustic piano tones," in *Proc. MOSART Workshop on Current Dir. in Computer Music Res.*, IUA-UPF, Barcelona, 2001, pp. 147–151.
- [10] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, and C. Uhle, "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. Speech and Audio Proc.*, vol. 14, no. 5, 2006, to appear.

A NEW ANALYSIS METHOD FOR SINUSOIDS+NOISE SPECTRAL MODELS

Guillaume Meurisse, Pierre Hanna, Sylvain Marchand

SCRIME – LaBRI, University of Bordeaux 1
351 cours de la Libération, F-33405 Talence cedex, France
firstname.name@labri.fr

ABSTRACT

Existing deterministic+stochastic spectral models assume that the sounds are with low noise levels. The stochastic part of the sound is generally estimated by subtraction of the deterministic part: It is assumed to be the residual. Inevitable errors in the estimation of the parameters of the deterministic part result in errors – often worse – in the estimation of the stochastic part. We propose a new method that avoids these errors. Our method analyzes the stochastic part without any prior knowledge of the deterministic part. It relies on the study of the distribution of the amplitude values in successive short-time spectra. Computations of the statistical moments or the maximum likelihood lead to an estimation of the noise power density. Experimentations on synthetic or natural sounds show that this method is promising.

1. INTRODUCTION

Many representations of musical sounds are based on spectral models and consider audio signals as sums of sinusoids whose amplitudes and frequencies evolve slowly with time [1]. Sinusoids+noise models [2] decompose natural sounds into two independent parts: the deterministic part and the stochastic part. The deterministic part is a sum of sinusoids evolving slowly, whereas the stochastic part corresponds to the noisy part of the original sound. This decomposition is usually required for performing several high-quality transformations such as time stretching or pitch shifting, because it allows two different treatments for the two parts. These hybrid models considerably improve the quality of the synthesized sounds.

Spectral models usually consider the stochastic part of the signal as residual or artifacts due to the analysis errors. Most of these techniques try to eliminate this stochastic part. In this paper, we are interested in noisy sounds: The stochastic part is considered as very important from a perceptual point of view. This assumption imposes new techniques and new approaches. Our method analyzes the stochastic part without any prior knowledge of the deterministic part.

After reviewing the representations of noise in existing spectral models in Section 2 and their limitations in Section 3, we present the theory about distribution functions in Section 4. The method proposed is then detailed in Section 5. Finally, the results of experimentations are given in Section 6.

2. NOISE IN SPECTRAL MODELING

Existing hybrid spectral models are specially dedicated to natural sounds with low noise levels. The stochastic part is composed of all the signal components that have not been considered as sinusoids whose amplitudes and frequencies evolve slowly with time.

It is assumed to be entirely defined by the time variations of the short-time spectral envelopes. Therefore, usual methods for the estimation of the noisy part are dependent on the analysis of the deterministic part. They require a high-precision analysis (in frequency, amplitude, and phase) of sinusoidal peaks. Detected sinusoids are then subtracted from the original sound in order to analyze the stochastic part. Limitations of these approaches appear if the frequencies and the amplitudes of the sinusoids are not precisely estimated: The errors of these estimations are added to the residual, and this part is thus badly estimated.

Recent works have shown the limitations of sinusoidal analysis methods [3]. The presence of high-level noise considerably degrades the quality of the results of the analysis methods. Furthermore, theory indicates that the precision of the frequency estimation is limited according to the Cramér-Rao lower bound [4, 5] which gives the limit of the variance on an estimator computing data that are corrupted by noise. As several real-world sounds (musical instruments, natural sounds, etc.) contain high noise levels, the analysis step cannot be precise enough. Errors cannot be avoided. Moreover, these errors result in an imprecise estimation of the stochastic part of the signal. For example, an error for the estimation of the frequency, amplitude, or phase of a sinusoid may imply the presence of this sinusoid in the residual. Furthermore, even if the sinusoid is correctly analyzed, residual analysis methods relying on a spectral subtraction [2] define the residual magnitude spectra as composed of several *holes*, at the frequency of the subtracted sinusoids. All these reasons explain why we think that analyzing the stochastic part of sounds after having estimated the deterministic part is not the most accurate technique. In our application, the stochastic part is the most important part of the analyzed sound. This part has not to be considered as a residual. We think that a new technique considering first the analysis of this stochastic part may certainly give more accurate results.

3. ANALYSIS OF THE STOCHASTIC PART

Several approaches for the extraction of the stochastic component have been proposed. These techniques rely mainly on the classification of spectral components (or peaks) into sinusoidal components or stochastic components induced by noise [6]. This decision is binary which implies that a component is always associated to a sinusoid or to noise. But a component cannot be assumed as a mix of sinusoid and noise. Moreover, the decision is made according to the values of audio descriptors (correlation with the window spectrum, duration, energy location, etc.) computed in the current analysis frame [6].

Such methods have limitations if the analyzed sound is with high noise levels: The Cramér-Rao bound theoretically indicates that errors cannot be avoided. Moreover we think that considering

only one short-time amplitude spectrum cannot be sufficient for a precise estimation of the level of the noise.

Another approach consists in considering a long-time analysis of the amplitude spectrum. Several short-time spectra are computed from several consecutive frames. The estimation thus relies on the study of the variations of the short-time amplitude spectra. The observation of successive short-time amplitude spectra shows significant differences between noise and sinusoidal spectral contributions. Amplitude spectra appear to be nearly constant in the case of sinusoidal sounds provided that frequencies and amplitudes do not highly vary over time (tremolo, vibrato). At the opposite, amplitude spectra of noisy sounds vary very rapidly with time.

Empirical methods based on these realizations have already been proposed [7] with some success. The first possibility is to consider for each Discrete Fourier Transform (DFT) bin the minimum of the amplitude spectra. In the case of noisy bins, this minimum may take values near zero whereas in the case of sinusoidal bins, this minimum approximates the amplitude of the sinusoid (slightly lower if noise exists). Thus, the noise level for each bin cannot be estimated.

Another similar idea is to consider the maximum of the amplitude spectra. Here again, this maximum approximates the amplitude of the sinusoid (slightly higher if noise exists). But if the energy of the analyzed bin is due to the presence of noise, the maximum of the amplitude spectra may have a very high value. Indeed, whatever the noise level is, there is a non-null probability that the amplitude of this bin is very high.

The last empirical method is to consider the average of the amplitude spectra. This method leads to errors in the case of noisy bins. We detail the explanations in the next section and we show how this method can be improved.

4. DISTRIBUTION OF THE AMPLITUDE SPECTRUM

The method introduced in this paper relies on a study of variations in the magnitude spectrum along the time axis. High variations seem to indicate the presence of noise whereas stationarity seems to characterize sinusoidal components. We propose here to revert to statistical considerations. The way these variations occur leads to a new analysis method for the stochastic part. We present in this section the theoretical distribution of the amplitude spectrum of noises.

4.1. Spectral Properties of Noises

Thermal noises can be described in terms of a Fourier series [8]:

$$x(t) = \sum_{n=1}^N [A_n \cos(\omega_n t) + B_n \sin(\omega_n t)] \quad (1)$$

where N is the number of frequencies, n is an index, and ω_n are equally-spaced frequency components. The random variables A_n and B_n are normally distributed with zero mean and variance σ^2 . The magnitude spectrum computed by the Fourier transform is defined by random variables C_n :

$$C_n = \sqrt{A_n^2 + B_n^2} \quad (2)$$

The amplitudes C_n are distributed according to a Rayleigh distribution with most probable value σ .

4.2. Rayleigh Distribution

Let us consider a complex random variable whose real and imaginary parts, denoted X_r and X_i , follow a Gaussian probability distribution (PD) with a standard deviation σ . The probability of the magnitude $M = \sqrt{X_r^2 + X_i^2}$ is given by the Rayleigh PD defined by:

$$p(M) = \frac{M}{\sigma^2} e^{-\frac{M^2}{2\sigma^2}} \quad (3)$$

where σ is the most probable value.

As explained in Section 4.1, the amplitude spectrum of any colored noise is defined by this probability density function: For each DFT bin, each amplitude value is a random variable that is distributed according to this Rayleigh distribution. Here, it is important to note that the probability that the amplitude of a bin reaches a very high or a very low value is not null.

4.3. Rice Distribution

If we add a complex Gaussian noise X with standard deviation σ to a complex value $A_r + jA_i$ of module A , the probability distribution of the magnitude $M = \sqrt{(X_r + A_r)^2 + (X_i + A_i)^2}$ is a random variable distributed according to the Rice distribution [9]. This distribution is defined by:

$$p_{A,\sigma}(M) = \frac{M}{\sigma^2} e^{-\frac{(M^2+A^2)}{2\sigma^2}} I_0\left(\frac{AM}{\sigma^2}\right) \quad (4)$$

where I_0 is the modified Bessel function of the first kind of order 0.

Figure 1 shows the Rice PD for $\sigma = 1$ and various A . The A value represents a fixed amplitude value due to the presence of a sinusoid. That is the reason why if A is zero, the Rice distribution turns into the Rayleigh distribution. At the opposite, if A is much greater than σ , the amplitudes are distributed according to a normal (Gaussian) distribution with standard deviation σ and mean A .

Whatever the noise level is, each bin amplitude is theoretically distributed according to the Rice law. Properties of this distribution can be exploited to extract information about the noise part of any signal.

5. NOISE ESTIMATION FROM AMPLITUDE DISTRIBUTION

We consider long-time stationary sounds: Noise power density and the frequencies and amplitudes of sinusoidal components are assumed to be constant in several consecutive frames.

The complex value observed at a bin of the spectrum of such sounds can be decomposed into a constant-magnitude component and a complex Gaussian noise. For each bin, the complex spectrum can be characterized with two parameters A and σ that respectively represent the magnitude induced by one or more sinusoidal peaks or side lobes, and the standard deviation induced by noise at this bin. Each bin is the realization of the Rice PD with parameters A and σ associated to this bin. When observing the magnitude on the same bin at different frames, a Rice-distributed set is obtained. This magnitude distribution may be analyzed in turn to determine the parameters A and σ associated to the studied bin.

The standard deviation σ indicates the energy of noise, while A indicates the amplitude of a sinusoid. So the noise power density of a sound can be obtained by the estimation of the standard

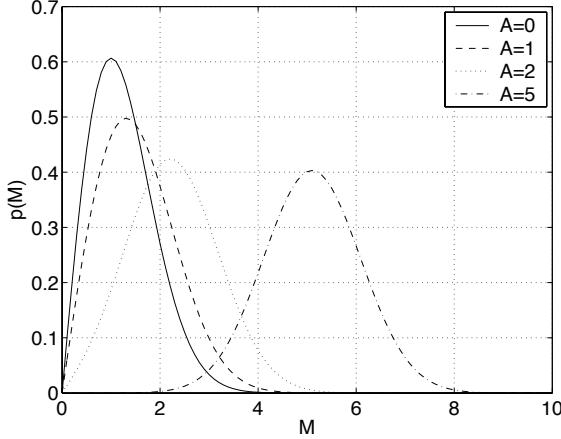


Figure 1: The Rice probability distribution function: If the parameter A of the function is zero, the Rice distribution turns into the Rayleigh distribution, whereas if A is much greater than σ , the Rice distribution turns into the normal (Gaussian) distribution.

deviation σ of noise at each bin. Since this distribution follows a Rice PD, two methods are proposed: It is possible to apply the relations on the moments of the Rice PD or to apply the likelihood method to estimate the standard deviation of the noise for each bin and thus obtain the noise power density.

Let us consider L non-overlapping consecutive frames. The discrete Fourier transform is computed for each frame. For each frequency bin the magnitude is computed for each frame in order to obtain a data set of L realizations per bin. The following methods are estimators for σ from a single set of realizations. The variance σ^2 for the whole bins leads to an estimation of the noise power density.

5.1. Moments Method

The first moment can be expressed in terms of the modified Bessel function:

$$E[M] = \left[\left(1 + \frac{A^2}{2\sigma^2}\right) I_{e0} \left(\frac{A^2}{4\sigma^2}\right) + \left(\frac{A^2}{2\sigma^2}\right) I_{e1} \left(\frac{A^2}{4\sigma^2}\right) \right] \times \sigma \sqrt{\frac{\pi}{2}} \quad (5)$$

where I_{e0} and I_{e1} are the scaled modified Bessel functions defined by:

$$I_{e0}(x) = e^{-x} I_0(x) \quad (6)$$

$$I_{e1}(x) = e^{-x} I_1(x) \quad (7)$$

I_0 and I_1 being the modified Bessel functions of the first kind of orders 0 and 1, respectively.

The second moment of the Rice PD can be expressed as polynomials in A and σ :

$$E[M^2] = A^2 + \sigma^2 \quad (8)$$

The standard deviation σ is evaluated by using any pair of moments and by finding the correct value that matches these moments.

The normalized mean μ is the mean computed on the data set normalized by the square root of the second moment. The normalized mean can be expressed only in terms of the signal-to-noise ratio (SNR). In this paper, the SNR is denoted γ and is defined by:

$$\gamma = A^2 / \sigma^2 \quad (9)$$

The normalized mean can be expressed as a function of γ [10] as:

$$\begin{aligned} \mu &= \frac{E[M]}{\sqrt{E[M^2]}} \\ &= \frac{\sqrt{\pi}}{2\sqrt{1+\gamma}} ((1+\gamma)I_{e0}(\gamma/2) + \gamma I_{e1}(\gamma/2)) \end{aligned} \quad (10)$$

In order to obtain an estimation of γ , it is possible to calculate the first and second moments and to find the value of γ that makes the calculated normalized mean match the theoretical normalized mean.

Expressing the first moment as a function of γ and σ leads to:

$$\hat{\sigma} = \sqrt{\frac{2}{\pi}} E[M] / \left[(1+\gamma) I_{e0} \left(\frac{\gamma}{2} \right) + \gamma I_{e1} \left(\frac{\gamma}{2} \right) \right] \quad (11)$$

5.2. Maximum Likelihood Method

For a probability distribution $p_{A,\sigma}(x)$ and a set M of L realizations following the probability density function denoted p , the likelihood function is given by:

$$\mathcal{L} = \prod_{i=1}^L p_{A,\sigma}(M_i) \quad (12)$$

where M_i is the i -th element of M .

Since M is a set of numerical values, L depends only of p . If p is the Rice PD, L can be considered as a function of A and σ . The log-likelihood function is given by:

$$\log(\mathcal{L}) = \sum_{i=1}^N \frac{M_i}{\sigma^2} I_0 \left(\frac{AM_i}{2\sigma^2} \right) - \frac{NA^2}{2\sigma^2} - \sum_{i=1}^N \frac{M_i^2}{2\sigma^2} \quad (13)$$

The amplitude and the standard deviation can be computed by maximizing the log-likelihood function [11]:

$$\{\hat{A}, \hat{\sigma}\}_{ML} = \underset{A, \sigma}{\operatorname{argmax}} \log(\mathcal{L}) \quad (14)$$

The maximization of the log-likelihood function on each data set gives us the parameters A and σ of the associated bin.

However, the maximization of 2-dimensional functions can be time consuming. So the maximization problem is here reduced to a 1-dimension problem by normalizing the data set M by the square root of the second moment (the second moment of the data set is an unbiased estimator of the Rice second moment). The log-likelihood function is then given by:

$$\begin{aligned} \log(\mathcal{L}) &= N \log(2(1+\gamma)) - N\gamma - (1+\gamma) \sum_{i=1}^N y_i \\ &\quad + \sum_{i=1}^N \log \left(I_0 \left(2y_i \sqrt{\gamma(1+\gamma)} \right) \right) \end{aligned} \quad (15)$$

where

$$y_i = \frac{M_i}{\sqrt{(E[M^2])}} \quad (16)$$

The maximization of this second log-likelihood function gives us the approximate location of the parameter γ :

$$\hat{\gamma} = \underset{\gamma}{\operatorname{argmax}} \log(\mathcal{L}) \quad (17)$$

A solution may be derived by solving:

$$\frac{\partial}{\partial \gamma} \log(\mathcal{L}) = 0 \quad (18)$$

5.3. Algorithms

Analysis of the theoretical distribution of the amplitude for each bin leads to the proposal of two algorithms for the estimation of the noise power density. The first method is based on the computation of two moments whereas the second one is based on the computation of the maximum likelihood.

The general algorithms for the estimation of the noise power density from L consecutive frames (of 1024 samples for example) are described here:

5.3.1. Moments Method

- L DFT with size $2N$ are computed;
- N distributions of L realizations are computed;
- For each distribution M_k ($k = 1, \dots, N$):
 - $\mu_k = \frac{E[M_k]}{\sqrt{E[M_k^2]}}$ is computed;
 - Equation 10 is applied to compute γ_k ;
 - Equation 11 is applied to compute σ_k .

5.3.2. Maximum Likelihood Method

- L DFT with size $2N$ are computed;
- N distributions of L realizations are computed;
- For each distribution M_k ($k = 1, \dots, N$):
 - Equation 17 is applied on M_k to compute the approximate value for γ_k ;
 - Finding the root of Equation 18 is applied to refine solution for γ_k ;
 - Equation 11 is applied to compute σ_k .

6. EXPERIMENTAL RESULTS

The two estimation methods described previously have been compared with various SNR and number of realizations. The conclusions of these experimentations indicate that the maximum likelihood and moment-based methods lead to the same precision.

Since the moments method has a better complexity, this method is preferred during our experimentations.

6.1. Number of Samples

Experimentations show that increasing the number of realizations reduces both error and bias. When using more than 1000 realizations, the spectral envelope obtained is smooth. However, using 1000 frames cannot be acceptable. Considering 1000 observations imposes a sound duration of at least 23 seconds when the sampling frequency is 44100 Hz and the DFT size is 1024. The signal is likely to change in a significant way during such a long period. Since the estimator is unbiased for 20 or more realizations, it can be a good choice to compute distributions on 20 frames and then recursively smooth values in time, according to:

$$\sigma(k, i) = \alpha\sigma(k, i - 1) + (1 - \alpha)\hat{\sigma}(k, i) \quad (19)$$

It should be also possible to use a temporal recursive computation of the moments instead of smoothing the estimated $\hat{\sigma}$. There are several advantages to do so. Complexity of the computation of the moments is simplified and there is no need to save all L magnitude spectra.

Figure 2 shows the estimated value for σ on a Rice-distributed set with $\gamma = 2$ and $\sigma = 1$ according to the number of realizations.

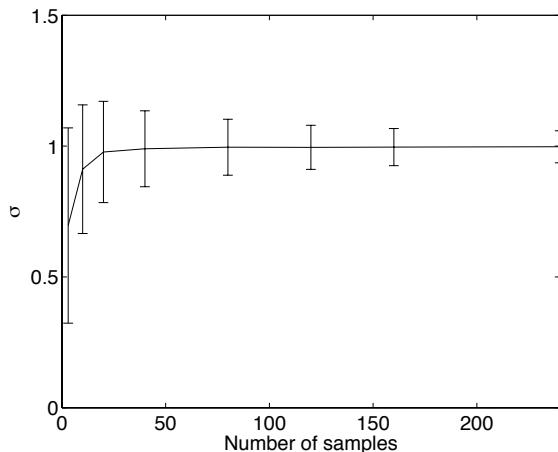


Figure 2: Estimated σ of a Rice-distributed set (theoretical values $\sigma = 1$ and $\gamma = 2$) with the moments method. Vertical bars indicate the standard deviation of the estimation.

6.2. Signal-to-Noise Ratio (SNR)

Experimentations show that the estimation is biased at low SNR. Figure 3 shows the estimated σ as a function of γ with different numbers of realizations. The distributions are computed with $\sigma = 1$, and A varies according to γ . For 20 samples, the estimator is biased for γ lower than 1. When $\gamma = 0$, the estimation is biased by 20%. For 1000 samples, the estimation is biased for γ lower than 0.5 and the bias is very low (5%). Therefore increasing the number of realizations reduces the bias. For low SNR, the Rice PD slightly changes. More observations are needed to fit closely the Rice PD. So, if the number of frame is not sufficient, errors in the estimation of γ are likely to occur. If the SNR is low, more observation are needed to avoid bias. If the SNR on a bin is *a priori* known to be $-\infty$ ($\gamma = 0$), the distribution follows the Rayleigh law and the computation of the first moment gives an unbiased estimator.

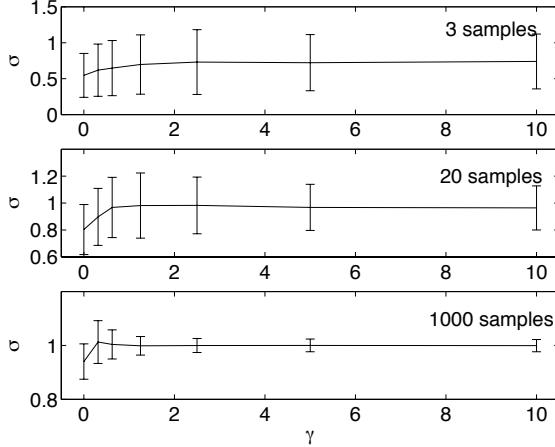


Figure 3: Estimated σ on Rice-distributed sets (theoretical values $\sigma = 1$, γ varying from 0 to 10). Vertical bars indicate the standard deviation of the estimation.

6.3. Effect of the Overlap

The moments and likelihood methods assume that the L realizations are statistically independent. Overlapping frames breaks this assumption and induces correlation in the data set. The following tests evaluate the effects of the induced correlation in the estimation of σ . Tests are made with different values of γ and different frame shifts.

The data sets are computed using a sound sampled at 44100 Hz, composed of a white noise of standard deviation $\sqrt{1024}/2$ and a sine wave of frequency 11025 Hz whose amplitude is $\sqrt{2\gamma}$. FFT are computed on 1024 samples. Data sets are computed on bin 256. In this way, the data set values obtained when the frames are not overlapping are realizations of a Rice distribution with standard deviation 1 and amplitude $f(\gamma)$. Several data sets are computed, with different frame shifts and γ (see Figure 4).

It has been observed that the first moment computed on data sets is constant when the frame shift varies. Since the first moment of the data set remains unbiased when overlapping frames, γ and σ are affected in the same way. So the effects of the overlap are only studied on the estimation of γ .

It has been observed that overlapping changed the magnitude distribution and biased noise estimation. Overlapping frames adds correlation in the computed magnitude data sets. So A may be overestimated while σ is underestimated. However, overlapping frames by 50% seems to have a small impact on the results. Bias and mean square error on the estimation of γ have been calculated using L non-overlapping frames on the one hand, and $2L - 1$ overlapping frames (50%) on the same duration on the other hand. It appears that overlapping frames reduces bias and mean square error for the estimation of γ . Computing σ on overlapping frames improves precision in time. It is even recommended to use a 50% overlap since it reduces both the bias and mean square error (see Figure 1).

It appears from our tests that using 21 overlapped frames gives the best results. Using less frames strongly degrades performances whereas increasing the number of frames improves slightly the results. Due to the under-estimation of the method at low SNR, it may be interesting to increase the number of frames if a bias at

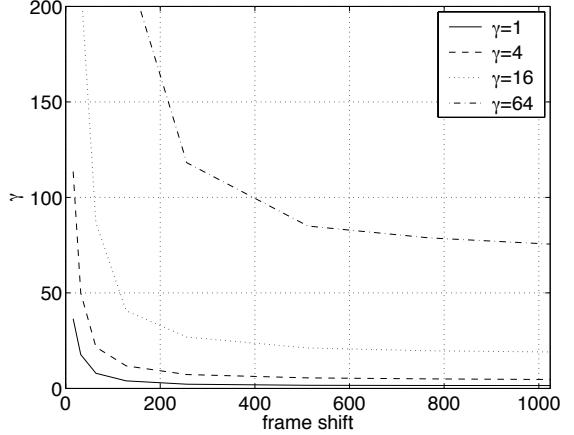


Figure 4: Estimated γ for several Rice-distributed DFT bins with various SNR according to the overlap rate. Distributions are computed with 20 frames. For a frame shift of 1024 samples, there is no overlap. Overlap increases when the frame shift decreases.

SNR(γ)	1	4	16	64
MSE with overlap	0.865	3.99	38.1	597
MSE without overlap	1.30	5.76	56.5	841

Table 1: Compared MSE for the estimation of σ with 20 non overlapping frames or 39 overlapping frames, for the same duration and for various γ .

low SNR is not acceptable.

6.4. Sound Tests

Sounds have been analyzed using the moments method. Data sets are computed on 21 overlapping frames of 1024 samples. Estimated σ values are smoothed in time using Equation 19 and $\alpha = 0.9$.

6.4.1. Synthetic Sound

Figure 5 shows the spectrogram of a synthetic sound and its stochastic component. This sound is composed of a pink noise and several sinusoids with various magnitudes. Due to the under-estimation of σ at low SNR, horizontal lines appear on the spectrogram. These lines are located on the frequency bins inhabited by the sines. However this error is hardly audible.

6.4.2. Natural Sound

The moments method has been tested on a sound composed of a saxophone sound and wind noise. Due to the length of the analysis frame, variations in the color of the noise are stretched in time while the attack and the release from the saxophone disturb the magnitude distribution. When the sound is nearly stationary during the analysis frame, sinusoids are correctly removed. Due to the under-estimation at low SNR and the small amplitude modulation of the harmonics of the saxophone sound, some estimation errors appear for the frequency bins inhabited by the sinusoids. This error is not disturbing, when the amplitude modulations are limited.

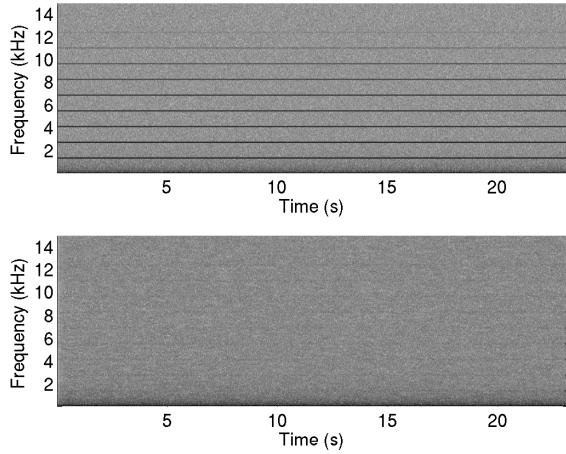


Figure 5: Spectrogram of a synthetic sound (23 s) composed of 9 stationary sinusoids (fundamental 1378 Hz) with a colored noise (top) and its analyzed stochastic component (bottom).

7. CONCLUSION

In this paper, we propose a new technique that estimates the stochastic part of the signal without having previously estimated the deterministic part. This method relies on a long-term analysis of the variation of the amplitude spectrum. It avoids errors due to the estimation of the sinusoidal parameters for the noisy bins. The limitations of this method is due to the assumption of stationarity for the analyzed signal. When sounds are nearly stationary, the method shows accurate results. Where classical methods require a short-term stationarity, our method requires that the sound is stationary over several frames. Due to the stochastic nature of noise, a single amplitude spectrum does not contain enough information to retrieve statistical properties of noise. That seems to be in agreement with perception. More time is needed to identify spectral content from noise than sinusoidal sounds. In the same way noise was ignored in early sinusoidal models, in this first approach we have neglected the variation of the sinusoids. Indeed, we are also interested in noise where sines could even be absent.

Applications of the methods proposed in this paper are numerous and concern essentially the improvement of the analysis method for the sinusoid+noise spectral models. Existing methods assume that each bin are either sinusoidal or stochastic, whereas the technique we introduce here estimates the proportion of noise – and thus the proportion of sinusoid – for each bin. In the future, improvements induced by this method on spectral model will be studied. Sound examples are available online¹.

8. REFERENCES

- [1] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [2] X. Serra and J. O. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic

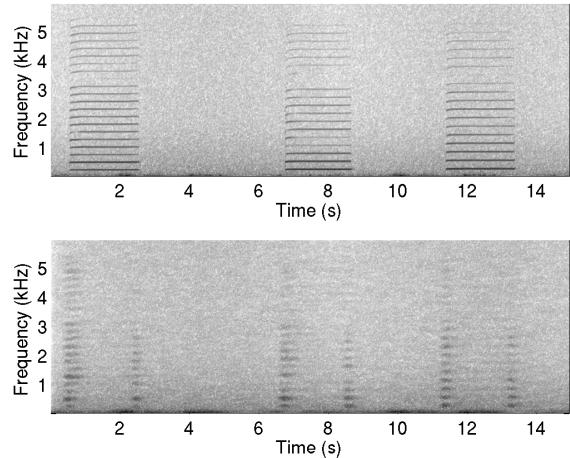


Figure 6: Spectrograms of a natural sound (15 s) composed of 3 notes (A#4, C#4, and D4) of saxophone with a background wind noise (top) and its analyzed stochastic component (bottom).

- plus Stochastic Decomposition," *Computer Music J.*, vol. 14, no. 4, pp. 12–24, 1990.
- [3] F. Keiler and S. Marchand, "Survey on extraction of sinusoids in stationary sounds," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, Sept. 2002, pp. 51–58. [Online]. Available: <http://citeseer.csail.mit.edu/keiler02survey.html>
- [4] D. C. Rife and R. R. Boorstyn, "Single-tone parameter estimation from discrete-time observations," *IEEE Trans. Information Theory*, vol. IT-20, pp. 591–598, 1974.
- [5] S. M. Kay, *Fundamentals of Statistical Signal Processing – Estimation Theory*, ser. Signal Processing Series. Prentice Hall, 1993.
- [6] A. Röbel, M. Zivanovic, and X. Rodet, "Signal Decomposition by Means of Classification of Spectral Peaks," in *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, Nov. 2004, pp. 446–449. [Online]. Available: <http://mediatheque.ircam.fr/articles/textes/Roebel04a/>
- [7] M. Okazaki, T. Kunimoto, and T. Kobayashi, "Multi-stage spectral subtraction for enhancement of audio signal," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)*, Montreal, Canada, vol. II, 2004, pp. 805–808.
- [8] W. M. Hartmann, *Signals, Sound, and Sensation*. Woodbury, NY: Modern Acoustics and Signal Processing, AIP Press, 1997.
- [9] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd ed. McGraw-Hill, 1984.
- [10] K. K. Talukdar and W. D. Lawing, "Estimation of the Parameters of the Rice Distribution," *J. Audio Eng. Soc.*, vol. 89, pp. 1193–1197, 1991.
- [11] J. Sijbers, A. den Dekker, D. V. Dyck, and E. Raman, "Estimation of Signal and Noise from Rician Distributed Data," in *Proc. Int. Conf. Sig. Proc. and Communications*, Feb. 1998, pp. 140–142. [Online]. Available: <http://citeseer.ist.psu.edu/article/sijbers98estimation.html>

¹<http://www.labri.fr/perso/hanna/Expe/sounds.html>

ADAPTIVE NOISE LEVEL ESTIMATION

Chunghsin Yeh, Axel Röbel

Analysis/Synthesis team
IRCAM/CNRS-STMS, Paris, France
{cyeh|roebel}@ircam.fr

ABSTRACT

We describe a novel algorithm for the estimation of the colored noise level in audio signals with mixed noise and sinusoidal components. The noise envelope model is based on the assumptions that the envelope varies slowly with frequency and that the magnitudes of the noise peaks obey a Rayleigh distribution. Our method is an extension of a recently proposed approach of spectral peak classification of sinusoids and noise, which takes into account a noise envelope model to improve the detection of sinusoidal peaks. By means of iterative evaluation and adaptation of the noise envelope model, the classification of noise and sinusoidal peaks is iteratively refined until the detected noise peaks are coherently explained by the noise envelope model. Testing examples of estimating white noise and colored noise are demonstrated.

1. INTRODUCTION

Many applications for audio signals such as speech and music require an estimation of the noise level that should be local in time and in frequency such that non-stationary and colored noise can be dealt with. Noise level estimation, or noise power spectral density estimation, is usually done by explicit detection of time segments that contain only noise, or explicit estimation of harmonically related spectral components (for nearly-harmonic signals). Since some of the noise is related to the signal, relying only on pure noise segments will not allow to properly detect the noise introduced with the source signal. Therefore, it has been proposed to include several consecutive analysis frames assuming that the time segment contains low energy portion and the noise present within the segment is more stationary than the signal [1] [2].

The other classical approach is to remove the sinusoids and estimate the underlying noise components afterwards [3]. This involves sinusoidal component identification, either in single frame [4] [5] or by tracking sinusoidal components across frames [6] [7]. We decide to follow this approach because the assumptions compared to the methods reviewed in [1] are released. We propose to classify the spectral peaks in each short-time spectrum independently because the costly tracking of sinusoidal components could then be avoided. Moreover, the spectral peak classification method proposed in [4] [5] allows to control the classification results such that a bias towards sinusoids or noise can be easily altered. After subtracting the sinusoidal peaks from the observed spectrum, we expect that there are few sinusoidal peaks left in the residual spectrum. Then, a bandwise noise distribution fit is performed using a statistical measure. The outliers of the observed noise peaks are excluded through an iterative process of distribution fit and noise level estimation. Upon the termination of the iterative approximation, the estimated noise level is thus defined.

This paper is organized as follows. First the problem of noise level estimation is defined. In section 3, we explain how the distribution of the magnitudes of narrow band noise can be modeled. An iterative algorithm to approximate the noise level is then presented in section 4. Lastly, different types of noise are used to demonstrate the effectiveness of the proposed method.

2. PROBLEM DEFINITION

A signal is called "white noise" if the knowledge of the past samples does not tell anything about the subsequent samples to come. The power density spectrum of white noise is constant. By means of filtering a white noise signal, correlations between the samples are introduced. Since in most cases the power density spectrum will no longer be constant, filtered white noise signals are generally called "colored noise". We define the "colored noise level" as the expected magnitude level of the observed noise peaks. A noise peak is defined as a peak that can not be explained as a stationary or weakly modulated sinusoid of the signal. The noise level could be represented as a smooth frequency dependent curve approximating the noise spectrum, as shown in Figure 1. The noise level should include most of the noise peaks and also follows smoothly the variation of the observed spectral magnitudes.

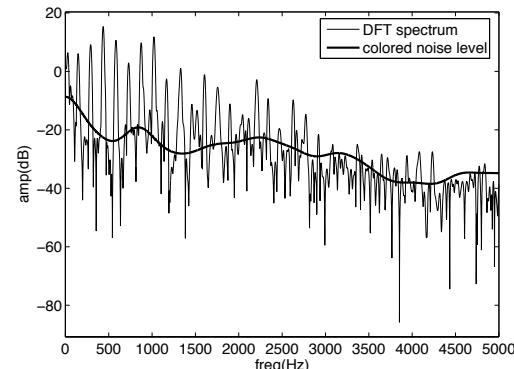


Figure 1: Colored noise level.

3. MODELING NARROW BAND NOISE USING RAYLEIGH DISTRIBUTION

Under the assumption that noise is nearly white within a considered frequency band, we choose Rayleigh distribution to fit the

distribution of the observed narrow band noise¹. The Rayleigh distribution was originally derived by Lord Rayleigh in connection with a problem in the field of acoustics. A Rayleigh random variable X has probability density function [8]:

$$p(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)} \quad (1)$$

with $0 \leq x < \infty, \sigma > 0$, cumulative distribution function

$$F(x) = 1 - e^{-x^2/(2\sigma^2)} \quad (2)$$

and the p th percentile

$$x_p = F^{-1}(p) = \sigma \sqrt{-2 \log(1-p)}, \quad 0 < p < 1 \quad (3)$$

In Figure 2, the probability density function is plotted for different values of σ ($\sigma = 0.5, 1, 1.5, 2, 2.5$ and 3). σ corresponds to the **mode** of the Rayleigh distribution, which is the most frequently observed value in X . Thus, $p(\sigma)$ corresponds to the maximum of the probability distribution. Notice that σ is not the usual notation for the variance of a distribution. The variance of Rayleigh distributed random variable is

$$\text{Var}(X) = \frac{4 - \pi}{2} \sigma^2 \quad (4)$$

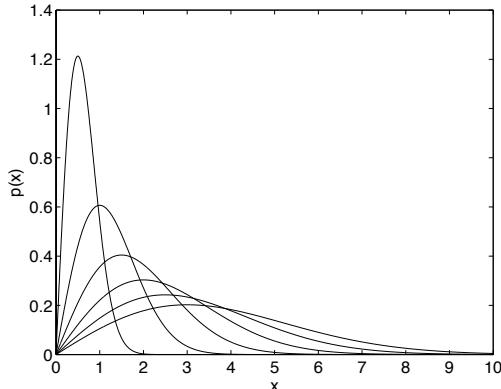


Figure 2: Rayleigh distribution with different σ .

Consider the Rayleigh random variable X as the observed magnitudes of spectral peaks in a narrow band, then σ represents the most frequent magnitude values of noise peaks. The mode of the Rayleigh distribution can then be used to derive the probability of an observed peak to belong to the background noise process. Comparing the magnitude of the spectral peak to σ we may conclude that peaks having amplitude below σ are most likely noise while for the spectral peaks having magnitudes larger than σ , the larger magnitudes they have, the less probable they are to be noise (and thus they are more likely related to the deterministic part of the signal).

¹In fact, Rice has shown in the Bell Laboratories Journal in 1944 and 1945 that Rayleigh distribution is suitable for modeling the probability distribution of a narrow band noise.

4. NOISE LEVEL ESTIMATION

For a given narrow band, e.g. each frequency bin k , the noise distribution can be modeled by means of Rayleigh with mode $\sigma(k)$. Once $\sigma(k)$ has been estimated for all k , the curve passing through these σ -value magnitudes defines a reference noise level \mathcal{L}_σ . Using eq. (3) it is now possible to adjust the noise threshold to a desired percentage of misclassified noise peaks. The related *noise envelope* \mathcal{L}_n can be estimated by simply multiplying the *estimated Rayleigh mode* \mathcal{L}_σ with $\sqrt{-2 \log(1-p)}$. Therefore, the problem comes to estimating the frequency dependent $\sigma(k)$.

It is known that the mean of a Rayleigh random variable X is

$$E[X] = \sigma \sqrt{\pi/2} \quad (5)$$

from which we have

$$\sigma = \frac{E[X]}{\sqrt{\pi/2}} \quad (6)$$

That is, the frequency dependent $\sigma(k)$ can be calculated if the mean noise magnitude $E[X]$, which is also frequency dependent, can be estimated.

However, estimation of the expected noise magnitude corresponding to each frequency bin requires sufficient observations for statistical evaluation. Most of the existing approaches [1] rely on observations from neighboring frames. Our approach relies on the assumption that the noise spectral envelope is changing only weakly with the bin index k such that we may use the observed spectral peaks in the predefined subbands² to estimate the (frequency dependent) *mean noise level* \mathcal{L}_m by means of a cepstrally smoothed curve over the noise peaks. We describe the noise level estimation procedures in the following.

4.1. Spectral subtraction of sinusoids

In [4], four spectral peak descriptors have been proposed to classify spectral peaks. The descriptors are designed to properly deal with non-stationary sinusoids. This method serves to classify sinusoidal and non-sinusoidal peaks in our algorithm. The sinusoidal peaks are then subtracted from the observed spectrum to obtain the residual spectrum that is assumed to contain mostly noise peaks.

To estimate the spectral parameters of each sinusoidal peak, the reassignment method proposed by F. Auger and P. Flandrin [9] is used to estimate the frequency slope [10]. Given a STFT (Short Time Fourier Transform), the frequency slope can be estimated by means of

$$\omega'(t, \omega) = \frac{\partial \hat{\omega}(t, \omega) / \partial t}{\partial \hat{t}(t, \omega) / \partial t}, \quad (7)$$

where $\hat{t}(t, \omega)$ and $\hat{\omega}(t, \omega)$ are the reassignment operators. Once the frequency and the frequency slope of each sinusoidal peak are estimated, the peak is subtracted from the observed spectrum. The optimal phase is estimated by means of the least square error criterion, i.e., the error between the original signal and the processed signal is minimized. However, if the estimated slope is larger than the maximal slope around the observed peak, it will not be considered as a consistent estimate and therefore be disregarded.

The main function of subtracting sinusoidal peaks is to provide sufficient residual peaks for a proper statistical measure of the magnitude distribution even if the frequency resolution is limited and sinusoidal peaks are very dense.

²We divide equally the subbands with the bandwidth 312.5 Hz.

4.2. Iterative approximation of the noise level

After obtaining the residual spectrum, denoted as S_R , the spectral peaks are again classified and then the iterative approximation of the noise level is carried out till the selected statistical measure of the noise distribution in all subbands fit that of Rayleigh distribution.

The reasons to use a statistical measure are: (i) the amount of the observed samples is usually not large enough to draw the underlying distribution, (ii) statistical measures are representative of a distribution and are more efficient for distribution fit.

We use skewness as the statistical measure for distribution fit. Skewness is a measure of the degree of asymmetry of a distribution [11]. If the right tail (tail at the large end of the distribution) extends more than the left tail does, the function is said to have positive skewness. If the reverse is true, it has negative skewness. If the two tails extend symmetrically, it has zero skewness, e.g. Gaussian distribution. The skewness of a distribution is defined as

$$Skw(X) = \frac{\mu_3}{\mu_2^{3/2}} \quad (8)$$

where μ_i is the i th central moment. And the skewness of Rayleigh distribution is independent of $\sigma(k)$:

$$Skw_{rayl} = \frac{2(\pi - 3)\sqrt{\pi}}{\sqrt{(4 - \pi)^3}} \approx 0.6311 \quad (9)$$

If the distribution of the noise magnitudes in a subband is assumed Rayleigh then we may test for misclassified sinusoids by means of the condition $Skw(X_n^b) > Skw_{rayl}$, where X_n^b are the noise magnitudes in the b th subband. Whenever this condition is true we assume that there are misclassified sinusoids that can be detected by observing their amplitude levels relative to the current estimate of $\sigma(k)$.

Note that the distribution of noise magnitudes in each subband will not be Rayleigh if $\sigma(k)$ in the subband is not constant. To improve the consistency of the skewness test we therefore rescale all noise magnitudes by means of normalizing with the current estimated Rayleigh mode \mathcal{L}_σ .

Assuming that for each subband in S_R there are a greater proportion of noise peaks and only a few sinusoidal peaks with dominant magnitudes remain. Then the noise level approximation can be realized by iterating the following processes:

- I. Calculate the cepstrum of the noise spectrum (constructed from interpolating the magnitudes of noise peaks). The cepstrum is the inverse Fourier transform of the log-magnitude spectrum and the d th cepstral coefficient is formulated as

$$c_d = \frac{1}{2} \int_{-\pi}^{\pi} \log |X_n(\omega)| e^{i\omega d} d\omega \quad (10)$$

By truncating the cepstrum and using the first D cepstral coefficients, we reconstruct a smooth curve representing the mean noise level \mathcal{L}_m as a sum of the slowly-varying components.

$$\mathcal{L}_m(\omega) = \exp(c_0 + 2 \sum_{d=1}^{D-1} c_d \cos(\omega d)) \quad (11)$$

The cepstral order D is determined in a way similar to that of [12]: $D = F_s / \max(\Delta f_{max}, BW) \cdot C$, where F_s is half the sampling frequency, Δf_{max} is the maximum frequency gap among all the noise peaks, BW is the subband bandwidth, and C is a parameter to set.

II. Then we have $\mathcal{L}_\sigma = \mathcal{L}_m / (\sqrt{\pi/2})$, the estimated Rayleigh mode across the analysis frequency range.

- III. For each subband, check if the distribution fit is achieved. If the distribution fit is not achieved in the subband under investigation, that is, $Skw(X_n^b / \mathcal{L}_\sigma^b) > Skw_{rayl}$ where \mathcal{L}_σ^b denotes the estimated Rayleigh mode in the b th subband, then the largest outlier is excluded (re-classifying the largest outlier in the subband as sinusoid).

When all the subbands meet the requirement of the skewness measure, the estimated Rayleigh mode \mathcal{L}_σ can be used to derive a probabilistic classification of all spectral peaks into noise and sinusoidal peaks. For this we suggest the p th percentile of Rayleigh distribution

$$\mathcal{L}_n = \mathcal{L}_\sigma \sqrt{-2 \log(1 - p)} \quad (12)$$

with a user selected value for p . Notice that if the underlying noise level varies very fast in such a way that the proposed model cannot capture the noise level evolution then the procedure may not converge or may not converge to a reasonable estimate.

5. TESTING EXAMPLES

To demonstrate the effectiveness of the proposed algorithm, we have tested two types of signals: white noise and a polyphonic signal with background noise. In both cases, the sampling frequency is 16 kHz and we set $C = 1$ for the cepstral order and $p = 0.8$ in eq. (12), that is, we allow 20% of the noise to be misclassified according to Rayleigh distribution.

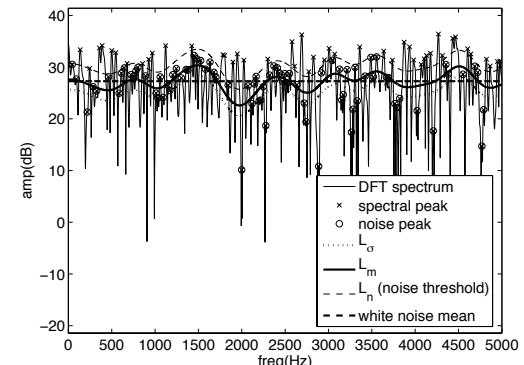


Figure 3: Estimated noise level for white noise (test 1).

In Figure 3, a white noise spectrum is shown with the estimated noise level \mathcal{L}_m . The estimated mean noise level \mathcal{L}_m does approximate the constant white noise mean. The estimated noise envelope \mathcal{L}_n is noted as “noise threshold” to notify that this is a user-adjustable level. The noise peaks are finally re-classified as the spectral peaks having magnitudes below this threshold.

To further demonstrate how the proposed algorithm works for polyphonic signals, we estimate the colored noise level of a polyphonic signal. Figure 4 shows the initial spectral peak classification result and Figure 5 shows the residual spectrum after subtracting the sinusoidal peaks. The dotted vertical lines represent the boundaries of the equally divided subbands. The estimated noise level is shown in Figure 6³. The proposed noise envelope model

³Additional peaks are shown to indicate possibly hidden sinusoidal peaks.

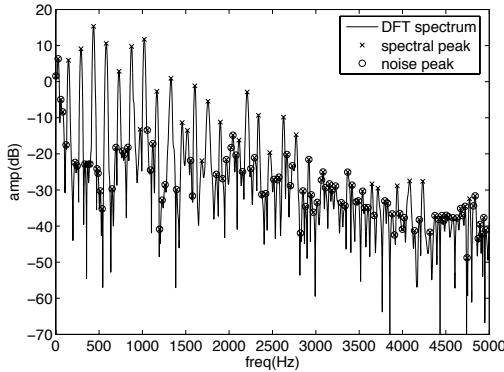


Figure 4: Initial spectral peak classification (test 2).

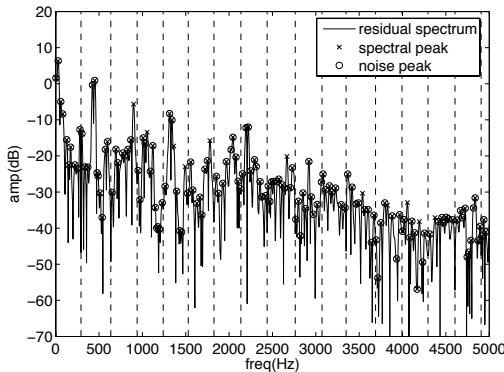


Figure 5: Residual Spectrum (test 2).

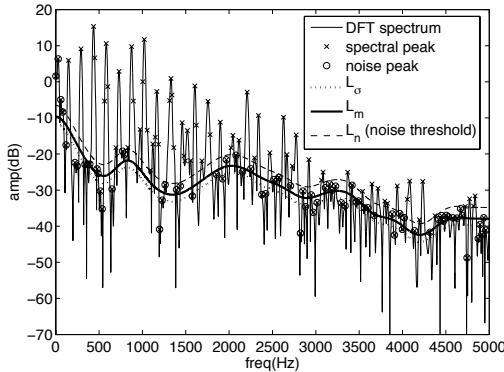


Figure 6: Estimated noise level for a polyphonic signal (test 2).

does follow well the variation of the observed spectrum. Moreover, it provides us the control over misclassified spectral peaks at the first stage.

6. CONCLUSIONS

We have presented an iterative algorithm for approximating the noise level local in time and in frequency. This algorithm is adaptive to the dynamics of the spectral variation. It neither includes additional information from the neighboring frames or pure noise segments, nor makes use of harmonic analysis. The proposed noise envelope model represents the instantaneous noise spectrum,

which can be used as a new feature for signal analysis.

Its ability to handle different types of signals has been demonstrated. However, there are several parameters to be studied: the number of subbands, the order (the number of cepstral coefficients) of the noise level curve, and the percentage of the noise in eq. (12) to be included according to Rayleigh distribution.

The proposed algorithm is useful for many signal analysis and synthesis applications, such as partial tracking, signal enhancement, etc. It has been implemented by the authors for estimating the number of quasi-harmonic sources in connection with the problem of multiple fundamental frequency estimation.

7. REFERENCES

- [1] C. Ris and S. Dupont, "Assessing local noise level estimation methods: application to noise robust ASR," *Speech Communication*, vol. 34, no. 2, pp. 141–158, 2001.
- [2] V. Stahl, A. Fischer, and R. Bippus, "Quantile based noise estimation for spectral subtraction and wiener filtering," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'00)*, Istanbul, Turkey, 2000, pp. 1875–1978.
- [3] M. Alonso, R. Badeau, B. David, and G. Richard, "Musical tempo estimation using noise subspace projection," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 2003, pp. 95–98.
- [4] A. Röbel, M. Zivanovic, and X. Rodet, "Signal decomposition by means of classification of spectral peaks," in *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, 2004, pp. 446–449.
- [5] G. Peeters and X. Rodet, "Sinusoidal characterization in terms of sinusoidal and non-sinusoidal components," in *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, 1998.
- [6] B. David, G. Richard, and R. Badeau, "An EDS modelling tool for tracking and modifying musical signals," in *Stockholm Music Acoustics Conf.*, Stockholm, Sweden, 2003, pp. 715–718.
- [7] M. Lagrange, S. Marchand, and J.-B. Rault, "Tracking partials for the sinusoidal modeling of polyphonic sounds," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'05)*, Philadelphia, USA, vol. 3, 2005, pp. 229–232.
- [8] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, 2nd ed. New York: John Wiley & Sons, Inc, 1994.
- [9] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *IEEE Trans. Sig. Proc.*, vol. 43, no. 5, pp. 1068–1089, 1995.
- [10] A. Röbel, "Estimating partial frequency and frequency slope using reassignment operators," in *Proc. Int. Comp. Music Conf. (ICMC'02)*, Gothenburg, Sweden, 2002, pp. 122–125.
- [11] A. Stuart and J. K. Ord, *Kendall's Advanced Theory of Statistics, Vol. 1: Distribution Theory*, 6th ed. New York: Oxford University Press, 1998.
- [12] 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, 2005, pp. 30–35.

CATEGORIES OF PERCEPTION FOR VIBRATO, FLANGE, AND STEREO CHORUS: MAPPING OUT THE MUSICALLY USEFUL RANGES OF MODULATION RATE AND DEPTH FOR DELAY-BASED EFFECTS

William L. Martens and Marui, Atsushi

Sound Recording Area
Schulich School of Music, McGill University
Montreal, Canada
wlm@music.mcgill.ca

ABSTRACT

Vibrato, Flange, and Stereo Chorus are perhaps the three most often used digital audio effects that are created by smoothly modulating the duration of a delay line at typically sub-audio rates. Common practice is to use a periodic or quasi-periodic modulation control signal with frequency roughly between 2 and 9 Hz, and both the rate and depth of delay modulation are typically adjusted according to the aesthetic criteria of a performer or by an audio production engineer. In order to establish norms for the musically useful range of modulation rate and depth for such delay-based effects, 25 listeners were asked to make categorical judgments regarding their perception of vibrato, flange, and stereo chorus effects. The results map out for these two modulation parameters three perceptual regions for these three related effects: the region in which modulation is too subtle for effective use, the parameter ranges that seem most musically useful, and the region in which it is too extreme for most musical applications. Of particular interest is the observed commonality between these perceptual regions for vibrato, flange, and stereo chorus effects.

1. INTRODUCTION

Among the most popular digital audio effects (DAFx) that are used in modern music production, there are three effects that are very frequently used, and which share a common underlying factor: They are created by smoothly modulating the duration of a delay line at typically sub-audio rates. The three DAFx are often identified as vibrato, flange, and stereo chorus [1][2][3, Chapter 3]. Although these three algorithms are well understood, there are no clear norms for the musically useful range of modulation rate and depth for delay lines underlying these effects, which must be based upon the perceptual attributes associated with the application of the effects. It should be clear that answering this question from the perspective of signal processing *per se* cannot be entirely satisfying. So the motivation for the study reported here was to map out for these two modulation parameters the distinct perceptual regions describing where they might be used most effectively. The method by which this was done was to ask a group of listeners to make categorical judgments regarding the perception of each effect as follows: the perceived modulation should be indicated to be either too subtle for effective use, quite suitable for musical use, or too extreme for most musical applications.

There is some precedent in the literature for this type of study, which should not be regarded as a study of just noticeable differences in frequency variation [4]; rather, this study is more closely

related to studies of fluctuation strength, which is one of the major psychoacoustic variables considered in sound-quality evaluation. The authors know of only one study, conducted by Wickelmaier and Ellermeier [5], in which the dependence of fluctuation strength on modulation rate and modulation depth was tested in a factorial design, as it was here. Of course, the current study did not investigate fluctuation strength directly, but only asked for categorical distinctions with regard to a variety of temporal fluctuations that are most definitely multidimensional in perceptual character, hence the use of the different terms vibrato, flange, and stereo chorus.

In order to regularize the signal processing used in stimulus preparation for this study, a particular structure was chosen as the basis for these three DAFx, which indeed may be described as linear, time-variant systems that are based upon similar algorithmic structures. But a few simple differences in structure result in the three DAFx having qualitatively different perceptual characteristics, which are well identified by their names. In the simplest of implementations, the three types of delay modulation effects can be created using only a single interpolated delay line as the core component. The only difference is in the presence and/or destination of the feed-forward signal path that may be added to the core component. (A fourth effect that has been popular in guitar effects processing, termed phase shifting, has been omitted from the current study primarily because it does not match the strict delay-modulation structure on which the other three are based [6].)

Vibrato, as implemented here, is a primarily a pitch-related effect that is based upon a simple delay modulation with no feed-forward signal. Flange is primarily a timbral effect that results from the zeros of a comb-filter sweeping through frequency due to the summation of the a feed-forward signal with the vibrato's delay modulation signal. Under some circumstances, the pitch variation can still be perceived here, but it is usually overshadowed by the timbral effect. The final option to be described here is when the feed-forward signal is sent to a different audio output channel than the vibrato's delay modulation signal. Again, the pitch variation virtually disappears, and what is left is perceived as a spatial effect that has been identified as a stereo chorus effect. In its most common implementation, stereo chorus effects are implemented with more than one delay line, but for the current investigation, only the simplest implementation has been considered in order to maximize similarity with the other two algorithms. In fact, feedback signals are also often included in the implementation of both flange and stereo chorus effects, and frequently play a role in the musical "sweetening" of reverberation algorithms [7].

Of course, by choosing only very simple implementations for these three effects, there is a chance that the results of the study will not be so readily generalized to other implementations. It is preferred to begin with these simple algorithms to establish a baseline for further work. It should also be noted that the stimuli may be characterized by the outputs of the effects processing, rather than in terms of the processing itself. This provides a basis for comparisons with other implementations that transcend the particular structures employed here. Other studies of these effects might include more complete characterization of their possibilities, such as that described in the review of different types of vibrato by Verfaillie et al. [8]. They explained that the acoustical effects of vibrato could be categorized into amplitude modulation, frequency modulation, and spectral envelope modulation. For the current investigation, pitch modulation alone was intended for the vibrato stimulus, in order to keep it as distinct as possible from the timbral effects of flange outputs that are associated with spectral envelope modulation, within the limitation of the simplistic model. Of course, vibrato effects that are created by simple delay modulation, as were those created for the current study, also exhibit a limited amount of modulation in spectral energy distribution. Nonetheless, for the guitar-sound inputs processed herein, which were relatively smooth spectrally, the slight variation in tone color (subtle brightening and darkening) was negligible, especially since the maximum depth of delay modulation never exceeded 1 ms (see below for details). The choice to limit the vibrato production to simple delay modulation (with all-pass interpolation for fractional-sample delay values) also served to keep the vibrato directly comparable to the other delay-modulation-based effects of interest here, most notably in terms of its simplistic signal processing implementation. It is also true that the output vibrato character that was introduced into the guitar performance that was used as the input to these effects was quite similar to that which can be produced by a human performer, with the exception that the modulation followed a perfect sinusoidal pattern.

The vibrato discussed in this paper is an instance of coherent vibrato [9] which means all the harmonic components in the output signal are modulated at the same rate and depth. (In the case of non-coherent vibrato, a particular frequency stands out. from the rest). In the current study, output signals of the vibrato process exhibit some dynamic frequency deviation relative to the input signal due to the same phenomenon as the Doppler shift (as if the input signal were “moving closer and farther” relative to the listener’s position). Frequency deviation of the output signal from the carrier frequency can be expressed as a function of modulation rate and modulation depth parameters. According to Dutilleux and Zölzer [3, Chapter 4], a *resampling factor* for sinusoidal modulation can be regarded as a pitch change ratio and equal to the ratio of the instantaneous output frequency divided by the carrier frequency. The pitch change ratio is a function of time, and it correspond to the frequency deviation, as follows:

$$\alpha(t) = 1 - 1/1000000 \cdot \text{DEPTH} \cdot \text{RATE} \cdot 2\pi \cdot \cos(\text{RATE} \cdot 2\pi t) \quad (1)$$

where DEPTH is the modulation depth in μs (thus $1/1000000$ in front of it to match the unit) and RATE is modulation rate in Hz (cycles/second). The minimum and the maximum values of the resampling factor are the ratios of the peak deviation of the instantaneous frequency from the carrier frequency, and the reach these extremes when the above cosine term reaches 1 and -1 , respectively.

$$1 \mp 1/1000000 \cdot \text{DEPTH} \cdot \text{RATE} \cdot 2\pi \quad (2)$$

Since this equation is a ratio of the instantaneous frequency divided by the carrier frequency, the peak deviation frequency can be obtained by multiplying the ratio times carrier frequency value.

2. METHOD

2.1. Stimulus Generation

A single short excerpt of an electric guitar performance was digitized at 44.1 kHz in 16-bit precision, and stored in a file for subsequent processing. The performance was a single note played on a electric guitar (the note G produced by hammering the string with the left-hand just below the 5th fret on the D string). The analogue signal from the guitar was submitted to mild compression and and overdrive effects to achieve a sound character similar to that used in popular music.

This signal was then submitted to each of the three delay-modulation-based effects processing algorithms, employing a factorial combination of delay modulation depths and rates to complete a two-dimensional palette of output character for each. Modulation rate was set to each of five frequencies: 2, 3, 4, 6, and 9 Hz. Modulation depth at peak was set to five different delay-line durations spaced logarithmically between 0.04 and 1.0 ms. The character of the output sound was naturally dependent on the shape of the modulation signal. Although a more musically pleasing result can be obtained through quasi-periodic modulation, a simple, gated sinusoidal modulation signal was employed in this study. The stimulus duration was 1.4 s, and the modulation signal was gated by a raised-cosine window of 200 ms rise and fall, leaving the modulation at its maximum for exactly 1 s.

For the flange and vibrato, the average delay value about which the delay modulated was 1 ms, which is of no consequence for pitch vibrato, but matters very much for the timbral modulation resulting for the flange stimuli. In the case of the flange effect, this means that the timbral variation was the most extreme possible (since at maximum modulation depth, the signals begin added exhibited very short delays). In the case of the stereo chorus effect, the non-modulated signal was delayed by .5 ms, which means that the delay modulated signal fluctuated between leading and lagging the non-modulated signal in time of arrival. This means that the spatial variation was like that associated with the variation in Interaural Time Delay (ITD) for a sound source taking a circular path around the head, since maximum modulation depths for ITD don’t often exceed .6 to .7 ms.

The stimuli were presented over either one or two JBL Control 8SR studio monitors in a large anechoic chamber, at a level producing 80 dB SPL at the listening position. For the vibrato and flange stimuli, subjects faced one of the two monitors. When listening to the stereo chorus stimuli, subjects oriented themselves to position the two monitors 30° to the left and right of the median plane.

2.2. Categorical Judgment Task

As a straightforward means for operationally defining the musical useful range of modulation rate and depth for delay-based effects, subjects were asked to make categorical judgments regarding their perception of vibrato, flange, and stereo chorus effects. The following three categories for response were explained prior to the experiment, the perceived modulation:

- would be too subtle for effective use,

- seems quite musically useful,
- would be too extreme for most musical applications.

While subjects were familiarized with the stimuli, it was explained that their responses should be limited to the three categories listed above. The three categorical judgment tasks for the three types of delay-modulation-based effects were completed in succession on a single day, but not in a randomized order. All subjects completed the task for the vibrato case first, followed by the flange case, and finally the stereo chorus case.

2.3. Subjects

Twenty-five subjects participated in the rating task. All 25 subjects were either undergraduate or master's level computer science students at the University of Aizu (Japan), with interest in audio effects processing but without strong musical background. Their age ranged between 18 and 24, and all reported that they enjoyed listening to popular music on a daily basis.

3. RESULTS AND ANALYSIS

In order to summarize the relative proportion of responses within each of the three response categories, logistic regression analysis (LRA) was employed to fit a curve to the observed categorical choice data summed across all 25 subjects. In commonly used linear regression models, continuous data are submitted for both independent variables and dependent variables. Although non-continuous variables can be used, evaluation of the linear regression results given such data have to be made with caution. LRA is an instance of nonlinear regression method suitable for analyzing a binary dependent variable against continuous independent variable [10, Chapter 13][11, Chapter 11]. An example of using LRA is fitting a probability curve on examination outcomes on test scores which the result of an exam is often binary (pass or fail) and predictor variable (examination score) is continuous. In LRA, a sigmoid (i.e., "S"-shaped) function called *logit* function shown in Equation 3 is used to fit to data.

$$\hat{Y} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots)}} \quad (3)$$

where X_i denotes a continuous independent variable, and \hat{Y} denotes a predicted dependent variable taking values between zero and one.

In the current investigation of delay modulation based effects, parameter values for modulation depth and modulation rate were submitted to the analysis as X and binary value of whether each rating category was chosen as Y . Since there were three response options, the analysis was performed first for the response of "too weak effect" versus other responses. Then the analysis was performed for the response of "too much effect" versus other responses. The curve for the proportion of "musically useful" responses was derived by subtracting the other two curves from unity, since the three response portions were constrained to sum to unity.

Results are shown in Figure 1 for the vibrato case, and results for the other two effects, flange and chorus, looked remarkably similar (and are not presented in this paper in order to reduce its length). The figure presents a plot showing the logit functions fit to the response-proportion distributions obtained from the three-category response task for one of tested stimulus sets. The case shows results of analyzing responses to vibrato stimuli presented

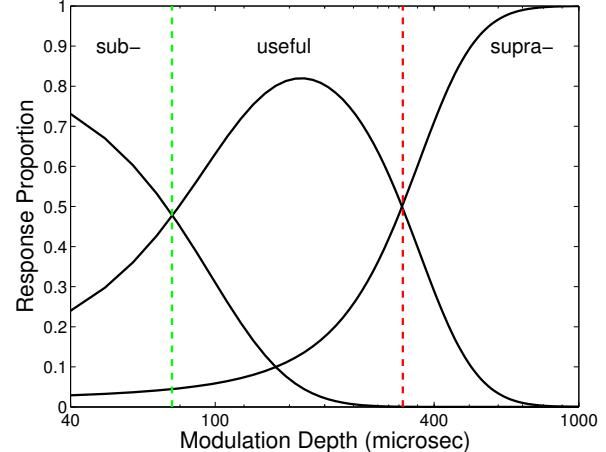


Figure 1: A response proportion plot showing the logit functions fit to the distributions obtained from the three-category response task, in this case given in response to vibrato stimuli with modulation depth varying between 40 and 1000 μ s, all presented at a fixed modulation rate of 6 Hz. Under the left-most curve, the term 'sub-' is written to indicate that for this range of modulation depth values, the resulting vibrato was perceived to be sub-useful. For the right-most curve, the term 'supra-' is written to indicate that these modulation depth values produce too strong a resulting vibrato to be judged as musically useful. The middle curve plots the remaining proportion of the responses that define the useful range of modulation depth values at this modulation rate. The green vertical dashed line marks the transition from sub-useful to useful, and the red vertical dashed line marks the transition from useful to supra-useful modulation depths.

at a fixed modulation rate of 6 Hz, and with modulation depth varying between 40 and 1000 μ s. Under the left-most curve, the term 'sub-' is written to indicate that for this range of modulation depth values, the resulting vibrato was perceived to be sub-useful. For the right-most curve, the term 'supra-' is written to indicate that these modulation depth values produce too strong a resulting vibrato to be judged as musically useful. The middle curve plots the remaining proportion of the responses that define the useful range of modulation depth values at this modulation rate. The green vertical dashed line marks the transition from sub-useful to useful, and the red vertical dashed line marks the transition from useful to supra-useful modulation depths.

The fact that a very similar pattern of results was observed for all three effects is underscored in the summary plot shown in Figure 2. In the figure, the boundaries between three perceptual regions are shown by a single common curve for three digital audio effects tested. These boundaries are plotting in the same manner as those in the contour plot shown in Figure 1. Transition points in the control space for the lower boundary of the musically useful region are marked using downward-pointing triangles as plotting symbols. Transition points for the upper boundary of the useful region are marked using upward-pointing triangles. Due to the curvilinear relation between modulation rate and the boundary values of modulation depth defining the useful region of modulation-based effects, the depth values were regressed upon modulation period (in seconds), the inverse of modulation rate (in Hz). Submitting the inverse data to regression effectively linearized the curve, and

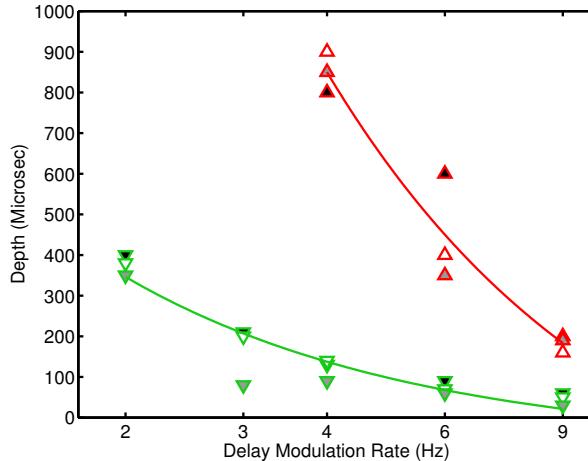


Figure 2: Plot showing curves fit to the transition points between adjacent perceptual response regions within the control space defined by two delay-modulation parameters, modulation rate and modulation depth. Points associated with the lower boundary of the musically useful region are plotted using downward-pointing triangles, and points associated with the upper boundary of the useful region are marked using upward-pointing triangles. Three different symbol colors denote effect types (black: vibrato, white: flange, and grey: chorus). For the modulation depth data derived for all three types of effects, a simple linear regression was executed to fit a prediction equation for the lower boundary of the useful region using modulation period as the independent variable. The resulting equation accounted for 85% of the variance in modulation-depth, with a slope of 814, and an intercept of -66 (these are in dimensionless units, but large because the independent variable, modulation period, was measured in seconds, and the dependent variable, modulation depth, was measured in μs). The F statistic for the lower boundary regression was 71.52, significant at $p < .01$. The equation fit to the upper boundary accounted for 94% of the variance, but could only be fit for modulation rates 4, 6, and 9 Hz, since at lower rates the modulation-depths at which these effects were tested always gave usable results (hence no boundary could be observed at 2 and 3 Hz). At least within this restricted range, the equation fit the derived data with a slope of 4800, and an intercept of -350. The F statistic for the upper boundary regression was 140.37, again significant at $p < .01$.

decreased the residual error by a factor of 2 and 3, for upper and lower boundaries, respectively). To depict the curvilinear relations, however, the figure shows the regression results plotted on modulation rate. The regression analysis is summarized in the figure caption.

4. CONCLUSION

The results map out, for two delay-modulation parameters, three perceptual regions for three related digital audio effects. Most importantly, the results indicate the parameter ranges that seem most musically useful for a representative group of young adult listeners. While the generality of the results may be somewhat limited by the narrow scope of the stimuli presented here, the success of the methods used here implies that similar tests might well answer

questions about optimal parameter ranges for other applications and implementations. An important caveat to make in this regard is that the musical usefulness of a particular delay-modulation-based effect must depend not only upon the processing parameters (e.g., depth and rate), but also upon input signal characteristics. This is a factor not addressed adequately in the current study, and calls for further investigation of the how to predict musical usefulness of delay-modulation-based effects from an analysis of the output signal itself. It is likely, however, that generalization is possible for guitar-sound inputs in regard to specifying the region of parameter values that would likely be found to produce a result judged too extreme for most musical applications. The fact that the same limits are observed for all three types of effects implies that most implementations of these effects will have similar limits, since many implementation will be less different from the three examples tested here than those examples are from one another. This observed commonality between vibrato, flange, and stereo chorus effects may be particularly helpful in predicting where limits will be for novel DAFx that are based upon modulated delay lines.

5. ACKNOWLEDGMENTS

This work was partially supported by McGill University's Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT), and by the National Sciences and Engineering Research Council of Canada (NSERC).

6. REFERENCES

- [1] C. Roads, *Computer Music Tutorial*. MIT Press, 1998.
- [2] J. Dattorro, "Effect design, part 2: Delay-line modulation and chorus," *J. Audio Eng. Soc.*, vol. 45, no. 10, pp. 764–788, Oct. 1997.
- [3] U. Zölzer, *DAFx—Digital Audio Effects*. John Wiley & Sons, 2002.
- [4] E. Zwicker and H. Fastl, *Psychoacoustics*, 2nd ed., ser. Springer Series in Information Sciences. Springer-Verlag, 1999, vol. 22.
- [5] F. Wickelmaier and W. Ellermeier, "Scaling the perceived fluctuation strength of frequency-modulated tones," *Ac. Soc. of Am.*, vol. 115, no. 5, p. 2600, May 2004.
- [6] W. M. Hartmann, "Flanging and phasers," *J. Audio Eng. Soc.*, vol. 26, no. 6, pp. 439–443, June 1978.
- [7] J. Dattorro, "Effect design, part 1: Reverberator and other filters," *J. Audio Eng. Soc.*, vol. 45, no. 9, pp. 660–684, Sept. 1997.
- [8] V. Verfaillie, C. Guastavino, and P. Depalle, "Perceptual evaluation of vibrato models," in *Proc. Colloq. Interdisciplinary Musicology (CIM'05)*, Montreal, Quebec, Canada, Mar. 2005.
- [9] W. M. Hartmann, *Signals, Sound, and Sensation*. Woodbury, NY: AIP Press, 1997.
- [10] J. Neter, M. H. Kutner, C. J. Nachtheim, and W. Wasserman, *Applied Linear Statistical Models*, 4th ed. WCB/McGraw-Hill, 1996.
- [11] J. L. Fleiss, B. Levin, and M. C. Paik, *Statistical Methods for Rates and Proportions*, 3rd ed. John Wiley & Sons, 2003.

INTER GENRE SIMILARITY MODELLING FOR AUTOMATIC MUSIC GENRE CLASSIFICATION

Ulaş Bağcı, Engin Erzin

MVGL, College of Engineering
Koç University, Istanbul, Turkey
{ubagci|erzin}@ku.edu.tr

ABSTRACT

Music genre classification is an essential tool for music information retrieval systems and it has been finding critical applications in various media platforms. Two important problems of the automatic music genre classification are feature extraction and classifier design. This paper investigates inter-genre similarity modelling (IGS) to improve the performance of automatic music genre classification. Inter-genre similarity information is extracted over the mis-classified feature population. Once the inter-genre similarity is modelled, elimination of the inter-genre similarity reduces the inter-genre confusion and improves the identification rates. Inter-genre similarity modelling is further improved with iterative IGS modelling(IIGS) and score modelling for IGS elimination(SMIGS). Experimental results with promising classification improvements are provided.

1. INTRODUCTION

Music genre classification is crucial for the categorization of bulky amount of music content. Automatic music genre classification finds important applications in professional media production, radio stations, audio-visual archive management, entertainment and recently appeared on the Internet. Although music genre classification is done mainly by hand and it is hard to precisely define the specific content of a music genre, it is generally agreed that audio signals of music belonging to the same genre contain certain common characteristics since they are composed of similar types of instruments and having similar rhythmic patterns. These common characteristics motivated recent research activities to improve automatic music genre classification [1, 2, 3, 4]. The problem is inherently challenging as the human identification rates after listening to 3sec samples are reported to be around 70% [5].

Feature extraction and classifier design are two pretentious problems of the automatic music genre classification. Timbral texture features representing short-time spectral information, rhythmic content features including beat and tempo, and pitch content features are investigated thoroughly in [1]. Another novel feature extraction method is proposed in [3], in which local and global information of music signals are captured by computation of histograms on their Daubechies wavelets coefficients. A comparison of human and automatic music genre classification is presented in a study [4]. Mel-frequency cepstral coefficients (MFCC) are also used for modelling and discrimination of music signals [1, 6].

Various classifiers are employed for automatic music genre recognition including K-Nearest Neighbor (KNN) and Gaussian Mixture Models (GMM) classifiers as in [1, 3], and Support Vector Machines (SVM) as in [3]. In a recent study, Boosting is used as a dimension reduction tool for audio classification [7].

In [8], we proposed *Boosting Classifiers* to improve the automatic music genre classification rates. In this study, in addition to inter-genre similarity modelling, two alternative classifier structures are proposed: i) Iterative inter-genre similarity modelling, and ii) Score modelling for inter-genre similarity elimination. Once the inter-genre similarity is modelled, elimination of those similarities from the decision process reduces the inter-genre confusion and improves the identification rates.

The organization of the paper includes a brief description of the feature extraction in Section 2. The discriminative music genre classification using the inter-genre similarity is discussed in Section 3. Later in Section 4 experimental results are provided following with discussions and conclusions in the last section.

2. FEATURE EXTRACTION

Timbral texture features, which are similar to the proposed feature representation in [1], are considered in this study to represent music genre types in the spectral sense. Short-time analysis over 25ms overlapping audio windows are performed for the extraction of timbral texture features for each 10ms frame. Hamming window of size 25ms is applied to the analysis audio segment to remove edge effects. The resulting timbral features from the analysis window are combined in a 17 dimensional vector including the first 13 MFCC coefficients, zero-crossing rate, spectral centroid, spectral roll-off and spectral flux.

3. MUSIC CLASSIFICATION USING INTER-GENRE SIMILARITY

The music signals belonging to the same genre contain certain common characteristics as they are composed of similar types of instruments with similar rhythmic patterns. These common characteristics are captured with statistical pattern recognition methods to achieve the automatic music genre classification [1, 2, 3, 4]. The music genre classification is challenging problem, especially when the decision window spans a short duration, such as a couple of seconds. One can also expect to observe similarities of spectral content and rhythmic patterns across different music genre types, and with a short decision window mis-classification and confusion rates increase. IGS modelling is proposed to decrease the level of confusion across similar music genre types. The IGS modelling forms clusters from the hard-to-classify samples to further eliminate the inter-genre similarity in the decision process of classification system. Inter-genre similarity modelling is further improved with iterative IGS modelling(IIGS) and score modelling for IGS elimination(SMIGS). IGS and its variations are described in the following subsections.

3.1. Inter-Genre Similarity Modelling

The timbral texture features represent the short-term spectral content of music signals. Since, music signals may include similar instruments and similar rhythmic patterns, no sharp boundaries between certain different genre types exist. The inter-genre similarity modelling (IGS) is proposed to capture the similar spectral contents among different genre types. Once the IGS clusters are statistically modelled with GMM, the IGS frames can be captured and removed from the decision process to reduce the inter-genre confusion.

Let $\lambda_1, \lambda_2, \dots, \lambda_N$ be the N different genre models in the database ($N = 9$ in our database). In this study, the Gaussian Mixture Models (GMM) are used for the class-conditional probability density function estimation, $p(f|\lambda)$, where f and λ are respectively the feature vector and the genre class model. The construction of the IGS clusters and the class-conditional statistical modelling (GMM) can be achieved with the following steps:

- i. Perform the statistical modelling of each genre with GMM in the database using the available training data of the corresponding music genre class.
- ii. Perform frame based genre identification task over the training data, (gmm test over training data), and label each frame as a true-classification or mis-classification.
- iii. Construct the statistical model, λ_{IGS} with GMM, through the IGS cluster over all the mis-classified frames among all the music genre types.
- iv. Update all the N -class music genre models, λ_n , using the true-classified frames.

The above construction creates N -class music genre models and a single-class IGS model. In the music genre identification process, given a sequence of features, $\{f_1, f_2, \dots, f_K\}$, which are extracted from a window of music signal, one can find the most likely music genre class, λ^* , by maximizing the weighted joint class-conditional probability,

$$\lambda^* = \arg \max_{\lambda_n} \frac{1}{\sum_k \omega_{kn}} \sum_{k=1}^K \omega_{kn} \log p(f_k | \lambda_n) \quad (1)$$

where the weights ω_{kn} are defined based on the class-conditional IGS model as following,

$$\omega_{kn} = \begin{cases} 1 & \text{if } p(f_k | \lambda_n) > p(f_k | \lambda_{IGS}), \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The proposed weighted joint class-conditional probability maximization eliminates the IGS frames for each music genre from the decision process. The inter-genre confusion decreases and the genre classification rate increases with the resulting discriminative decision process. Experimental results, which are supporting the discrimination based on the IGS elimination, are presented in Section 4.

3.2. Iterative Inter-Genre Similarity Modelling

Inter-genre similarity modelling can be repeatedly used to extend the detection of hard-to-classify samples in the training data. In each iteration a new IGS model is formed over the new set of mis-classified samples. In the decision process a frame is eliminated if

it matches any one of the IGS classes. The construction of the T -step iterative inter-genre similarity (IIGS) models can be defined with the following steps:

- i. Perform IGS modelling, and get λ_{IGS_1} and updated music genre models λ_n for all N -class. Set $t = 2$.
- ii. Perform frame based genre identification task with IGS model over the training data, and label each frame as a true-classification, mis-classification or true-IGS classification.
- iii. Construct the statistical model, λ_{IGS_t} , over all the mis-classified frames among all the music genre types.
- iv. Update all the N -class music genre models, λ_n , over the true-classified frames.
- v. Increment iteration counter t , and if $t \leq T$ go to step [ii.]

The above construction creates N -class music genre models and T -class IGS model. In the music genre identification process, the decision is taken by maximizing the weighted joint class-conditional probability in Equation 1. In IIGS modelling the weights ω_{kn} are re-defined based on the IGS models as following,

$$\omega_{kn} = \begin{cases} 1 & \text{if } p(f_k | \lambda_n) > p(f_k | \lambda_{IGS_t}) \text{ for any } t = 1, \dots, T \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

3.3. Score Modelling for IGS Elimination (SMIGS)

In IGS modelling, the elimination of likely mis-classified frames is performed with a hard thresholding. That is, if IGS model produces the highest likelihood score for a frame, that frame is eliminated even though the best genre model is the true class and has a very close likelihood score to the IGS likelihood. Alternatively, IGS may score lower than the best likelihood score, while the best likelihood score belongs to a false-class. For such possible wrong decisions, the relative likelihood differences can be used as a reliability factor for the IGS elimination process. For example, if the likelihood difference between best two likelihood scores of music genre classes is low, one can claim that the decision is not reliable and the frame should be eliminated even though the IGS likelihood score is not the highest one. Hence, rather than taking a hard decision based on IGS likelihood score, one can model frame elimination based on the likelihood score distributions of the best M genre classes and the IGS class. Score modelling for IGS elimination is built on this idea, and described with the following procedure:

- i. Perform IGS modelling, and get λ_{IGS} and updated music genre models λ_n for all N -class.
- ii. Perform frame based genre identification task for each genre λ_n over the training data, extract the highest class conditional likelihood score that belongs to a class λ^* other than λ_n , $p(f_k | \lambda^*)$, $p(f_k | \lambda_n)$ and $p(f_k | \lambda_{IGS})$. Form a 3-dimensional likelihood score difference vector,

$$s_k = [\Delta_k^1 \Delta_k^2 \Delta_k^3]$$

where

$$\begin{aligned} \Delta_k^1 &= (p(f_k | \lambda_n) - p(f_k | \lambda^*)), \\ \Delta_k^2 &= (p(f_k | \lambda_n) - p(f_k | \lambda_{IGS})), \\ \Delta_k^3 &= (p(f_k | \lambda^*) - p(f_k | \lambda_{IGS})). \end{aligned}$$

- iii. Construct the statistical models of the likelihood difference vectors for each genre λ_n over the true-classified and mis-classified frames of IGS modelling in step [i.], respectively as λ_{n_1} and λ_{n_0} .

The above construction creates N -class music genre models and for each genre true- and mis- classification models of the likelihood differences are extracted. In the music genre identification process, the decision is taken by maximizing the weighted joint class-conditional probability in Equation 1. In score modelling for IGS elimination the weights ω_{kn} are re-defined as following,

$$\omega_{kn} = \begin{cases} 1 & \text{if } p(f_k|\lambda_{n_1}) > p(f_k|\lambda_{n_0}) \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

4. EXPERIMENTAL RESULTS

Evaluation of the proposed classification algorithms is performed over a music genre database that includes 9 different genre types: classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. The songs in the database had been collected from CD collection of the authors and most of the songs are recorded from randomly chosen broadcast radios on the internet. The database includes totally 566 different representative audio segments of duration 30sec for all 9 music genre types, resulting a total duration of $566 \times 30 = 16980$ seconds. All the audio files are stored mono at 16000Hz with 16-bit words. The resulting timbral texture feature vectors are extracted for each 25msec audio frame with an overlapping window of size 10msec. The music genre classification is performed based on the maximization of the class-conditional probability density functions, which are modelled using the Gaussian mixture models (GMM). In experiments two-fold cross validation is used: the database is split into two partitions, each partition includes half of the audio segments from each genre type, where they are used in alternating order for training and testing of the music genre classifiers, and the average correct classification rates are reported in the following.

Decision	Correct Classification Rates (%)			
Window	Flat	IGS	SMIGS	IIGS
0.5s	44.61	49.95	52.70	50.46
1s	46.74	54.08	55.62	54.35
3s	49.22	58.50	61.99	59.16
30s	55.73	62.56	62.57	64.71

Table 1: The average correct classification rates of the flat, IGS clustering, score modeling for IGS elimination (SMIGS) and iterative IGS clustering, using 8-mixture GMM modeling for varying decision window sizes.

The three proposed music genre classification schemes, which are variations of inter-genre similarity elimination to reduce inter-genre confusion, are evaluated and compared with a flat classifier structure. Tables 1, 2 and 3 present correct classification rates of flat, IGS, SMIGS and IIGS classifiers for respectively 8, 16 and 32 mixture GMM modelling. Note that, IGS classification improves flat classification rates for all decision windows. We also observe further improvement using IIGS and SMIGS classifiers over IGS classifier. While the IIGS improvements are observed to be better for larger decision window sizes, the SMIGS improvements

are better for smaller decision window sizes with 8-mixture GMM modeling. This behavior is expected, since iterative IGS expands the inter-genre similarity modelling, which causes elimination of more frame decisions, IIGS needs larger decision window sizes to bring further improvement.

Decision	Correct Classification Rates (%)			
	Flat	IGS	SMIGS	IIGS
0.5s	46.87	53.49	55.35	55.09
1s	49.32	56.80	57.06	58.71
3s	53.18	61.89	61.53	64.23
30s	57.78	66.91	67.26	72.48

Table 2: The average correct classification rates of the flat, IGS clustering, score modelling for IGS elimination (SMIGS) and iterative IGS clustering, using 16-mixture GMM modelling for varying decision window sizes.

However, with increasing number of GMM mixtures as presented in Tables 2 and 3, iterative IGS classifier is observed as the clear winner of these three discriminative music genre classification schemes at all decision window sizes. The classification rate improvements, which are achieved with IGS based classifiers, are significant, especially when the challenging automatic music genre classification task is considered with 70% human identification rate over 3s decision windows and 61% identification rate reported in [1].

Decision	Correct Classification Rates (%)			
	Flat	IGS	SMIGS	IIGS
0.5s	48.83	61.05	62.03	62.57
1s	51.81	65.28	66.64	66.96
3s	54.83	73.25	71.00	74.43
30s	59.90	83.16	81.58	84.41

Table 3: The average correct classification rates of the flat, IGS clustering, score modelling for IGS elimination (SMIGS) and iterative IGS clustering, using 32-mixture GMM modelling for varying decision window sizes.

5. CONCLUSION

Automatic music genre classification is an important tool for music information retrieval systems. Feature extraction and the classification design are the two significant problem of music genre classification systems. In feature extraction, a set of widely used timbral features are considered. In this work, we investigate three novel classifier structures for discriminative music genre classification. In proposed classifier structure, inter-genre similarities are captured and modelled over the mis-classified feature population for the elimination of the inter-genre confusion. The proposed iterative IGS model expands the inter-genre similarity modelling to better eliminate these similarities for the decision process. In score modelling for IGS elimination, a novel scheme based on statistical modelling of decision region of each genre for capturing IGS frames is presented. Experimental results with promising identification improvements are obtained with classifier design based on

the similarity measure among genres. Both in [8] and in this study we observed that IGS improves the flat classifiers and we also investigated the two extension of IGS modelling which increments the identification rates further depending on the number of mixture for GMM or size of the decision window.

6. REFERENCES

- [1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio Proc.*, vol. 10, no. 5, pp. 293–302, July 2002.
- [2] D. Pye, "Content-based methods for managing electronic music," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'00)*, Istanbul, Turkey, 2000, pp. 2437 – 2440.
- [3] T. Li, M. Ogihara, and Q. Li, "A comparative study on content-based music genre classification," in *Proc. 26th Int. ACM SIGIR conf. Research and Development in Information Retrieval*, 2003, pp. 282–289.
- [4] S. Lippens, J. P. Martens, T. D. Mulder, and G. Tzanetakis, "A comparison of human and automatic musical genre classification," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)*, Montreal, Canada, vol. 4, May 2004, pp. 233–236.
- [5] D. Perrot and R. Gjerdingen, "Scanning the dial: An exploration of factors in identification of musical style," in *Proc. Soc. Music Perception Cognition*, 1999, p. 88.
- [6] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *Proc. Int. Symp. Music Information Retrieval (ISMIR'00)*, Plymouth, Massachusetts, USA, 2000, pp. 138–147.
- [7] S. Ravindran and D. Anderson, "Boosting as a dimensionality reduction tool for audio classification," in *Proc. Int. Symp. Circuits and Systems (ISCAS '04)*, vol. 3, May 2004, pp. 465–468.
- [8] U. Bağcı and E. Erzin, "Boosting classifiers for music genre classification," in *20th Int. Symp. on Computer and Information Sciences (ISCIS 2005)*, also appeared in *Lecture Notes in Computer Science (LNCS)* by Springer-Verlag, Istanbul, Oct. 2005, pp. 575–584.

VARIABLE PRE-EMPHASIS LPC FOR MODELING VOCAL EFFORT IN THE SINGING VOICE

Karl I. Nordstrom, Peter F. Driessens

Music Intelligence and Sound Technology Interdisciplinary Centre (MISTIC)
University of Victoria, Canada
knordstr@uvic.ca, peter@ece.uvic.ca
<http://karlnordstrom.ca>, <http://www.ece.uvic.ca/~peter/>

ABSTRACT

In speech and singing, the spectral envelope of the glottal source varies according to different voice qualities such as vocal effort, lax voice, and breathy voice. In contrast, linear prediction coding (LPC) models the glottal source in a way that is not flexible. The spectral envelope of the source estimated by LPC is fixed and determined by the pre-emphasis filter. In standard LPC, the formant filter captures variation in the spectral envelope that should be associated with the source. This paper presents variable pre-emphasis LPC (VPLPC) as a technique to allow the estimated source to vary. This results in formant filters that remain more consistent across variations in vocal effort and breathiness. VPLPC also provides a way to change the envelope of the estimated source, thereby changing the perception of vocal effort. The VPLPC algorithm is used to manipulate some voice excerpts with promising but mixed results. Possible improvements are suggested.

1. INTRODUCTION

Linear prediction coding (LPC) estimates a voice source with a fixed spectral envelope. The true voice source has a spectral envelope that varies. As a result, part of the perceived voice quality [1, 2, 3] that should be in the source ends up in the LPC filter [4]. This paper presents variable pre-emphasis LPC (VPLPC) for separating the spectral envelope into two components: formant filter and source envelope. The paper describes why the true source varies between high-effort and breathy voices, why standard LPC estimates a formant filter that captures source variation, and how VPLPC can estimate a formant filter that is more consistent across different voice qualities. An attempt is then made to manipulate the perceived vocal effort with the VPLPC algorithm.

1.1. Variation in the glottal source

The spectral envelope of the glottal source does not remain constant but varies according to changes in glottal voice quality such as tense voice, lax voice and breathiness [5]. This variation in glottal quality often happens within a single spoken or sung phrase. When a voice is tense, it requires more effort to phonate. The resulting voice has more high frequency content than the same voice in a relaxed state. When a voice is relaxed (known as lax voice), the vocal folds move freely resulting in vibrations that appear almost sinusoidal. The lower harmonics are much stronger relative to the upper harmonics. Air often leaks between the vocal folds when the voice is relaxed. When air leakage causes significant aspiration noise and the vocal folds are relaxed, it is known as a breathy voice.

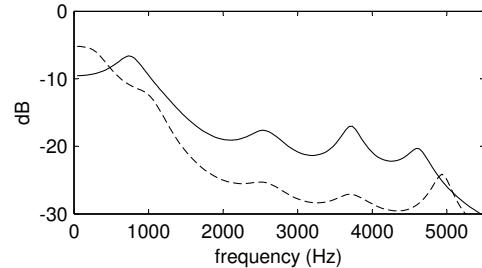


Figure 1: Frequency spectra of LPC filters for breathy voice (dashed line) and high-effort voice (solid line). The same voice is singing the same vowel on the same pitch.

Vocal effort is a subjective term that describes a strained or tense voice quality. The perception of vocal effort has been associated with compression of the vocal folds and a reduced open quotient [6]. When a voice exhibits vocal effort, pressure builds up behind the vocal folds. When the pressure exceeds the resistance of the vocal folds, they open, releasing a short burst of air before quickly closing again. The voice source looks like a series of impulses and the spectral envelope is flatter than the envelope from a lax or breathy voice. The difference in the spectra between voices with high and low effort can be seen in Figure 1.

There are a number of indicators of vocal effort. While the overall sound level is significant, people are able to identify the effort a person is putting into speaking or singing even when the sound level has been normalized [7]. Vocal effort is also associated with higher pitches and changes in the phrasing [8]. We are looking at singing voices and therefore many of these factors are less influential. The music specifies pitch and phrasing, and the sound level is normalized to the recording. In this situation, the spectral balance between low frequencies and high frequencies is more significant. Voices with more vocal effort have more high-frequency content. Another indicator of vocal effort is the mix of harmonics and noise in the voice signal. Voices with vocal effort have little aspiration noise (as long as the source has not become aperiodic). In contrast, relaxed voices have a more aspiration noise.

1.2. Problem with LPC

Standard LPC models the voice in a way that does not appropriately capture variation in the source. This subsection describes the problem.

The source-filter concept provides a perceptual approximation of the glottal source and vocal tract and is widely used for voice

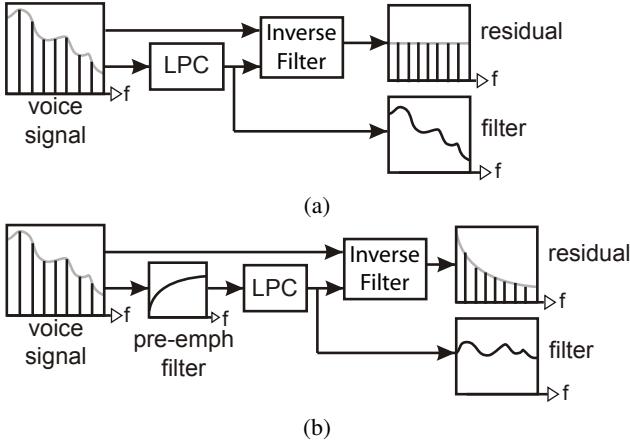


Figure 2: (a) LPC analysis algorithm (b) LPC analysis algorithm with pre-emphasis filter. Note that the tilt of the residual spectrum is the inverse of the pre-emphasis filter.

analysis and synthesis [9]. The most common technique for doing this is linear prediction coding (LPC). The operation of the LPC algorithm [10] and its relation to the human voice [11] have been greatly discussed in the literature. LPC finds a filter to fit the spectrum of the input signal. If we apply the inverse of this filter to the original signal, we can extract the LPC residual. This residual represents the glottal source.

LPC attempts to minimize the error between the spectrum of the signal and the frequency response of the filter. As a result, the LPC residual has a flat spectrum as seen in Figure 2(a). Most LPC algorithms compensate for lip radiation with a pre-emphasis filter. Pre-emphasis boosts the high frequencies, resulting in slightly better formant matching at the high frequencies and fewer scaling issues in fixed-point algorithms. The tilt of the LPC residual is the inverse of the pre-emphasis filter as seen in Figure 2(b). This is closer to the expected appearance of the glottal source. However, the pre-emphasis algorithm does not estimate the spectral envelope of the source. The slope of the residual's spectrum is fixed.

The LPC algorithm does not take into account spectral changes to the glottal source. Whether the voice has much or little vocal effort, whatever the shape of the glottal spectrum, the pre-emphasis filter remains the same and the LPC residual has the same spectral envelope. This means that variation in the envelope of the glottal spectrum is captured by the LPC filter instead of the LPC residual [4, 12]. This appears to be an inherent part of LPC that has not been clearly addressed in the speech literature.

2. VARIABLE PRE-EMPHASIS LPC

This paper proposes that the pre-emphasis filter be made variable. We already know that the pre-emphasis filter determines the spectral envelope of the LPC residual. Variable pre-emphasis results in a residual that responds to broad changes to the spectral envelope. As long as this variation in the spectral envelope does not affect the perception of formants, the assumption is that variable pre-emphasis captures a glottal voice quality. We cannot verify this by physiological measurement but we can perceptually evaluate this influence.

If we compare VPLPC formant filters from high-effort and breathy voices, we should find the VPLPC formant filters to be more consistent than the corresponding formant filters from stan-

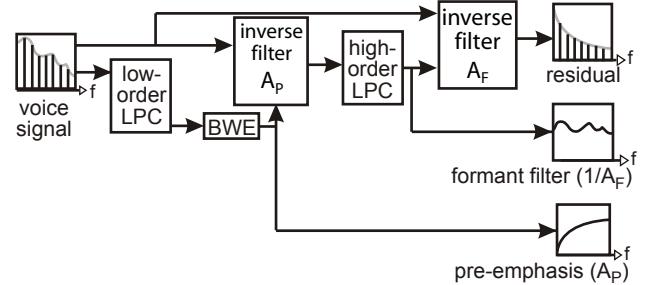


Figure 3: Variable pre-emphasis LPC analysis. Low-order LPC plus bandwidth expansion (BWE) determines the variable pre-emphasis filter A_P . Following pre-emphasis, high-order LPC determines the formant filter $1/A_F$.

dard LPC. Variable pre-emphasis should reduce variation in the formant filters by allowing glottal voice qualities to pass through to the residual.

Variable pre-emphasis is not a new idea. Some LPC algorithms use variable pre-emphasis to improve voice compression or speech recognition [13]. This paper adds to that research by suggesting that there is a physiological reason why variable pre-emphasis works and by attempting to use it to manipulate the perception of vocal effort.

2.1. Low-order LPC

One way to estimate an appropriate pre-emphasis filter is to carry out low-order LPC. The low-order LPC analysis method is presented in Figure 3. Because LPC tends to produce filters that are peaky, bandwidth expansion (BWE) is carried out on the filter coefficients using pole scaling [14].

From experimentation, an order of three appeared to work best while using a sampling rate of 22050 Hz. In standard LPC, one pole is located at 0 Hz to represent lip radiation. This can be thought of as one pole in the pre-emphasis filter. Adding another pole pair enables the algorithm to capture a broad resonance in the spectrum around 2–3 kHz that can happen in high-effort voices. A couple of example pre-emphasis filters are shown in Figure 4.

It may seem strange that there is an upper resonance in the pre-emphasis filter in Figure 4(b). Most voice analysis in linguistic research takes place at lower sampling frequencies, truncating the plot at approximately 5 kHz. This makes changes in vocal effort appear as a tilt. Plotting the spectrum up to 11 kHz reveals that many high-effort voices have a hump in the spectrum.

After pre-emphasis, the signal is fed into high-order LPC to estimate the formant filter. The order of the formant filter was informally adjusted to the order that perceptually seemed to work best. Orders between nineteen and twenty-four at a sampling rate of 22050 Hz roughly correspond to the length of a human vocal tract when LPC is interpreted as a physical model. Since LPC does not typically estimate a true vocal tract, we weren't constrained by this range. We used an order of thirty for the sound excerpts that we modified. To extract the residual, the original signal is inverse filtered by the formant filter.

2.2. Synthesis method

To modify the perceived vocal effort, the spectral envelope of the residual has to be modified and resynthesized. The process of

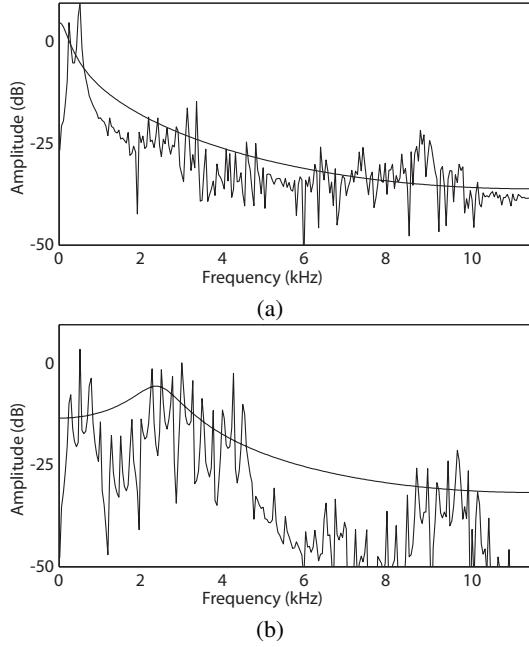


Figure 4: Inverse of the pre-emphasis filter (A_P) estimated by low-order LPC for (a) breathy and (b) high-effort voice excerpts.

resynthesizing the voice is illustrated in Figure 5. First, the spectral envelope of the residual is flattened by filtering with the pre-emphasis filter. The flat spectrum makes it easier to add aspiration noise if required. Two matched butterworth filters were used to blend aspiration noise, one filter applied to the flattened residual and one filter applied to the white noise. Then, a filter representing the desired spectral envelope is applied to the mix of the flattened residual and aspiration noise. The resulting signal is the modified source with the desired spectral envelope and aspiration noise when required. The modified source is fed through the formant filter to synthesize the voice.

Voices with less vocal effort have less aspiration noise. In the algorithm, the aspiration was generated as gaussian noise. This noise was pulsed in sync with the frequency of the voice using a square envelope with a 50% duty cycle. The pitch was estimated using Praat software [15]. The energy level of the noise was adjusted to be the same as the energy level of the flattened residual. Matched, first-order butterworth filters were used to low-pass the residual and high-pass the pulsed noise. This blended the noise into the flattened residual.

3. RESULTS

One of the advantages that VPLPC should provide is a formant filter (A_F) that is more consistent over varying voice qualities. We tested this aspect of VPLPC by exciting the formant filters from high-effort voices and breathy voices with the same excitation. For raw voice data, we had three pairs of voice excerpts. Within each pair, the same person sang the same vowel at the same pitch varying only their voice quality between high effort and breathiness. To remove the influence of the source, we used the same LF model [16] as the excitation for all voices. The LF model is the most popular model for glottal excitation. We also carried out the same procedure using standard LPC. Due to the nature of arti-

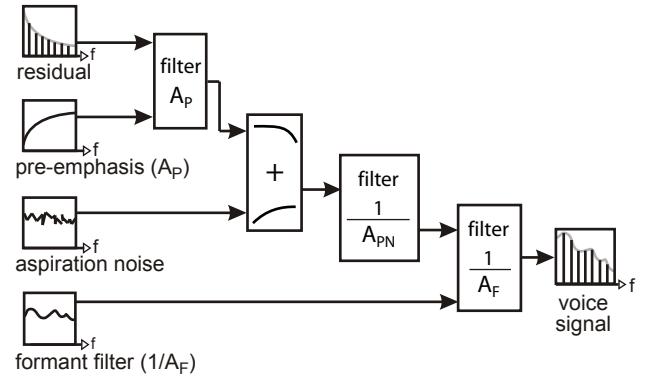


Figure 5: VPLPC synthesis configured to modify the perception of vocal effort. The tilt of the residual is removed by the variable pre-emphasis filter (A_P), leading to a flat spectrum. Matched filters blend in aspiration noise if required. A new pre-emphasis filter (A_{PN}) applies the desired spectral envelope for the glottal source. The signal is then filtered by the formant filter ($1/A_F$) to achieve the new voice signal.

cial excitation, some artifacts were present in the data; however, it was still easy to hear relative differences in voice quality between samples.

When the same LF model excited the VPLPC formant filters (A_F) from the high-effort and the breathy voices, both synthesized voices took on similar voice qualities. The biggest difference between the two voices was that the formant filter for the breathy voice had more jitter and shimmer associated with it, resulting in more artifacts and slightly more perceived breathiness. The LPC formant filters sounded more different from each other. The LPC formant filter from the high-effort voice carried a significant perception of vocal effort. In comparison, the LPC formant filter from the breathy voice carried a significant perception of breathiness.

3.1. Manipulating the perception of vocal effort

In the next stage, we attempted to use VPLPC to manipulate the perception of vocal effort. The shape of the spectral envelope for the desired source was estimated from the excerpts. Using VPLPC analysis, A_P from the breathy voice provided A_{PN} for the high-effort voice. A_P from the high-effort voice provided A_{PN} for the breathy voice. Doing this gives the spectral envelope of the breathy residual to the spectral envelope of the high-effort residual and vice versa.

First, vocal effort was removed from the high-effort voice. The spectral envelope of the high-effort residual was replaced by the spectral envelope of the breathy residual. This reduced the high-frequency content of the voice. Some of the perception of vocal effort was reduced. However, the new voice did not sound as relaxed as the original breathy voice. Although the spectral envelope changed, the mix of harmonics and noise did not change. A relaxed voice should have fewer harmonics and more aspiration noise.

To further reduce the perception of vocal effort, we added noise to the residual as described in section 2.2. The addition of aspiration noise made the synthesized voice sound more natural, which improved the perception of a more relaxed voice. However, the voice that was originally perceived to have high effort was not fully transformed into a relaxed breathy voice. It was difficult to

blend much noise into the residual without creating an unnatural sounding voice. Perceptually, the noise easily separated from the source, sounding like a separate stream of noise rather than part of the voice.

Next, vocal effort was added to the breathy voice. The spectral envelope of the breathy residual was replaced by the spectral envelope of the high-effort residual. This boosted the high-frequency content of the voice. The resulting voice was perceived to have a higher degree of vocal effort but there was too much aspiration noise and not enough harmonics. The voice sounded noisy and unnatural due to the amplified noise.

Original and synthesized voice samples are available online at:
<http://www.ece.uvic.ca/~knordstr/dafx06>

4. CONCLUSIONS

This paper presented the VPLPC algorithm as a method to estimate formant filters that are more neutral across varying voice qualities. While formant filters from standard LPC contain a significant perception of vocal effort, VPLPC appeared able to remove the perception of vocal effort from formant filters that were excited by an LF model.

The VPLPC algorithm was able to partially change the perception of vocal effort by manipulating the residual. The transformation was not complete because the mix of harmonics and noise did not change along with changes to the spectral envelope. When reducing vocal effort, it helped to add pulsed noise into the residual. Unfortunately, only a little noise could be added before there were problems with stream separation between the noise and the residual. In transformations to high effort, the residual did not have enough harmonic content, resulting in voices that sounded noisy.

There are a couple of ways that the artifacts could be reduced. VPLPC, as presented here, involved two-stages of inverse filtering. The opportunity for artifacts increases with each stage, especially when the filters are dynamic like those from LPC. It may be possible to eliminate the pre-emphasis inverse filter by extracting key information from a standard LPC filter. Given that the variable pre-emphasis filter is of a low order, it should be possible to convert an analysis of pole locations into a single measure of vocal effort. This parametric measure of vocal effort could then be used to control the model.

Methods other than low-order LPC could be used to estimate a pre-emphasis filter. A larger number of voice excerpts could provide a better indication for an appropriate estimation method.

5. ACKNOWLEDGEMENTS

I send my thanks out to the following people. Laura Anne Bateman provided some voice excerpts from her research [17]. George Tzanetakis provided some feedback on the paper. Thanks to Glen Rutledge for useful discussions about voice modeling and John Esling for improving my understanding of phonetics. Thanks to IVL Technologies and TC Helicon for starting me in this research path and lending me some audio equipment.

6. REFERENCES

- [1] F. Thibault and P. Depalle, "Adaptive processing of singing voice timbre," in *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE2004)*, Niagara Falls, Ontario, Canada, 2004, pp. 871-874.
- [2] L. Fabig and J. Janer, "Transforming singing voice expression – the sweetness effect," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 70–75.
- [3] A. Loscos and J. Bonada, "Emulating rough and growl voice in spectral domain," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 1998, pp. 188–91.
- [4] K. I. Nordstrom, P. F. Driessens, and G. A. Rutledge, "Influence of the LPC filter upon the perception of breathiness and vocal effort," in *IEEE Int. Symposium on Signal Processing and Information Technology (ISSPIT06)*, Vancouver, Canada, Aug. 2006.
- [5] A. N. Chasaide and C. Gobl, *Handbook of Phonetic Sciences*. W. J. Hardcastle and J. Laver Eds. Blackwell, 1997, ch. Voice source variation, pp. 427–461.
- [6] D. G. Childers and C. K. Lee, "Vocal quality factors: Analysis, synthesis, and perception," *J. Acoust. Soc. Am.*, vol. 90, no. 5, pp. 2394–2410, 1991.
- [7] J.-S. Liénard and M.-G. D. Benedetto, "Effect of vocal effort on spectral properties of vowels," *J. Acoust. Soc. Am.*, vol. 106, no. 1, pp. 411–422, July 1999.
- [8] H. Traunmüller and A. Eriksson, "Acoustic effects of variation in vocal effort by men, women and children," *J. Acoust. Soc. Am.*, vol. 107, no. 6, pp. 3438–3451, June 2000.
- [9] P. R. Cook, "Toward the perfect audio morph? Singing voice synthesis and processing," in *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, 1998, pp. 223– 230.
- [10] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561–580, Apr. 1975.
- [11] J. D. Markel and A. H. Gray Jr., *Linear Prediction of Speech*. New York: Springer-Verlag Berlin Heidelberg, 1976.
- [12] K. I. Nordstrom, G. A. Rutledge, and P. F. Driessens, "Using voice conversion as a paradigm for analyzing breathy singing voices," in *Pacific Rim Conf. on Communications, Computers and Sig. Proc. (PACRIM05)*, Victoria, Canada, 2005, pp. 428 – 431.
- [13] S. E. Bou-Ghazale and J. H. L. Hansen, "A comparative study of traditional and newly proposed features for recognition of speech under stress," *IEEE Trans. Speech and Audio Proc.*, vol. 8, no. 4, pp. 429–442, July 2000.
- [14] P. Kabal, "Ill-Conditioning and bandwidth expansion in linear prediction of speech," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'03)*, Hong Kong, China, 2003, pp. 824–827.
- [15] P. Boersma and D. Weenink, "Praat: A system for doing phonetics by computer," *Glot International*, vol. 5, no. 9/10, pp. 341–345, 2001.
- [16] G. Fant, J. Liljencrants, and Q. Lin, "A four-parameter model of glottal flow," *STL-QPSR*, Vol. 4, pp. 1–13, Tech. Rep., 1985.
- [17] L. A. Bateman, "Soprano, style and voice quality: Acoustic and laryngographic correlates," Master's thesis, University of Victoria, 2004.

FREQUENCY-DEPENDENT BOUNDARY CONDITION FOR THE 3-D DIGITAL WAVEGUIDE MESH

Antti Kelloniemi

Telecommunications Software and Multimedia laboratory
Helsinki University of Technology, Finland
antti.kelloniemi@panphonics.fi

ABSTRACT

The three-dimensional digital waveguide mesh is a method for modeling the propagation of sound waves in space. It provides a simulation of the state of the whole soundfield at discrete timesteps. The updating functions of the mesh can be formulated either using physical values of sound pressure or particle velocity, also called the Kirchhoff values, or using a wave decomposition of these instead. Computation in homogenous media is significantly lighter using Kirchhoff variables, but frequency-dependent boundary conditions are more easily defined with wave variables. In this paper a conversion method between these two variable types has been further simplified. Using the resulting structure, a novel method for defining the mesh boundaries with digital filters is introduced. With this new method, the reflection coefficients can be defined in a frequency-dependent manner at the boundaries of a Kirchhoff variable mesh. This leads to computationally lighter and more realistic simulations than previous solutions.

1. INTRODUCTION

The best method for room acoustic modeling depends on the frequencies of interest and size and complexity of the modeled space. At frequencies below the Schroeder frequency, the wave phenomena such as interference and diffraction of sound have to be taken into account [1]. Geometrical methods typically fail in this when non-trivial shapes are considered. In contrary, good results are accomplished by modeling the wave propagation and scattering in discrete time and space, for example with the digital waveguide (DWG) mesh method as presented in Section 2.

Digital waveguides have been widely used for musical sound synthesis [2, 3]. For example, in 1-D a DWG provides an effective model of a vibrating string. The method has been extended to the multi-dimensional case using the DWG mesh to provide models of vibrating membranes in 2-D [4, 5] and acoustic enclosures in 3-D [6, 7, 8].

The mesh scheme has two formulations: The computation can be done using traveling wave variables or Kirchhoff (*K*) variables [2, 6]. While these formulations are technically equivalent, one variable type is often better suited than the other for certain applications. When operating with multi-dimensional models, the traveling wave formulation consumes significantly more computational and memory resources than the finite-difference time-domain scheme with *K* variables. On the other hand, wave variables are needed when digital filters are designed for modeling frequency-dependent boundary conditions or air absorption, for example [9, 10]. The conversion between the two variable types was introduced by Karjalainen and Erkut [11]. A simplified KW-converter structure based on their results is introduced in Section 3.

The KW-converter makes it possible to define the reflection characteristics of the boundaries of a *K* variable mesh by digital filters. A small variation in its design enables use of a linear-phase FIR filter without adding extra delays compared to a single real valued reflection coefficient, as shown in Section 4. In this design, the filter coefficients are related to the reflection coefficient values in a very straightforward way. In Section 5, a case study is presented, where by use of this filter structure the T_{60} values of a modeled lecture hall are matched with measurement results.

Conclusion of the results and discussion of the future improvements end this paper in Section 6.

2. THE DIGITAL WAVEGUIDE MESH

2.1. The 1-D Digital Waveguides

In one dimension, a DWG is constructed of two delay lines passing the signal in opposite directions and of scattering junctions between them. With wave variables the value at a junction is calculated as a sum of incoming wave variable values p_l^+ :

$$p_c(n) = \frac{2 \sum_{l=1}^2 Y_l p_l^+(n)}{\sum_{l=1}^2 Y_l}, \quad (1)$$

and the outgoing wave variable values are then updated as:

$$p_l^-(n) = p_c(n) - p_l^+(n), \quad (2)$$

where p is the signal value at a junction, subscript *c* denotes the junction to be calculated, n is the time step index, Y is the characteristic admittance of an interconnection, and index *l* denotes the unit delays connected to the junction *c* [4]. The transformation from outgoing to ingoing values happens by passing the values through the unit delays at the interconnections.

In Kirchhoff variables, the updating of the 1-D junction may be written as:

$$p_c(n) = \frac{2 \sum_{l=1}^2 Y_l p_l(n-1)}{\sum_{l=1}^2 Y_l} - p_c(n-2), \quad (3)$$

where index *l* runs now through the neighboring junctions instead of interconnections.

2.2. The 3-D Triangular Mesh

A multi-dimensional DWG structure is constructed by connecting multiple delay lines together. A DWG mesh is defined by a regular grid of scattering junctions separated by bi-directional unit delays. The grid can be constructed in many topologies. For this study, a 3-D triangular structure, also known as 3-D dodecahedral or hexagonal close packed structure[12, 13], was chosen as it

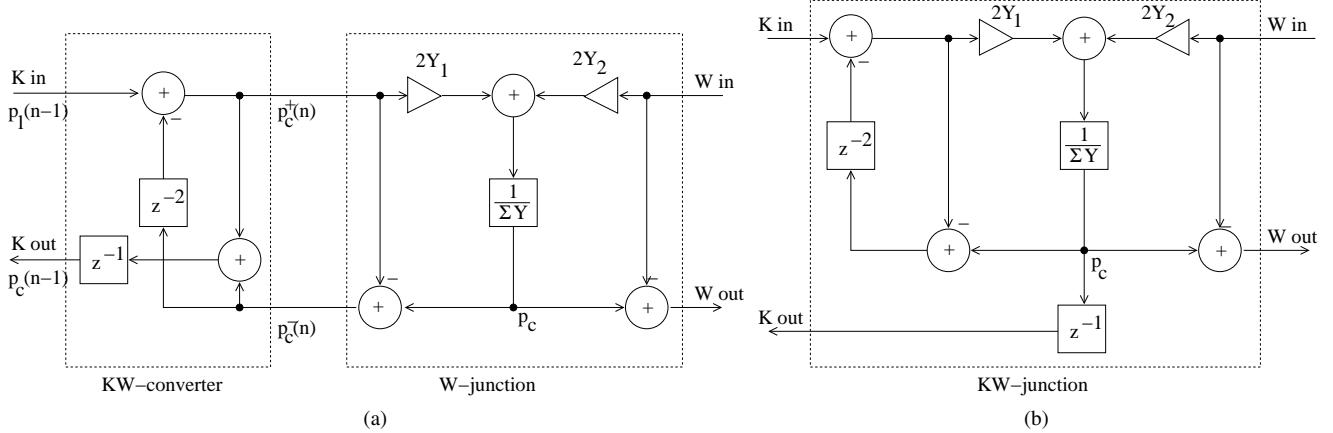


Figure 1: (a) KW-converter: Filter block for changing between Kirchhoff (K) and traveling wave (W) variables is shown with a wave variable junction [11]. (b) KW-junction: The K to wave variable conversion is included at the K input of the junction, scattering computation is done with wave variables and K variable output is provided by delaying the junction value p_c by one time step.

causes less numerical dispersion than the original rectilinear mesh and is more efficient in 3-D than for example the interpolated rectilinear mesh [14, 15].

Kirchhoff variables are used in the interior of the mesh for computational efficiency. When the mesh is considered to be homogenous, the updating function of a 3-D triangular scattering junction with 12 neighbors is written as

$$p_c(n) = \frac{2}{12} \sum_{l=1}^{12} p_l(n-1) - p_c(n-2). \quad (4)$$

When the triangular topology is used, the maximum simulation frequency f is restricted to one third of the sampling frequency f_s of the mesh [16].

The nominal wave propagation speed in a DWG mesh is determined as

$$c = \frac{f_s \Delta x}{\sqrt{N}}, \quad (5)$$

where Δx is the spatial step length and N is the number of spatial dimensions. In a 3-D mesh, wave speed is thus $c = \sqrt{1/3}$ spatial steps per time step [14, 17].

3. CONVERSION BETWEEN KIRCHHOFF AND TRAVELING WAVE VARIABLES

A converter between the two variable types in a digital waveguide mesh is a useful tool, as characteristics of both formulations are needed for creating efficient yet flexible models [10, 11]. Simple rules can be derived for the KW-conversion shown in Fig. 1(a). Firstly, a Kirchhoff variable $p_l(n-1)$ at the K input is converted into a traveling wave variable by $p_c^+(n) = p_l(n-1) - p_c^-(n-2)$. Secondly, for reverse conversion, the junction value p_c at current time instant n is calculated also in a wave variable mesh as shown in (1). An input for a neighboring K junction is created from this by passing the value through a unit delay.

A new kind of building block for mixed modeling, namely the KW-junction, is introduced in Fig. 1(b). Before summing with the wave variables, the K variables are filtered just as earlier and the wave variable ports are constructed as in pure wave variable junctions. More ports with their respective admittances and input

and output signals could be added. The novelty is that the K output is computed only once and not separately for each port as in Fig. 1(a). If the junction would have m K ports, m summations and $m - 1$ unit delays would be saved by the new KW-junction in comparison to the use of earlier proposed KW-converter structure.

4. THE NEW BOUNDARY FILTER FOR K MESH

To approximate the behavior of the wavefront when it coincides with walls, furniture, or other boundaries, the mesh edges have to be defined in a controllable manner. For the K mesh, this has been possible only by defining simple real valued reflection coefficients. In contrary, the reflection characteristics of real walls are significantly frequency-dependent both in magnitude and in phase.

A new construction for defining the reflection coefficients for the K mesh is based on the structure shown in Fig. 1(b). At the boundary, only the K port on the left hand side of the junction is connected to the mesh and there is no signal connected to the wave variable port on the right hand side. The admittance of the K port Y_1 is the same as that of the homogenous mesh, and the reflection coefficient is defined by variable Y_2 . Thus, from (3) we get

$$p_b(n) = \frac{2Y_1 p_1(n-1)}{Y_1 + Y_2} - p_b(n-2), \quad (6)$$

where p_b is the value at the boundary junction and p_1 is the value at the neighboring junction inside the mesh.

The admittance value can be replaced with reflection coefficients: $Y_2 = (1 - R)/(1 + R)$. Using this, and if admittance Y_1 is normalized to 1, (6) can be written as

$$p_b(n) = (1 + R)p_1(n-1) - p_b(n-2). \quad (7)$$

In Fig. 2, the boundary junction structure is shown in a general form. It includes the conversion from K variables to wave variables and a boundary filter $H(z)$. It can be noted that when comparing with Fig. 1(b), one unit delay has been omitted from the signal line passing the signal to the first summation after K variable input and also from the K output line. The scheme is reorganized so that now this unit delay is implemented at the second vertical signal line from left and also inside the filter $H(z)$.

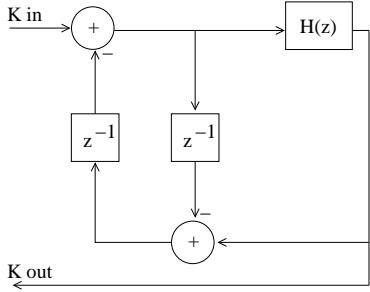


Figure 2: The boundary filter for K mesh: Conversion to wave variables is performed by filtering the input signal. The location of a unit delay has been altered compared to the conversion block in Fig. 1(b) and the multiplications with admittance coefficients have been replaced with filter $H(z)$.

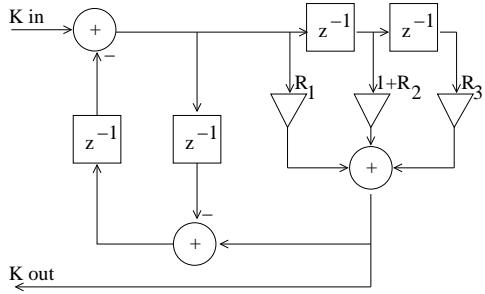


Figure 3: The boundary filter for K mesh: A 3-tap FIR filter is realized as an example.

If $H(z) = (1 + R)z^{-1}$, (7) is implemented with real-valued reflection coefficient R . The reflection coefficient should get values $-1 \leq R \leq 1$ for stable operation. Other possible filters include fractional delays for adjusting the boundary location [13], different lengths of delay lines for diffusive structures [18], or high-order IIR or FIR filters [9].

A very basic example for a boundary filter is shown in Fig. 3. The FIR filter with $H(z) = R_1 + (1 + R_2)z^{-1} + R_3z^{-2}$ causes no extra delay to the reflected signal if $R_1 = R_3$. With this kind of linear-phase filter, the boundary location in the model is not altered in comparison to the real valued reflection coefficient realization. The reflection coefficient value at DC and a simple lowpass or highpass behavior can be defined with the two free parameters of the filter.

More complex boundaries can be simulated using higher order FIR or IIR filters, for which the magnitude and the phase of the responses can be defined at all simulation frequencies.

5. APPLICATION: MODELING OF A LECTURE HALL

As a case study, the lecture hall T3 at Helsinki University of Technology was modeled and the results compared with measured responses. As shown in Fig. 4, the left side wall and the back wall are sloping. A soft absorbent plate is hung at 0.40 m below the ceiling, covering the full area from the back wall to 3.00 m from the front wall.

Boards and stools were present in the hall during the measurements. The sound source was located 2.70 m away from the left side wall and from the front wall equally, at a height of 1.20 m.

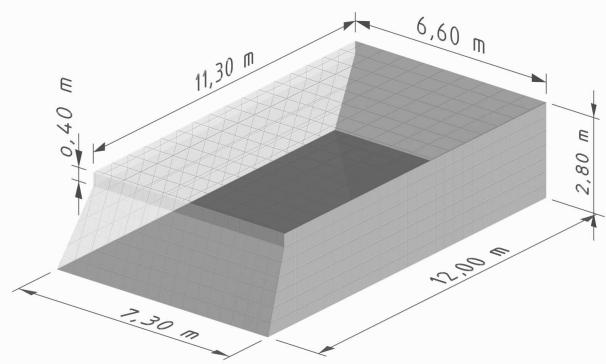


Figure 4: Dimensions of the lecture hall T3 at Helsinki University of Technology, where reference response data was measured.

The impulse response was recorded at 5.50 m from the left wall and 10.00 m from the front wall at a height of 1.70 m.

The hall was modeled using the 3-D triangular DWG mesh with junction spacing $\Delta x = 0.20$ m corresponding to $f_s = 2.875$ kHz. The simulation was run for 5000 time steps, which is equal to about 1.74 s. The reflection coefficients of the boundaries were approximated using two 3-tap FIR filters such as depicted in Fig. 3. One filter was used for the soft area of the ceiling, another for other surfaces. The filter coefficients R_1 and R_3 were bound to have equal values to make the filter linear phase. Thus there were 2 parameters to be optimized per filter. A Nelder-Mead optimization algorithm was run in Matlab to find the coefficients best matching with the mean T_{60} values measured on octave bands with central frequencies 62.5, 125, 250, and 500 Hz. As the low-frequency cutoff of the sound source used for the measurements was at 69 Hz, there was no reliable data to match at lower frequencies.

The sums of the coefficients of each filter were bound to values below one to keep the simulation stable and the soft ceiling was bound to be more absorbing or equal with the other surfaces. The optimization was first done by hand and multiple runs of the optimization algorithm were tested with different starting values. The best results were obtained with $R_1 = R_3 = 0.019577$ and $R_2 = 0.8401$ for the hard surfaces and $R_1 = R_3 = 0.0089562$ and $R_2 = 0.85918$ for the ceiling. The resulting T_{60} values are listed in Table 1 and the simulated and measured responses are plotted with 5 Hz smoothing in Fig. 5. The reverberation times match fairly well, as do many of the peaks and the notches in the responses.

As a linear-phase filter was used, the wall locations were set only by the mesh size in increments of unit delays. This resulted in a good match with all dimensions for the floor and the ceiling, but the sloping walls could be modeled exactly only at certain heights. Also the height of the modeled space was overestimated by 0.08 m. With the fact that the lowered ceiling was not modeled as a separate boundary, this caused error in the simulated frequencies and magnitudes of the modes, seen as shifting of some peaks in Fig. 5. More detailed models could be built and the frequency-dependent reflection characteristics of the boundaries could be simulated more precisely using higher order filters.

Frequency / Hz	Measured T ₆₀ / s	Simulated T ₆₀ / s
62.5	1.62	1.44
125	1.23	1.41
250	0.75	0.64
500	0.67	0.50

Table 1: Measured and simulated reverberation time values of the lecture hall T₃ on four octave bands.

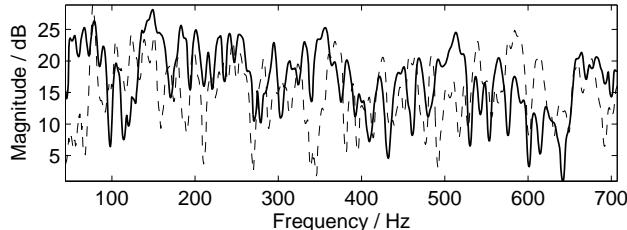


Figure 5: Measured (—) and simulated (—) responses of the lecture hall.

6. CONCLUSIONS AND FUTURE WORK

A simplified structure for KW-conversion has been introduced. It saves computation, particularly if multiple K-ports are connected to a junction at an interface of the two variable types. The new KW-junction has been used in the special case of a mesh boundary, and a way of defining the reflection coefficients as a digital filter has been derived.

A second order linear-phase FIR-filter was used for modeling frequency-dependent reflection characteristics. Higher order filters are needed for more detailed control over the resulting reflection magnitude and phase. As a first addition, fractional delay could be utilised for fine-tuning the boundary locations. For directional reflection control, the boundary junction should be extended to having more than one port connected to the mesh. This kind of extension is also required, if the proposed boundary structure is to be used with the interpolated rectilinear mesh topology.

Finally, a large question that is out of the scope of this paper remains: What are the reflection and diffusion coefficient values of real surfaces at all frequencies and incident angles of interest? Good experimental results are essential for defining the filter characteristics for the simulated boundaries.

7. ACKNOWLEDGMENTS

The work was supported by the Academy of Finland (project no. 201050) and the Nokia Foundation. The author would like to thank Dr. Tapio Lokki for the measurement data, Mr. Seppo Paulin for Fig. 4, and Dr. Damian Murphy, Prof. Matti Karjalainen, Prof. Lauri Savioja, and Prof. Vesa Välimäki for their comments.

8. REFERENCES

- [1] M. R. Schroeder, "The "Schroeder frequency" revisited," *J. Acoust. Soc. Am.*, vol. 99, no. 5, pp. 3240–3241, May 1996.
- [2] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music J.*, vol. 16, no. 4, pp. 74–91, 1992.
- [3] 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.
- [4] S. A. Van Duyne and J. O. Smith, "Physical modeling with the 2-D digital waveguide mesh," in *Proc. Int. Comp. Music Conf. (ICMC'93)*, Tokyo, Japan, Sept. 1993, pp. 40–47.
- [5] F. Fontana and D. Rocchesso, "Physical modeling of membranes for percussive instruments," *Acustica United with Acta Acustica*, vol. 84, pp. 529–542, May/June 1998.
- [6] L. Savioja, T. Rinne, and T. Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. Int. Comp. Music Conf. (ICMC'94)*, Århus, Denmark, Sept. 1994, pp. 463–466.
- [7] S. A. Van Duyne and J. O. Smith, "The 3D tetrahedral digital waveguide mesh with musical applications," in *Proc. Int. Comp. Music Conf. (ICMC'96)*, Hong Kong, Aug. 1996, pp. 9–16.
- [8] P. Huang, S. Serafin, and J. O. Smith, "A waveguide mesh model for high-frequency violin body resonances," in *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, Sept. 2000, pp. 86–89.
- [9] J. Huopaniemi, L. Savioja, and M. Karjalainen, "Modeling of reflections and air absorption in acoustical spaces — a digital filter design approach," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, Oct. 1997.
- [10] M. J. Beeson and D. T. Murphy, "Virtual room modelling using hybrid digital waveguide mesh techniques," in *Proc. 147th Meeting Acoust. Soc. Am.*, New York, USA, vol. 115(5), May 2004, p. 2515, abstract.
- [11] M. Karjalainen and C. Erkut, "Digital waveguides versus finite difference structures: Equivalence and mixed modeling," *EURASIP Journal on Applied Signal Processing*, vol. 7, pp. 978–989, June 2004.
- [12] F. Fontana, D. Rocchesso, and E. Apollonio, "Using the waveguide mesh in modelling 3D resonators," in *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-00)*, Verona, Italy, Dec. 2000, pp. 229–232.
- [13] J. A. Laird, *The physical modelling of drums using digital waveguides*, ser. PhD Thesis. University of Bristol, Nov. 2001.
- [14] S. A. Van Duyne and J. O. Smith, "The tetrahedral digital waveguide mesh," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, Oct. 1995, pp. 234–237.
- [15] L. Savioja and V. Välimäki, "Interpolated rectangular 3-D digital waveguide mesh algorithms with frequency warping," *IEEE Trans. Speech and Audio Proc.*, vol. 11, no. 6, pp. 783–790, Nov. 2003.
- [16] G. R. Campos and D. M. Howard, "On the computational efficiency of different waveguide mesh topologies for room acoustic simulation," *IEEE Trans. Speech and Audio Proc.*, vol. 13, no. 5, pp. 1063 – 1072, Sept. 2005.
- [17] L. Savioja, J. Huopaniemi, T. Lokki, and R. Väänänen, "Creating interactive virtual acoustic environments," *J. Audio Eng. Soc.*, vol. 47, no. 9, pp. 675–705, Sept. 1999.
- [18] K. Lee and J. O. Smith, "Implementation of a highly diffusing 2-D digital waveguide mesh with a quadratic residue diffuser," in *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, Nov. 2004, pp. 309–315.

SOME PHYSICAL AUDIO EFFECTS

Edgar Berdahl, Julius O. Smith

CCRMA, Stanford University

Stanford, CA 94305, USA

{eberdahl|jos}@ccrma.stanford.edu

ABSTRACT

This paper presents a survey of various audio effects that can be physically applied to a rigidly-terminated vibrating string. The string's resonant behavior is described, and then the ability of active feedback control to "reprogram" the physics of the string is explained. Active damping, which is a direct result of applying classical control techniques, provides for an effect based on amplitude modulation (AM). Traditional electric guitar sustain techniques are elaborated upon, which suggest another approach for ensuring marginal stability of the system even in the presence of an arbitrary nonlinear and/or time-varying effect unit in the feedback loop. This approach involves placing a dynamic range limiter in the feedback loop and does not introduce significant harmonic distortion other than that due to the effect unit. The maximum RMS level of the system's output can be easily bounded if reasonable conditions are met by the dynamic range limiter. Finally, nonlinear and time-varying feedback control loops are applied experimentally to artificially induce frequency modulation (FM) at a low rate and AM at a high rate. These effects can be interpreted musically as vibrato and as a sort of resonant ring modulation, respectively.

1. INTRODUCTION

The study of *modal stimulation* is the study of actively controlling the vibrating structures in a musical instrument with the intent of altering its musical behavior [1]. Although it is possible to design an instrument such that most aspects are easily controllable, this study instead applies control engineering to a core component found in the guitar, the vibrating string. Many different controllers are conceivable, so we will focus on some of those relating to audio effects that have traditionally been popular among musicians. In developing some effects, we strive to preserve the musical qualities of the string that have historically been optimized during the evolution of stringed instruments. In developing other effects, we apply nonlinear and time-varying feedback loops to break free from the standard behavior of a vibrating string. In a sense, we are "reprogramming" the physics of the vibrating string. However, to simultaneously preserve the identity of the stringed instrument, we wish for the controller parameters to be as orthogonal as possible to the string length, which is adjusted by the musician during play.

2. PRIOR WORK

Various forms of actively-controlled musical instruments have been designed. For instance, the problem of indefinitely sustaining string vibration has long been investigated, especially in the framework of the electric guitar. Musicians have used acoustic feedback from loudspeakers to re-excite their electric guitar

strings [2]; however, due to the complex nature of the transfer functions involved and the nonlinear nature of the amplifiers, this approach has proven difficult to control precisely. The commercially-available Sustainiac has mitigated these problems somewhat using a phase-locked loop [3]. In a similar manner, Weinreich and Caussé have electromechanically induced the Helmholtz "stick-slip" bowing motion in a vibrating string without using an actual bow [4]. Besnainou applied active feedback control techniques to a violin, a snare drum, a pipe organ, and a marimba bar [1]. For example, he changed the damping time and pitch of a marimba bar using Proportional-Integral-Derivative (PID) control.

3. OVERVIEW

To simplify matters, this study involves only a single guitar string. It is actuated electromagnetically and sensed piezoelectrically to avoid direct feedback from the actuator to the sensor [5]. The system block diagram is shown in Figure 1. $g(t)$ is the impulse response of the string and changes whenever the musician alters the length or damping of the string. k_G is the loop gain, and $r(t)$ is the plucking excitation for the string produced by the musician.

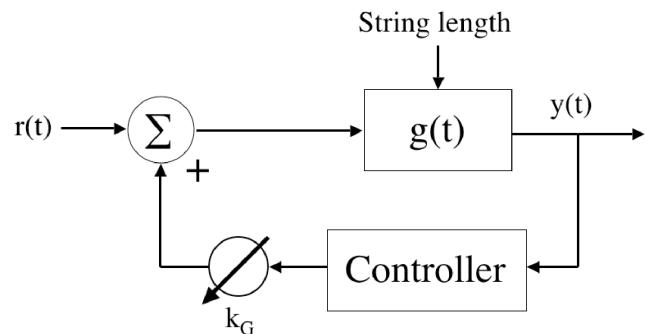


Figure 1: System block diagram for one string.

When no control is applied ($k_G = 0$), the string can be seen as a parallel bank of resonances at the partial overtone frequencies [5], which are approximately integer multiples of the lowest resonance frequency ("harmonics"). Applying white noise to the actuator causes all of the string's harmonics to be excited. On the other hand, large sinusoidal inputs can produce interesting behavior due to the guitar string's nonlinear nature. For example, even though a single input sinusoid can be tuned to only one of the harmonics, nonlinear interactions cause some of the energy to bleed over into the other harmonics. We can observe many of the same nonlinear effects in our guitar string that Roger Hanson has observed in a

brass harpsichord string [6].

4. ACTIVELY DAMPING THE STRING

4.1. Summary

In general, removing energy from a system is more difficult than adding energy, so active damping is a problem one should consider when placing actuators and sensors [7]. Figure 2 shows how the actuator (see Figure 2a) is placed near the string termination and the sensor (see Figure 2b) is placed at the string termination. This allows the musician to change the length of the string without greatly affecting the relative distance between the actuator and sensor.

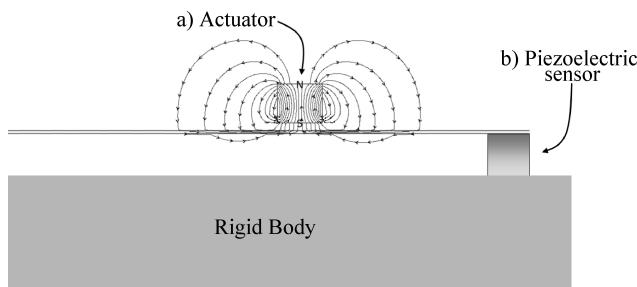


Figure 2: Sensor and actuator placement.

By exerting a force proportional to the integral of the string's displacement, the string can be damped quickly. This method damps the lowest harmonics the fastest. In particular, the change in the decay rate of a resonance (*i.e.*, the change in the inverse of the decay time τ) due to integral control is approximately inversely proportional to f^2 , where f is the natural frequency of the resonance.

$$\Delta(1/\tau) \propto \frac{k_G}{f^2} \quad (1)$$

4.2. Amplitude Modulation

Approximate amplitude modulation (AM) of the string's displacement can be induced by varying k_G at f_c Hz:

$$k_G(t) = g \cdot \cos(2\pi f_c t) + k_{GO} \quad (2)$$

The parameter g can be used to control the intensity of the effect. Choosing $k_{GO} > 0$ ensures that plucked tones decay over time. Guitar effect units that implement this effect for f_c in the range $\frac{1}{2} < f_c < 5$ Hz refer to it as *tremolo*.

5. SUSTAINING THE STRING'S VIBRATION

5.1. Using Integral Control

One might imagine that by carefully adapting $k_G(t)$ over time while restricting $k_G(t) < 0$, one could sustain a plucked string's vibration arbitrarily long. This is equivalent to introducing a dynamic range limiter (a signal processor that limits the RMS signal level flowing through it) into the feedback loop [8]. However, due to idiosyncrasies in the system, one harmonic mode always dominates, having a tendency to grow faster than the other modes. The limiter enforces a bound on the signal level, and so eventually

only the dominant harmonic remains. By this time, the sustained pluck no longer sounds like an instrument because it is devoid of rich harmonic content. This explains why the Sustainiac contains a phase-locked loop (PLL). After enough time following a pluck elapses, only the dominant harmonic will retain significant energy, and so the control algorithm may as well be simplified such that it excites the string with a sinusoid [3].

5.2. Using An Output Power-Limiting Nonlinearity

Electric guitar players have a solution for sustain in which they increase the output volume of their power amplifiers until the sound waves from the loudspeaker are powerful enough to excite the guitar strings, effectively sustaining their vibration. These systems tend to be quite complicated because they include delays, such as the air transmission delay T_d corresponding to the distance between the electric guitar player and the loudspeaker [9]. In addition, a real power amplifier has a maximum output power level, implying that the amplifier becomes nonlinear at large amplitudes. This means that at large amplitudes, the effective loop gain decreases until the energy in the system stops growing. Figure 3 shows the block diagram for the nonlinear guitar amplifier control system [10].

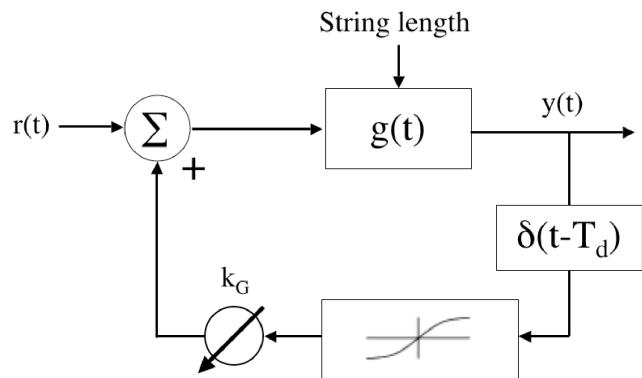


Figure 3: Nonlinear amplifier controller configuration.

Because the system is so nonlinear, the output does not decay to a pure sinusoid, but rather to a harmonic mixture, which may be more or less rich-sounding depending on the particular system parameters. This mixture is often desirable for artistic reasons; however, significant harmonic distortion changes the system's dynamic behavior drastically. In order to investigate other feedback effects more transparently, free from the side effects of the nonlinear amplifier method for ensuring marginal stability, an approach involving a dynamic range limiter is more palatable.

6. SUSTAINING AN ARBITRARY EFFECT

6.1. Using A Dynamic Range Limiter

The nonlinear amplifier element can be replaced by a dynamic range limiter, which adjusts the gain more slowly in order to avoid introducing significant harmonic distortion. Figure 4 depicts the system block diagram that allows for sustaining an arbitrary effect, where only the effect unit (depending on what type of signal processing it implements) can cause significant harmonic distortion.

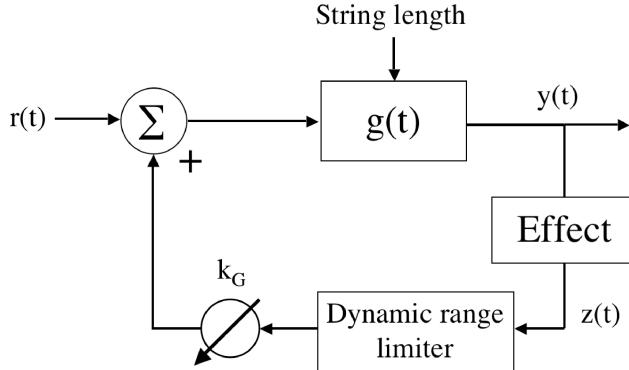


Figure 4: Configuration for sustaining an arbitrary effect.

6.2. Marginal Stability

In order to provide sustain, the controller should make the system marginally stable. The conditions for marginal stability are however more complicated than sufficient conditions for avoiding system instability, which will be derived here. For simplicity, we will assume that the dynamic range limiter adjusts the gain according to z_{RMS} , the RMS level of $z(t)$. We will further assume that the static compression curve of the (feedforward) limiter is upper-bounded by l and that the attack time for the limiter is essentially zero seconds. This means that the input to the string has a maximum RMS level of $r_{\text{RMS}} + k_G l$, where r_{RMS} is the RMS level of $r(t)$. Because the guitar string itself is passive, we have that $y_{\text{RMS}} \leq G_{\text{MAX}}(r_{\text{RMS}} + k_G l)$, where $G_{\text{MAX}} < +\infty$ is the maximum RMS gain of the actuator-string-sensor system represented by $g(t)$. r_{RMS} must surely be upper-bounded because a physical string can only be plucked or struck finitely hard, so y_{RMS} must also be upper bounded. Thus, we avoid system instability.

6.3. Frequency Modulation

The musical effect of applying frequency modulation (FM) to the harmonics of a guitar string is termed *vibrato* for carrier frequencies f_c roughly in the range $\frac{1}{2} < f_c < 5$ Hz. FM can be implemented using PID control of the string displacement; however, this method is quite sensitive to the particular decay rates of the harmonics of $g(t)$ [7]. Another method involves placing the cascade of an integrator and a vibrato effect unit in the feedback path. Without the integrator, the system would easily become unstable; the integrator ensures that string plucks decay quite quickly. A good compromise involves placing a dynamic range limiter in series with the effect unit (comprised of vibrato and integrator) as shown in Figure 4. Since the limiter will not be able to completely even out the volume level, some additional AM is expected.

Our implementation includes the vibrato circuit in Anderton's book [11] and the DOD Compressor Sustainer FX80B. The FX80B has an attack time of about 12ms at 500Hz, which is short enough to ensure marginal stability of the relatively well-behaved effect units that we use. That is to say, the parameters of the effect units vary slowly enough in time, and the effect units do not have wildly-varying gains due to nonlinearities. The sonogram of a guitar pluck with $f_0 \approx 250$ Hz is displayed in Figure 5. The decay of the string pluck is induced by gradually decreasing k_G to zero. It can be readily seen that the frequencies of the lowest

four harmonics vary periodically. Note that the energy at 120 Hz is measurement noise.

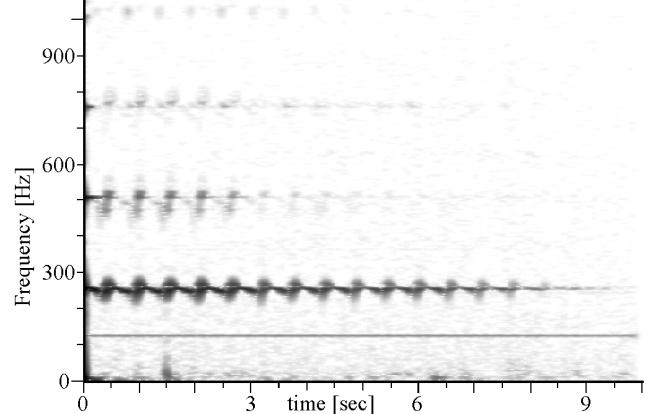


Figure 5: Sonogram of a guitar string pluck with vibrato.

6.4. Resonant Ring Modulation

When f_c for AM becomes large enough that a resonant frequency in a particular critical band of hearing splits by more than about 15 – 20% of the band's center-frequency, the human ear perceives the resulting sound quite differently [12]. Rather than perceiving "beating" or "roughness," the ear resolves the resonant frequencies independently, which may no longer be harmonic. The standard formulation of this effect has been termed *ring modulation* [13]. However, in this case the behavior is somewhat different due to the resonant properties of the string, which cause the resulting effect to sound less inharmonic than standard ring modulation. For instance, if the relationship between f_c and f_0 is such that the string would be driven at unnatural frequencies, then these frequencies are damped considerably. On the other hand, if f_c is chosen to be related by a ratio of small integers to the fundamental frequency f_0 of the string, sets of resonant frequencies may be achieved that equivalently could be mapped to a "pseudo-fundamental" subharmonic of f_0 . For these reasons, we refer to this effect as *resonant ring modulation*.

We implemented the modulation using a Max/MSP patch and a FirePOD sound interface, which introduced additional delay, but the limiter was still able to enforce marginal stability. Figure 6 shows a sonogram of the results of playing an ascending chromatic scale starting from $f_0 = 85$ Hz over one octave. The constant carrier frequency $f_c = 3 \times f_0 = 255$ Hz was chosen to be harmonically-related to f_0 . While the resonant frequencies corresponding to the natural harmonic series of the string increase as the scale progresses, additional resonant frequencies decrease in pitch due to the ring modulation. In this sense, this effect shares some characteristics with foldover and aliasing.

7. SUMMARY

This paper presented a survey of various audio effects that can be physically applied to a rigidly-terminated vibrating string using feedback control techniques. Damping, AM, tremolo, sustain, FM, vibrato, and resonant ring modulation were discussed, and experimental results were presented. The ability to apply many of these

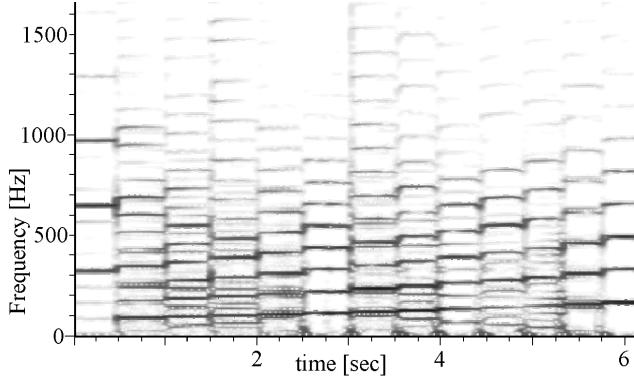


Figure 6: *Resonant ring modulated scale.*

nonlinear and/or time-varying effects relied on the placement of a dynamic range limiter in the feedback loop. The waveforms for the examples in this paper and other related examples can be found on the following website:

<http://ccrma.stanford.edu/~eberdahl/Projects/PhysicalEffects/>

8. ACKNOWLEDGEMENTS

We gratefully acknowledge Adrian Freed, Peter Lindener, Thomas D. Rossing, Bill Verplank, and CCRMA for their assistance. We thank the Stanford Graduate Fellowship program for supporting this work.

9. REFERENCES

- [1] C. Besnainou, “Transforming the voice of musical instruments by active control of the sound radiation,” in *Proc. Int. Symp. Active Noise and Vibration Control*, Fort Lauderdale, Florida, Dec. 1999.
- [2] G. Heet, “String instrument vibration initiator and sustainer,” 1978, u.S. Pat. 4,075,921.
- [3] G. Osborne and A. Hoover, “Sustainer for a musical instrument,” 1999, u.S. Pat. 5,932,827.
- [4] G. Weinreich and R. Caussé, “Digital and analog bows: Hybrid mechanical-electrical systems,” *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'86)*, Tokyo, Japan, vol. 2, pp. 1297–1299, Apr. 1986.
- [5] E. Berdahl, S. Backer, and J. O. Smith III, “If I had a hammer: Design and theory of an electromagnetically-prepared piano,” in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, Sept. 2005, pp. 81–84.
- [6] R. J. Hanson, J. M. Anderson, and H. K. Macomber, “Measurements of nonlinear effects in a driven vibrating wire,” *J. Acoust. Soc. Am.*, vol. 96, no. 3, pp. 1549–1556, 1994.
- [7] E. Berdahl and J. O. Smith III, “Active damping of a vibrating string,” in *Proc. Int. Symp. Active Noise and Vibration Control, Adelaide, Australia*, to appear in Aug. 2006.
- [8] J. Abel and D. Berners, “On peak-detecting and RMS feedback and feedforward compressors,” in *115th Conv. Audio Eng. Soc.*, New York, USA, 2003, preprint 5914.
- [9] C. R. Sullivan, “Extending the Karplus-Strong algorithm to synthesize electric guitar timbres with distortion and feedback,” *Computer Music J.*, vol. 14, pp. 26–37, 1990.
- [10] J. O. Smith III, *Physical Audio Signal Processing: For Virtual Musical Instruments and Audio Effects*, June 2006, [Online] <http://ccrma.stanford.edu/~jos/pasp/>
- [11] C. Anderton, *Electronic Projects for Musicians*, second edition ed. New York: Amsco Publications, 1980.
- [12] J. O. Smith III and J. Abel, “Bark and ERB bilinear transform,” *IEEE Trans. Speech and Audio Proc.*, vol. 7, no. 6, pp. 697–708, Nov. 1999.
- [13] C. Roads, *The Computer Music Tutorial*. Cambridge, MA: MIT Press, 1996.

TABLE LOOKUP OSCILLATORS USING GENERIC INTEGRATED WAVETABLES

Günter Geiger

Music Technology Group
 Universitat Pompeu Fabra, Barcelona, Spain
 ggeiger@iua.upf.edu

ABSTRACT

Table lookup oscillators form a basic building block of most software synthesis systems. Several of the classical digital and analog synthesis techniques require their use. The classical table lookup oscillator [1, 2] is commonly discussed regarding its amplitude error, but another source of error is mostly disregarded, although it influences the sound quality to a much higher degree, especially when running the oscillators at elevated frequencies. This error is due to the aliasing effect that occurs when shifting (or resampling) the original lookup table at higher rates.

One obstacle that has been encountered when trying to emulate classical analog synthesizers, is that the naive implementation of waveforms for subtractive synthesis generate the same aliasing frequencies. Over the years, several solutions for the problem of generating high quality band-limited waveforms for emulating classical analog synthesis have been proposed, solving the problem for specific waveforms. Never-the-less, the general question of how to manage aliasing behavior efficiently in general table lookup oscillators is still not resolved. This article reviews existing techniques for analog synthesis and tries to expand them for the general case.

1. INTRODUCTION

A general table lookup oscillator consists of two elements, the table, which specifies a sampled version of the oscillator function at a specific frequency (f_s/N , where N is the table size) and an interpolation scheme, which is used to resample the stored function in a way that it has a new duration $1/f$, the period of the desired oscillator. If N is the number of samples in the table, f_s the systems sampling frequency, f the oscillator frequency, a interpolation scheme I can be described in the following form.

$$x[n] = I\left(\frac{nf}{f_s N}\right) \quad (1)$$

The interpolation scheme is commonly a lowpass filter that acts on several samples of the lookup table. Interpolating the table values this way leads to a reduction of the amplitude error, but because of the fact that samples of the original table get actually skipped, it acts only as a post-resampling low-pass filter, and therefore it doesn't improve the signal to noise ratio of "real" against aliased frequency components.

Implementing a table lookup oscillator leads to aliasing effects, even if the original signal in the table does not show aliasing. This is noticeable when operating the oscillator at higher frequencies, as for a naive implementation frequencies beyond the Nyquist frequency get mirrored around $f_s/2$ and reappear as aliasing frequencies.

Figure 1 shows a spectrum of such an aliased waveform. Our goal is to devise an interpolation scheme that is efficient and general and allows for the implementation of general wave-table lookup oscillators with minimal aliasing effects.

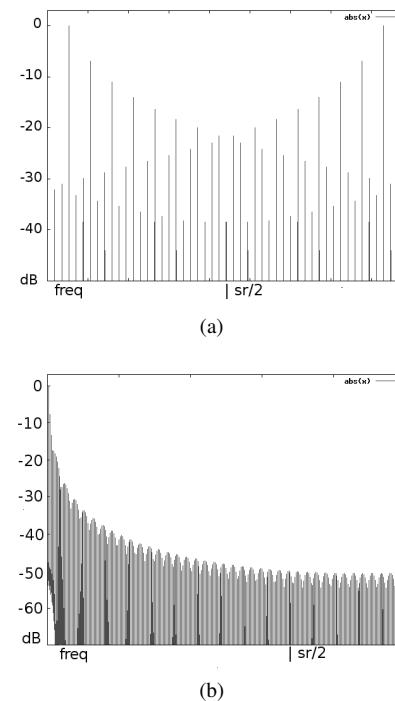


Figure 1: (a) Aliasing through resampling of a wave-table. Partials are mirrored at half the sampling frequency and disturb the original signal. (b) Spectral content of a sawtooth at frequency 44100/1024, the amplitude at half sampler-rate is about 54 dB lower.

Several algorithms are known to reduce aliasing effects, mainly targeted at the recreation of analog synthesizer waveforms such as square and sawtooth. We try to give a short overview of these techniques.

- Oversampling
- Precalculation of band limited signals using resampling and table interpolation.
- Synthesis based on band limited impulse trains (BLIT) [3]
- band limited step (BLEP), based on the idea of BLIT [4]
- Differentiated Parabolic Wave (DPW) [5]
- full-wave rectified sine waves [6]

1.1. Oversampling

The method used when oversampling is straight forward, best results give sinc based FIR anti-aliasing filters which are used for the resampling step. This method is fairly expensive because of the high oversampling factors but it can be generally applied with high quality.

Prefiltering the lookup table in order to be able to shift to higher frequencies is a special case of the oversampling method. In this case the oversampling factor is determined by the length of the table and by the target frequency. The wave-table can be seen as such an oversampled representation of the function. Low-pass filtering the whole table at the desired frequency ($k f_s / N$) where $k = f_{\text{target}} N / f_s$ is the pitch shift factor and resampling [7] the resulting table in order to achieve a higher frequency leads to a high quality implementation. N samples have to be filtered, regardless of the resulting period which could be just a few samples for high frequencies. Therefore the method gets more expensive the higher the frequency. Oversampling lookup tables this way yields oscillators of constant high quality, but the quality has to be paid with efficiency losses at higher frequencies.

1.2. Precalculating band-limited wave-tables

For the general case of anti-aliased waveforms, this method can either be implemented by analytically computing the waveforms at different frequencies, by constructing their Fourier series with varying bandwidth, or by using the oversampling method as a pre-calculation step. It helps to decrease the performance problems of the oversampling method but increases memory consumption considerably [8].

1.3. Band Limited Impulse Trains

This method achieves to synthesize some of the analog waveforms by the interpolation of a band-limited Impulse Train, computed by the following formula:

$$x[n] = \frac{\sin(2\pi f/f_s)}{f/f_s} \quad (2)$$

The BLIT method is effective and can be computed without additional memory, but being an analytic method it is only applicable for specific waveforms.

1.4. The minBLEP method

An enhancement of the BLIT method is the band limited step function. In this method the discontinuity in the analog waveforms such as the sawtooth gets precalculated as such and stored in a wave-table. The advantage is a higher stability of the resulting waveform, the implementations efficiency is dependent on the frequency that has to be synthesized.

1.5. Differentiated parabolic wave

This method does not try to synthesize a perfect band-limited signal, but it allows some aliasing to occur, and minimizes it in those regions where it is most likely to be noticed by a listener. The method describes the synthesis of a sawtooth by integration of a parabolic wave, but it can be expanded to the general case. The method, being analytic can be expanded to the general numeric case as we will see later.

1.6. Full-wave rectified sine wave

Lane [6] proposes an efficient implementation of the sawtooth waveform based on full-wave rectifying a sine wave and filtering the resulting signal with two digital filters, one as a 1 pole filter (with a zero at DC) and a higher order butterworth low-pass. The goal of the method is reduction of aliasing and not its complete suppression. The methods proposed are very much focused on generating specific waveforms and are hard to apply to the general case.

2. THE INTEGRATED WAVE-TABLE

2.1. DPW analysis

We might take a closer look at the differentiated parabolic wave method. Besides the relatively expansive oversampling method, the DPW method is a candidate to be expanded for general waveforms, as its idea does not rely on dealing directly with the properties of specific waveforms, but on the low-pass characteristics of the integration.

First we will try to take a closer look at the DPW method in order to understand how it can be applied in the general case. We start from a sawtooth wave, described by $x[n] = nf/f_s \bmod 1$, from Fourier analysis we know that this waveform has harmonics which decay at a ratio of $1/n$, which means 6dB per octave. To calculate the amplitude of the nearest aliased bin we use the ad-hoc formula

$$A_a(f_0) = 20 * \log \left(\frac{f}{f_s - f_0} \right) \quad (3)$$

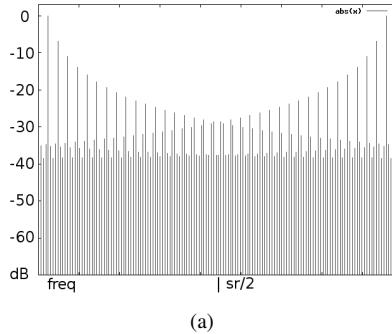
with f_0 being the fundamental frequency of our generated sawtooth wave, A_a the amplitude in dB of a aliased bin at f_0 . We assume the closest aliased bin to appear in the first octave above f_0 . For example, assuming a frequency f of 1323 Hz, and considering the mirror effect of aliasing at 22050 Hz the closest aliased bin appears at 1764 Hz with an RMS value 30 dB below the fundamental. Depending on the frequency of the fundamental these aliasing frequencies are either directly audible or produce roughness in the signal through interference. When sweeping the frequency of the oscillator one can hear the additional disturbing effect that the aliased components are moving in the opposite direction of the sweep.

The DPW method creates an analytically integrated waveform, hence filtering the sawtooth with an integrator, achieving an additional attenuation of 6 dB/octave, resulting in an amplitude of -60 dB for the aliasing bin in the previous example.

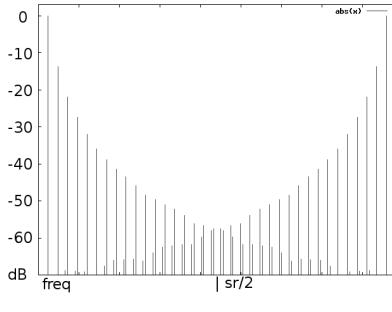
In order to restore the original signal, in DPW the generated waveform gets sampled and differentiated. As the amplitude changes for the integrator and differentiator are linearly dependent on the frequency, the aliasing effect gets reduced, because the attenuation took place at a higher frequency, whereas the boosting trough differentiation takes place at the lower, aliased frequency, resulting in an overall diminution of aliasing. This effect is stronger the bigger the difference of the aliasing frequency and the Nyquist frequency.

2.2. The generic integrated wave-table

In order to demonstrate the effectiveness of the numerical integration Figure 2 shows a sawtooth wave stored in a lookup table and the spectral content of its integrated wave. Its not astonishing that the numerical method shows a similar improvement in terms



(a)



(b)

Figure 2: (a) Generating a sawtooth waveform from a lookup table of size $N = 1024$, frequency f_0 is 1200 Hz. (b) Spectral content of a integrated lookup table of the same sawtooth.

of suppression of aliasing than the analytic generation of the sawtooth used in DPW.

Figure 3 (a) shows the extreme case of a randomly generated function used as the basis of an oscillator. As can be seen the aliasing effect with this function is such, that the original fundamental frequency of 1200 Hz is not distinguishable anymore. Instead the frequency bins are more or less equal in amplitude and only show a harmonic pattern because of the mapping of aliased bins onto already existing ones, reflecting the real periodicity of the random wave-table, which is a common divisor of the Nyquist frequency and the oscillators frequency.

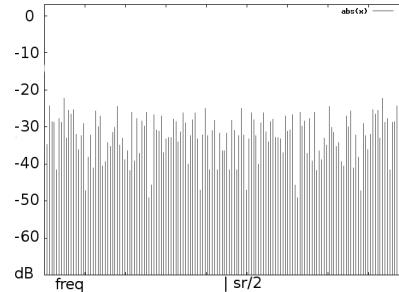
The integrated wave-table method can be compared to the method of using several band-limited wave-tables, the difference is that the low-pass function is not used to remove aliasing completely, but to diminish it over all frequencies. This way one low-pass filtered wave-table is enough.

2.3. Restoring the original wave-table

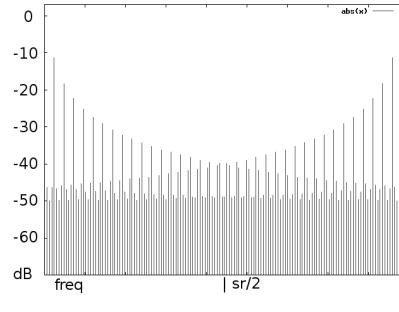
Using integrated wave-tables, the target waveform has to be restored by differentiating the synthesized waveform. Unlike the integration step this has to be done in real-time. As differentiation boils down to a simple subtraction in the digital domain, this method is very efficient.

The method will be better, the farther away from the Nyquist frequency the aliased components lie. The differentiation has the following properties as a function of frequency:

$$H(k) = kX(k) \quad (4)$$



(a)



(b)

Figure 3: (a) Generating a random periodic waveform from a lookup table of size $N = 1024$, frequency f_0 is 1200 Hz. (b) Spectral content of a integrated random lookup table of the same function, the fundamental frequency of 1200 Hz can be recognized.

Whereas the components close to the Nyquist frequency will be fully restored to their original value, the amplitude will fall off by $1/(n_{\text{Nyquist}} - n)$, therefore achieving good results at lower frequencies where aliasing effects are more disturbing.

Figure 4 shows the spectral content of the differentiated signal, based on the integrated sawtooth. Compared with result with Figure 2 (a) the integrated wave-table method exposes a considerably improved SNR, where the aliasing noise floor of -35 – 40 dB is only visible at values lower than -50 dB and only for frequencies higher than about 4000 Hz.

The extreme case of a white noise based integrated wave-table is shown in Figure 4 (b). Although the noise floor is still very high in this case (-20 dB), at least the fundamental frequency is recognizable.

3. CONCLUSIONS

We presented a generic, efficient method to reduce aliasing artifacts in generic table lookup oscillators. By storing the desired wave-table function at a low frequency (long table) in its integrated form we achieve a considerable improvement of the signal to noise ratio. The method does not eliminate aliasing completely, but it improves the quality of table lookup synthesis considerably, and is directly applicable to implementations of oscillators.

We show that the problem of aliasing is not limited to the simulation of analog synthesis, but is a general problem of the table lookup oscillator, with a negative influence superior to the better investigated problem of sample accuracy in interpolation schemes.

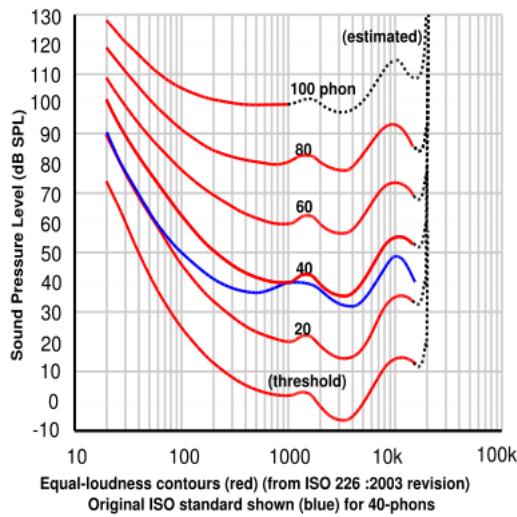


Figure 5: Curves of equal loudness (diagram from wikipedia) suggest that aliasing that occurs above 15 kHz only has a limited influence upon the signal frequencies between 200 Hz and 8 kHz are critical.

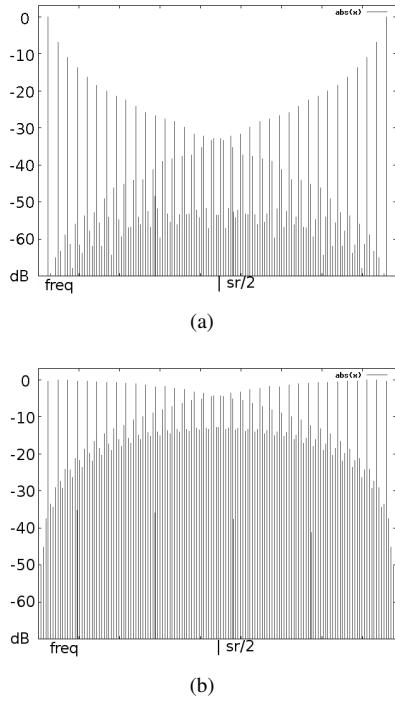


Figure 4: (a) The spectrum of the resynthesized sawtooth, based on the integrated wave-table method. (b) The hardest case: a resynthesized noise wave-table.

Our method is based on the Differentiated Parabolic Wave (DPW) method, but it expands it to the general case, and therefore we propose to name it generic integrated wavetable (GIW)

method, reflecting how the wavetable is stored in memory (in its integrated form) than how it is resynthesized (by differentiation).

We conducted our work without any interpolation (except simple rounding) on wave-tables of a length of 1024 values in order to discard influences of interpolation and the resulting low-pass filtering. This interpolation scheme amounts to a maximal noise level due to amplitude errors of about -60 dB [9]. In a high quality implementation the table size has to be increased and a classic interpolation scheme has to be used in order to avoid amplitude errors.

4. FUTURE WORK

Although the generic integrated wave-table is a considerable improvement in the implementation of table lookup oscillators, the quality of the resulting signal might still be improved. The most obvious path for improvement is to work with higher sample rates. In this case a 6 dB SNR improvement would require the double of calculations.

We didn't make quantitative comparisons with other methods yet, although we consider it more important to check our method with listening tests, as pleasing the human ear is the final goal of our efforts.

Other methods that solve the aliasing problem, like precalculated waveforms still have to be investigated and compared to this method in terms of accuracy and speed.

5. ACKNOWLEDGEMENTS

Thanks go to Vesa Välimäki for his idea about differentiated parabolic waveforms [5], which was the main inspiration for this work.

6. REFERENCES

- [1] C. Roads, *The Computer Music Tutorial*. Cambridge, MA.: MIT Press, 1996.
- [2] F. R. Moore, *Elements of computer music*. Englewood Cliffs, NJ.: Prentice Hall, 1990.
- [3] T. Stilson and J. Smith, "Alias-free digital synthesis of classic analog waveforms," in *Proc. Int. Comp. Music Conf. (ICMC'96)*, Hong Kong, 1996, pp. 332–335.
- [4] E. Brandt, "Hard sync without aliasing," in *Proc. Int. Comp. Music Conf. (ICMC'01)*, Havana, Cuba, 2001, pp. 365–368.
- [5] V. Välimäki, "Discrete-time synthesis of the sawtooth waveform with reduced aliasing," *IEEE Sig. Proc. Letters*, vol. 12, no. 3, pp. 214–217, march 2005.
- [6] J. Lane, D. Hoory, E. Martinez, and P. Wang, "Modeling analog synthesis with DSPs," *Computer Music J.*, vol. 21, no. 4, pp. 23–41, may 1997.
- [7] J. O. Smith, "Digital audio resampling home page," [Online] <http://www-ccrma.stanford.edu/~jos/resample/>, Jan. 28 2002.
- [8] J. Wright, "Synthesising band limited waveforms using wavetables," [Online] <http://www.musicdsp.org/files/bandlimited.pdf>, Retrieved June 29th, 2006.
- [9] M. S. Puckette, "Theory and techniques of electronic music," [Online] <http://www-crcr.ucsd.edu/~msp/>, 2006.

ERROR COMPENSATION IN MODELING TIME-VARYING SINUSOIDS

Wen Xue, Mark Sandler

Centre for Digital Music, Dept. Elec. Eng.

Queen Mary, Univ. of London, UK

{xue.wen|mark.sandler}@elec.qmul.ac.uk

ABSTRACT

In this article we propose a method to improve the accuracy of sinusoid modeling by introducing parameter variation models into both the analyzer and the synthesizer. Using the least-square-error estimator as an example, we show how the sinusoidal parameters estimated under a stationary assumption relate to the real nonstationary process, and propose a way to reestimate the parameters using some parameter variation model. For the synthesizer, we interpolate the parameters using the same model, with the phase unwrapping process reformulated to adapt to the change. Results show that the method effectively cuts down the systematic error of a conventional system based on a least-square-error estimator and the McAulay-Quatieri synthesizer.

1. INTRODUCTION

Sinusoid modeling expresses a pitched sound as the linear combination of time-varying sinusoids. This involves an analyzer and a synthesizer. The analyzer does sinusoidal parameter estimation and the synthesizer rebuilds the signal from the estimated parameters. However, since parameter estimators rarely consider the in-frame dynamics, most current sinusoid modeling systems carry an inborn error even if the signal being analyzed strictly matches the sinusoid model. This error is often ignorable when a clean residue is not crucial, but becomes significant when one tries to subtract a sinusoid from the original. The systematic error is a combination of an analyzer error and a synthesizer error. In section 2 we give a short review of sinusoid modeling. Section 3 explains how the analyzer error occurs and how it can be mended. Section 4 gives an example on what we can do to cut down the synthesizer error.

2. SPECTRAL MODELING SYNTHESIS

The complete sinusoid modeling was first presented by McAulay and Quatieri [1] and later refined by several. The parameter estimator has been improved by more accurate parameter estimation methods, such as those summarized in [2]. Partial tracking has been improved by using more natural tracking methods to connect spectral peaks in consecutive frames [3, 4]. The model itself has been extended to involve non-stationary noise [5]. On the synthesizer side, the reconstruction process proposed in [1] remains the same. Suppose we have a single time-varying complex sinusoid

$$x(n) = a(n)e^{j\varphi(n)}, \text{ where } \varphi(n) - \varphi(n-1) = 2\pi \int_{n-1}^n f(t) dt \quad (1)$$

where n is a sampled version of the continuous time, t . The instantaneous amplitude $a(n) > 0$ and frequency $f(t) > 0$ are slow-varying functions of time. The complete sinusoid model is constructed by summing up multiple sinusoids (*partials*) in the form of (1). For each sinusoid, the analyzer estimates its amplitudes,

frequencies and phase angles at a set of points n_0, n_1, \dots, n_F , $F + 1$ being the total number of measurements. We denote the estimates as $\hat{a}(n_0), \hat{f}(n_0)$, etc. We use the term *estimator error* to refer to the difference between the estimates and their true values, such as between $\hat{a}(n_1)$ and $a(n_1)$. The analyzer also includes a peak tracker which forms sinusoids from local spectral peaks. When the signal has two or more partials, there may also be a peak tracker error.

The McAulay-Quatieri synthesizer connects two consecutive measure points with a sinusoid segment by interpolating the parameters. As the true parameter variation laws usually do not coincide with the interpolation laws, this interpolation introduces a synthesizer error.

Using all the $F + 1$ measured parameter sets, we can reconstruct a sinusoid covering the interval $[n_0, n_F]$. We denote the rebuilt signal $\hat{x}(n)$, $n_0 \leq n \leq n_F$, and define the *relative modeling error* as

$$e = \frac{1}{n_F - n_0 + 1} \sum_{n=n_0}^{n_F} \left\| \frac{\hat{x}(n) - x(n)}{a(n)} \right\|^2 \quad (2)$$

in which the difference of corresponding values is normalized by the instantaneous amplitude. When the signal being analyzed strictly fits the model, this final error (2) is a combined result of estimator, peak tracker, and synthesizer errors.

3. PARAMETER REESTIMATION

Most sinusoid modeling systems use a frame-based spectral analyzer. For each frame, the amplitude and frequency are assumed constant during the whole frame and calculated from the short-time Fourier transform, along with a phase angle [2]. In most cases the results are intuitively assigned to be the instantaneous parameters at the frame centre.

Let N be the frame width. As any estimate is calculated from N data points, it depends on N instantaneous parameter sets rather than equals its instantaneous value at $N/2$, the frame centre. Instead of using the estimates directly, we try to recover the instantaneous values from a sequence of multiple estimates, using parameter variation information derived from the estimates. That is, given a sequence of parameter estimates, we try to find a sinusoid in the form of (1) which, when fed into the analyzer, generates the given estimates. To do this we need to study the quantitative relation between the true parameters and the frame-based estimates. Naturally the relation depends how the parameters are estimated.

In this article we use a least-square-error (LSE) estimator for parameter measurement, which minimizes the square error between the spectrum of a pure sinusoid and that of a narrow-band signal. In short, for any pure sinusoid with parameter set (a, f, φ) , we can calculate its spectrum $ae^{j\varphi} H_f(k)$. Given a narrow-band spectrum $X(k)$, the LSE estimator finds the frequency \hat{f} that

maximizes the inner product $\langle X, H_f \rangle$ by the norm, which can be shown to yield the least square error. The amplitude and phase angle are then given by $\hat{a}e^{j\hat{\varphi}} = \langle X, H_f \rangle / \|H\|^2$, where $\|H\|^2$ is a normalizing factor determined by the window function. In this article the window function is always real, symmetric and lowpass.

Let our data frame be

$$x(n) = a(n)e^{j(\varphi_c + 2\pi \int_{N/2}^n f(t)dt)}, \quad 0 \leq n < N \quad (3)$$

Let the window function be $w(n)$, $\varphi_{mn} = 2\pi \int_m^n f(t)dt$, $\varphi_n = \varphi_{\frac{N}{2}, n}$, then the short-time Fourier transform of $x(n)$ is

$$X_k = \sum_{n=0}^{N-1} w(n)a(n)e^{j(\varphi_c + \varphi_n - 2\pi k \frac{n}{N})}, \quad 0 \leq k < N \quad (4)$$

The Fourier transform of a pure zero-phase unit sinusoid is

$$H_k = \sum_{n=0}^{N-1} w(n)e^{j2\pi(f_0(n-\frac{N}{2}) - \frac{kn}{N})}, \quad 0 \leq k < N \quad (5)$$

Define $b(n) \equiv w(n)^2 a(n)$, $\Delta\varphi_{mn}(g) = \varphi_{mn} - 2\pi(n-m)g$, we calculate the square norm of $\langle X, H \rangle$

$$\begin{aligned} \|\langle X, H \rangle\|^2 &= N^2 \left(\sum_{n=0}^{N-1} b(n)^2 \right. \\ &\quad \left. + 2 \sum_{n=1}^{N-1} \sum_{m=0}^{n-1} b(n)b(m) \cos \Delta\varphi_{mn}(f_0) \right) \end{aligned} \quad (6)$$

To maximize the above we set its derivative regarding f_0 to 0:

$$\begin{aligned} \frac{d\|\langle X, H \rangle\|^2}{df_0} &= N^2 2\pi \sum_{n=1}^{N-1} \sum_{m=0}^{n-1} (n-m)b(n)b(m) \sin \Delta\varphi_{mn}(\hat{f}) \\ &= 0 \end{aligned} \quad (7)$$

where \hat{f} is the frequency that maximizes $\|\langle X, H \rangle\|^2$, i.e. the LSE estimate. Let $w_{mn} = (n-m)w(m)^2w(n)^2$. After some math we get

$$\hat{f} = \frac{\sum_{l=0}^{N-2} \eta_l(\hat{f}) \int_0^1 f(l+t)dt}{\sum_{l=0}^{N-2} \eta_l(\hat{f})} \quad (8)$$

where $\eta_l(g) = \sum_{n=l+1}^{N-1} \sum_{m=0}^l w_{mn} a(n)a(m) \text{sinc} \frac{\Delta\varphi_{mn}(g)}{\pi}$, and the sinc function $\text{sinc}(x) = \frac{\sin \pi x}{\pi x}$. The amplitude and phase estimates \hat{a} and $\hat{\varphi}$ are given by

$$\hat{a}e^{j\hat{\varphi}} = \frac{e^{j\varphi_c} \sum_{n=0}^{N-1} w(n)^2 a(n) e^{j2\pi \left(\int_{N/2}^n f(t)dt - (n-\frac{N}{2})\hat{f} \right)}}{\sum_{n=0}^{N-1} w(n)^2} \quad (9)$$

Eq. (8) implies that the frequency estimate is a weighted average of the instantaneous frequency over the frame. The weights depend on the window function, instantaneous amplitudes, and the instantaneous frequencies themselves. In particular, if the frequency remains constant, it equals the estimate regardless of the amplitude.

3.1. Pure Amplitude Change

The easiest case of parameter dynamics is amplitude change while the frequency stays constant. The signal can be written as

$$x(n) = a(n)e^{j(2\pi f n + \varphi)} \quad (10)$$

As stated above, the estimated frequency shall equal f . An immediate result is that the phase estimate is accurate as well. The amplitude estimate can be easily expressed using (9):

$$\hat{a} = \frac{\sum_n a(n)w(n)^2}{\sum_n w(n)^2} \quad (11)$$

The symmetry if (11) implies that only the even-symmetric part of $a(n)$, regarding the frame centre, contributes to the estimate. Pure amplitude change happens to stable-pitch sound sources such as piano or oboe. If a sinusoid is assumed to have a constant frequency, we can re-estimate the centre amplitude using (11).

3.2. Frequency and Amplitude Change

Instantaneous frequency change rarely happens without accompanying amplitude change. The general form of such a sinusoid is

$$x(n) = a(n)e^{j(\varphi_c + 2\pi \int_{N/2}^n f(t)dt)} \quad (12)$$

The frequency estimate is a weighted average of the instantaneous frequency during the frame. Calculation shows that when the amplitude is constant, the averaging weights η_l is symmetric and slightly more concentrated towards the frame centre than the square of the window function.

Like the constant frequency case, when the amplitude is constant during a frame, only the even part of $f(t)$ contributes to the frequency estimate. This implies that the frequency measurement is exact for a linear chirp. Considering amplitude change, a more general result is that the even part of $f(t)$ contributes to the frequency estimate when the even part of $a(n)$ is non-zero, and the odd part of $f(t)$ contributes to the frequency estimate when the odd part of $a(n)$ is non-zero.

3.3. Reestimation of Parameters

We formulate the error compensation task as follows: given $F+1$ measure points, say, n_0, n_1, \dots, n_F , and $F+1$ parameter measurements $(\hat{a}_m, \hat{f}_m, \hat{\varphi}_m)_{m=0,1,\dots,F}$, find a series of parameters (a_m, f_m, φ_m) , $m = 0, 1, \dots, F$, that generate the estimates through the analyzer.

Our key equation (8) involves $2N-1$ unknowns, i.e. $a(n)$, $0 \leq n < N$, and $\int_0^1 f(l+t)dt$, $0 \leq l < N-1$. This is too many to recover from the given estimates. We introduce parametric models $f(t, \Sigma_f)$ for frequency and $a(n, \Sigma_a)$ for amplitude variation, so that all the unknowns can be calculated from a sequence of $F+1$ parameter sets. Denote such a sequence $\mathbf{P} = \{(a_m, f_m, \varphi_m)^P | m=0, \dots, F\}$, and the corresponding variation model parameters $\Sigma(\mathbf{P})$. (8) and (9) relates a parameter set \mathbf{P} to its estimate $\hat{\mathbf{P}}$ in the form of

$$\hat{\mathbf{P}} = \mathcal{P}(\Sigma(\mathbf{P})) \quad (13)$$

That is, given any \mathbf{P} , we can estimate $\Sigma(\mathbf{P}) = \{\Sigma_f(\mathbf{P}), \Sigma_a(\mathbf{P})\}$, then calculate $\hat{\mathbf{P}}$ by (8) and (9). The reestimation task is just the opposite: given $\hat{\mathbf{P}}$ we try to find the original \mathbf{P} . We rewrite (13) as

$$\mathbf{P} = \hat{\mathbf{P}} - \mathcal{P}(\Sigma(\mathbf{P})) + \mathbf{P} \quad (14)$$

A recurrent method for solving (13) is derived from (14) as follows:

- 0° Define distance function $D(\mathbf{P}_1, \mathbf{P}_2)$, convergence threshold ε , and the maximal number of iterations MAX; set $\mathbf{P}_0 = \hat{\mathbf{P}}$;
- 1° for $n = 1, 2, \dots, \text{MAX}$, do 2°-5°;
- 2° estimate the variation parameters $\Sigma(\mathbf{P}_{n-1})$;
- 3° calculate $\hat{\mathbf{P}}_{n-1} = \mathcal{P}(\Sigma(\mathbf{P}_{n-1}))$ using (8) and (9);
- 4° if $D(\hat{\mathbf{P}}, \hat{\mathbf{P}}_{n-1}) < \varepsilon$, output \mathbf{P}_{n-1} , return;
- 5° calculate $\mathbf{P}_n = \hat{\mathbf{P}} - \hat{\mathbf{P}}_{n-1} + \mathbf{P}_{n-1}$;
- 6° output \mathbf{P} with non-convergence tag.

We ignore phase estimates during this stage to avoid phase wrapping problem. The phase angle can be reestimated as

$$\varphi = \hat{\varphi} - \arg \sum_{n=0}^{N-1} w(n)^2 a(n) e^{j2\pi \left(\int_n^{\frac{N}{2}} f(t) dt - (n - \frac{N}{2}) \hat{f} \right)} \quad (15)$$

which is an immediate result of (9).

3.4. Test Examples

We use cubic splines to model amplitude and frequency variation. A cubic spline is a piecewise trinomial with continuous 1st and 2nd derivatives. Accordingly, the phase angle function is a quartic polynomial.

Let $F + 1$ be the total number of estimates (frames). Parameters are estimated at points $0, N/2, \dots, NF/2$. The cubic spline fills the gaps between 0 and $NF/2$. However, the reestimation of the first point 0 requires half a frame before zero. In this case we extrapolate the spline half a frame beyond its effective interval. The same is done to the last frame.

The test signals are synthesized sinusoids for which we have the "true" instantaneous parameters at any point. The error between the true parameter set \mathbf{P} and an estimate $\tilde{\mathbf{P}}$ is defined as

$$ERR(\tilde{\mathbf{P}}, \mathbf{P}) = \frac{1}{F+1} \sum_{m=0}^F \sum_{n=0}^{N-1} w(n)^2 \left(\tilde{a}_m \cos(\tilde{\varphi}_m + 2\pi \tilde{f}_m(n - N/2)) - a_m \cos(\varphi_m + 2\pi f_m(n - N/2)) \right)^2 / \frac{a_m^2}{2} \sum_{n=0}^{N-1} w(n)^2 \quad (16)$$

We run tests on three types of signals: exponential-decay amplitude with constant frequency, constant amplitude with sinusoid-modulated frequency, and exponential-decay linear chirp. The first two are simplified cases of real sounds, and the last is included to represent combined amplitude-frequency change. Reestimated results are compared with the original. Tests show that the error hardly depends on the absolute signal level, frequency or phase. In all the tests we set central frequencies of three concurrent sinusoids to 0.151, 0.251, 0.351 (the Nyquist frequency being 0.5), and phase angles to 0. 11 frames are extracted with a Hann window of size 1024. So $F = 10$. The maximal iteration count is set at 25.

1. Exponential amplitude The main variable in this test is the rate of amplitude decay, defined as $\lambda_a = \frac{a(0)}{a(N/2)}$, i.e. the amplitude drops by a factor of a per hop size. In the test λ_a varies between 1 and 4. The results are given in Figure 1(a), in which line ① is the error calculated for the LSE estimate, and line ② from the reestimates. Line ② lies below ① between $\lambda_a = 1$ and $\lambda_a = 3$, an improvement of 15–25 dB for most of the interval. When $\lambda_a > 3$, the cubic spline can no longer keep up with the global signal dynamics (e.g. 95 dB for $\lambda_a = 3$), and the reestimation fails. Line ③ gives the result we get after one iteration. It is shown that most improvement is achieved by the first iteration of up to 25.

2. Sinusoid-modulated frequency The frequency modulator has three parameters: amplitude a_M , frequency f_M and phase angle. We fix the modulator phase to 0 at time 0, f_M to 0.2 and 0.33 per frame, and vary a_M between 0 and 10 bins. The maximal frequency change rate is $2\pi a_M f_M$ bins per frame. Results are given in Figures 1(b) and 1(c). We see that the first iteration is normally enough when the modulation is small, but more are needed when the modulation is high.

3. Exponentially decreasing linear chirp In this example we have two variable parameters: the linear frequency change and exponential amplitude change rates. We measure the frequency

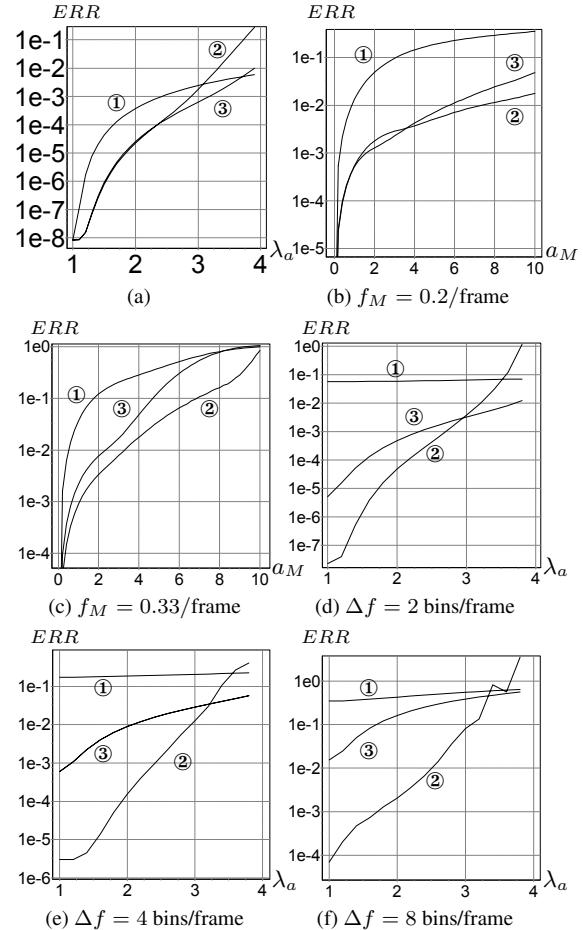


Figure 1: Testing analyzer error. (a) exponential amplitude; (b)(c) modulated frequency; (d)(e)(f) exponential-decay linear chirp.

change rate Δf in bins per frame, and the amplitude change decay rate λ_a as in the first example. We let Δf be 2, 4, 8, and vary λ_a between 1 and 4. Results are given in Figures 1(d)–1(f). Again we see that the reestimation fails for high decay rates. It is also shown that the more the frequency varies, the less the first iteration contributes to the total improvement.

4. RESYNTHESIS

We consider only the resynthesis *with phase*. As the amplitude and frequency variations are modeled with cubic splines, it is natural to use this model in resynthesis instead of the standard interpolation method in [1]. The sample-wise amplitudes can be derived from the cubic spline interpolation. For the phase angles, we can compensate for the model-to-measurement difference as follows.

Let the cubic spline frequency 0 and N be

$$f(t) = a t^3 + b t^2 + c t + d \quad (17)$$

and the phase estimate be $\varphi(0) = \varphi_0, \varphi(N) = \varphi_N$. The model phase function can be written as

$$\tilde{\varphi}(t) = 2\pi \left(\frac{a}{4} t^4 + \frac{b}{3} t^3 + \frac{c}{2} t^2 + d t \right) + \varphi_0 \quad (18)$$

We write the final interpolation function φ as

$$\varphi(t) = \tilde{\varphi}(t) + \theta(t) \quad (19)$$

that satisfies

$$\begin{aligned}\varphi(0) &= \varphi_0, \varphi(N) = \varphi_N + 2k\pi, \\ \varphi'(0) &= \tilde{\varphi}'(0), \varphi'(N) = \tilde{\varphi}'(N)\end{aligned}\quad (20)$$

where $k \in \mathbb{Z}$ is chosen to minimize $\theta(t)$. We write (20) in term of $\theta(t)$:

$$\begin{aligned}\theta(0) &= 0, \theta(N) = \varphi_N - \tilde{\varphi}(N) + 2k\pi, \\ \theta'(0) &= 0, \theta'(N) = 0\end{aligned}\quad (21)$$

The four conditions of (21) suggests the use of a trinomial for $\theta(t)$, i.e. $\theta(t) = pt^3 + qt^2 + rt + s$. By solving (21) we get

$$\begin{aligned}p &= \frac{-2}{N^3}d(k), q = \frac{3}{N^2}d(k), r = s = 0, \\ d(k) &\equiv \varphi_N - \tilde{\varphi}(N) + 2k\pi\end{aligned}\quad (22)$$

To minimize $\theta(t)$ we choose the integer k that minimizes $d(k)$, which is simply the integer nearest to $(\tilde{\varphi}(N) - \varphi_N)/2\pi$. This is very similar to the phase unwrapping process in [1], which can be regarded as a *linear spline* version.

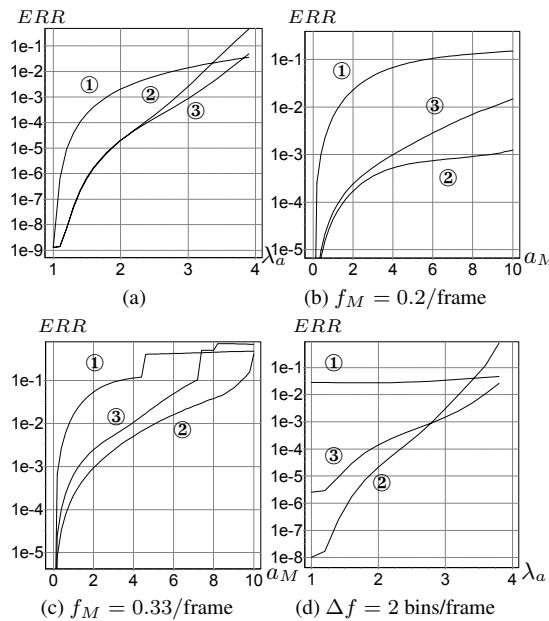


Figure 2: Testing synthesizer error. (a) exponential amplitude; (b)(c) modulated frequency; (d) exponential-decay linear chirp.

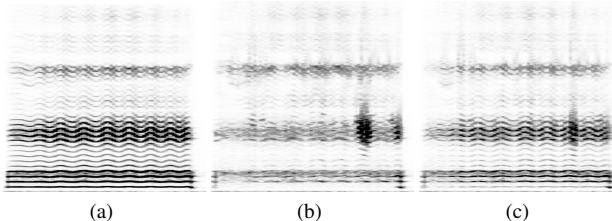


Figure 3: Testing vocal vibrato. (a) original; (b) cubic spline modeling; (c) LSE-MQ modeling.

4.1. Test Examples

In this section we compare the synthesizer error to the original one in [1] for signals used in section 3.4, plus a real recorded excerpt. For the synthesized sounds an extra result is given for using the [1] resynthesizer the true parameters to give some idea of pure synthesizer error. Errors are evaluated using equation (2).

1. Tests on synthesized signals The results are given in Figure 2(a)-2(d) for the first four test settings in Figures 1. The synthesizer errors show similar development trends to the analyzer errors in Figures 1. Both the absolute value and the measured improvements are slightly better than the analyzer error values, thanks to the use of interpolation.

2. Test on recording For this example we take a recording of soprano vibrato from the RWC database [6]. The spectrogram of the original is given in Figure 3(a). Figure 3(b) is the residue we get by subtracting the resynthesized result using cubic spline modeling, while Figure 3(c) is the result derived from the old system. The new system outperforms the old one except for the part where the residue shows some transient. Averaging over frames, the new modeling achieves 22 dB SNR, compared to 16 dB for the old one. We also see that the new residue is less sinusoidal, or more noise-like, than the old one.

5. CONCLUSION

In this article we have addressed the problem of estimating parameters from non-stationary sinusoids. Rather than directly using the results obtained using a stationary assumption, we propose to reestimate the parameters incorporating frequency and amplitude variation models. Although we have chosen an LSE estimator and a cubic spline model for our system, this reestimation framework is open to many other methods. For the resynthesizer part we have reformulated the phase unwrapping process in the context of cubic spline interpolation. Tests show the improvement in the accuracy of parameters, as well as in the resynthesized signals.

6. ACKNOWLEDGEMENTS

This work was supported by EU-FP6-IST-507142 project SIMAC (Semantic Interaction with Music Audio Contents) and Centre for Digital Music.

7. REFERENCES

- [1] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [2] F. Keiler and S. Marchand, "Survey on extraction of sinusoids in stationary sounds," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 51–58.
- [3] P. Depalle, G. Garcia, and X. Rodet, "Tracking of partials for additive sound synthesis using Hidden Markov Models," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'93)*, Minneapolis, USA, 1993, pp. 225–228.
- [4] M. Lagrange, S. Marchand, M. Raspaud, and J.-B. Rault, "Enhanced partial tracking using linear prediction," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003, pp. 141–146.
- [5] X. Serra and J. O. Smith, "Spectral modeling synthesis:a sound analysis/synthesis based on a deterministic plus stochastic decomposition," *Computer Music J.*, vol. 14, no. 4, pp. 14–24, 1990.
- [6] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical, and jazz music databases," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'02)*, Paris, France, 2002, pp. 287–288.

A STOCHASTIC STATE-SPACE PHASE VOCODER FOR SYNTHESIS OF ROUGHNESS

Doug Van Nort, Philippe Depalle

Sound Processing and Control Laboratory
 Music Technology Area
 Schulich School of Music
 McGill University, Montreal, Canada
 {doug|depalle}@music.mcgill.ca

ABSTRACT

This paper presents an implementation of the phase vocoder within a Gaussian state-space framework. Rather than formulate the problem as a deterministic evolution of frequencies centered around a given bin, this evolution is treated stochastically by introducing noise into the dynamics matrix of the recursive state equation. This produces effects on the roughness of the input sound, which vary depending on the position within the matrix where the noise is added, how it is propagated throughout the matrix and further by the variance of the noise input.

1. INTRODUCTION

The phase vocoder is a widely used tool for the analysis, transformation and synthesis of audio signals. It began as a way to efficiently code and transmit voice signals using filterbanks [1], was later represented by the Short-Time Fourier Transform (STFT) [2] and then began to find use in musical applications [3],[4]. The most common effects generated by the use of the phase vocoder are pitch shifting and time scaling, which are achieved through altering the time/frequency block increment size between the analysis and synthesis step and then interpolating. If the step increment for both analysis and synthesis is subjected to certain constraints based on the type of windowing function used in the STFT, then the input signal is perfectly reconstructed upon re-synthesis. However the phase vocoder becomes musically interesting when the signal is distorted by transformations such as the aforementioned pitch/time scaling or others in which the amplitude and phase of each frequency bin are modified over time. As the representation itself is purely deterministic and able to capture the signal entirely, these distortions are externally applied to the spectral data in an intermediate (*i.e.* between analysis and synthesis) step. While this technique is very powerful and is the basis for most spectral processing used in computer music compositions, we have found that some interesting effects can be had by embedding a stochastic representation within the phase vocoder itself. This is achieved through the use of a state-space representation.

2. STOCHASTIC STATE-SPACE VOCODER

Rather than model a signal as an Autoregressive-Moving Average (ARMA) process, a stochastic process $x[n]$ that is governed by a linear dynamical system can be expressed by the state-space equations

$$s[n+1] = As[n] + w[n] \quad (1)$$

and

$$x[n] = Bs[n] + v[n] \quad (2)$$

where the sequence $s[n]$ is the state of process x at time n , and Equation (1) represents the internal dynamics of the process as governed by dynamics matrix A . Equation (2) transfers the state vector, which may be hidden, into a vector of observable output variables. Both w and v are Gaussian white-noise processes. The first affects the progression of the state while the second is additive noise present in the output process x . This latter use of noise in the modeling of an audio signal can be found in the Spectral Modeling Synthesis (SMS) approach [5]. Our interest here is in the effect of including noise in the state process equation of a representation based on the the phase vocoder. In particular, we include noise in the state *matrix* rather than simply in the state vector as is done in equation (1). Therefore, we may re-write the state equation as follows

$$s[n+1] = (A + W_n)s[n] \quad (3)$$

where W_n is a time-varying matrix of Gaussian random variables. This matrix will be described in more detail in section 2.2.

2.1. Related work

A state-space approach to analysis/synthesis was presented in [6] in which the real and imaginary components of p sinusoidal partials, tracked over time, were represented in the state vector. The observation matrix summed across the real components of the partials, and the addition of observation noise generated a sinusoid+noise re-synthesis. Thus, this model represents a hybrid state-space / sinusoidal model.

Similarly, a recursive state-space formulation is presented in [7] in which the state is comprised of the real and imaginary components for N evenly spaced frequency bins. Thus, this implementation maintains all of the data from the phase vocoder while the aforementioned work tracks only partials and is closer to a sinusoidal model. The motivation differs in this latter work as well, with the goal being the interpolation of missing audio samples whereas the former research was concerned with building an analysis/synthesis scheme for audio transformations. This current work is situated between these two in the sense that our motivation is towards musical transformations, yet we work on the lower-level representation of the phase vocoder's spectral frames. However our state-space representation differs from [7] in reflection of the differing motivations: the desire to track time-domain signals and interpolate missing values lead to a stochastic representation as in equation (1), in order to build uncertainty into the time-varying

signal. We add noise to the dynamics matrix in order to perturb the structure of the *system itself* in order to explore the complex couplings that result.

Each of these approaches are based on a recursive description of the Discrete Fourier Transform. That is, the complex exponentials of the DFT can be expressed as

$$e^{jn\theta} = e^{j\theta} e^{j(n-1)\theta} \quad (4)$$

for time n and frequency θ . Thus the DFT matrix and its inverse can be expressed as a first-order recursion, a necessity in order to work within the state-space framework.

2.1.1. A Note on "Roughness"

The term "roughness" has taken on a specific meaning in the psychoacoustics literature [8]. Its introduction in this context can be dated back to Helmholtz in the late 19th century where it was linked to the subjective notion of dissonance. While this latter term has somewhat changed itself with musical periods, psychoacoustical roughness relates to dissonance in the classical sense of beating/modulating sounds such as those that result from certain pitch ratios (*e.g.* a minor second). In this sense it has been shown to change as a function of the depth and frequency of modulation of a sound in both amplitude and frequency. In this work, we do no use the word in this strict sense, and yet our results relate qualitatively to those sounds that would be considered psychoacoustically "rough". Certain results could as easily be labeled as "noisy" or "textural."

2.2. Current Implementation

Again, our state-space phase vocoder is built from a state vector comprised of the real and imaginary components of the audio signal. The nature and size of said state varies depending on whether the analysis or synthesis step is being performed. For the analysis step, given an input block of real signal $x = \{x_1, \dots, x_N\}$, the state vector is thus

$$s = [x_1, 0, \dots, x_N, 0]^T \quad (5)$$

which represents the initial state vector for the current block of N samples. The state is re-initialized with a new input block at each signal boundary (each N samples), and during the state recursion s is propagated by the dynamics matrix

$$A = \mathbf{DIAG}(R(\theta_0), \dots, R(\theta_{N-1})) \quad (6)$$

where **DIAG** represents a block diagonal matrix and

$$R(\theta_k) = \begin{pmatrix} \cos(\frac{2\pi k}{N}) & \sin(\frac{2\pi k}{N}) \\ -\sin(\frac{2\pi k}{N}) & \cos(\frac{2\pi k}{N}) \end{pmatrix}. \quad (7)$$

The observation matrix

$$B = \begin{pmatrix} 1 & 0 & \dots & 1 & 0 \\ 0 & 1 & \dots & 0 & 1 \end{pmatrix} \quad (8)$$

produces an output vector¹

$$\hat{s} = (s_{0,r}, s_{1,r}, s_{1,i}, \dots, s_{\frac{N}{2}-1,r}, s_{\frac{N}{2}-1,i}, s_{\frac{N}{2},r}) \quad (9)$$

¹Strictly speaking, the analysis step produces a $2 \times N$ matrix. These values are then rearranged and the trivial imaginary values at $\theta_0, \theta_{\frac{N}{2}}$ discarded in order to form \hat{s} .

which is comprised of the real and imaginary components of the spectrum of input block x . We assume that x is real, and so only the first $\frac{N}{2}$ frequency bins are generated by the analysis state equations.

Now, the observed process \hat{s} becomes the state vector for the synthesis step, where the synthesis dynamics matrix is defined by

$$\hat{A} = \mathbf{DIAG}(1, R^{-1}(\theta_1), \dots, R^{-1}(\theta_{\frac{N}{2}-1}), 1). \quad (10)$$

The new observation matrix

$$\hat{B} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 \end{pmatrix} \quad (11)$$

produces output signal \hat{x} . In the absence of noise added to the state or observation equations for analysis and synthesis, this two-step recursion provides a perfect reconstruction. However, the addition of noise into the state equations at various points in the analysis/synthesis process and in various ways can introduce different roughness qualities into the input sound that can then be controlled.

2.2.1. Introduction of Process Noise

In the standard state-space formulation the noise that is added into the state and/or observation equation is a vector w_n whose dimension is the same as that of the state. In our implementation we introduce an $M \times M$ matrix of Gaussian noise, where M is the size of the state. In particular the matrix is decomposed as follows:

$$W_n = \alpha W_n^d + \beta W_n^r \quad (12)$$

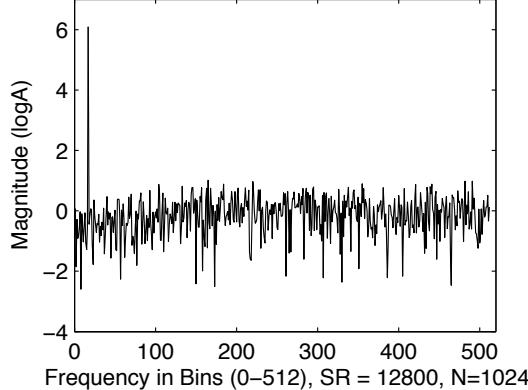
where W_n^d is the block diagonal that corresponds to the non-zero values of the dynamics matrix A and W_n^r provides the Gaussian values for the remaining upper and lower triangular parts of the matrix. α and β are free parameters that allows one to tune the contribution of these two parts of the matrix. We do not include an observation noise vector as this simply adds white noise to the output. Rather, the above matrix is added to the state dynamics as in equation (3), which can produce interesting frequency effects.

In particular, the matrix W_n^d is added to the sinusoidal components of the dynamic rotation matrix, causing an uncorrelated and random fluctuation of amplitude and/or phase in each element of the state. This effect can be very precise and localized in certain cases. For example, if the same noise values \hat{N} are added to a sub-block along the diagonal of the synthesis-step state matrix \hat{A} as follows

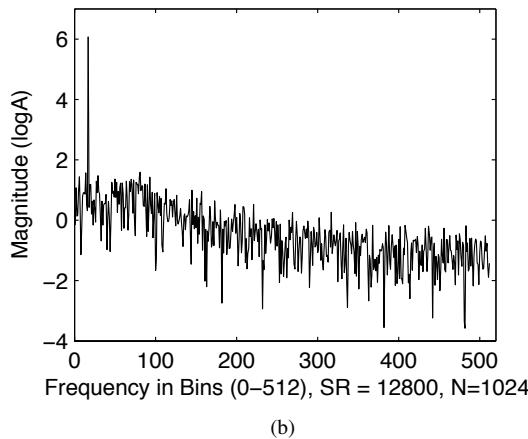
$$R^{-1}(\theta_k) + \hat{N} = \begin{pmatrix} \cos(\frac{2\pi k}{N}) + n_1 & -\sin(\frac{2\pi k}{N}) + n_2 \\ \sin(\frac{2\pi k}{N}) + n_1 & \cos(\frac{2\pi k}{N}) + n_2 \end{pmatrix} \quad (13)$$

then there will be a random modulation in the amplitude of partial k while the phase remains unchanged. The amplitude or phase of a given frequency can also be modified by converting the corresponding members of the state vector into polar form, acting on the appropriate values and then converting back to rectangular form before re-inserting them into the state equation. In this way one can *e.g.* introduce concurrent random modulations between partials that can affects a given sound's "texture" [9].

Now, the matrix W_n^r causes a random fluctuation which behaves quite differently. Random values added in this part of the matrix introduce a non-linear distortion, and noise can be added at specific matrix locations in order to introduce a coupling between



(a)



(b)

Figure 1: Log FFT plot for stochastic state-space processed sinusoid. Gaussian noise added to analysis dynamics matrix in outer triangular portion (a) and to diagonal (b).

two frequencies. This can be non-physical — such as if the frequency at bin i is coupled to bin j but j is not coupled to i — or it can maintain a physical meaning if frequencies remain coupled in a bi-directional manner. We experimented with different process noise behaviors towards the end of creating musically interesting roughness effects.

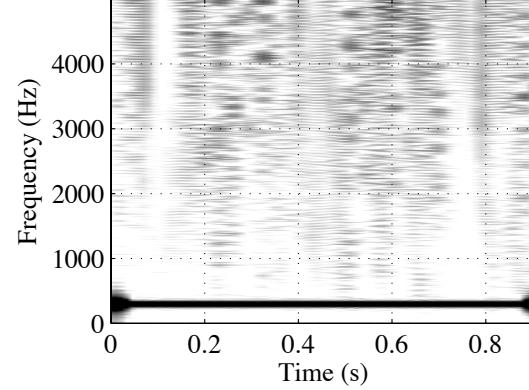
3. RESULTS

Different roughness effects are observed depending on several factors: where in the dynamics matrix noise was introduced, if and how it was propagated in time through the matrix, and whether it was added during the analysis or synthesis step. Some results are summarized below.

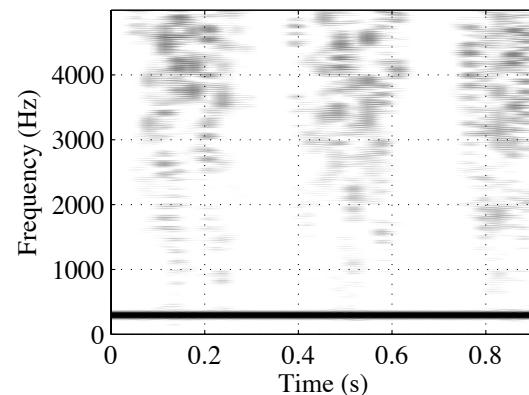
3.1. Noise in Analysis Step

Given an input sinusoid at frequency f , and when noise was added to the outer triangular regions of the matrix only — that is, when $W_k = \beta W_k^r$ — a nearly white noise component was added to the

²We use index k here to underscore the fact that the state recursion in the analysis step is a function of frequency rather than time.



(a)



(b)

Figure 2: Spectrogram of sinusoid affected by noise propagated through analysis state matrix at one column alone (a) and coupled between a column/row pair (b). Sampling frequency = 11.025 kHz.

entire signal, with a slight increase in energy at higher frequencies. In contrast, when noise was added to the matrix diagonal and $W_k = \alpha W_k^d$, a band of noise was introduced whose energy was concentrated around frequency f and fell off at higher frequencies. This difference is illustrated in Figure 1.

In order to create more interesting roughness qualities, we propagated noise through various parts of the matrix to create a time-varying effect. In particular, noise was propagated through a given column or row of the matrix. When noise was propagated down a single row or column, a beating noise with several small peaks was introduced. The rate of the beating can be controlled by the speed at which the Gaussian scalar noise value is sent through the given row/column. This modulating behavior can be seen in Figure 2b.

While this time-varying single perturbation produced a more musical result, it was not physically accurate: the noise value caused an interaction between the frequency located at the given column where the propagation occurred and each other frequency bin at the instant that the noise was swept past it in the matrix. However, this was not truly a coupling between frequencies as it did not occur in both directions. Thus, to make the effect more physical, we sent the same noise value down both column and row. Therefore, at time t if the input noise value was added to matrix

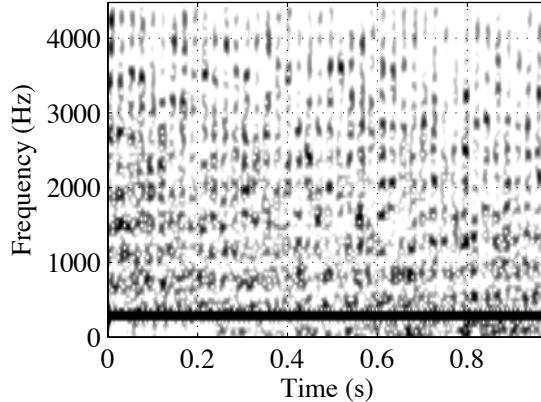


Figure 3: Spectrogram of sinusoid at 300 Hz affected by noise propagated through synthesis state matrix. Sampling frequency = 8.5 kHz.

value $A(i,j)$, it was further added to the value at $A(j,i)$. This coupled time-varying effect proved to create a much more sophisticated stochastic component to the sound — one that possessed more “texture” and resembled the sound of fire. For input sinusoid with frequency f , this effect was most prominent when it occurred at the column/row associated with the highest-energy frequency bin, namely $k = \frac{N*f}{F_s}$ where F_s is sampling frequency and N is the size of the input signal block. The difference between the “abstract” and physical roughness effects can be seen in Figure 2. Beyond having more high frequency content, the coupled example of Figure 2a possesses a spectral fine-structure that is present throughout the spectrum and which likely contributes to the overall textural quality.

3.2. Noise in Synthesis Step

It is important to remember that the state vector is not the same between analysis and synthesis steps. During analysis, the state is initialized with real and imaginary components of an input signal block of size N . At the synthesis stage, the state vector is initialized with the real and imaginary spectral values that are generated by the first $N/2$ iterations (assuming a real-valued input) of the recursive analysis process. Therefore, the addition of noise to the state matrix affects the dynamics of either the complex modulation or demodulation process associated with the DFT/iDFT and there is no reason to assume that the addition of the Gaussian noise vector would produce the same sonic result at each stage. Indeed, the addition of noise values in the synthesis state matrix — of coupled noise propagated down a column/row pair — produce a strong modulation effect not present in the previous examples. While the others produced fluctuations and a beating effect, this synthesis-step noise introduces a spectrotemporal modulation illustrated by the spectrogram of figure 3. In this representation one can see a quasi-periodic emergence of strong spectral peaks which modulate throughout the spectrum.

4. CONCLUSION AND FUTURE WORK

We have introduced a phase vocoder in which a stochastic element has been built into the representation via a state-space framework. Through the embedding of noise within the system representation itself (rather than as input to the state or observation equations) a sound can be re-synthesized with an added “roughness” quality. Preliminary results illustrate the potential of using this approach to generate interesting effects. With this simple linear systems framework, musically useful nonlinear distortions can be introduced and different effects can be created and controlled by altering the Gaussian noise matrix over time. One simple physically-inspired effect was suggested. We intend to explore more complex processes by exploiting the interaction between the analysis and synthesis state vectors, by introducing harmonic distortion via couplings between harmonically related frequency components and by constructing more elaborate time-based control curves for the noise parameters.

5. REFERENCES

- [1] J. L. Flanagan and R. M. Golden, “Phase vocoder,” *Bell System Tech. J.*, vol. 45, pp. 1493–1509, Nov. 1966.
- [2] J. B. Allen and L. R. Rabiner, “A unified approach to short-time Fourier analysis and synthesis,” *Proc. IEEE*, vol. 65, pp. 1558–1564, 1977.
- [3] J. A. Moorer, “The use of the phase vocoder in computer music applications,” *J. Audio Eng. Soc.*, vol. 26, no. 1, pp. 42–45, 1978.
- [4] M. Dolson, “The phase vocoder: A tutorial,” *Computer Music J.*, vol. 10, no. 4, pp. 14–27, 1986.
- [5] X. Serra and J. O. Smith, “Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition,” *Computer Music J.*, vol. 14, no. 4, pp. 14–24, 1990.
- [6] H. D. Thornburg and R. J. Leistikow, “Analysis and resynthesis of quasi-harmonic sounds: An iterative filterbank approach,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003, pp. 129–134.
- [7] A. T. Cemgil and S. J. Godsill, “Probabilistic phase vocoder and its application to interpolation of missing values in audio signals,” in *13th European Sig. Proc. Conf.*, Antalya, Turkey, 2005, [Online] <http://www-sigproc.eng.cam.ac.uk/~atc27/papers/cemgil-godsill-em-restore-eusipco.pdf>.
- [8] P. Daniel and R. Weber, “Psychoacoustical roughness: Implementation of an optimized model,” *Acustica*, vol. 83, pp. 113–123, 1997.
- [9] S. Dubnov, N. Tishby, and D. Cohen, “Influence of frequency modulating jitter on higher order moments of sound residual with applications to synthesis and classification,” in *Proc. Int. Comp. Music Conf. (ICMC’96)*, Hong Kong, 1996, pp. 378–385.

FAST ADDITIVE SOUND SYNTHESIS USING POLYNOMIALS

Matthias Robine, Robert Strandh, Sylvain Marchand

SCRIME – LaBRI, University of Bordeaux 1
 351 cours de la Libération, F-33405 Talence cedex, France
 firstname.name@labri.fr

ABSTRACT

This paper presents a new fast sound synthesis method using polynomials. This is an additive method, where polynomials are used to approximate sine functions. Traditional additive synthesis requires each sample to be generated for each partial oscillator. Then all these partial samples are summed up to obtain the resulting sound sample, thus making the synthesis time proportional to the product of the number of oscillators and the sampling rate. By using polynomial approximations, we instead sum up only the oscillator coefficients and then generate directly the sound sample from these new coefficients. Most of computation time is consumed by a data structure that manages the update of the generator coefficients as a priority queue. Practical implementations show that Polynomial Additive Sound Synthesis (PASS) is particularly efficient for low-frequency signals.

1. INTRODUCTION

We present a method for additive sound synthesis with a complexity that is proportional to the sum of the frequencies of the oscillators, as opposed to the traditional method whose complexity is proportional to the number of the oscillators, regardless to their frequencies. Thus, the new method is especially efficient for low frequencies. Perhaps more surprisingly, this method is not so dependent on the sampling frequency. This makes our new method especially well-suited for high sampling frequencies.

First, we review in Section 2 the principles of classic additive synthesis, as well as the methods proposed for real-time implementations. Then we present in Section 3 our new method using polynomials, with a lower complexity. We explain how to approximate with polynomials the sine functions of the sinusoidal oscillators, and how to generalize this approximation for all the partials. All the polynomial coefficients from these approximations are then summed, becoming coefficients of a global polynomial generator. Thus, sound samples are computed from a single polynomial.

As the approximation of each partial is limited in time (on a part of a period of the sine function), its coefficients must be updated. We propose in Section 4 to manage all these update events with a priority queue, efficiently implemented as a binary heap. The element with the highest priority is always the next update event to be processed. We explain how to use this data structure for our method. We show then in Section 5 how this priority queue can again be useful to manage the change of the amplitudes and frequencies of the partials at the right moment.

Finally we present some performance results in Section 6 and we compare our method with other techniques proposed in the literature.

2. ADDITIVE SOUND SYNTHESIS

Additive synthesis (see for example [1]) is the original spectrum modeling technique. It is rooted in Fourier's theorem, which states that any periodic function can be modeled as a sum of sinusoids at various amplitudes and harmonic frequencies. For stationary pseudo-periodic sounds, these amplitudes and frequencies evolve slowly with time, controlling a set of pseudo-sinusoidal oscillators commonly called *partials*. This is the well-known McAulay-Quatieri representation [2] for speech signals, also used by Serra [3] in the context of musical signals. As they evolve slowly in time, we consider the frequencies and amplitudes as constant for a short length. The audio signal s can be calculated from the sum of the partials using:

$$s(t) = \sum_{i=1}^N a_i \sin(2\pi f_i t + \phi_i) \quad (1)$$

where N is the number of partials in the sound and the parameters of the model are f_i , a_i , and ϕ_i , which are respectively the frequency, amplitude, and initial phase of the partial number i . This equation is valid if the frequency is constant. However, for practical sound examples, both the frequency and the amplitude must be updated regularly. Equation 1 then holds for each sound segment between two update times.

In the general approach derived from Equation 1, for each sample the partials are processed separately, and then summed. Thus the complexity of the method is proportional to the product of the number of partials and the sample rate. Computing the sine function for every partial and every sound sample can be very time-consuming. Using additive synthesis to synthesize a whole orchestra is a big challenge, and even more so if we want to do it in real time. This is why we need to reduce the computation time of the additive synthesis, while keeping the control of all the parameters of the sound partials in time.

The most straightforward – rather naive – way to calculate a partial contribution is to use the sine function. But it consumes a lot of computation time. Other techniques are possible, such as the use of the digital resonator method (see for example [4, 5]), which computes the samples of each separate partial with an optimal number of operations. In this method, the sine is calculated with an incremental algorithm that avoids computing the sine function for every sample. We proposed the use for fast additive synthesis of the digital resonator with floating point arithmetic in [6, 7]. For each partial the resonator is initialized as Equation 2 shows, with F_s the sampling rate of the synthesis, a , f , and ϕ respectively the amplitude, frequency, and initial phase of the partial, and Δ_ϕ the phase increment. The incremental computation of each oscillator

sample requires only 1 multiplication and 1 addition.

$$\begin{cases} \Delta_\phi &= \frac{2\pi f}{F_s} \\ s[0] &= a \sin(\phi_0) \\ s[1] &= a \sin(\phi_0 + \Delta_\phi) \\ C &= 2 \cos(\Delta_\phi) \\ s[n+1] &= C \cdot s[n] - s[n-1] \end{cases} \quad (2)$$

This algorithm is optimal in a sense that 1 multiplication with no addition will lead to a geometric progression, whereas no multiplication with 1 addition will lead to an arithmetic one; none of these progressions being a sine function. Again for the purpose of real-time additive synthesis, we then proposed to limit the number of partials to be synthesized by removing inaudible ones, using a psychoacoustic model together with an efficient data structure [8].

In order to efficiently synthesize many sinusoids simultaneously, Freed, Rodet, and Depalle propose in [9] to use the inverse Fourier transform, provided that the oscillator parameters vary extremely slowly. The idea is to reconstruct the short-term spectrum of the sound at time t , by adding the band-limited contribution of each partial, then to apply the Inverse Fast Fourier Transform (IFFT or FFT^{-1}) in order to obtain the temporal representation of the sound, and finally to repeat the same computation further in time, thus performing a kind of “inverse phase vocoder”. The gain in complexity is when the number of oscillators is large in comparison to the number of samples to compute at each frame. This approach is very interesting, because its complexity is no more the product of the number of partials and the sampling rate. However, the control of the additive parameters is more complex.

3. USING POLYNOMIALS

Polynomials have been traditionally used in order to model the parameters of the sinusoidal model [10, 11, 12]. Here, we propose to use polynomials to replace the sine function. Our method consists of first calculating a set of polynomial coefficients for each partial. Polynomial values from polynomials computed with these coefficients approximate the signal of the partial on a part of its period. The classic approach would evaluate the polynomial associated to each oscillator, and then sum up the results, which is quite inefficient. The idea is yet to sum the coefficients in a polynomial generator, then to evaluate the resulting polynomial only once. Indeed, summing polynomials leads to another polynomial of the same degree. The sound samples can be computed from this single resulting polynomial, with a fairly low degree – independent of the number of partials to synthesize. The general process is illustrated by Figure 1.

3.1. Partial Approximation

The time-domain signal generated by each partial is defined by a sine function. We propose to approximate this function by a polynomial. To get the polynomial coefficients that can approximate any partial of a sound, we decide to first approximate a unit signal u with amplitude $a = 1$, frequency $f = 1$, and phase $\phi = 0$, i.e.:

$$u(t) = \sin(2\pi t)$$

We have to choose a part of the period where we will do the approximation. We call this part the validity period p of the polynomial coefficients. Thus, if we approximate a half period of u , then $p = 1/2$.

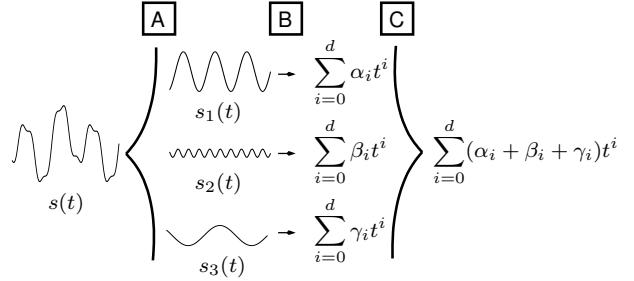


Figure 1: PASS. Step A: A periodic signal can be divided into sinusoidal components. Step B: Computing polynomial coefficients to approximate the signal for each partial. Step C: A polynomial generator is obtained by summing the coefficients from the polynomials of the partials. The values computed by the generator will be the samples of the sound signal.

Measuring the performance of the approach of a signal $u(t)$ by a polynomial $U(t)$ on a validity period p can be done using the SNR ratio given by:

$$\text{SNR} = 10 \log_{10} \frac{\int_0^p u^2(t) dt}{\int_0^p (u(t) - U(t))^2 dt}$$

For a given polynomial degree, we propose to find the polynomial coefficients that minimize the value of the denominator of the SNR:

$$\int_0^p (u(t) - U(t))^2 dt$$

These coefficients have to respect other constraints to maintain a piecewise continuity. For example, with a 2-degree polynomial U and $p = 1/2$, to impose a C^1 continuity, it is sufficient that $U(0) = U(1/2) = 0$, and the coefficients a_i that minimize:

$$\int_0^{\frac{1}{2}} (\sin(2\pi t) - (a_0 + a_1 t + a_2 t^2))^2 dt$$

are:

$$\begin{cases} a_0 = 0 \\ a_1 = 240/\pi^3 \\ a_2 = -480/\pi^3 \end{cases}$$

To approach the sine function, we can use in this case alternately U_a for a first half period and U_b for the second:

$$\begin{cases} U_a(t) = a_1 t + a_2 t^2 \\ U_b(t) = -a_1 t - a_2 t^2 \end{cases}$$

The choice of the validity period, and the highest degree of the polynomials to use, have a big influence on the performance of the approximation, as shown in Table 1. We explain in 3.3 how a high polynomial degree can lead to numerical instability of the method, or in Section 6 why the choice of a short validity period increases the computation time. We can note that using a validity period $p = 1/2$ with a polynomial degree $d = 2$ is particularly suited for a very fast synthesis, and that using a validity period $p = 1/2$ with a polynomial degree $d = 4$ is suited for a fast synthesis with good quality.

The coefficients we compute define a unit polynomial $U(t)$ by validity period. When the unit polynomial is found, every partial can be approximated from it. In the general case of a partial i with

amplitude a_i , frequency f_i , and initial phase ϕ_i , the approximating polynomial P_i is then given by:

$$P_i(t) = a_i U \left(f_i t + \frac{\phi_i}{2\pi} \right)$$

Notice that the amplitude, frequency, or phase parameters do not modify the approximation error given in Table 1. In addition to the sinusoid, the polynomial approximation generates a noise consisting of harmonics of this sinusoid. The magnitudes of these harmonics are small, and depend on the validity period p and the polynomial degree d .

p	d	C^0 SNR (dB)	C^1 SNR (dB)
1/4	2	36	28
1/4	3	57	28
1/4	4	79	59
1/4	5	102	59
1/2	2	28	28
1/2	3	28	28
1/2	4	59	59
1/2	5	59	59
1	4	17	17
1	5	42	42

Table 1: Error of polynomial approximation of a partial. For two different continuity requirements (C^0 and C^1), the Signal-to-Noise Ratio (SNR) obtained with the approximation error $u - U$ compared to the target signal u are shown as functions of the validity period p and the polynomial degree d .

Since for now we consider only constant parameters for the partials, the generated functions are periodic. It is thus possible to compute the polynomial coefficients for only one period of any partial.

Each set of polynomial coefficients is valid for a part of the period of the sine function. For example, if we choose to approximate sine functions using a fourth of their period, we need to compute four sets of coefficients by partial. As long as the amplitude and the frequency of a partial are constant, we can continue with the same pre-calculated sets. During the sound synthesis, the coefficients that approximate the partials must be updated regularly (the rate depending on the frequency of each partial), and must also be changed if the parameters have changed.

3.2. Incremental Calculation of Polynomials

To avoid the problem of computing a polynomial with large time values, leading to numerical imprecision, we propose to use the Taylor's theorem to compute it. The polynomial can be evaluated at every instant $t_0 + \Delta_t$ by using the value and the values of its derivatives at a preceding instant t_0 :

$$P^k(t_0 + \Delta_t) = P^k(t_0) + \sum_{i=k+1}^d \frac{\Delta_t^{i-k}}{(i-k)!} P^i \quad (3)$$

where P^k is the k -th derivative of the polynomial function (P^0 being the polynomial itself). The number of necessary values depends on the degree of the polynomial (e.g., three values with a 2-degree polynomial).

With the polynomial coefficients of the partials obtained according to the method presented in Section 3.1, we compute the first value of the polynomial and of its derivatives. To compute each of the following values, we use Equation 3 with a step Δ_t corresponding to the time between two time events. A time event is either the time of a sound sample or of a scheduled update of the coefficients. When time reaches or exceeds the validity period we have chosen, the coefficients are updated, and the incremental algorithm goes on with the new coefficients.

3.3. Polynomial Generator

Using this incremental method for each individual partial would be very expensive in terms of computation time. For that reason, we propose a technique in which we sum the coefficients to compute only a global polynomial, the generator. During the synthesis, the generator is computed incrementally. When a partial reaches the end of its validity period, the different values of the generator (value and values of the derivatives) are summed with the new values from the partial. When a sound sample must be produced, the generator is computed to get the sound sample value.

As the generator is computed incrementally, we have to care about numerical precision: using the preceding values to compute new ones accumulates floating-point precision errors in the result. Thus, there is a validity limit for updating the generator. According to the polynomial degree used, to the number of partials in the sound, and the floating-point precision we can use, we need to re-initialize the generator coefficients regularly with the authentic sine function.

The complexity of our method is dominated by the management of the update events from individual partials. To optimize this process, we propose the use of an efficient data structure, in a way similar to that of [8] which uses a skip-list to increase the performance of additive synthesis. In the PASS method, we use a priority queue implemented as a binary heap to manage the update events from the partials.

4. DATA STRUCTURE

4.1. Using a Heap as a Priority Queue

A priority queue is an abstract data type supporting two operations: *insert* adds an element to the queue with an associated priority; *delete-max* removes the element from the queue that has the highest priority, and returns it. We use a priority queue to manage update events from partials of the sound. During synthesis, update events are regularly inserted in, or removed and processed.

The standard implementation of priority queues is based on binary heaps (see for example [13]). With this implementation, queue operations have $O(\log(N))$ complexity, with N the number of elements in the queue. A binary heap is a binary tree satisfying two constraints:

1. the tree is either a perfect binary tree, or, if the last level of the tree is not complete, the nodes are filled from left to right;
2. each node is greater (in priority) than or equal to each of its children. The top of the binary heap is always the next event to process.

4.2. Heap Optimization

Most of the computation time of the PASS method is due to the management of the heap. Consequently, heap primitives need to be highly optimized. Our first approach was to *delete-max* the priority queue, to process the update event and to *insert* a new one in the queue. With this approach, the heap is substantially reorganized twice for each insert / delete pair of operations.

To improve the performance, we replaced the insert / delete primitives with *top* and *replace*. *top* returns the element of the queue that has the highest priority, without removing it. It is possible to process an update event without removing it from the queue. The *replace* method replaces the element from the queue that has the highest priority with an other element by initially putting it on top of the heap, and then letting it trickle down according to normal heap-reorganization primitives. In the worst case, the *replace* method needs $O(\log(N))$ operations, N being the number of elements in the heap. The heap is reorganized only once per update.

Partials with high frequencies must be updated more often than the others, because their validity period is smaller. With the *replace* method, the update events concerning high frequencies stay near to the top of the heap. Using fewer operations for the most frequently updates improves the complexity of our method. Figures 2 and 3 give an example of heap management, where the *delete-max* then *insert* methods use 4 elements swaps, whereas the *replace* method takes only 1 swap.

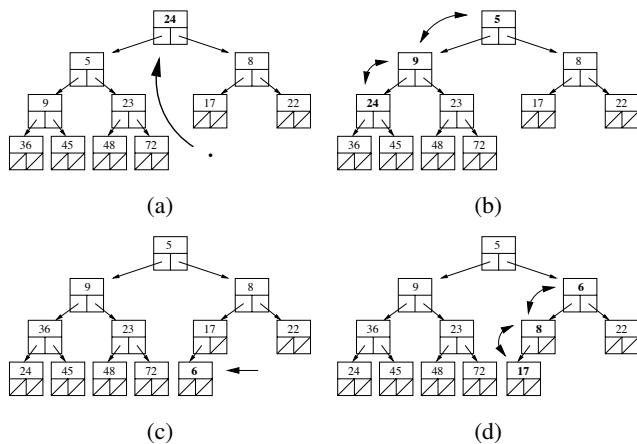


Figure 2: *delete-max then insert* priority queue methods within a heap. (a) *Delete-max* of the element with highest priority. (b) Heap reorganization. (c) Insert of the new element. (d) Heap reorganization.

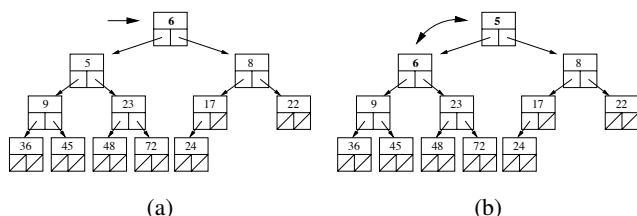


Figure 3: *replace* priority queue method with a heap. (a) Replace of the element with highest priority by the new one. (b) Heap reorganization.

5. CHANGE OF SOUND PARAMETERS

We consider that the parameters of the partials are constant on a short length. But since they change, they have to be updated regularly. In [14] we indicate the best time in a period to change parameters of a partial, as illustrated by Figure 4 and 5. The best moment to change the amplitude is when the signal is minimal, to preserve the continuity of the signal. And the best moment to change the frequency is when the signal is maximal, to preserve the continuity of the signal derivative.

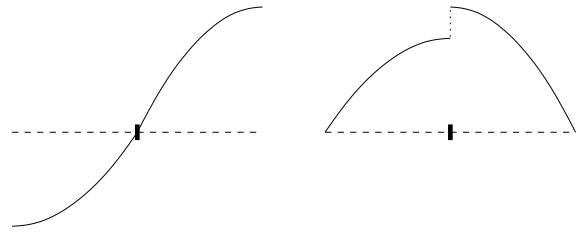


Figure 4: Changing the amplitude either when the signal is minimal (left) or maximal (right). It appears that the left case is much better, since it avoids amplitude discontinuities (clicks).

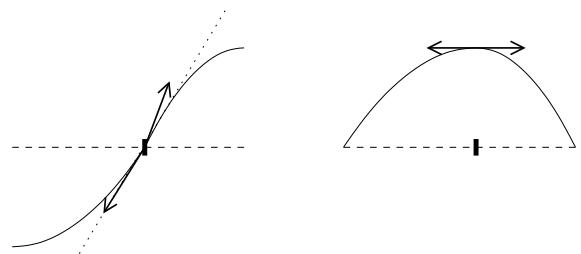


Figure 5: Changing the frequency either when the signal is minimal (left) or maximal (right). It appears that the right case is better, since it avoids derivative discontinuities (clicks).

Changing parameters of partials with the PASS method consists in changing the polynomial coefficients of an oscillator when this oscillator must be normally updated, because of the end of its validity period. Thus, this change does not need more computation time than without changing the parameters. The best case is with the validity period $p = 1/4$. In this case we can update the parameters at the best moment we described before for the frequency or the amplitude. Otherwise an update event will be added in the priority queue, to indicate the time to change the parameters of the partials. But few events will be added regarding to the normal updates of the coefficients, and it will not really affect the general computation time. The parameters are updated in the right moment to avoid clicks in the sound. This moment is different for every partials, regarding to their frequencies.

6. COMPLEXITY AND RESULTS

The complexity of the PASS method depends on the use of the priority queue. For one second of synthesis, the priority queue will be used f/p times per partial, where p is the validity period chosen ($1/2$ for a half of a sine period), and f the frequency of the

partial. For every partial, the queue operations are called X times per second, where:

$$X = \frac{1}{p} \sum_{i=1}^N f_i = \frac{N\bar{f}}{p} \quad (4)$$

with N being the number of partials in the sound, p the validity period, f_i the frequency of the partial i , and we denote by \bar{f} the mean frequency of the partials.

If we consider that each priority queue operation is done with $O(\log(N))$ complexity for an update event, with N the number of elements simultaneously in the queue (*i.e.* the number of partials in the sound), the complexity C_1 due to the priority heap managing is given by:

$$C_1 = O\left(\frac{N\bar{f}}{p} \log N\right) \quad (5)$$

We can note that this complexity is very dependent from the mean frequency of the partials. It explains why the method is particularly efficient for low-frequency signals. If the validity period p is doubled (from a fourth to a half of period for example), the performance of the PASS method is doubled too. The higher is p and the better is the complexity of the method. But if we want to increase p , we need to use higher polynomial degree to approximate the sine function. And it leads to numerical instability. We have to find a trade-off between complexity and stability.

In addition to C_1 is the C_2 complexity, to produce the samples of the sound. If we use F_s as the sampling rate and d the polynomial degree of the generator, it is given by:

$$C_2 = O(dF_s) \quad (6)$$

Thus, the general complexity of PASS is:

$$C_{\text{PASS}} = O\left(\alpha \frac{N\bar{f}}{p} \log N + dF_s\right) \quad (7)$$

where α is some constant which is architecture-dependent (in practice, the synthesis methods were implemented in C language, compiled using the GNU C compiler (`gcc`) versions 4.0 and 4.1, and executed on PowerPC G4 1.25-GHz and Intel Pentium 4 1.8-GHz processors). This global complexity is a function of the sum of the frequencies of the partials, and we notice that it does not strongly depend on the sample rate anymore. Increasing the sample rate from 44.1 kHz to 96 kHz does not really affect the computation time, as illustrated by Table 2.

One might reasonably ask how our method compares to other methods for additive synthesis. Recently, we showed in [7] that the digital resonator method was a little better than the synthesis using the FFT^{-1} method. But later, Meine an Purnhagen [15] compared different methods and concluded that the fastest additive synthesis is now the FFT^{-1} method. In fact, this highly depends on the implementation details and computer used, as well as the number of partials and sampling rate. However, the digital resonator easily allows the fine control of each partial of the sound, which is not really the case the FFT^{-1} method. The PASS method allows the same fine control, and thus we have compared the performance of our method with that of the digital resonator method.

And if we have just noted that for PASS method the sample rate does not really affect the computation time, it is not the same with the digital resonator. The complexity C_{DR} of the digital resonator method is given by:

$$C_{\text{DR}} = O(NF_s) \quad (8)$$

with F_s the sampling rate, and N the number of partials in the sound. Here N and F_s are multiplied. This difference of complexity between digital resonator and PASS methods is illustrated by Table 2.

As shown in Table 3 or in Figure 6, the method we present is clearly better than the digital resonator for low frequencies. Using a 2-degree polynomial to approximate a half of a period of each partial, PASS is better for 2500 partials when the mean frequency of the partials is under 300 Hz, and even 500 Hz for a 96-kHz sampling rate. Real-time synthesis can be achieved with 5000 partials with a frequency of 150 Hz for example.

N	\bar{f}	F_s (Hz)	DR	PASS
4000	300	22050	3.2 s	6.6 s
4000	300	44100	6.3 s	6.6 s
4000	300	96000	13.7 s	6.6 s

Table 2: Comparison of the computation time of the Digital Resonator (DR) and PASS methods using different sampling rates, for 5 seconds of sound synthesis, implemented in C language, compiled using the GNU C compiler (`gcc`) version 4.1, and executed on an Intel Pentium 4 1.8-GHz processor. The PASS method is used with 2-degree polynomials and a validity period $p = 1/2$. N is the number of partials, \bar{f} is the mean frequency of the partials, and F_s is the sampling rate.

N	\bar{f}	DR	PASS
2500	200	3.9 s	2.0 s
2500	300	3.9 s	3.0 s
2500	400	3.9 s	4.0 s
2500	500	3.9 s	5.0 s
5000	200	7.9 s	7.3 s
5000	300	7.9 s	10.6 s
5000	400	7.9 s	14.4 s

Table 3: Comparison of the computation time of the Digital Resonator (DR) and PASS methods, for 5 seconds of sound synthesis with a sampling rate of 44100 Hz, implemented in C language, compiled using the GNU C compiler (`gcc`) version 4.1, and executed on an Intel Pentium 4 1.8-GHz processor. The PASS method is used with 2-degree polynomials and a validity period $p = 1/2$. N is the number of partials, \bar{f} is the mean frequency of the partials.

7. CONCLUSION AND FUTURE WORK

We have presented PASS, a new additive synthesis method using polynomials. The computation time of this method depends mainly on the sum of the frequencies of the partials. We have shown that the method is fast, and particularly efficient for signals with low frequency partials, as well as for high sampling rates.

In the near future, we plan to tune the trade-off between complexity and stability for our method on the fly, by combining the advantages of the PASS and Digital Resonator (DR) methods, since these methods both manipulate oscillators. For this hybrid method, the idea is, for a given partial, to use either PASS or DR depending

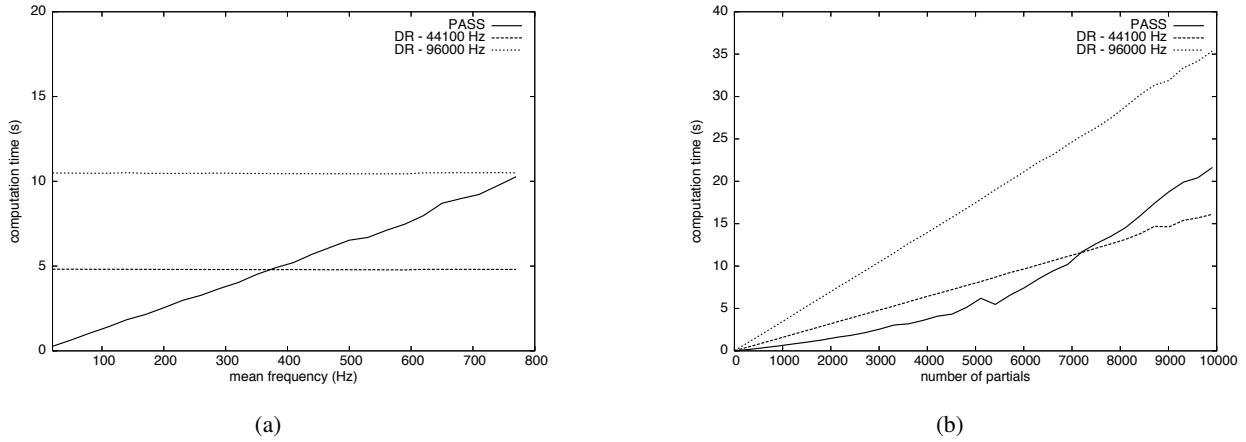


Figure 6: Comparison of the computation times of the Digital Resonator (DR) and PASS method, for 5 seconds of sound synthesis. (a) Computation times are functions of the mean frequency \bar{f} , with a fixed number of partials $N = 3000$. (b) Computation times are functions of the number of partials N , with a fixed mean frequency $\bar{f} = 200$ Hz. The PASS method is used with 2-degree polynomials and a validity period $p = 1/2$. For this comparison, both methods were implemented in C language, compiled using the GNU C compiler (gcc) version 4.0, and executed on a PowerPC G4 1.25-GHz processor.

on the frequency of the partial. For low frequencies, PASS will be preferred. Also, the DR method will take advantage of the priority queue to schedule its optimal update times (see Section 5). At each update time, for the concerned oscillator the decision of switching from PASS to DR or from DR to PASS could be decided. And since at this update time, the amplitude, frequency, and also phase of the oscillator is known, the switch of method is really straightforward.

8. REFERENCES

- [1] J. A. Moorer, "Signal processing aspects of computer music – a survey," *Computer Music J.*, vol. 1, no. 1, pp. 4–37, 1977.
- [2] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [3] X. Serra and J. O. Smith, "Spectral Modeling Synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music J.*, vol. 14, no. 4, pp. 12–24, 1990.
- [4] J. W. Gordon and J. O. Smith, "A sine generation algorithm for VLSI applications," in *Proc. Int. Comp. Music Conf. (ICMC'85)*, Burnaby, Canada, 1985, pp. 165–168.
- [5] J. O. Smith and P. R. Cook, "The second-order digital wavetable oscillator," in *Proc. Int. Comp. Music Conf. (ICMC'92)*, San Francisco, USA, 1992, pp. 150–153.
- [6] S. Marchand and R. Strandh, "InSpect and ReSpect: Spectral modeling, analysis and real-time synthesis software tools for researchers and composers," in *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, 1999, pp. 341–344.
- [7] S. Marchand, "Sound models for computer music (analysis, transformation, and synthesis of musical sound)," Ph.D. dissertation, University of Bordeaux 1, France, 2000.
- [8] M. Lagrange and S. Marchand, "Real-time additive synthesis of sound by taking advantage of psychoacoustics," in *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, 2001, pp. 5–9.
- [9] A. Freed, X. Rodet, and P. Depalle, "Synthesis and control of hundreds of sinusoidal partials on a desktop computer without custom hardware," in *Proc. ICSPAT*, 1992, pp. 98–101.
- [10] Y. Ding and X. Qian, "Processing of musical tones using a combined quadratic polynomial-phase sinusoid and residual (QUASAR) signal model," *J. Audio Eng. Soc.*, vol. 45, no. 7/8, pp. 571–584, 1997.
- [11] L. Girin, S. Marchand, J. di Martino, A. Röbel, and G. Peeters, "Comparing the order of a polynomial phase model for the synthesis of quasi-harmonic audio signals," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 2003, pp. 193–196.
- [12] M. Raspaud, S. Marchand, and L. Girin, "A generalized polynomial and sinusoidal model for partial tracking and time stretching," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005, pp. 24–29.
- [13] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, ser. Series in Computer Science and Information Processing. Addison-Wesley, 1983, pp. 392–407.
- [14] R. Strandh and S. Marchand, "Real-time generation of sound from parameters of additive synthesis," in *Proc. Journées d'Informatique Musicale*, 1999, pp. 83–88.
- [15] N. Meine and H. Purnhagen, "Fast sinusoid synthesis for MPEG-4 HILN parametric audio decoding," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 239–244.

SYNTHESIS BY ARCS ON THE UNIT CIRCLE

Vittorio Cafagna[#], Domenico Vicinanza^{#, b}

[#]DMI - Musica Inaudita, University of Salerno, Italy

^bCERN, Genève, Switzerland

{cafagna|dvcinanza}@unisa.it

ABSTRACT

Take a set of n arcs on the unit circle and consider the characteristic function of the set of n arcs. Think of this as a waveform, taking only the values 0 and 1. Produce a sound by generating a wavetable by sample and hold. You can think of this procedure as a very simple synthesizer. The question is: which sounds can be generated in this way. The surprising answer is: all of them. Thanks to a flourishing of very interesting papers from the Twenties to the early Fifties of last century, by Szegő, Verblunsky, Szasz, Friedman, Geronimus and Ghizzetti, among others, one can prove that it is possible to define sets of n arcs, whose characteristic functions have preassigned Fourier coefficients c_0, \dots, c_{n-1} . In this paper we introduce such a synthesis procedure, give a sketch of the theorem, on which the procedure rests and highlight differences and similarities with Fourier synthesis and Shannon's sampling theorem. We also discuss the mechanical principles on which a device performing this new kind of synthesis rests.

1. INTRODUCTION

Let $S^1 = \{z \in \mathbb{C} : |z| = 1\}$ be the unit circle in the complex plane. As is customary, we think of S^1 as the quotient $\mathbb{R}/2\pi\mathbb{Z}$, or, equivalently, as the unique compact manifold whose universal cover is \mathbb{R} , the covering map being obviously $\mathbb{R} \ni t \mapsto e^{it} \in S^1$. When the covering map is restricted to a single *sheet* of the covering, e.g. $[0, 2\pi)$, the map is bijective. Therefore we feel free to think of a function $f : S^1 \rightarrow \mathbb{C}$ as a 2π -periodic function $f : \mathbb{R} \rightarrow \mathbb{C}$ and to think of the integral of a summable f on S^1 , i.e. measurable with respect to μ , the normalized ($\mu(S^1) = 1$) Lebesgue measure on S^1 , $\int_{S^1} f d\mu$ as $\frac{1}{2\pi} \int_0^{2\pi} f(t) dt$. (By *abus de langage* we will keep calling by the same name f both the function on the universal cover \mathbb{R} and the function on the covered manifold S^1 .) From a mathematical point of view, a periodic waveform is a summable function f on S^1 . Assume also that the amplitude is finite, so that f is bounded. After rescaling, you can of course assume that $0 \leq f(t) \leq 1$. Let \mathbb{I} be the unit interval $[0, 1]$ and define $L^1(S^1, \mathbb{I})$ as the set of summable functions with values in \mathbb{I} . Remark that $L^1(S^1, \mathbb{I})$ is not a vector space: all you can use from a functional analysis point of view is the structure of a metric space inherited from that of $L^1(S^1, \mathbb{R})$. In this sense, the theory which we are going to discuss is essentially nonlinear. To every $f \in L^1(S^1, \mathbb{I})$ associate the sequence $\{c_k(f)\}_{k \in \mathbb{Z}}$ of its Fourier coefficients defined by

$$c_k(f) = \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{ikt} dt$$

Remark that, being f \mathbb{R} -valued, one has $c_{-k} = \bar{c}_k$. Remark also that we adopted the convention of calling c_k what is more fre-

quently indicated with c_{-k} . The choice is of course, conceptually irrelevant.

Consider now the very special functions in $L^1(S^1, \mathbb{I})$ which are the characteristic functions of a set of n arcs on the unit circle. To fix the notations, let ξ_r, η_r , ($r = 1, \dots, n$) be $2n$ real numbers such that $0 = \xi_1 < \eta_1 < \dots < \xi_n < \eta_n < 2\pi$ and associate to them n arcs on the unit circle whose endpoints are, respectively, $e^{i\xi_r}$ and $e^{i\eta_r}$, ($r = 1, \dots, n$). Let χ be the characteristic function of the set of arcs, defined (on the universal cover $[0, 2\pi)$) by

$$\chi(t) = \begin{cases} 1 & \text{if } t \in \cup_{r=1}^n [\xi_r, \eta_r] \\ 0 & \text{otherwise} \end{cases}$$

From now on we will refer to a characteristic function of n arcs, as a n -rectangular wave. We are now in a position to state the following

Theorem 1 *If $f \in L^1(S^1, \mathbb{I})$, then there exist a sequence $\{\chi_n\}_{n \in \mathbb{N}}$ of n -rectangular waves such that*

$$c_k(f) = c_k(\chi_n), \quad k = 0, \dots, n-1.$$

The sequence $\{\chi_n\}_{n \in \mathbb{N}}$ is not unique. Actually each χ_n can be chosen freely in a 1-parameter family $\{\chi_{n,\lambda}\}$, where $\lambda \in [0, 2\pi)$.

Theorem 2 *Each of the sequences $\{\chi_{n,\lambda}\}_{n \in \mathbb{N}, \lambda \in [0, 2\pi)}$, whose existence is assured by Theorem 1, weak * converges to f in L^1 .*

The consequences for sound synthesis are rather surprising: you can approximate any spectrum up to order n with a spectrum of a very simple rectangular waveform, taking only the values 0 and 1. It is rather easy to implement a software able to associate to any set of n arcs the sound produced by the characteristic function. The situation is somewhat similar to that of FM synthesis: one has a machine potentially capable of producing rich spectra by means of a very simple device. The problem is to *tame the monster*, that is to say, to find procedures to orient the timbre in the desired directions, by manipulating the parameters. Said that, there is a big difference with respect to FM synthesis: in this case, in principle, one is able to synthesize every possible sound, while, as far as we know, no characterization of FM sounds is available. We refer to [1] for a discussion of some aspects of the intricacies of mathematical FM theory.

In the next section we will indicate the main steps of a proof of Theorem 1. The exposition is very sketchy and uneven: we indulge in some easy, but, to our advice illuminating, computations and completely bypass some difficult steps. The intention is to convey the flavor of the mathematical tools employed and to highlight the noticeable fact that the proof is actually constructive.

A totally different proof of Theorem 1 can be given, following the guideline of Friedman ([2], see also [3]), which uses differential topology, instead of complex analysis, as the main tool. The differential-topological proof is maybe simpler and has the advantage to allow a remarkable generalization to compact manifolds other than S^1 . The big drawback, as far as sound synthesis is concerned, is that it is not constructive and does not, at least for the moment, lead to implementable algorithms.

We do not discuss the proof of Theorem 2. It is just a brilliant exercise in pure mathematical analysis and does not seem (once more, at least for the moment) to shed any light on the synthesis algorithm. For the interested reader, we refer to the original proofs of Verblunsky [4] and Ghizzetti [5].

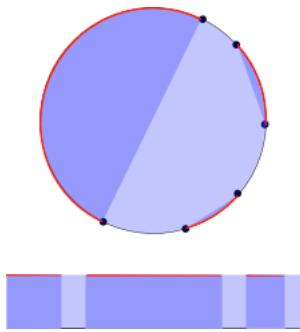


Figure 1: Points defining intervals on the unit circle and associated rectangular wave (left boundaries: 0.0, 0.2, 0.8, right boundaries: 0.1, 0.7, 0.87).

2. SKETCH OF THE PROOF OF THEOREM 1

We outline the main steps of the proof of the theorem. For the (many) missing details, we refer to the original papers of Szegő [6, 7], Verblunsky [4], Szasz [8], Friedman [2], Geronimus [9] and Ghizzetti [5], or to the (somewhat more friendly) rational reconstruction contained in the PhD thesis of G. Caterina [10] (mainly concerned with the, also rather surprising, implications of the theorem for fuzzy logic.)

2.1. Fourier coefficients of n -rectangular waves

Let us start by computing the Fourier coefficients of a n -rectangular wave χ defined by the arcs $[e^{i\xi_r}, e^{i\eta_r}] \subset S^1$, ($r = 1, \dots, n$):

$$\begin{aligned} c_0(\chi) &= \frac{1}{2\pi} \int_0^{2\pi} \chi(t) dt = \frac{1}{2\pi} \sum_{r=1}^n (\eta_r - \xi_r), \\ c_k(\chi) &= \frac{1}{2\pi} \int_0^{2\pi} \chi(t) e^{ikt} dt \\ &= \frac{1}{2\pi k i} \sum_{r=1}^n (e^{ik\eta_r} - e^{ik\xi_r}), \quad k \in \mathbb{Z}^*. \end{aligned}$$

To prove the theorem means, given n numbers c_0, \dots, c_{n-1} (which are the Fourier coefficients of a $f \in L^1(S^1, \mathbb{I})$). For a discussion of the necessary conditions see the references above), to find $2n - 1$ numbers $0 = \xi_1 < \eta_1 < \xi_2 < \eta_2 \dots < \xi_n < \eta_n <$

2π , such that

$$\begin{cases} \sum_{r=1}^n (\eta_r - \xi_r) &= 2\pi c_0 \\ \sum_{r=1}^n (e^{i\eta_r} - e^{i\xi_r}) &= 2\pi i c_1 \\ \vdots &\vdots \\ \sum_{r=1}^n (e^{i(n-1)\eta_r} - e^{i(n-1)\xi_r}) &= 2\pi i c_{n-1} \end{cases}$$

We are not aware of any simple way of solving these n nonlinear equations in $2n - 1$ unknown. The solution which we will describe requires a rather complicated machinery.

2.2. A nonlinear Fourier transform

To every $f \in L^1(S^1, \mathbb{I})$, we associate a function h_f holomorphic in the unit circle $D = \{z \in \mathbb{C} : |z| < 1\}$, defined by $h_f(z) = \exp(-2\pi i \sum_{k=1}^{\infty} c_k(f) z^k)$. Consider the power series expansion $h_f = \sum_{k=1}^{\infty} s_k z^k$ and define a sequence $s : \mathbb{Z} \rightarrow \mathbb{C}$ by $s(k) = s_k$, $k > 0$, $s(0) = 2 \sin \pi c_0$ and $s(-k) = \bar{s}_k$. From now on we will refer to s_k ($k \in \mathbb{Z}$) as the k -th *nonlinear coefficient* of f . Taking the logarithmic derivative of both sides of the identity

$$e^{-2\pi i \sum_{k=1}^{\infty} c_k z^k} = \sum_{k=1}^{\infty} s_k z^k$$

one obtains

$$-2\pi i \left(\sum_{k=1}^{\infty} k c_k z^{k-1} \right) \left(\sum_{k=1}^{\infty} s_k z^k \right) = \sum_{k=1}^{\infty} k s_k z^{k-1}.$$

From the last identity one sees that the nonlinear coefficients are given as polynomials in the Fourier coefficients, and, reciprocally, that the Fourier coefficients are given as polynomials in the nonlinear coefficients:

$$s_k = P(c_1, \dots, c_{k-1}), \quad c_k = Q(s_1, \dots, s_{k-1})$$

Let us call \mathcal{N} the map which associates to a function f its nonlinear coefficients s_k . One could think of this map as a *nonlinear Fourier transform* and of the set of nonlinear coefficients as a *nonlinear spectrum*. The abstract scheme is illustrated by the following commutative diagram

$$\begin{array}{ccc} L^1(S^1, \mathbb{I}) & & \\ \downarrow & \searrow & \\ \mathbb{C}_H^{\mathbb{Z}} & \longrightarrow & \mathbb{C}_H^{\mathbb{Z}} \end{array}$$

where $\mathbb{C}_H^{\mathbb{Z}}$ is the space of all the hermitian sequences, the vertical arrow is the Fourier transform, the diagonal arrow the nonlinear Fourier transform \mathcal{N} and the horizontal arrow the algebraic map between Fourier and nonlinear coefficients, given by the polynomial P .

2.3. The holomorphic function associated to a n -rectangular wave

A first hint on the usefulness of the procedure described above comes from an explicit calculation of the holomorphic function associated to a n -rectangular wave. Let us, as usual, denote by χ the rectangular wave defined by the arcs $[e^{i\xi_r}, e^{i\eta_r}] \subset S^1$, ($r = 1, \dots, n$):

$1, \dots, n$). Then the associated holomorphic function h_χ is given by $\exp(-2\pi i \sum_{k=1}^{\infty} c_k(\chi) z^k)$. Let us start by computing:

$$\begin{aligned} \sum_{k=1}^{\infty} c_k(\chi) z^k &= \sum_{k=1}^{\infty} \left(-\frac{1}{2\pi k i} \sum_{r=1}^n \left(-e^{ik\eta_r} + e^{ik\xi_r} \right) z^k \right) \\ &= -\frac{1}{2\pi i} \sum_{r=1}^n \sum_{k=1}^{\infty} \left(-e^{ik\eta_r} + e^{ik\xi_r} \right) \frac{z^k}{k} \\ &= -\frac{1}{2\pi i} \sum_{r=1}^n \sum_{k=1}^{\infty} \left(\frac{(-ze^{i\eta_r})^k}{k} - \frac{(-ze^{i\xi_r})^k}{k} \right) \\ &= -\frac{1}{2\pi i} \sum_{r=1}^n \sum_{k=0}^{\infty} (-1)^k \left(\frac{(-ze^{i\eta_r})^{k+1}}{k+1} - \frac{(-ze^{i\xi_r})^{k+1}}{k+1} \right) \end{aligned}$$

Recalling that, for $|w| < 1$,

$$\ln(w+1) = \sum_{k=0}^{\infty} (-1)^k \frac{w^{k+1}}{k+1}$$

and applying it to, respectively, $w = -ze^{i\eta_r}$ and $w = -ze^{i\xi_r}$, $|z| < 1$, we see that the last expression in our computation is equal to

$$\begin{aligned} \frac{1}{2\pi i} \sum_{r=1}^n \ln(-ze^{i\eta_r} + 1) - \ln(-ze^{i\xi_r} + 1) &= \\ -\frac{1}{2\pi i} \sum_{r=1}^n \ln \frac{1 - ze^{i\eta_r}}{1 - ze^{i\xi_r}}. \end{aligned}$$

Exponentiating allows to get rid of the unpleasant logarithm in the last formula. To sum up, for the holomorphic function associated to the rectangular wave χ one computes the expression

$$\begin{aligned} \exp \left(\sum_{r=1}^n \ln \frac{1 - ze^{i\eta_r}}{1 - ze^{i\xi_r}} \right) &= \prod_{r=1}^n \exp \left(\ln \frac{1 - ze^{i\eta_r}}{1 - ze^{i\xi_r}} \right) \\ &= \prod_{r=1}^n \frac{1 - ze^{i\eta_r}}{1 - ze^{i\xi_r}} \end{aligned}$$

which is indeed very nice. Observe that the function, holomorphic on the (open) disk, has poles at the left endpoints and zeroes at the right endpoints of the arcs.

2.4. Toeplitz matrices of nonlinear coefficients

Associate to the nonlinear coefficients s_k , ($k \in \mathbb{Z}$) of a $f \in L^1(S^1, \mathbb{I})$ a sequence $\{T_k\}_{k \in \mathbb{N}^*}$ of $(k+1 \times k+1)$ Toeplitz matrices (i.e. constant on the NW to SE diagonals), by

$$T_k = \begin{pmatrix} s_0 & s_1 & s_2 & \dots & s_k \\ s_{-1} & s_0 & s_1 & \dots & s_{k-1} \\ \dots & \dots & \dots & \dots & \dots \\ s_{-k} & s_{-k+1} & s_{-k+2} & \dots & s_0 \end{pmatrix}$$

Observe that T_k is hermitian and therefore $D_k = \det T_k$ is a real number. One can use the Toeplitz determinants D_k to characterize the n -rectangular waves, according to the following

Proposition 1 $f \in L^1(S^1, \mathbb{I})$ coincides with a n -rectangular wave almost everywhere if and only if $D_k > 0$ for $k = 0, 1, \dots, n-1$ and $D_k = 0$, $\forall k \geq n$. If this is not the case, then $D_k > 0$, $\forall k \in \mathbb{N}$.

The above proposition establishes a bijection between n -rectangular waves and collections of n Toeplitz matrices verifying the above conditions. But one can prove more. An explicit description of the bijection is given by the following

Proposition 2 The n -rectangular wave associated to the collection T_k , ($k = 1, \dots, n$) of Toeplitz matrices verifying the conditions of Proposition 1, is the one defined by the arcs $[e^{i\xi_r}, e^{i\eta_r}]$, whose left endpoints $e^{i\xi_r}$ and right endpoints $e^{i\eta_r}$ are, respectively, the roots of the following two polynomials of degree n :

$$\begin{vmatrix} 1 & z & z^2 & \dots & z^n \\ s_0 & s_1 & s_2 & \dots & s_n \\ s_{-1} & s_0 & s_1 & \dots & s_{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ s_{-n+1} & s_{-n+2} & s_{-n+3} & \dots & s_1 \end{vmatrix},$$

$$\sum_{r=1}^n \mu_r e^{i\xi_r} \prod_{s=1}^n (z - e^{i\xi_s})$$

(the superscript (r) meaning that the r -th factor is omitted) where the coefficients μ_r are the solutions (real and positive) of the linear system

$$s_k = \sum_{r=1}^n \mu_r e^{ik\xi_r}.$$

2.5. Construction of n -rectangular waves with first n Fourier coefficients preassigned

Let us remark that the intractable problem of solving the nonlinear equations relating the Fourier coefficients to the geometric data of rectangular waves has been translated, thanks to the nonlinear Fourier machinery, into a purely algebraic problem about Toeplitz matrices. Namely, given a sequence $\{T_k\}_{k \in \mathbb{N}^*}$, such that $D_k > 0$, $\forall k$, to find $\forall n$, a complex number \tilde{s}_n , such that the $(n+1) \times (n+1)$ matrix \tilde{T}_n defined by

$$\tilde{T}_n = \begin{pmatrix} s_0 & s_1 & s_2 & \dots & \tilde{s}_n \\ s_{-1} & s_0 & s_1 & \dots & s_{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ \tilde{s}_{-n} & s_{-n+1} & s_{-n+2} & \dots & s_0 \end{pmatrix}$$

has the determinant $\tilde{D}_n = 0$. Done that, the n -rectangular function associated to the Toeplitz collection $T_1, T_2, \dots, T_{n-1}, \tilde{T}_n$ will have the same nonlinear coefficients (and therefore the same Fourier coefficients) as the function $f \in L^1(S^1, \mathbb{I})$, whose nonlinear coefficients are the entries of the matrices T_k .

To solve this algebraic problem, it is useful to associate to every T_k a new $(k \times k)$ Toeplitz matrix W_k , obtained from T_k by suppressing the first column and the last line:

$$W_k = \begin{pmatrix} s_1 & s_2 & s_3 & \dots & s_k \\ s_0 & s_1 & s_2 & \dots & s_{k-1} \\ \dots & \dots & \dots & \dots & \dots \\ s_{-k+2} & s_{-k+3} & s_{-k+4} & \dots & s_1 \end{pmatrix}$$

Observe that the W_k 's are not hermitian, so that their determinants $F_k = \det W_k$ are complex numbers. Nonetheless, one has the following relationship between absolute values of D_k 's and F_k 's:

Lemma 1 $|F_k| = |D_{k-1}|$. More explicitly, there exists a $\lambda \in [0, 2\pi]$, such that $F_k = e^{i\lambda} D_{k-1}$.

From the lemma, one immediately has that $\forall \lambda \in [0, 2\pi)$ there is one and only one solution to the following equation in the unknown s

$$\begin{vmatrix} s_1 & s_2 & s_3 & \dots & s \\ s_0 & s_1 & s_2 & \dots & s_{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ s_{-n+2} & s_{-n+3} & s_{-n+4} & \dots & s_1 \end{vmatrix} = e^{i\lambda} \begin{vmatrix} s_0 & s_1 & s_2 & \dots & s_{n-1} \\ s_{-1} & s_0 & s_1 & \dots & s_{n-2} \\ \dots & \dots & \dots & \dots & \dots \\ s_{-n+1} & s_{-n+2} & s_{-n+3} & \dots & s_0 \end{vmatrix}$$

Putting in the matrix \tilde{T}_n , \tilde{s}_n equal to the solution of the equation above, and using the well-known (in the Toeplitz world) fact that $D_{n-1}^2 - D_{n-1}\tilde{D}_n = |F_n|$, one concludes that $D_n = 0$, and the theorem is proved.

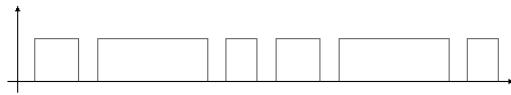


Figure 2: Rectangular waveform associated to the set of arcs shown in fig. 1.

3. DIFFERENCES AND SIMILARITIES WITH FOURIER SYNTHESIS

In Fourier analysis one has a set of *elementary* signals, the trigonometric functions, by means of which one reconstructs every periodic signal. In our case, the elementary signals are the characteristic functions of sets of n arcs on the unit circle, by means of which one can reconstruct every bounded periodic signal. Reconstruction in Fourier world means that a signal is an infinite linear combination of elementary signals. In our case the reconstruction is completely different. We still have a sequence of elementary signals, completely determining the original signal, but, instead of obtaining the original signal as the sum of a series, one obtains it by a limit process: the original signal is the weak * limit in L^1 of the elementary signals.

For what concerns approximation, which is the relevant aspect for sound synthesis, an actual signal is spectrally approximated up to order n by a trigonometric polynomial of order n in Fourier synthesis, while it is approximated by a n -rectangular wave in our case. The obvious difference being that, while a rectangular wave is a much simpler function, from both algebraic and computational points of view, a trigonometric polynomial is band-limited, while a rectangular wave is not. Therefore, while in Fourier case one has a simple estimate of the approximation error: the tail of the series of the original signal, in our case the error is given by a much more complicated object: the difference of the tails of the original series and the series of the n -rectangular wave.

This brings in a third fundamental difference between ours and the Fourier scheme: while the approximating trigonometric polynomial of order n is unique, in our case, according to the theorem, one has a choice of the approximating n -rectangular wave, depending on a parameter $\lambda \in [0, 2\pi)$. The wide-open question is how to use this freedom in the choice of the approximating rectangular waves to the scope of minimizing the error. The problem

seems to be a difficult theoretical one. Some first results are contained in [11].

Experimenting with sounds generated by n -rectangular waves is currently carried over at the sound laboratory *Musica Inaudita* at the University of Salerno, by means of a synthesizer, called *Arcosinti*¹, whose mechanical principles are discussed in section 5.

4. FORMAL ANALOGIES WITH SHANNON'S SAMPLING THEOREM

There is an interesting formal analogy between Theorems 1 and 2, and Shannon's sampling theorem that seems to deserve some comments. While the concrete framework is totally different (in Shannon's case one deals with bandlimited signals on the real line, in ours with, so to say, *stripvalued* periodic signals) there is a common abstract framework. In both cases you restrict to a subset of signals and you end up with a reconstruction theorem in terms of a privileged class of signals (*sinc* functions and rectangular waves, respectively.)

The reconstruction is, of course, totally different: a bandlimited signal as the sum of a series of *sinc* functions, having as coefficients samplings of the signal, the stripvalued signal as weak * limit of a sequence of rectangular waves.

Yet the philosophy is similar: selecting a special class of signals which can be totally described in terms of a very special *elementary* subclass.

5. N-RECTANGULAR WAVE GENERATION: A MECHANICAL INSIGHT

The theorem stated in this paper suggests the possibility to approximate a given spectrum with a rectangular wave obtained as a characteristic function of a set of arcs on the unit circle.

From a purely mechanical point of view the problem of a good approximation (up to the n -th term) of the Fourier spectrum, is turned to the problem of the production of such a mechanical wave. The more flexible (customizable) and practical way to build a rectangular wave is by using the diaphragm of a loudspeaker, moving between two fixed endpoints. Think of these two endpoints as the rest position of the membrane and its position at a fixed mechanical tension. The rest position corresponds to a zero driving current (the loudspeaker electromagnet, when the driving current is zero, is inactive so the membrane is naturally relaxed). The fixed position corresponds to a fixed-value driving current, for example the maximum elongation (i.e. the maximum strength of the diaphragm) corresponds to the maximum current with which it is possible to feed the device. Using a loudspeaker as a transducer *electric current → position of a mechanical membrane*, we changed the problem to the one of producing a current which changes between two values. The changes in time of the current between the two endpoints will drive automatically switches of the membrane between two positions at the same rate, producing a rectangular wave. The limits on the frequencies physically obtainable with this method are of course the ones of the response in time of the membrane.

For example, to build a square wave whose period is 10 ms (frequency $f = 1/0.01 = 100$ Hz), duty cycle 50% (i.e. the wave assumes the value 1 for one half of the period and 0 for the

¹The name intends to be a homage to the architect Paolo Soleri, the visionary builder of the utopian Renaissance-like city of Arcosanti in the Arizona desert.

second half), we have to produce a current which has exactly the same properties, alternating between zero and a fixed value with the same duty cycle and in the same time intervals. Maybe the best (surely one of the most customizable and cost effective) way to generate such a current, able to correctly drive loudspeakers, is by employing a computer-controlled sound card. The personal computer, through a software description of the waveform will instruct the sound card to produce the desired current. Sound cards take as input a stream of digital values (representing the waveform) from a program and through an interpolation process (typically linear fit between adjacent values, or sample and hold) it produces the driving current. Suppose we are working with a 44.1 KHz sound card, and suppose that we want to build the 50% duty cycle rectangular wave, we were talking about above. 44.1 KHz means that one second of signal is described by a series of 44100 values. To tell the sound card to produce the right switching current we have to provide a sequences of 22050 ones followed by a sequence of 22050 zeroes.

Summarizing the whole process to produce a mechanical n -rectangular wave, we start from a program (the virtual synthesizer) able to produce the right series of values to represent (taking into account the work frequency of the sound card) the rectangular wave; the sound card will translate this series (through a sample-and-hold procedure) into a current with exactly the same functional profile; finally the loudspeaker will turn this current into a mechanical wave.

It could be useful to spend a few more words about the sound card working frequency. If f_{sc} (sc stands for *sound card*) is such a frequency, the maximum number of variations per second we can describe is $f_{sc}/2$. For example, if $f_{sc} = 44100$, we cannot switch between 0 and 1 more than 22050 times per second (i.e. 1010101010...) The upper limit for the switching frequency has a direct influence on the accuracy of the spectral approximation capabilities of the synthesized waveform. The Fourier spectrum of the n -rectangular wave generated following the procedure described in the proof of the theorem, has exactly the first n spectral coefficients identical to the ones of the function $f \in L^1(S^1, \mathbb{I})$. Each arc is identified by its endpoints, so to each arc are associated two flips in the characteristic function diagram, the first one from 0 to 1, and the second one from 1 to 0. If the working frequency of the sound card is f_{sc} , the shortest possible arc describable is one of length $1/f_{sc}$. Therefore we cannot have more than $f_{sc}/2$ arcs of length $1/f_{sc}$, spaced by minimal intervals of length $1/f_{sc}$. If the sound card frequency is 44100 Hz, this means that the maximum number of arcs is 22050. This is also the upper limit to the number of Fourier coefficients that the rectangular waveform will have in common with the function. The spectral accuracy limit is set to $f_{sc}/2$ number of partials, which allows for a more than satisfactory synthesis procedure, even on a typical PC sound card.

6. CONCLUSIONS

We proposed a new sound synthesis method based on a rather complicated pure-mathematical machinery. Still, we think that the outcome is worth the price: a virtual machine able to synthesize any sound with *rectangular waves*, the simplest possible functions from a logical point of view (boolean functions). We discussed differences and similarities with additive synthesis and underlined some possible draw-backs of our method. A lot more of experimental and theoretical work is required. The good news, from a pure-mathematical point of view, are that the framework in which

the theorem we stated stands, the Szegő theory of trigonometric polynomials on the unit circle, longtime considered as a beautiful but sort of a marginal subject, is now back in the mainstream of mathematical research. This opens up the possibility, for people interested in sound synthesis, to go *fishig* for hints in the hundreds of papers published in the last few years on the subject (see the recent authoritative treatise of Barry Simon [12, 13], in two volumes and more than thousand pages.) On the experimental side, a software implementation in Java is under development and will be available at <http://www.musicinaudita.it/arcosinti>. At the same URL, one will be also able to find the sonic results of ongoing playing-around with distributions of arcs on the unit circle. Along with the virtual synthesizer, a hardware implementation with discrete components ($2n$ monostables) is currently under study, in collaboration with A. De Nardo and G. Lisi of the department of Electronic Engineering (DIIIE) of the University of Salerno.

7. REFERENCES

- [1] V. Cafagna and D. Vicinanza, "Synthesis by mathematical models: elliptic functions and lacunary series," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 339–344.
- [2] B. Friedman, "Fourier coefficients of bounded functions," *Bull. Amer. Math. Soc.*, vol. 47, pp. 84–92, 1941.
- [3] V. Cafagna and G. Caterina, "On a theorem of B. Friedman," in preparation.
- [4] S. Verblunsky, "On the Fourier constants of a bounded function," *Proc. Cambridge Philos. Soc.*, vol. 32, pp. 201–211, 1936.
- [5] A. Ghizzetti, "Sui coefficienti di Fourier di una funzione limitata, compresa fra limiti assegnati," *Ann. Scuola Norm. Sup. Pisa*, vol. 4, pp. 131–156, 1950.
- [6] G. Szegő, "Beiträge zur theorie der Toeplitzschen formen, I," *Math. Z.*, vol. 6, pp. 167–202, 1920.
- [7] ——, "Beiträge zur theorie der Toeplitzschen formen, II," *Math. Z.*, vol. 9, pp. 167–190, 1921.
- [8] O. Szasz, "On the partial sums of certain Fourier series," *Amer. J. Math.*, vol. 59, no. 3, pp. 696–708, 1937.
- [9] J. Geronimus, "On the trigonometric moment problem," *Ann. of Math.*, vol. 47, no. 4, pp. 742–761, 1946.
- [10] G. Caterina, "Fuzzy sets: some analytic, algebraic and geometric aspects," Ph.D. dissertation, University of Salerno, Mar. 2006.
- [11] V. Cafagna, G. Caterina, and D. Vicinanza, "Strategies of spectral approximation by boolean functions," in preparation.
- [12] B. Simon, *Orthogonal Polynomials on the Unit Circle, Part 1: Classical Theory*, ser. AMS Colloquium Publications. Providence, RI: American Mathematical Society, 2004, vol. 54.
- [13] ——, *Orthogonal Polynomials on the Unit Circle, Part 2: Spectral Theory*, ser. AMS Colloquium Publications. Providence, RI: American Mathematical Society, 2004, vol. 54.

CIRCLE MAPS AS A SIMPLE OSCILLATORS FOR COMPLEX BEHAVIOR: II. EXPERIMENTS

Georg Essl

Deutsche Telekom Laboratories, TU-Berlin
georg.essl@telekom.de

ABSTRACT

The circle map is a general non-linear iterated function that maps the circle onto itself. In its standard form it can be interpreted as a simple sinusoidal oscillator which is perturbed by a non-linear term. By varying the strength of the non-linear contribution a rich array of non-linear responses can be achieved, including wave-shaping, pitch-bending, period-doubling and highly irregular patterns. We describe a number of such examples and discuss their subjective auditory perception.

1. INTRODUCTION

Circle maps are a particularly simple yet rather general example of a mapping that exhibits many important aspects of complex dynamical behavior. A circle map is capable of demonstrating such behaviors as mode and phase-locking, period doubling and subharmonics, quasi-periodicity as well as routes to chaos via repeated period doubling or via disruption to quasi-periodicity [1].

Circle maps are also attractive because they have served as an important “simplest case” example of iterated dynamics in the study of these dynamics among mathematicians and physicists. They also are related to already proposed sound synthesis methods that worry about introducing functional iterations or non-linearities.

The circle map is particularly suitable for the study and generation of sustained undamped sounds as the map confines the space of possible iterations exactly to functions of this nature by construction.

This paper is the second in a series of papers describing the circle map for sound synthesis purposes. The aim is to describe the properties in a systematic fashion to allow easy use. Additionally it is an example of classifying synthesis methods with respect to perturbation of parameters. The purpose of this paper is to discuss computational results of the circle map. Specifically we are interested in properties which are relevant for the use of the circle map for sound synthesis. The basic properties of the circle map were already described in [2] and we will only give a brief review of the method here and refer the reader to this reference for more detailed introductory discussions.

2. BACKGROUND

Non-linearities have played an ongoing important role since very early. Risset introduced [3], and Arfib and Le Brun refined wave-shaping, a method where a pre-existing signal would be fed through a non-linear function, hence modifying the sound [4, 5]. The method is able to create complex though generally only perfectly non-chaotic, periodic signals and the control is well understood.

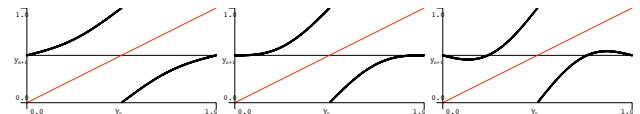


Figure 1: The mapping $y_{n+1} = \phi(y_n)$ for the standard circle map at values $k = 0.5$ (left), $k = 1.0$ (center) and $k = 1.5$. The horizontal line at y_{n+1} marks the boundary between invertible and non-invertible maps. At $k = 1$ the map becomes tangent to this limit line. For $k = 1.5$ the line is crossed more than once and the map is non-invertible.

Chaos itself became a focus of attention in the late 80s and early 90s. The use of iterated functions that can lead to rich non-linear and chaotic behavior falls into broad categories: (1) The use of periodic pattern in the generation of music structure and (2) for direct sound synthesis purposes. Within the first category Pressing studied logistic maps [6]. Gogins [7] investigated randomly switched sets of functions in his iterations. Bidlack introduced physically motivated maps of either dissipative or conservative character using Lorenz-type and Hénon-Heiles type iterations [8]. The second category was developed by Truax [9] and Di Scipio [10, 11] motivated directly by iterated maps. DiScipio considers what he calls the sine map, an iterated sinusoid without coupling to a linear function. Rodet considered Chua’s network and its time-delayed extension for sound synthesis who also draws connections to nonlinearities in a physical context [12, and references therein]. Dobson and Fitch considered iterated complex quadratic maps [13] experimentally. Manzolli et al consider a set of two-variable iterations which are variations of the so-called *standard map* which in turn is related to the circle map [14]. Recently Valsamakis and Miranda consider a family of two variable coupled oscillator with sine waves in the feedback loop [15]. The most widely cited reference of chaos theory is the computer music literature is [16]. It does contain a description of the circle map but gives little interpretation or motivation of the map. Maybe for the lack of emphasis of the specific properties of the circle map, it has not been widely considered as a desirable model for iterative synthesis and sequence construction in the above mentioned literature.

3. ITERATED MAPS FROM THE CIRCLE TO ITSELF

The most general form of the circle map is

$$y_{n+1} = \phi(y_n) \quad (1)$$

where the defining property is that ϕ is a mapping from a bound interval to a bound interval of the same size, or alternatively

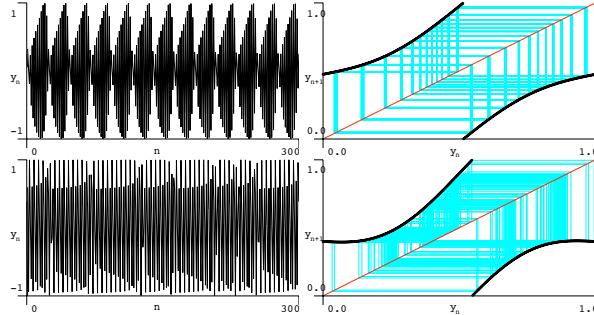


Figure 2: Top: Beating pattern at $\Omega = 0.475$, $k = 0.6$, Bottom: Jitter pattern at $\Omega = 0.4$, $k = 1.1$. Left shows the waveforms, right shows the mapping function and the iteration trace. The diagonal line are the fixed point locus.

of a periodically repeating interval. A periodically repeating interval is topologically equivalent to a circle, hence the name of the map.

If we want to model a perfectly sinusoidal oscillator that is perturbed by some coupled non-linear function, this turns into:

$$y_{n+1} = \left(y_n + \Omega - \frac{k}{2\pi} f(y_n) \right) \mod 1 \quad (2)$$

where Ω is a constant that is the fixed angular progression of the sinusoidal oscillator, and k is the coupling strength of the non-linear perturbation $f(\cdot)$. y_0 is the starting phase. In principle, the choice of $f(\cdot)$ is very flexible and examples of discontinuous functions can be found in the literature as well as smooth cases. We will consider a number of examples later. The canonical theoretical example is the *standard circle map*:

$$y_{n+1} = \left(y_n + \Omega - \frac{k}{2\pi} \sin(2\pi y_n) \right) \mod 1 \quad (3)$$

In order to study the long-term behavior of the iterated map $\phi(\cdot)$ we can look at the *winding number*

$$W = \lim_{n \rightarrow \infty} \frac{y_n - y_0}{n} \quad (4)$$

which measures the average angle added in the long term. If this added angle notated over the interval $[0, 1]$ is a rational number p/q with $p, q \in \mathbb{N}$ then after q iterations we will have a recurrence and hence the map is periodic. Irrational winding numbers are called *quasi-periodic*.

Throughout this paper we will call a response *singular* if any perturbation to the parameters results in a qualitative change of the response, otherwise it is *stable*. We will call a response *generic* if most, but not necessarily all responses under variation of one or more parameter stays qualitatively the same.

We will call a closed path an *orbit*. For orbits where the number of iterations until repetition is known we say the path is an n -orbit where n is a positive integer. We call a path *regular* if it is periodic or quasi-periodic. Highly irregular patterns will loosely be called *chaotic*.

Ω is a phase progression, but of course is essentially the frequency of the unperturbed oscillator which is calculated as $f_\Omega =$

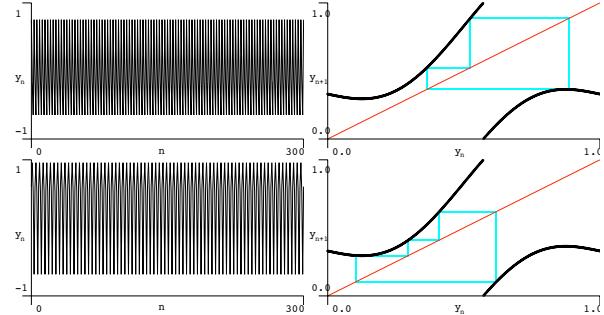


Figure 3: Top: First one two bistable simple orbits at $\Omega = 0.33$, $k = 1.42$, Bottom: Second of two bistable simple orbits at $\Omega = 0.33$, $k = 1.42$. Left shows the waveforms, right shows the mapping function and the iteration trace. The diagonal line are the fixed point locus.

$\Omega \cdot S$ where S is the sampling rate, or time interval between two time steps for $\Omega \in [0, 0.5]$. Therefore we will call Ω a frequency throughout this paper. If $\Omega > 0.5$ we get aliasing and the effective frequency decreases again, which opposite phase sign.

4. OBSERVED WAVEFORMS IN THE PARAMETER PLANE (Ω, K)

The parameter k defines the strength of the influence of the non-linear term is on the overall iteration. If k is small or vanishes, we get behavior close to or equivalent to a pure sinusoidal oscillator. With increasing values of k the non-linear term starts to dominate. The specific change in behavior of k also depends on the choice of the linear oscillator frequency Ω as we shall see below. In this section all rendered examples are for the standard circle map of equation 3.

Waveforms were generated by the result of a given iteration n into a sine function:

$$Y_n = \sin(2\pi y_n) \quad (5)$$

4.1. Quantitative Change with Varied Coupling k

In the case of the standard circle map the behavior with respect to k can be roughly classified into three regions. If $|k| < 1$ then the transfer function is invertible, that is a unique y_n maps to a unique y_{n+1} . At $|k| = 1$ a self-tangency forms and at $|k| > 1$ the mapping isn't invertible anymore as multiple values of y_n map to the same y_{n+1} . This can be seen in Figure 1 where the left case of $k = 0.5$ no self-overlap parallel to the y_n -axis, whereas the right case clearly does. We will call the case $|k| < 1$ example of *small k* whereas the case of $|k| > 1$ constitutes examples of *large k*. The case $|k| = 1$ is singular and not of practical interest.

For the same reason, for values of $|k| < 1$ the sign simply means an inversion of the waveform about the line $y_n = 0$ while for $|k| > 1$ one gets qualitatively different behavior for positive and negative coupling constants.

4.1.1. Behavior for Small k

For small k one can already see non-linear behavior. Most prominent are waveform deformations, reminiscent of wave-shaping or

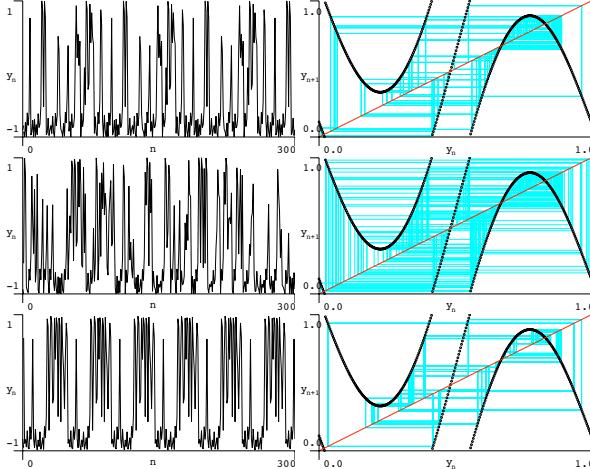


Figure 4: Sensitivity to iteration start position y_0 at $\Omega = 0.11$, $k = 6.4$. Two orbits (top, bottom) are stable, one (center) is chaotic. All orbits are non-singular.

phase distortion synthesis [17]. Also beating can occur. An example of both these phenomena can be seen in the top example of Figure 2.

Another phenomenon of practical interest is the relation of the mapping function to the line of fixed points. In all figures, the center diagonal line depicts the fixed points of the map, that is points where

$$\phi(y_n) = y_{n+1} = y_n = y_f. \quad (6)$$

Clearly once a mapping hits a fixed point it will stick to this point. With respect to fixed points a number of basic properties can again be observed: If the relative angle of the intersection at the fixed point y_f is mild, that is if $f'(y_f) < 1$ then starting points of the iteration in the neighborhood of that fixed point will converge to the fixed point y_f [18, p. 482 for a related discussion for the logistic map]. This has practical implication for the qualitative behavior of the standard circle map. As k is increased, the amplitude of the sinusoidal non-linearity is increased and hence the slope of the intersection that is possible. For small k the slope can generally be expected to be small and hence, generically, $f'(y_f) < 1$ when there is an intersection with the fixed point line. Convergence to fixed points translate into rapid decay to silence and hence these cases are not of practical interest. These silent regions are likely for small values of k and occur whenever the iterated function $\phi(x)$ intersects the fixed point line. If an intersection happens as a function of both Ω and of k , Ω defines a constant offset from the fixed point line, hence larger values of Ω translate into larger possible values of k before intersections happen.

For intuition one may say, that oscillatory waveforms for small k are limited by the frequency Ω . For low frequency oscillation one has less room for adding non-linear contributions than for higher frequencies.

Qualitatively the same is true if other functions than sinusoids are used as the iteration nonlinearity $f(\cdot)$. The defining property is the angle of the function at an intersection with the fixed point line.

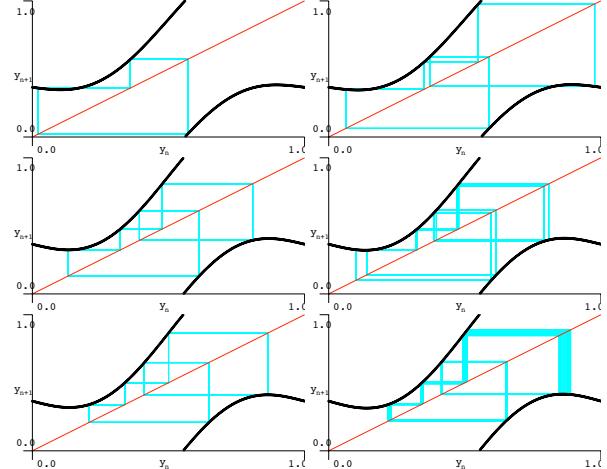


Figure 5: Left to right, top to bottom: Mapping function and iteration trace of period doubling and first complex pattern formation for $\Omega = 0.365$, $k = 1.26, 1.28, 1.48, 1.5, 1.52, 1.56$. The diagonal line are the fixed point locus.

4.1.2. Behavior for Large k

As said earlier, for large k , the map is non-invertible. Also from the previous discussion, the map function $\phi(\cdot)$ is much more likely to intersect the fixed point line. Every actual intersection point the map with the fixed point line:

$$\phi(x_n) = x_{n+1} = x_n \quad (7)$$

forms a sparse set of point for which the iteration results in a constant wave form. Due to the sparsity this does not have many practical implications, though it is important to know that by accident a poorly chosen starting point can result in silence.

For small k silence is a rather frequent response. However, with increasing k , the slope of, for example, the standard circle map at intersection points successively increases. Eventually it exceeds the slope necessary for convergence and non-attracting orbits away from the intersection fixed point become possible again. In fact silence becomes rare with very large k .

This is especially interesting for low frequencies Ω , where the potentially interesting responses of the standard circle map are rather confined. This confinement is lifted for sufficiently large k and the non-linear response also observed for larger Ω can be found equally for low frequencies.

4.2. Some Observed Phenomena

This section discussion some observed phenomena using the circle map, with an eye towards phenomena that are relevant for synthesis. For large k the behavior can change drastically for small perturbations of the parameters and hence at least naive control of the map is very difficult. The ultimate goal is to define ways match parameters with specific phenomenological responses. This task is however, future work and here this paper confines itself to giving examples of parameters that illustrate certain properties are seem interesting for sound synthesis.

Examples of phenomena can be found:

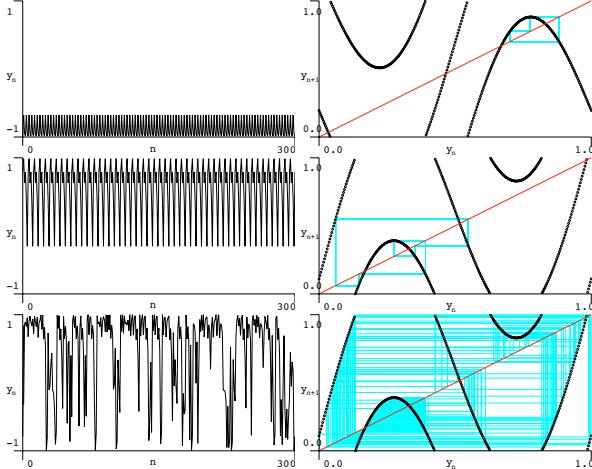


Figure 6: Simple orbits for large k . Top: A stable simple orbit at $\Omega = 0.195, k = 5.8$. Center: A singular orbit at $\Omega = 0.11, k = -6.4$. Bottom: Chaotic regime for all other initial values of y_n at $\Omega = 0.11, k = -6.4$.

- Mode locking: Mode locking is a feature for small k and becomes more prominent with increasing k [2, for some more details]. Mode locking means that around frequencies with rational frequencies, the oscillator tends to lock to those frequencies.
- Wave distortion: Wave fronts can be distorted from a sinusoidal form. Qualitatively this is similar to wave-shaping or phase distortion synthesis. This can be seen in the top example of Figure 2.
- Beating: A wave form is overlaid with a lower frequency envelope. The top part of Figure 2 shows a beating pattern for fairly mild coupling.
- Jitter: Wave fronts have disrupted phases or jittery phases. The bottom part of Figure 2 shows a jitter pattern.
- Pseudo-Noise: Wave fronts become strongly disrupted becoming perceptually noisy. All of Figure 4 shows this, as well as the bottom example of Figure 6.

Additionally, for large k the map becomes possibly but not necessarily sensitive to initial values of the iteration. Figure 3 shows such a case. A single point at $\Omega = 0.33$ and $k = 1.42$ in the parameter plane, which is bistable. This means that for different initial points of the iterations y_0 two different stable orbits are possible. Those are however stable. The first has a 3-orbit and the second is a 4-orbit. Such a bistable configuration is rare. A more typical example of the sensitivity to initial values of the map is shown in Figure 4. All three states in this case are for the same point in the parameter plane but with different starting positions y_n all three orbits are stable, hence can easily be achieved with a wide variety of starting points. But only two of the points are regular.

An important path to chaotic behavior is the successive bifurcation of resonant frequencies with increasing non-linearity k . The map goes through successive steps where the period doubles, which indicated by a doubling of the length of an orbit. An example of such a bifurcation is illustrated in Figure 5. We see that a single orbit separates into two similar but connected orbit, hence doubling the period.

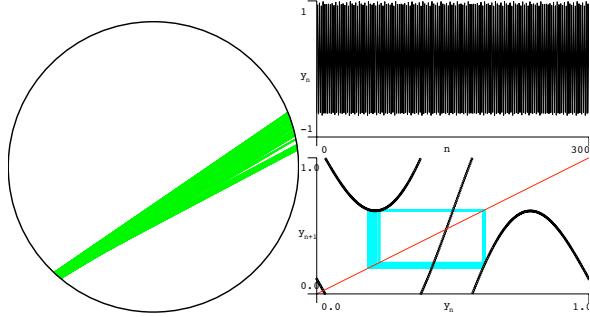


Figure 7: Refocusing: The map only occupies a restricted angular segment of the circle and repeatedly refocuses on this area for $\Omega = 0.11, k = 4.6$. The fine structure of the area is sensitive to initial conditions. Right: Values of the iteration y_n are mapped onto a circle to show the confinement of the map more clearly. Top left: Wave form of the response. Bottom left: The mapping function and the iteration trace.

Simple patterns can be observed even for large k . Figure 6 shows such simple orbits. One is stable for all initial values. The other is singular, embedded in an otherwise chaotic regime.

Figure 7 shows refocusing behavior. A beam of iterations only occupy a narrow area of two segments on the circle and repeatedly refocus on this area. Interestingly, the fine structure of this beam is not stable with initial values of the iteration.

5. VARIATION IN THE NON-LINEAR FUNCTION

The function $f(\cdot)$ of equation (2) significant influence on the actual wave form. This is often not emphasized in the mathematical literature, because one can show that despite variation of this transfer function only certain specific properties of the function have an influence how period-doubling and eventually chaos occurs. Specifically the degree of the turning points of the map is such a determining factor, but the shape of the function away from turning points is not [1].

However this change in function still has drastic influence to the sound and also change the qualitative behavior if turning points change. For this reason we describe a few experiments with variation of these transfer function.

We consider three functions in addition to the sine for $f(\cdot)$ in equation (2):

$$f(y_n) = \begin{cases} 4 \cdot y_n & \text{if } 0 \leq y_n < 1/4 \\ (1/4 - y_n) \cdot 4 + 1 & \text{if } 1/4 \leq y_n < 3/4 \\ (y_n - 3/4) \cdot 4 - 1 & \text{otherwise.} \end{cases} \quad (8)$$

$$f(y_n) = \begin{cases} \frac{y_n + T}{1 + 2 \cdot \epsilon \cdot T} & \text{if } 0 \leq y_n < B \\ \frac{y_n + (1 - 2 \cdot \epsilon) \cdot T}{1 - 2 \cdot \epsilon \cdot T} & \text{if } b \leq y_n < 1 - T \\ \frac{y_n + T - 1}{1 + 2 \cdot \epsilon \cdot T} & \text{if } 1 - T \leq y_n \leq B + 1 \\ \frac{y_n + (1 - 2 \cdot \epsilon) \cdot T - 1}{1 - 2 \cdot \epsilon \cdot T} & \text{otherwise.} \end{cases} \quad (9)$$

$$f(y_n) = \frac{1}{A} \sum_{m=1}^4 a_m \sin(2\pi m y_n) \quad (10)$$

With $T = 0.5$, $\epsilon = 0.25$ and $B = 0.5 + (\epsilon - 1) \cdot T$ in (9) and with $a_m = \{1, \frac{1}{2^2}, \frac{1}{3^2}, \frac{1}{4^2}\}$ and $A = a1 + a2 + a3 + a4$ in (10).

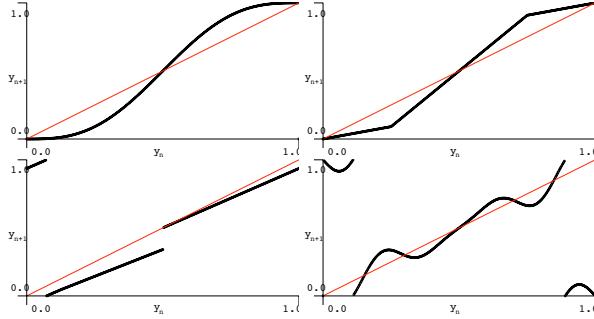


Figure 8: *Four non-linearity functions explored: Top, left: Sine, Top; right: Triangle; Bottom, left: Piecewise linear cardiorespiratory coupling model [19]; Bottom, right: truncated Fourier-series. All depicted at $k = 1, \Omega = 0$.*

Equation (8) is a triangle function. Equation (9) is a piecewise linear function from the biomedical literature [19] and equation (10) is a Fourier-series composition with four terms. The functions are shown in Figure 8.

For small k the main effect of variation in the two-fold. One is the precise occurrence of intersection with the fixed-point line. The second is the wave-shaping character of the function.

For large k all four example exhibit chaotic properties, though for the same parameter values, the behavior can naturally be very different. Figure 9 shows the same parameter point $\Omega = 0.2, k = 16$ for all four functions. Note that the standard circle map exhibits a stable periodic orbit, which has however a large orbit period. The triangle map shows a chaotic pattern. The cardiorespiratory model shows a stable orbit, including a dense attractor with some jitter, and the Fourier series also displays a chaotic regime.

6. SONIC RESPONSES

This section gives a subjective description of the sounds heard for most figures presented so far. To make the results accessible to the audible range, an iteration frequency of 22050 was used and the result of function (5) was downsampled by a factor of ten. The purpose of this section is to give a rough idea of the sonic character of the responses that certainly are not well captured in visual depiction. Some figures while looking very similar have rather different sounding responses.

- The example of bifurcation depicted in Figure 5 corresponds to a sine wave that successively gains new partials, eventually a high frequency noise develops, the noise becomes more broadband and eventually dominates the spectrum.
- The jitter pattern from Figure 2 is perceived as a sine wave with some underlaying noise.
- The bistable sounds of Figure 3 are two sine waves which are at a perfectly tuned fourth interval.
- The bottom two examples of the singular orbit within the noise regime sounds like a pure oscillator with some partials in the singular case, and sounds like noise with irregular rhythmic subpatterns in the chaotic case.
- The different initial conditions of the case depicted in Figure 4 is a sine wave with a low frequency pitch bending

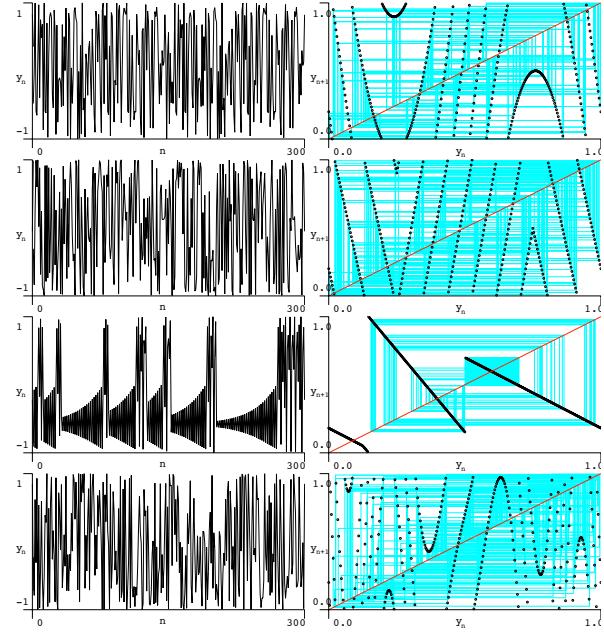


Figure 9: *The result of the general circle map for (top to bottom) sine, triangle, piecewise linear and Fourier series functions at $\Omega = 0.2, k = 16$.*

modulation for the first regular case, while the other non-chaotic orbit has a rich spectrum. The chaotic case sound like a low frequency band with a rhythmic subpattern.

- The example of variation of the nonlinear functions of Figure 9 sound like a simple sound with a light pitch bend for the sine function, rhythmic noise for the triangle function, a single sound with complex spectrum for the piecewise linear function and broadband noise with a mild periodic pitch bend for the Fourier series.
- Other interesting examples (without visual depiction) can be found at $\Omega = 0.4, k = 1.1$ for high pitched narrow-band noise. Narrowband low-frequency noise with and without rhythmic subpatterns can be found at $\Omega = 0.36, k = 16.0$ and $k = 15.8$.

Overall it is important to note, that pure noise responses are rather rare, even “noisy”-sounding responses often have additional features, for example underlaying rhythmic patterns, they may be band-limited and be subject to pitch-bending phenomena. The noisy aspect of the sound may be mixed with pitched or narrowband sounds. Hence chaotic patterns and perceptual noise are typically not the same.

7. CONCLUSIONS

We have demonstrated a number of properties of the circle map for sound synthesis by experimentation with both the parameters of the map and by varying the non-linearity. Some of the sonic results of non-linear iterative maps have already been discussed in the context of granular synthesis [9, 10]. The advantage of considering the circle map is the transition of linear and familiar behavior into rather complex regions which exhibits varied responses, like

chaotic, noisy-sounding responses, to pure and mixed sine wave, amplitude modulation, and pitch bending. One intriguing feature of these maps is their evident computational efficiency, requiring one function lookup, one multiplication, 2 additions and one additional memory lookup per time step. By the rich diversity of possible responses, and guaranteed stability this makes iterated maps attractive. The main disadvantage of the method is the difficulty of control. The purpose of this work is to prepare for eventual recommendations control of perceptually relevant responses through these parameters, which is planned.

The circle map is also attractive because by its one parameter perturbation between linear and non-linear behavior allows for classification of the method with respect to the two end of this behavioral spectrum.

8. ACKNOWLEDGMENTS

Thanks to Martin Roth and Matthias Rath for discussions and pointers to related literature. All figures were rendered using Processing by Ben Fry and Casey Raes and SimplePostScript by Marius Watz.

9. REFERENCES

- [1] J. A. Glazier and A. Libchaber, "Quasi-periodicity and dynamical systems: An experimentalist's view," *IEEE Trans. Circuits and Systems*, vol. 35, no. 7, pp. 790–809, 1988.
- [2] G. Essl, "Circle maps as a simple oscillators for complex behavior: I. Basics," submitted to *Int. Comp. Music Conf. (ICMC'06)*, New Orleans, USA, 2006.
- [3] J.-C. Risset, "Catalog of computer synthesized sound," Murray Hill, Bell Telephone Laboratories, 1969.
- [4] D. Arfib, "Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves," *J. Audio Eng. Soc.*, vol. 27, no. 10, pp. 757–779, 1979.
- [5] M. Le Brun, "Digital waveshaping synthesis," *J. Audio Eng. Soc.*, vol. 27, no. 4, pp. 250–266, 1979.
- [6] J. Pressing, "Nonlinear maps as generators of musical design," *Computer Music J.*, vol. 12, no. 2, pp. 35–46, 1988.
- [7] M. Gogins, "Iterated functions systems music," *Computer Music J.*, vol. 15, no. 1, pp. 40–48, 1991.
- [8] R. Bidlack, "Chaotic systems as simple (but complex) compositional algorithms," *Computer Music J.*, vol. 16, no. 3, pp. 33–47, 1992.
- [9] B. Truax, "Chaotic non-linear systems and digital synthesis: an exploratory study," in *Proc. Int. Comp. Music Conf. (ICMC'90)*, Glasgow, Scotland, 1990, pp. 100–103.
- [10] A. Di Scipio, "Composition by exploration of non-linear dynamic systems," in *Proc. Int. Comp. Music Conf. (ICMC'90)*, Glasgow, Scotland, 1990, pp. 324–327.
- [11] ———, "Synthesis of environmental sound textures by iterated nonlinear functions," in *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-99)*, Trondheim, Norway, 1999.
- [12] X. Rodet, "Nonlinear oscillations in sustained musical instruments: Models and control," in *Proc. Euromech*, Hamburg, Germany, 1993.
- [13] R. Dobson and J. Fitch, "Experiments with chaotic oscillators," in *Proc. Int. Comp. Music Conf. (ICMC'95)*, Banff, Canada, 1995, pp. 45–48.
- [14] J. Manzolli, F. Damiani, P. J. Tatsch, and A. Maia, "A non-linear sound synthesis method," in *Proc. 7th Brazilian Symp. on Computer Music*, Curitiba, 2000.
- [15] N. Valsamakis and E. R. Miranda, "Iterative sound synthesis using cross-coupled digital oscillators," *Digital Creativity*, vol. 38, no. 4, pp. 331–336, 2005.
- [16] W. Lauterborn and U. Parlitz, "Methods of chaos physics and their application to acoustics," *J. Acoust. Soc. Am.*, vol. 84, no. 6, pp. 1975–1993, 1988.
- [17] C. Roads, *The Computer Music Tutorial*. Cambridge, MA: MIT Press, 1996.
- [18] L. N. Hand and J. D. Finch, *Analytical Mechanics*. Cambridge, UK: Cambridge University Press, 1998.
- [19] M. McGuinness and Y. Hong, "Arnold tongues in human cardiorespiratory systems," *Chaos*, vol. 14, no. 1, pp. 1–6, 2004.

FEEDBACK IMPLEMENTATION WITHIN A COMPLEX EVENT GENERATION SYSTEM FOR EMERGENT SONIC STRUCTURES

Sinan Bökesoy

CICM, University of Paris VIII

Paris, France

sinan@sonic-disorder.com

ABSTRACT

This paper discusses the implementation of a complex event generation model with a simple feedback loop and its sound synthesis results while investigating the overall system behaviour. The system is based on the *Cosmos* model, which is a self similar structure, and distributes events on different time-scales with certain interdependency. The user intervenes with the system in real-time by inputting a live sound source and interacting with the user interface by controlling the parameters for the time scales *macro*, *meso* and *micro*. Because of the complex dynamic behaviour and modulation scheme, it is possible to create a timbre space of unique textures.

1. INTRODUCTION

Complex systems become increasingly important in synthesizing sonic structures. The understanding and controlling the phase space of such systems is a challenge. We have explored before the use of stochastic formulae inspired by Xenakis's models¹ for waveform synthesis, granular synthesis [1], other linear mapping models for organizing sonic atoms [2], and complex mapping models [3]. Applications using non-standard synthesis models with iterative functions [4] and interdependency in their structure [5] introduce non-linear mapping systems with dynamic behavior resulting to emergent phenomena. The interaction design is based on controlling their parameter space which controls the inner processes. By constraining the synthesis parameter space of dynamic systems for obtaining perceptually meaningful results, it becomes possible to produce a sound palette with desired specifications for compositional purposes. My perspective is gained by considering a macro-sound object as an evolving system of micro structures. The granular representations are especially useful in the analysis and perceptual modeling of dynamical events of peculiar complexity. [6] By introducing a hierarchy of multiple time scales for the event distribution process and modulation sources constructed with stochastic/deterministic functions, it is possible to form sonic creations that cannot be handled by traditional synthesis methods. The integration of gained experiences through different approaches into a coherent structure is my basic motivation.

First I will introduce the *Cosmos* model [7] shortly and follow with the structural facts such as how it gains the property of an iterative function by adding the recursive element becoming an interactive signal processing tool [8].

¹Xenakis' works such as Achorripsis, AnalogiqueA/B, Gendy3.

1.1. About the *Cosmos* model

The *Cosmos* model is a real-time dynamic event distribution system, which generates sonic textures with an the event distribution process on multiple time scales. The discrete events of certain density are distributed in a time space with their onset time and duration parameter, which are calculated with stochastic/deterministic functions (Figure 1).

Each event in the macro space defines the duration of a meso space, and the sub events are distributed inside this space length. The same organization is also true for the events of meso space and micro space. As new spaces are created, new events and therefore new sub spaces are distributed while constructing the interdependency between layers.

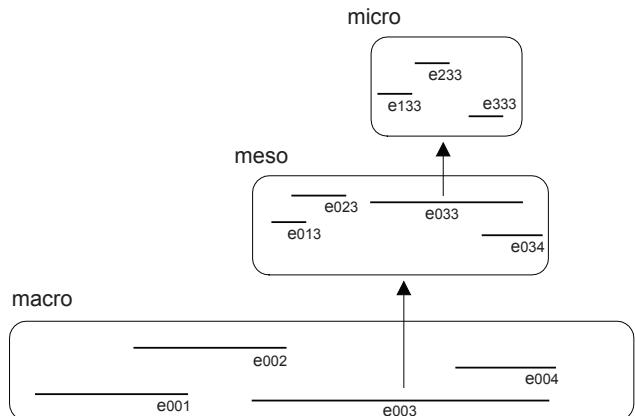


Figure 1: *Self similarity in the structure of the event generation mechanism of Cosmos.*

The modulation generators in 'Cosmos' are the stochastic modulation sources addressing the synthesis parameters, which are assigned to sonic entities defined within the micro event space. There are four different curve generators for each of the macro, meso and micro space events. Finally, we can combine together different modulation sources belonging to three different time scales, from macroscopic to microscopic, to obtain higher complexity. Each curve generator can be assigned to a unique destination, and if there are more than one modulation source for one destination, they will be superimposed as one modulation source with the complexity of representing the evolution of multiple time scales. The original version of *Cosmos* is based on the idea of assigning a waveform, sonic grain, to the micro events, where the meso and macro level distribution forms the sonic texture on multiple layers

with various modulation schemes. The version presented in this paper will be explained in the next step.

2. ADDING THE RECURSIVE ELEMENT ON THE COSMOS MODEL

My purpose here is to convert the *Cosmos* model into a system, which transforms the waveform assigned for the sonification of micro-events with its inner dynamics, where it is able to generate a chain of actions without the intervening of human action. The only necessary human action is setting the initial conditions of the system and injecting a starting waveform (also it can be a continuous flow of audio). The audio input will be delivered on a delay line into the system (see Figure 3), which will be tapped with different delay times according to the equation (1).

$$X(n) = \sum_{l=1}^{macro_d} w_m(n) \cdot \sum_{m=1}^{meso_d} w_{mm}(n) \cdot \sum_{l=1}^{micro_d} f(n - D_{l,m,n}) \cdot w_{mmm}(n) \quad (1)$$

$f(n)$ represents the input waveform as a series of sampled values. $macro_d$, $meso_d$ and $micro_d$ are event space densities defined by the user or by stochastic functions. The scalar $D_{l,m,n}$ representing the delay time in samples in equation (1) is calculated at the point of each macro-space initialization. This occurs with a mapping of the onset time values calculated for the micro events as in equation (2).

$$T_{on} e_{x,y,z} = \sum_{m=0}^z \sum_{l=0}^y \sum_{k=0}^x t_{on} e_{k,l,m} \quad (2)$$

$t_{on} e_{x,y,z}$ in equation (2) is the onset time interval between $e_{k,l,m}$ and $e_{k,l,m-1}$. It can be calculated with statistical functions like gaussian, exponential, uniform, triangular, chauchy distributions or deterministic functions like arithmetic, geometric series or constant values. The functions use the space length and density as input parameters and distribute the onset time values and durations for the events inside the particular space. $T_{on} e_{x,y,z}$ represents the absolute time between the initialization of the macro space and the onset time of x th micro event in y th micro-space of z th meso-space.

This value will be mapped as the delay time in equation (1) by multiplying it with the sampling frequency which is 32 kHz in this case.

The delayed input waveform is multiplied with a window function $w_{mmm}(n)$ which is a trapezoid function with the duration equal to the micro-event length having the attack and decay times of 1 ms. The meso-events will be multiplied with $w_{mm}(n)$ and the macro-events with $w_m(n)$ as in equation (1). This procedure implements a complex delay network created by the event data which is generated following a top-down approach from macro to micro structure. The bottom up organization of the delay network will be realized by sampling this delay line with the duration defined by the length of micro and meso spaces of *Cosmos* and passing the audio to the delay lines on higher levels (Figure 3). Thus, each organization of micro events produces the smallest delay network and will be passed as micro-space waveform data to the meso-space.

In Figure 2 one can notice an input waveform belonging to a xylophone sample as organized by the micro-space distribution

with a space length of 100 ms. We remind here that each micro-space corresponds to a meso-event. Then the meso-events organize the meso-spaces corresponding to macro-events. Finally the macro-events (meso-spaces) together realize a macro-space. The audio output is delivered back to the micro-event with a connection of the macro-space output to the input delay line which creates a feedback loop. In Figure 2, we see a direct mapping of the input material to the micro-events forming up the meso-event.

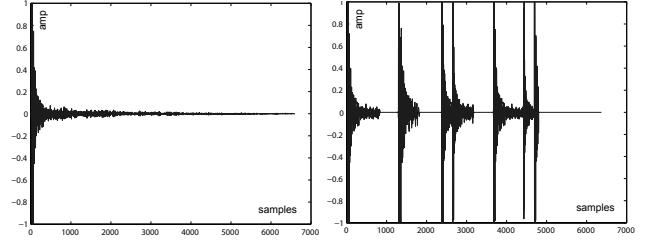


Figure 2: The input waveform above has been mapped by the micro-space distribution onto the micro-events. The window function applies a 1 ms fade out on each event.

This introduces local densities contributing rhythm to pitch phenomena depending on the distribution functions in the event spaces, but also the possibilities of *Cosmos* present certain modulation source generators which are capable of introducing morphological change on the waveform data.

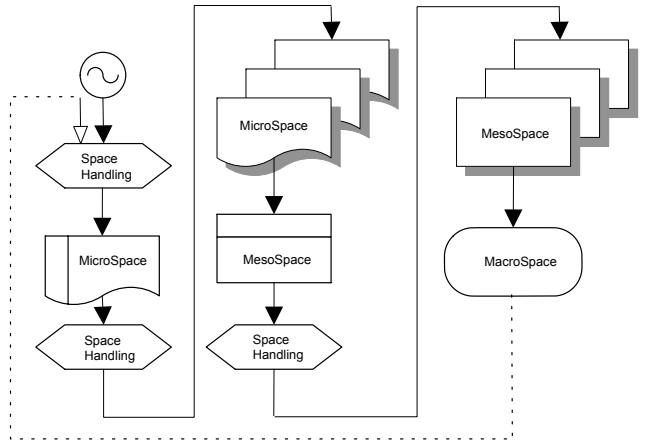


Figure 3: This is the schematic overview of the audio routing and the event space organization inside the application. The audio enters top-left. The dotted line is the feedback line.

The applied modulation sources on micro, meso and macro events individually are as following:

For macro-events: delay tap mod., intensity mod.,

spectral filter spatialization

For micro-events: delay tap mod., intensity mod.,

For meso-events: delay tap mod., intensity mod., spatialization, spectral filter mod.

These are continuous modulation sources applied to each event along its duration. They are constructed with mathematical functions inspired by the *Achorripsis* model of I. Xenakis [1] (see Figure 4). Depending on the speed of the modulations applied on the

audio data, the effect can be non-linear such as the in the delay tap modulation, which can produce complex pitch modulation in this case. In this version, I had to consider the CPU usage in order not to compromise the real-time aspect of interaction.

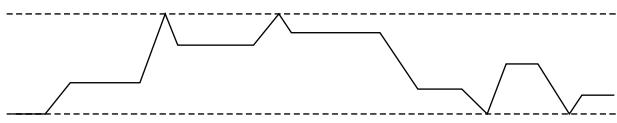


Figure 4: A continuous modulation constructed with a stochastic function and lo-hi barriers which fold the curve back from the limits as in the Xenakis models.

The application of the model is implemented on Max/MSP². The delay network is constructed with *tap.in/out* structure. All the duration and onset time information is calculated on Java-script. The parameters from the user interface are updating the variables in the script during the real-time operation, and the calculated event parameters are written to the memory locations, which are accessed by the MSP patch accompanying for DSP routines. The information necessary to render the audio in these functions are delivered in audio-rate to the relevant MSP objects for precise handling of the data. Each event is synchronized with at audio-rate *adsr* (AttackDecaySustainRelease) function serving as a window function with the duration equal to event duration. Those frames are captured and organized as in Figure 3.

I have limited the macro space density with 2 macro-events, the meso-space density with 4 meso-events and the micro-space density with 5 micro-events. Hence, inside one macro-space there can be $2 \times 4 \times 5 = 40$ micro events at once and the accompanying modulation sources would have the quantity equal to $2 \times 2 + 8 \times 4 + 40 \times 2 = 116$. This gives an idea about the complexity the system is offering between one waveform input and the macro sound output. We could substitute the windowing functions in the equation (1) to include the modulation functions in macro, meso and micro-space domains.

The system in Figure 3 proposes a single audio input/output structure with controllable parameters. A complex system is not *linear time-invariant*. Time invariance here means that whether we apply an input to the system now or t seconds from now, the output will be identical, except a time delay of the t seconds. We could force the model to periodic behavior. We can describe the next macro-space according to the input parameters explicitly when using only deterministic functions like geometric series or fixed values for the decision of the onset time and duration distributions which effect directly the delay time distribution and deterministic modulation sources. We would know then how the events are distributed and modulated inside one macro-space, which can be also represented as the phase space of the macro-space with regarding its input parameters and state controls.

2.1. A case study using the recursive *Cosmos* model

Now I will explicitly formulate a special deterministic case as an example and explain how the system becomes an iterative function by applying the feedback loop. The onset time decisions in macro, meso and micro events are made with arithmetic series function and the event duration is defined with the distance coefficient a

²a programming environment distributed by *Cycling74*.

of the arithmetic series of Equation (3). With the use of the discrete time domain equation, it becomes possible to watch the phase space with continuing the iteration on the equation. Consider the arithmetic series with:

$$y_n = 0 + a + 2a + 3a + \cdots + na \quad (3)$$

We define an event space with density n , where y_n equals to the event space length. The macro space length M_{Cl} and the densities macro, meso and micro spaces (d_m, d_{mm}, d_{mmm}) are deterministic input parameters for the system.

$$M_{Cl} = \frac{d_m(d_m + 1)}{2}a \implies y_{M_n} = \frac{n(n + 1)M_{Cl}}{d_m(d_m + 1)} \quad (4)$$

y_{M_n} in equation (4) will provide us the macro space event onset times, such as for $1 \leq n \leq d_m$:

$$e_{Mon_n} = y_{M_{n-1}} \text{ and } e_{Mdur_n} = a = \frac{2M_{Cl}}{d_m(d_m + 1)} \quad (5)$$

For example a 1 sec. long macro space with 3 events would have the event onsets at 0 ms, 166 ms and 500 ms with the event durations of 166 ms, according the arithmetic series distribution function used here. The event duration of the macro events define the meso space length and the meso space event durations define the micro space length in the self similar structure of *Cosmos*. Therefore it is easy to substitute the equations to find the meso and micro level distributions. For the meso space, the onset and duration equations are:

$$\begin{aligned} e_{MMdur_i} &= \frac{4M_{Cl}}{d_m(d_m + 1)d_{mm}(d_{mm} + 1)}; \quad 1 \leq i \leq d_{mm} \\ y_{mm_l} &= \frac{2l(l + 1)M_{Cl}}{d_m(d_m + 1)d_{mm}(d_{mm} + 1)} \quad e_{MMon_l} = y_{mm_{l-1}} \end{aligned} \quad (6)$$

The micro, meso and macro spaces are modulated by their individual modulation sources, however in this example we discard the modulation sources and perform the iteration for an impulse input. In order to define the delay times for the micro events, we need to formulate the absolute onset time according to Eq. (2):

$$\begin{aligned} e_{Mon_n} + e_{MMon_l} + e_{MMMon_k} &= e_{MMMon_{n,l,k}}(abs) \\ 1 \leq n \leq d_m \quad 1 \leq l \leq d_{mm} \quad 1 \leq k \leq d_{mmm} \end{aligned} \quad (7)$$

Equation (8) represents the recursive delay network with the feedback gain G . A unity gain represents the direct connection of *Cosmos* output with its input. The experiment can follow by inserting an impulse function as an input and listen to the iterated y values.

$$f \left(x - \sum_{n=1}^{d_m} \sum_{l=1}^{d_{mm}} \sum_{k=1}^{d_{mmm}} e_{MMMon_{n,l,k}}(abs) \right) + G y(x-1) = y(x) \quad (8)$$

2.2. Sound examples and some interpretations

The first result is a mapping of the arithmetic series in the self-similar structure of *Cosmos* as a rhythmic pattern while $G = 0$ (*exampleA*³). The change in G won't destruct this pattern as new

³The audio examples can be downloaded at URL:
<http://www.sonic-disorder.com/cosmosfdb-examples.zip>

macro-spaces follow each other using the same mapping (*exampleB*). Hence, regarding the impulse response of this system, we observe the simplest case of sound morphology carried by the feedback loop to the input delay line. It is helpful to follow the transformation by ear, and do some interpretations according the hearing experience.

The audio signal is not being analyzed in order to create control signals and there are no functions which manipulate the system using this kind of analysis [7]. Any change in micro or meso-level will be reflected to the higher level which becomes the interdependency inherent in the system design. In *exampleC*, a uniformly distributed tap delay modulation has been applied to the meso events by keeping the same onset time and duration distribution and $G = 1$. The skeleton remains the same but the skin does change.

In *exampleD*, a stochastic delay time modulation is applied on the micro event level while keeping the settings in *exampleC*. Then in *exampleE*, more disorder has been created in the system by changing the macro and meso event durations with the exponential distribution. In *exampleF* a very low frequency sawtooth waveform with a single cycle has been delivered as the initial signal.

These examples show the response of the application to a short waveform input (a 100 ms xylophone sample) according to the change applied to its parameter space. After certain amount of iterations, we can hardly recognize anymore the departure sound having been injected to the system. Due to the system dynamics, emergent properties result textural sound phenomena which is evident in the *exampleC*, *D*, *E* and *F*. Such results are extremely difficult to obtain with *montage* or any other hand driven methods since the delivered complexity and the computation necessary is high.

I stressed myself to use short sounds as the input material and the model organizes a macro sound structure from its own sound output depending on its internal conditions at the moment it captures the output signal on the micro-level. It creates a skeleton of events organized on different timescales simultaneously. The overall texture becomes a complex one exhibiting various modulation sources transforming the audio material from micro to macro levels.

I heard many comments from my colleagues about how this system could successfully deliver an organic character like the sounds of unknown creatures especially even when the input material is very simple and the model transforms it quickly to various emergent forms within its dynamic structure.

3. PRELIMINARY CONCLUSIONS

With the combination of stochastic distribution mechanisms and the self similar structure of the *Cosmos* model, it is impossible to predict the precise output of any configuration of values. A tool has been developed in order to bring a periodic behaviour by freezing the macro level mechanisms but letting the system operate further on meso and micro levels.

In this case, the ear can easily follow and memorize the macro level outline of the sonic structure. It is also a quick process to add new dynamic functions or user defined rhythmic patterns to the Javascript code in order to control the onset time and event duration distribution. A new tool is also being developed, where one could morph between two distribution functions on the fly such as a mapping between determinism and indeterminism occurs. Many

experiments should be tried to gain perceptual experience with such systems where the user interacts in order to explore the sonic output by controlling the parameter space externally.

I have tried to explain the implementation and the design aspects of a recursive structure based on the *Cosmos* model. The feedback of the sound output (macro level) into the system on the micro level adds a considerable gain to the emergent behaviour. In musical contexts, there are important aspects which cannot be left aside such as the question of interacting with a complex synthesis application and a powerful user interface for efficient mapping of the parameter space based on external manipulations [9]. One could transform this model into an instrument within real time implementation.

A continuous MIDI controller is very useful to assign its pots to various parameters in the application interface, as the operation with mouse is not very intuitive. As a composer, my intention is continuing this research by trying to realize compositional ideas by developing a mapping interface which translates the dynamics of acoustic sound sources/instruments to the parameter space of the model.

(The author would like to realize an augmented percussion instrument coupled with this model to process the audio/control data captured from the percussion by sensors and microphone.)

4. REFERENCES

- [1] I. Xenakis, *Formulized Music*, revised ed. New York: Pendragon Press, 1992.
- [2] C. Roads, "Sound composition with pulsars," *J. Audio Eng. Soc.*, vol. 49, no. 3, pp. 134–147, Mar. 2001.
- [3] M. Hamman, "Mapping complex systems using granular synthesis," in *Proc. Int. Comp. Music Conf. (ICMC'91)*, Montréal, Canada, 1991, pp. 475–478.
- [4] A. Di Scipio, "Iterated nonlinear engines as a sound-generating engine," *Leonardo Journal*, vol. 34, no. 3, pp. 249–254, May 2001.
- [5] P. Bowcott, "High level control of granular synthesis using concepts of inheritance and social interaction," in *Proc. Int. Comp. Music Conf. (ICMC'90)*, Glasgow, Scotland, 1990, pp. 50–52.
- [6] A. Bregman, *Auditory Scene Analysis*, 2nd ed. Cambridge: MIT Press, 1999.
- [7] S. Bökesoy, "The *Cosmos* model, an event generation system for synthesizing emergent sonic structures," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 259–262.
- [8] A. D. Scipio, "'Sound is the interface': Sketches of a constructionistic ecosystemic view of interactive signal processing," in *Proc. XIV Colloq. Musical Informatics (CIM-03)*, Firenze, Italy, 2003, pp. 259–262.
- [9] I. Choi, "A chaotic oscillator as a musical signal generator in an interactive performance system," *J. New Music Research*, vol. 26, no. 1, pp. 17–47, Sept. 1997.

JOINT ACOUSTIC SOURCE LOCATION AND ORIENTATION ESTIMATION USING SEQUENTIAL MONTE CARLO

Maurice Fallon*, Simon Godsill

Signal Processing Group
University of Cambridge, UK
{mff25|sjg30}@cam.ac.uk

Andrew Blake

Microsoft Research
Cambridge, UK
ablake@microsoft.com

ABSTRACT

Standard acoustic source localization algorithms attempt to estimate the instantaneous location of a source based only on current data from a microphone sensor array. This is done regardless of previous location estimates. However more recent Sequential Monte Carlo based approaches have instead posed the problem using an evolving state-space framework. In this paper we take this approach further by exploiting the directionality of human speech sources. This allows us to estimate the orientation of the source within the room. Finally combining previous source localization methods with this work we outline how both parameters - location and orientation - may be estimated jointly. Examples are given of performance in a typically reverberant real office environment for both a stationary and a moving source.

1. INTRODUCTION

Localization and tracking of sources is a topic of great importance in many fields from seismology to sonar. Location and orientation estimation and tracking in an audio environment, and more specifically speech, will be the focus of this paper. The use of these estimates has become important in several applications including automatic camera or microphone steering. The setup envisaged in this work is of several microphones spatially distributed around a typically reverberant room such as an office or living room. Traditionally localization algorithms have been based upon time delay estimation (TDE) on signals from multiple microphone pairs followed by often complex triangulation of the resultant estimates [1].

While sufficient in low noise and mildly reverberant conditions these methods break down in even moderately reverberation conditions - giving false peaks which lead to incorrect location estimates. As a result recent attention has turned to Sequential Monte Carlo (SMC) methods [2], commonly known as particle filters, which provide a simple yet adaptable framework for parameter estimation and tracking. Advantages of these methods include robust operation in adverse noise conditions and it allows for an elegant integration of the non-linear relationship between the TDE values and source position. Also SMC forms an easily extendable framework - which is illustrated by the orientation extension introduced in this paper.

The remainder of this paper details this extension as follows: Section 2 illustrates the effect that speech directivity has on signal correlation and how this information can be used to estimate orientation. Section 3 details the generic time delay framework as used

in previous literature. Section 4 incorporates these estimates into a particle filtering framework from which a novel likelihood function for orientation is then formed in Section 5. Finally in Section 6 experimental results are detailed. Conclusions and discussion follow in Section 7.

2. SPEECH DIRECTIONALITY AND THE GCC

Speech Directionality, as experimentally explored by [3], is the effect of non-uniform radiation of sound from the mouth and its absorption by the head and torso. For human speakers this effect causes as much as 15dB front/back attenuation differential at high frequencies, which is illustrated in Figure 1.

For clustered sensor arrays using the far-field assumption, such as those used in coherent signal processing, directionality is negligible. However when one chooses to instead use a sparse distributed sensor network the effect of directionality becomes very relevant. Indeed the setups envisaged in [4, 5] have the source surrounded by the sensors.

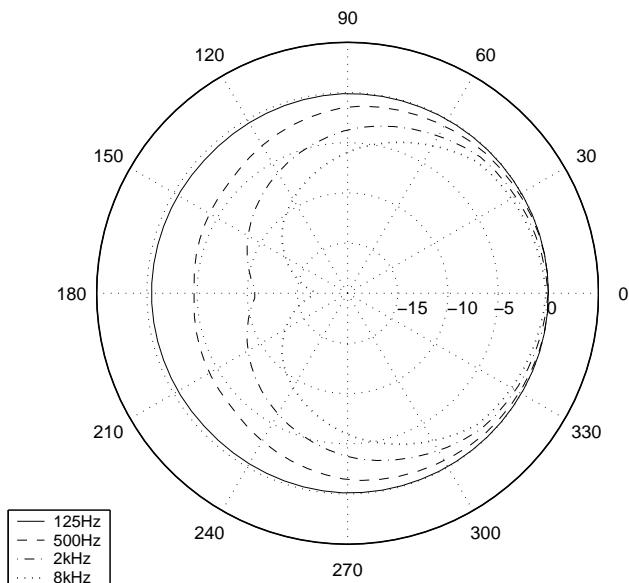


Figure 1: Human speaker directional response (dB) obtained by least squares fitting of data from [3]. Graph taken from [6].

This effect causes reduced correlation between signals recorded at sensors positioned at different angles relative to the mouth. There

* This work was supported by Microsoft Research. Preliminary results available at <http://www.sigproc.eng.cam.ac.uk/~mff25/research.html>

are two different reasons for this. Firstly the attenuation effect varies greatly with frequency which leads to non-uniform absorption of the sound by the source's head. Secondly as the relative angle increases the energy in the direct path portion of the signal falls in comparison to that of the multi-path portion caused by reflections - in effect increasing the reverberation portion of the recorded signal. While this effect is very difficult to quantify because of the numerous factors, it is illustrated in Figure 2. In Figure 3 we go on to show that with an increasing relative angle between the source's orientation and the microphone pair there is a corresponding fall in signal cross-correlation. In other words microphones near to front of a speaker's head record the source signal with greater correlation. Using this effect we will now introduce a model which utilizes this effect to estimate speaker orientation.

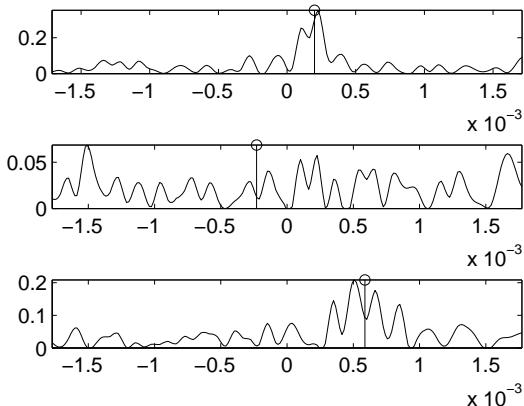


Figure 2: Effect of orientation on signal correlation: GCC functions formed from the same source signal recorded at microphone pairs situated at 0° (top), 180° (middle) and 90° (bottom) relative to source orientation. The true source delay is indicated with a vertical line, the x-axis is relative delay between the recorded signals (in seconds) while the y-axis is the cross-correlation magnitude.

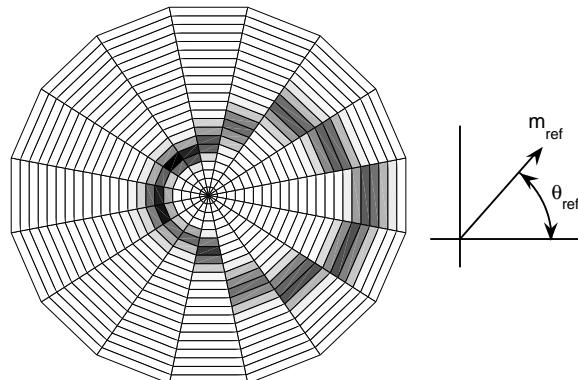


Figure 3: Plot of the distribution of GCC peak magnitude, m_{rel} , versus relative angle between source heading and microphone pair, θ_{rel} , illustrating that there is greater signal correlation when the relative angle is small. The source signal was white noise and the environment was a typical office room.

3. MEASUREMENT MODEL

We will concern ourselves with the problem of tracking the location and orientation of a single speaker in the XY-plane using TDOA measurements taken from a set of N_m spatially distributed microphone pairs. While the paper concentrates on the 2-D case it should be stated that any solution may be quite easily extended to full 3-D tracking. The assumed environment is a typically noisy and reverberant room and the microphones are assumed to be omnidirectional.

The core mathematical framework used in this paper follows that introduced by [7]. The received signal at sensor m is related to the transmitted source as follows

$$x_m(t) = h_m(t) * s(t) + e_m(t) \quad (1)$$

where $s(t)$ is the source signal, $h_m(t)$ is the impulse response and $e_m(t)$ is additive noise. In theory having estimated the impulse response it would then be possible to evaluate the source position. Unfortunately in challenging reverberant conditions estimating a rapidly changing time varying impulse response is a very difficult task. While there do exist some methods which attempt to do this, such as adaptive eigenvalue decomposition (AED, [8]), it has yet to be demonstrated in challenging multi-path environments. As a simplification [7, 9] and others ignore the multi-path portion of the signal and instead assume a simple direct path model only. The signal model then becomes

$$x_m(t) = \alpha_m s(t - \tau_m) + e_m(t) \quad (2)$$

where signal received at a particular microphone is simply a delayed and attenuated version of the source signal. The amplitude parameter, α_m , and the delay parameter τ_m , are then a simple function of the distance between source and sensor, d_m , as follows

$$\alpha_m \propto \frac{1}{d_m} \quad \tau_m = \frac{d_m}{c} \quad (3)$$

where c is the speed of sound. Because of speech's highly variable nature the amplitude parameter, α_m , will not be considered for estimation. Meanwhile various techniques to estimate time delay parameter, τ_m , will be discussed in the following section.

3.1. Time Delay Estimation Model

A vast body of literature on the subject of time delay estimation exists. For multichannel delay estimation, a number of techniques attempt to estimate the set of delays over all microphones which best fit the recorded data. This approach is used for the SRP-PHAT beamforming localizer, which is essentially a multichannel cross-correlation [1]. This is expanded upon and used as the measurement model for the importance sampling particle filter in [4]. This method assumes that by correlating the data from all microphones we can make our estimate in some way more robust than a set of pairwise delay estimates alone. However since the previously mentioned directivity effect causes non-adjacent recordings to be relatively uncorrelated, it may not be particularly useful in practice for distributed microphones.

As a result we will concentrate on the most basic of the time delay estimation methods - pairwise cross-correlation using the phase transformed generalized cross-correlation (PHAT-GCC) as introduced by [10]. Taking synchronized frames of data with length L samples at time frame k from sensor m , that is

$$\mathbf{x}_m(k) = [x_m(kL), x_m(kL+1), \dots, x_m(kL+L-1)], \quad (4)$$

we form a N_m by L matrix as follows

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{x}_1(k) \\ \vdots \\ \mathbf{x}_{N_m}(k) \end{bmatrix}. \quad (5)$$

A generalized TDE function makes a transformation between this raw data vector and a set of time delays, represented as $\mathbf{T}_k = \mathbf{f}(\mathbf{X}_k)$. For the GCC this transformation may be made as follows: First pairing up the sensors into N_p pairs, we evaluate the signal Fourier transforms, where for example $X_{p,2}(\omega)$ represents the Fourier transform of the audio recorded by second sensor in pair p . Next we form the GCC for pair p as follows

$$R_p(\tau) = \frac{1}{2\pi} \int \Psi(\omega) S_{x_{p,1}x_{p,2}}(\omega) e^{j\omega\tau} d\omega \quad (6)$$

where $S_{x_{p,1}x_{p,2}}(\omega) = E[X_{p,1}(\omega)X_{p,2}^*(\omega)]$ is the signal cross power spectrum. $E[\cdot]$ and $(\cdot)^*$ stand for the expectation and complex conjugate operators respectively. The weighting function will be the commonly used phase transform

$$\Psi_p(\omega) = \frac{1}{\|X_{p,1}(\omega)X_{p,2}^*(\omega)\|}. \quad (7)$$

This time delay estimator is used because of its robust behavior in the face of changing speech source signal characteristics although it is noted that performance is much reduced in strong reverberation [11]. Having formed the GCC function the corresponding set of promising candidate time delay estimates, $T_k^{(p)} = (\hat{\tau}_1^{(p)}, \dots, \hat{\tau}_{N_c}^{(p)})$, are then isolated. These are simply the delays corresponding to the largest N_c peaks in the GCC function.

The entire set of delay measurements for a particular time frame k is then $\mathbf{T}_k = (T_k^{(1)}, \dots, T_k^{(N_p)})$. The expected delay, at microphone pair p , for a given source position, $l = (x, y)$, is then given by

$$\tau^{(p)} = c^{-1} (\|l - l_{p,1}\| - \|l - l_{p,2}\|). \quad (8)$$

Note that the TDE measurements are limited by the separation of the intra-pair microphones; where the positions of the microphones in pair p are

$$\begin{aligned} l_{p,1} &= (x_{p,1}, y_{p,1}) \\ l_{p,2} &= (x_{p,2}, y_{p,2}), \end{aligned} \quad (9)$$

The maximum possible time delay is $\tau_{\max}^{(p)} = c^{-1} \|l_{p,1} - l_{p,2}\|$.

Meanwhile as indicated by Figure 3 the **magnitude** of the GCC peaks are indicative of the relative angle between the source heading and the microphone pair direction. This information will be used to estimate orientation. For the set of delay measurements identified above the matching GCC function magnitudes are identified as follows

$$\hat{m}_c = R_p(\hat{\tau}_c^{(p)}) - m_{\text{th}} \quad (10)$$

where m_{th} is an experimentally determined constant to correct for the GCC function noise floor and obviously candidate peaks are chosen to have magnitudes greater than the noise floor.

In the same way that the delay measurement vector was constructed above the magnitude measurement vector for microphone pair p is $M_k^{(p)} = (\hat{m}_1^{(p)}, \dots, \hat{m}_{N_c}^{(p)})$. The set of magnitude measurements for a particular time frame k will then become $\mathbf{M}_k =$

$(M_k^{(1)}, \dots, M_k^{(N_p)})$. Finally the overall combined measurement vector is then a combination of the above, $\mathbf{D}_k = (\mathbf{T}_k, \mathbf{M}_k)$.

How we decide upon which observations to consider is open to some discussion - for example thresholding of the GCC function is the simplest method. It is also useful to limit the number of candidate peaks to a reasonable maximum using a constant $N_{c,\max}$. Either way the set of candidate peaks should contain all of those likely to be due to a speech source.

4. PARTICLE FILTER FRAMEWORK

In this section a general particle filter framework for localization is introduced. We will use a framework adapted from that initially proposed by [7] and also used by [5]. The source state vector at time k will be defined as

$$\alpha_k \triangleq (\alpha_{l,k}, \alpha_{o,k})$$

where the source state vector is made up of two different sections: one each for location and orientation respectively. These are in turn defined as

$$\begin{aligned} \alpha_{l,k} &\triangleq (x_k, \dot{x}_k, y_k, \dot{y}_k) \\ \alpha_{o,k} &\triangleq (\theta_k, \dot{\theta}_k) \end{aligned} \quad (11)$$

where, for example, (x_k, \dot{x}_k) are source position and velocity, in the \mathcal{X} -dimension. Estimation of this state vector will jointly provide us with source position (x_k, y_k) and orientation (θ_k) . The time delay, $\tau_k^{(p)}$, that one would expect for this source state vector, for microphone pair p , is found by substituting the location parameter pair, (x_k, y_k) , into Equation 8. The set of resultant source time delays is then $\mathbf{T}_{\alpha,k} = (\tau_{\alpha,k}^{(1)}, \dots, \tau_{\alpha,k}^{(N_p)})$.

Source movement in the \mathcal{X} , \mathcal{Y} and θ dimensions is assumed to be independent and can be decoupled as a result. The state dynamics will be modeled by a first-order Markov process specified by its initial state and state transition distributions which are of the form $p(\alpha_0)$ and $p(\alpha_k | \alpha_{k-1})$ respectively, whose specifics can be found in [7].

The tracking problem itself involves recursive estimation of the posterior filtering distribution, $p(\alpha_k | \mathbf{D}_{1:k})$, using Bayes' Theorem as follows

$$\begin{aligned} p(\alpha_k | \mathbf{D}_{1:k-1}) &= \int p(\alpha_k | \alpha_{k-1}) p(\alpha_{k-1} | \mathbf{D}_{1:k-1}) d\alpha_{k-1} \\ p(\alpha_k | \mathbf{D}_{1:k}) &\propto p(\mathbf{D}_k | \alpha_k) p(\alpha_k | \mathbf{D}_{1:k-1}). \end{aligned} \quad (12)$$

The first step is the *prediction step* which will use the combined dynamics model, $p(\alpha_k | \alpha_{k-1})$, introduced in the succeeding sections to propagate the previous posterior, $p(\alpha_{k-1} | \mathbf{D}_{1:k-1})$, so as to give an estimate of the predictive distribution $p(\alpha_k | \mathbf{D}_{1:k-1})$. The second step is the *update step*, where the likelihood, $p(\mathbf{D}_k | \alpha_k)$, is combined with the predictive distribution to obtain the posterior filtering distribution at time k .

This problem is by nature non-linear/multi-modal and as such no closed-form filtering framework exists. However, Sequential Monte Carlo (SMC) methods provide accurate yet simple and computationally efficient estimation strategies for this framework. Essentially SMC, also known as particle filtering, involves a Monte Carlo implementation of the recursions in Eq. (12) by using a large set of discrete samples, or particles, with associated weights.

Having propagated the particles in the first step (prediction) and reweighting them with the likelihood function in the second

step (update) the particles are then resampled according to the new weights to have uniform weight distribution. Simple bootstrap filtering [2] and stratified resampling are used here. Other more elaborate schemes could easily be incorporated into our framework.

We will now go into some detail to explain the dynamics models used for orientation and location and finally the likelihood functions, including the novel orientation likelihood function.

4.1. Localization Movement Model

The motion, in the \mathcal{X} -coordinate, will be as described by Equation 1 in [7]. The parameter values in that paper are retained, viz $\beta_{\mathcal{X}} = 10\text{Hz}$ and $\bar{v}_{\mathcal{X}} = 1\text{m/sec}$. Movement in the \mathcal{Y} -coordinate will be independent to this but will use an identical framework. Note that a more accurate and complex model could be trained on a set of representative movements but this has not been done here so as to maintain generality and simplicity.

4.2. Orientation Movement Model

For the orientation movement model we will again use the model from Equation 1 [7] where the orientation portion of the state, at time k , is defined as $\alpha_{o,k} \triangleq [\theta_k, \dot{\theta}_k]$ and the parameters $\beta_{\mathcal{X}}$ and $\bar{v}_{\mathcal{X}}$ are replaced by $\beta_{\theta} = 10\text{Hz}$ and $\bar{v}_{\theta} = 0.1\text{rad/sec}$. Note that as in all the following orientation calculations, particle orientations are limited to the range $[-\pi : \pi]$ with wrap-around applied to all particles propagated to positions outside this range.

5. MEASUREMENT LIKELIHOOD MODEL

The next step is to outline the likelihood function for joint orientation and location tracking. Movement in each domain (i.e. orientation or location) is assumed to be independent and as a result is it possible to create separate likelihood functions for each. The overall likelihood function is simply the product of the individual likelihood functions

$$\begin{aligned} p(\mathbf{D}_k | \alpha_k) &= p(\mathbf{D}_k | \alpha_{l,k}, \alpha_{o,k}) \\ &= p(\mathbf{T}_k | \alpha_{l,k}) p(\mathbf{M}_k | \alpha_k). \end{aligned} \quad (13)$$

Note that to track speaker location alone, as per [7], it is simply a case of setting $p(\mathbf{M}_k | \alpha_k) \propto 1$ and removing all orientation parameters from the state vector. While conversely orientation-only tracking, with a fixed and known source location, is possible by setting $p(\mathbf{T}_k | \alpha_{l,k}) \propto 1$ and removing the location parameters. This is further discussed in Section 5.2 and experimentally illustrated in the Section 6.

Note: For the remainder of this section the time index k has been suppressed for ease of reading.

5.1. Localization Likelihood Functions

The localization likelihood function will follow a similar form to that introduced in [7]. This framework allows two hypotheses: either one of measurements is due to the source and the rest are due to clutter, \mathcal{H}_0 , or alternatively all of the measurements are due to clutter, \mathcal{H}_1 .

Hypothesis \mathcal{H}_0 : For microphone pair p , if a particular candidate delay measurement, $\hat{\tau}_c^{(p)}$, is due to the true source we will represent the resultant likelihood function using a normal distribution as follows

$$p(\hat{\tau}_c^{(p)} | \alpha_l, \mathcal{H}_0) = c_{\alpha_l} \mathcal{N}(\hat{\tau}_c^{(p)}; \hat{\tau}_\alpha, \sigma_l^2) \quad \text{for } |\hat{\tau}_c^{(p)}| \leq \tau_{\max}^{(p)} \quad (14)$$

where c_{α_l} is a normalizing constant due to the limited admissible delay region. The TDE is assumed to be corrupted by Gaussian observation noise with variance σ_l^2 . This assumption is further discussed in [7]. The normalizing constant is obtained using the Gaussian error function.

Hypothesis \mathcal{H}_1 : The likelihood of one of the measurements being associated with clutter is given by a uniform distribution within the admissible interval

$$p(\hat{\tau}_c^{(p)} | \alpha_l, \mathcal{H}_1) = \mathcal{U}_{\mathcal{D}}(\hat{\tau}_c^{(p)}) \quad (15)$$

These measurements are summed together to give a final likelihood function for microphone pair p :

$$p(T^{(p)} | \alpha_l) = \sum_{c=0}^{N_c} q_c p(T^{(p)} | \alpha_l, \mathcal{H}_c) \quad (16)$$

where q_c are the prior hypothesis probabilities which are commonly held to be equal but may also be chosen to reflect confidence in the measurements. Finally the overall likelihood function is the product of the likelihood functions for each microphone pair

$$p(\mathbf{T} | \alpha_l) = \prod_{p=1}^{N_p} p(T^{(p)} | \alpha_l) \quad (17)$$

5.2. Proposed Orientation Likelihood Functions

In this section a novel algorithm is proposed which estimates an orientation likelihood function, $p(\mathbf{M} | \alpha_o)$, for the particle set at the current time. (A number of different algorithms were implemented however space restrictions inhibit their discussion here). It uses information from the measurement magnitude vector, \mathbf{M} , as well as particle locations and orientations to create a representative likelihood function for each microphone pair, which are then combined to give an overall likelihood function.

As mentioned above if the source is stationary then orientation-only tracking may be implemented if the source position, (\mathbf{x}, \mathbf{y}) , is known *a priori*. Examples of each type of tracking are shown in Section 6.

The hypotheses of the measurements being due to clutter or the true source are similar to that used in the previous section and are calculated for each microphone pair. Note that for this algorithm the maximum number of candidate pairs is limited, $N_{c,max} \leq 1$, therefore the individual microphone pair measurement vectors contain only a single element i.e. $M^{(p)} = \hat{m}^p$.

Hypothesis \mathcal{H}_0 : Consider microphone pair p , with microphones positions defined as $l_{p,1}$ and $l_{p,2}$. The midpoint of the microphone pair is then simply

$$l_{p,\text{mid}} = \frac{l_{p,1} + l_{p,2}}{2}. \quad (18)$$

Now consider a particle with position $l = (x, y)$; its position relative to the midpoint of microphone pair p is given by

$$r^{(p)} = l - l_{p,\text{mid}}. \quad (19)$$

Using this vector the likelihood function takes the form of a normal distribution as follows:

$$p(\hat{m}^{(p)} | \alpha, \mathcal{H}_0) \propto \mathcal{N}(\hat{\theta}^{(p)}; \theta, \sigma_{\hat{\theta}}^{(p)}). \quad (20)$$

where mean and variance are determined using the relevant magnitude measurement data, $\hat{m}^{(p)}$, as

$$\hat{\theta}^{(p)} = \angle(r^{(p)}), \quad \sigma_{\hat{\theta}}^{(p)} = \frac{a}{\|\hat{m}^{(p)}r^{(p)}\|} \quad (21)$$

where a is a constant experimentally determined using the data in Figure 3. As such the mean of the likelihood function is the relative angle between source and microphone pair while the standard deviation is inversely proportional to the measurement magnitude vector, $\hat{m}^{(p)}$, and the distance between the source and the microphone pair.

Hypothesis \mathcal{H}_1 : The clutter hypothesis for this algorithm is again a uniform distribution but must be formed for each microphone pair p :

$$p(M^{(p)}|\alpha, \mathcal{H}_1) = \mathcal{U}_{\mathcal{D}}(M^{(p)}). \quad (22)$$

From this the likelihood function, due the measurements from a particular microphone pair, is as follows

$$p(M^{(p)}|\alpha) = q_T p(M^{(p)}|\alpha, \mathcal{H}_0) + (1 - q_T)p(M^{(p)}|\alpha, \mathcal{H}_1) \quad (23)$$

where q_T is again the prior probability that the source is present in microphone pair p . Finally the complete likelihood function is product of the individual microphone pair likelihood functions

$$p(\mathbf{M}|\alpha) = \prod_{p=1}^{N_p} p(M^{(p)}|\alpha). \quad (24)$$

This algorithm has the advantage that the overall angle estimate need not be limited to within the angular distribution of the microphones. In other words it can produce an estimate in the full range $[-\pi : \pi]$ independent of microphone positioning.

6. REAL AUDIO EXPERIMENTS

6.1. Experimental Setup

A recording environment was a typical room measuring roughly 5.5m x 6.1m x 2.8m containing typical office furniture. (Note that none of the experiments were carried out on simulated data because of the ease of gathering real data). A set of 12 microphones, arranged into pairs, were set up at the same horizontal height - 1.2m. The intra-pairing spacing was uniformly 60cm. Accurate ground-truth location and orientation of the source and the microphones was provided via a commercial motion capture system. Accuracy of this system is estimated to be sub-millimeter [12]. The source used was a computer loudspeaker transmitting typical conversational speech. The samples were taken from digital recordings of BBC radio presenters. The duration of the audio samples varied from 20 seconds to several minutes. The recorded signals were band-passed to remove interfering components outside of the 200-6000Hz range.

The following parameters were used in the particle filter algorithm: Audio Sampling Rate, 16kHz; Number of particles, $N_p = 150$; Resampling Threshold, $N_{th} = 0.5$; Frame Length, $L = 512$ samples; Frame Overlap percentage, 50%; Update Rate, $\Delta T = 31.25\text{Hz}$; $a, 0.2$; $q_T, 0.2$ and $\sigma_l, 6e^{-5}$ (for the joint tracker only). While the number of particles used is much more than that used by [5] it is still sufficiently small to allow for real time performance and as such no optimization to reduce the number has been attempted.

6.2. Typical Tracking Examples

Two examples of typical recordings are discussed in this section. Both of these examples are typical tracking results using the algorithm proposed in Section 5.2. The first is an example of orientation-only tracking of a stationary source as seen in Figure 4. The particles are seen to clearly track the source's changing orientation.

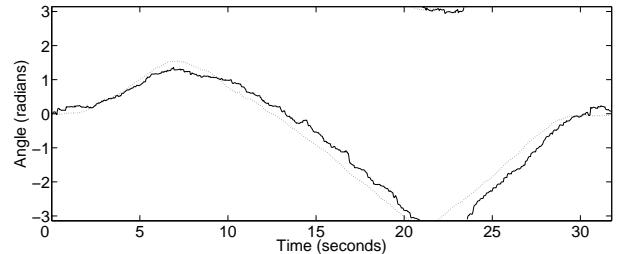


Figure 4: Example of orientation tracking performance (solid line) of a real source which is turning but stationary at the center of an office room. The dotted line represents the ground truth orientation.

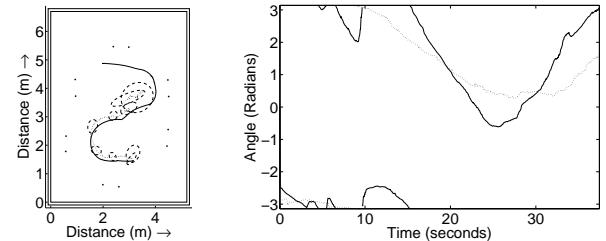


Figure 5: Example of joint location (left) and orientation (right) tracking performance (solid line) of a real moving and turning source (dotted line). Particle position and tracking performance is indicated by the loops of equal variance (dashed lines, top). The microphone positions and the room boundary are also shown.

The second example illustrated is of the source simultaneously moving and turning around the room. Figure 5 shows the results for joint tracking. The particle filter comfortably tracks the movement and orientation for the first two-thirds of the signal - as expected. The remainder of the test illustrates the performance limitations of bearings only tracking - and why speaker directivity and must be considered at a deeper level. At this source location and orientation only the microphone pair at the top-right of the room can provide bearings estimates - causing reduced performance radially relative to this microphone pair as the GCC functions from the other microphone pairs show very little correlation because of the directivity of the source.

6.3. Comparative Results

During this final section the source also turns quickly anti-clockwise causing difficulty for the orientation tracker. The particle filter does regain the correct track after a period of inaccuracy. The cause of this inaccuracy is twofold: firstly the impulse responses will be non-stationary across the data frames - leading to poor orientation estimates. Secondly the source orientation is changing rapidly - which the tracking system does not respond to because its behavior is outside of the typical movement envisaged by the movement model. Both of these problems and possible solutions are discussed in the Conclusions section.

Path	1	2	3	4	5
Location MSE	0.104	0.061	0.013	0.024	-
Location MSTD	0.051	0.070	0.06	0.074	-
Orientation MSE	0.018	0.098	0.311	0.113	0.145
Orientation MSTD	0.073	0.227	0.115	0.377	0.300

Table 1: Results for experimental tracking for each recording. The units of the performance measures are as follows: m^2 , m , rad^2 , and rad .

To evaluate performance the proposed algorithm was tested on 5 different audio recordings. The paths the speaker took in four of these recordings are shown in Figure 6. The source movement was approximately constant at typical walking speeds. Orientation was maintained in the direction of movement. The fifth recording is of a stationary source located at the co-ordinates (2.7m,3.9m) turning on its axis as shown in Figure 4. Because of the nature of the microphone pair spacings and the different paths and source signals used performance is expected to vary from recording to recording. Each algorithm was run 50 times and the statistics were then averaged to give the results in Table 1. The statistics evaluated are mean square error (MSE) and mean standard deviation (MSTD) of the particle cloud as suggested by [5]. The first statistic give an indication of tracking performance while the second is an estimate of tracking stability.

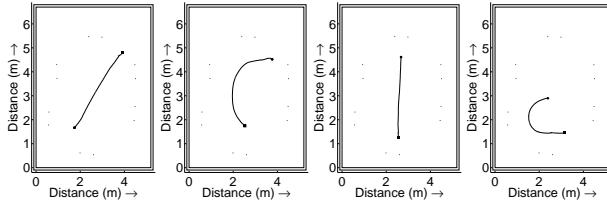


Figure 6: Source paths used to evaluate performance statistics. Each path runs from the square marker to the circle marker and lasts approximately 20-30 seconds, the room boundary and microphone positions are also shown. The paths are numbered 1-4 from left to right.

The results in Table 1 illustrate that the algorithm can robustly estimate both orientation and location jointly. As expected Path 1 has better orientation estimation performance as it does not change orientation while it moves across the room. Paths 2-5 all show similar orientation estimation performance. Paths 2 and 3 illustrate good tracking in the face of changes in both orientation and location.

7. CONCLUSIONS

In this paper the problem of orientation and location estimation of a moving speech source has been proposed and discussed. We introduced an algorithm to estimate and track speaker orientation while maintaining the same framework as previous localization literature. This allows us to simultaneously estimate both parameters jointly.

Results of real audio experiments were promising; however future work is still necessary. For example it is necessary to correct for the bias caused by an uneven distribution of microphones around the source. The likelihood function is by no means perfect

and as such a good deal of experimentation is required to achieve optimal performance. In a situation where only one microphone pair gave active measurements the performance was predictably poorer. A solution to this problem may be possible by considering the effect of speaker directivity within the localization estimator.

8. REFERENCES

- [1] J. DiBiase, H. Silverman, and M. Branstein, *Robust Localization in Reverberant Rooms in Microphone Arrays: Signal Processing Techniques and Applications*. New York: Springer, 2001, ch. 8, pp. 157–180.
- [2] N. Gordon, D. Salmond, and A. F. M. Smith, “Novel approach to nonlinear and non-Gaussian Bayesian state estimation,” *Proc. IEE-F*, vol. 140, pp. 107–113, 1993.
- [3] H. K. Dunn and D. W. Farnsworth, “Exploration of pressure field around the human head during speech,” *J. Acoust. Soc. Am.*, vol. 10, pp. 184–199, Jan. 1939.
- [4] E. A. Lehmann and R. C. Williamson, “Particle filtering design using importance sampling for acoustic source localisation and tracking in reverberant environments,” *EURASIP J. on Applied Sig. Proc.*, vol. 2006, Special issue on Advances in Multi-Microphone Speech Proc., article ID 17021, 9 pages, Jan. 2006.
- [5] D. B. Ward, E. A. Lehmann, and R. C. Williamson, “Particle filtering algorithms for tracking an acoustic source in a reverberant environment,” *IEEE Trans. Speech and Audio Proc.*, vol. 11, no. 6, pp. 826–836, Nov. 2003.
- [6] T. Betlehem and R. C. Williamson, “Acoustic beamforming exploiting directionality of human speech sources,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP’03)*, Hong Kong, China, april 2003, pp. 365–368.
- [7] J. Vermaak and A. Blake, “Nonlinear filtering for speaker tracking in noisy and reverberant environments,” *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, vol. 5, pp. 3021–3024, 2001.
- [8] J. Chen, J. Benesty, and Y. A. Huang, “Time delay estimation in room acoustic environments: an overview,” *EURASIP J. on Applied Sig. Proc.*, vol. 2006, Special issue on Advances in Multi-Microphone Speech Proc., article ID 26503, 19 pages, 2006.
- [9] D. N. Zotkin and R. Duraiswami, “Accelerated speech source localization via a hierarchical search of steered response power,” *IEEE Trans. Speech and Audio Proc.*, vol. 12, no. 5, pp. 499–508, Sept. 2004.
- [10] C. H. Knapp and G. C. Carter, “The generalized correlation method for estimation of time delay,” *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 24, pp. 320–327, Aug. 1976.
- [11] B. Champagne, S. Bedard, and A. Stephenne, “Performance of time-delay estimation in the presence of room reverberation,” *IEEE Trans. Speech and Audio Proc.*, vol. 4, no. 2, pp. 148–152, Mar. 1996.
- [12] PhaseSpace Inc., “PhaseSpace motion capture,” Retrieved June 29th, 2006, [Online] <http://www.phasespace.com>.

ON THE USE OF IRREGULARLY SPACED LOUDSPEAKER ARRAYS FOR WAVE FIELD SYNTHESIS, POTENTIAL IMPACT ON SPATIAL ALIASING FREQUENCY

Étienne Corteel

Room acoustics team
IRCAM, Paris, France
etienne.corteel@ircam.fr

sonic emotion
Oberglatt, Switzerland
etienne.corteel@sonicemotion.com

ABSTRACT

Wave Field Synthesis (WFS) is a physical based sound reproduction technique. It relies on linear arrays of regularly spaced omnidirectional loudspeakers. A fundamental limitation of WFS is that the synthesis remains correct only up to a corner frequency referred to as spatial aliasing frequency.

This paper addresses irregular spacing of loudspeaker array for WFS. Adapted driving functions are defined. New formulations of the spatial aliasing frequency are proposed. It is shown that the use of logarithmically spaced loudspeaker arrays can significantly increase the spatial aliasing frequency for non focused virtual sources.

1. INTRODUCTION

Wave Field Synthesis (WFS) is a holophonic technique that relies on the reproduction of physical properties of sound fields in an extended listening area [1]. Its original formulation relies on simplifications of the Rayleigh 1 integral. These approximations reduce the amount of required loudspeakers to a finite number of regularly spaced loudspeakers on a segment. They enable for the synthesis of the target sound field within a large portion of the horizontal plane up to a corner frequency referred to as "spatial aliasing frequency".

Irregular or "random" transducer spacing is currently employed in sound reproduction [2] or sound recording [3]. However, they have not been considered in the context of Wave Field Synthesis. This paper proposes to explore the potential benefits of the use of irregularly spaced arrays for WFS. Two test geometries are considered: "randomly spaced arrays and "symmetrical logarithmically" spaced arrays.

First, WFS driving functions for irregularly spaced arrays are proposed and the performance of the test arrays at low frequencies are analyzed. Accurate definitions of the spatial aliasing frequency are then given for finite length arrays considering both regular and irregular spacing of the transducers. Finally, potential improvements on the value of the spatial aliasing frequency compared to regular loudspeaker spacing are studied for various types of irregularly spaced loudspeaker arrays.

2. WAVE FIELD SYNTHESIS FOR IRREGULARLY SPACED LOUDSPEAKER ARRAYS

2.1. Wave Field Synthesis for continuous loudspeaker array

WFS relies on simplifications of the Rayleigh 1 integral [4]. This surface integral defines an infinite plane of "secondary" sources

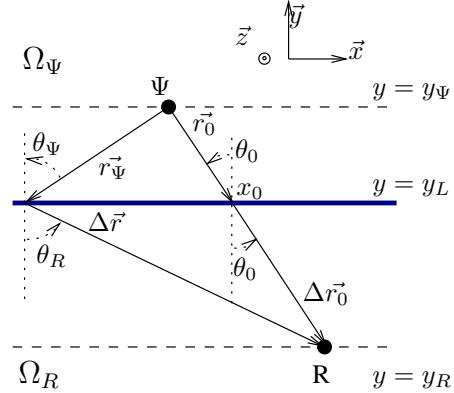


Figure 1: Synthesis of a virtual source using WFS, source/loudspeakers geometrical description.

that splits the space into two subspaces (cf. figure 1): a source subspace Ω_Ψ in which "primary" or virtual sources Ψ are, and a reproduction subspace Ω_R where the sound field they radiate is to be synthesized. WFS filters are derived by using the so-called stationary phase approximation as:

$$U(x_L, k) = F(k)G_\Psi(x_L)e^{-j(k\tau_\Psi(x_L)c)}, \quad (1)$$

for a given loudspeaker located at $x = x_L$ on an infinite horizontal line for the synthesis of an omnidirectional source Ψ (cf. figure 1). $F(k)$ is a filter introduced by the stationary phase approximation, which realizes a 3dB per octave attenuation and $\frac{\pi}{4}$ phase shift:

$$F(k) = \sqrt{\frac{k}{2\pi}}e^{j\frac{\pi}{4}}. \quad (2)$$

$\tau_\Psi(x_L)$ is a delay that accounts for natural propagation of the wave front from Ψ :

$$\tau_\Psi(x_L) = \frac{r_\Psi}{c}. \quad (3)$$

$G_\Psi(x_L)$ is a gain factor that stands for the natural attenuation of Ψ and compensates for level inaccuracies due to the natural attenuation characteristics of a linear array:

$$G_\Psi(x_L) = \cos(\theta_\Psi)\sqrt{\frac{|y_L - y_{R_{av}}|}{|y_{R_{av}} - y_\Psi|r_\Psi}}. \quad (4)$$

By definition, the synthesized level is thus only correct at an average listening depth $y_{R_{av}}$.

In a limit case, sources Ψ may also be located in Ω_Ψ by inverting

natural propagation delays. Synthesized wave fronts are converging to the target source position and thus propagate from this position in the rest of the reproduction subspace Ω_R . Such sources are therefore referred to as focused sources.

2.2. Wave Field Synthesis for sampled loudspeaker array

We consider a finite length continuous loudspeaker array parallel to the x axis ($z = 0, y = y_L$) such that $x \in [x_A, x_B]$. Its frequency response $H_\Psi(\vec{r}_R, k)$ at position \vec{r}_R for the synthesis of a virtual source Ψ using WFS filters (cf. equation 1) is given by:

$$H_\Psi(\vec{r}_R, k) = \int_{x_A}^{x_B} U(x_L, k) \frac{e^{-jk\Delta r(\vec{r}_R, x_L)}}{4\pi\Delta r(\vec{r}_R, x_L)} dx_L. \quad (5)$$

We define N sampling positions x_n , positions of the loudspeakers, and rewrite the previous equation as:

$$H_\Psi(\vec{r}_R, k) = \sum_{n=1}^N \int_{x_n - \Delta x_n^-}^{x_n + \Delta x_n^+} U(x_L, k) \times \frac{e^{-jk\Delta r(\vec{r}_R, x_L)}}{4\pi\Delta r(\vec{r}_R, x_L)} dx_L, \quad (6)$$

where Δx_n^- and Δx_n^+ determine a certain interval around x_n . The sum of these intervals spans the entire line L . Sampled driving functions $U_{\text{samp}}(x_n, k)$ may therefore be derived such that:

$$U_{\text{samp}}(x_n, k) \frac{e^{-jk\Delta r(\vec{r}_R, x_n)}}{4\pi\Delta r(\vec{r}_R, x_n)} \simeq \int_{x_n - \Delta x_n^-}^{x_n + \Delta x_n^+} U(x_L, k) \frac{e^{-jk\Delta r(\vec{r}_R, x_L)}}{4\pi\Delta r(\vec{r}_R, x_L)} dx_L \quad \forall n \in [1, N]. \quad (7)$$

The latter should remain valid at any listening position \vec{r}_R in Ω_R and for a certain frequency range.

We propose here to consider simple sampled WFS filters expressed as:

$$U_{\text{samp}}(x_n, k) = F(k) \frac{|x_{n+1} - x_{n-1}|}{2} G_\Psi(x_n) e^{-j(k\tau_\Psi(x_n)c)}. \quad (8)$$

These driving functions account for the local spacing of successive loudspeakers on the array. For regularly sampled arrays, the proposed formula remains coherent with known WFS filters. Additional attenuation factors may be introduced for loudspeakers located at extremities of the loudspeaker array in order to limit diffraction effect due to finite length of the array [4].

3. WAVE FIELD SYNTHESIS AT LOW FREQUENCIES

In this part, performances of irregularly spaced loudspeaker arrays for WFS rendering at low frequencies (below 1000 Hz) are compared with those of a reference regularly spaced loudspeaker array. The analysis considers a large number of sources and listening positions. The comparison is realized using perceptually relevant criteria.

3.1. Rendering accuracy evaluation

We simulate and compare for a listening position \vec{r}_j (x_j, y_j) the frequency response of the system $H_\Psi(\vec{r}_j, k)$ with an "ideal" WFS response $A_\Psi(\vec{r}_j, t)$:

$$A_\Psi(\vec{r}_j, k) = Att_\Psi^{wfs}(\vec{r}_j) e^{-jk d_\Psi^j}, \quad (9)$$

where Att_Ψ^{wfs} stands for the real attenuation of a WFS source synthesized with a linear loudspeaker array [5]:

$$Att_\Psi^{wfs}(\vec{r}_j) = \sqrt{\frac{|y_L - y_{R_{av}}|}{|y_L - y_j|}} \sqrt{\frac{|y_i - y_\Psi|}{|y_{R_{av}} - y_\Psi|}} \frac{1}{4\pi d_\Psi^j}. \quad (10)$$

A quality function $Q_\Psi(\vec{r}_j, f)$ that describes the deviation of the synthesized response from an ideal response can be defined in the frequency domain as:

$$Q_\Psi(\vec{r}_j, k) = \frac{H_\Psi(\vec{r}_j, k)}{A_\Psi(\vec{r}_j, k)} \quad (11)$$

Magnitude deviation $MAG_\Psi(\vec{r}_j, m)$ and group delay deviation $GD_\Psi(\vec{r}_j, m)$ are then calculated for $ERB_N(m)$ frequency bands [6]. They are simply obtained by averaging the corresponding quantities derived from $Q_\Psi(\vec{r}_j, k)$ in the equivalent frequency band.

The calculation considers 96 ERB_N bands for the entire audible frequency range. For the low frequency evaluation, it is however limited to frequency bands having their center frequency between 100 and 1000 Hz.

3.2. Test setup

We consider a test setup of 24 loudspeakers arranged in a 3.6 m long array. This corresponds to a regular spacing of 15 cm. Two alternative loudspeaker arrays of same length are considered:

- a "randomly" spaced loudspeaker array,
- a symmetrical "logarithmically" spaced array.

The latter is defined such that loudspeaker positions x_n are obtained from:

$$\begin{aligned} x_{n+1} - x_n &= (x_n - x_{n-1}) \times a^b \text{ if } n \geq 12 \\ x_{n+1} - x_n &= (x_n - x_{n-1}) \times a^{-b} \text{ otherwise,} \end{aligned} \quad (12)$$

We define a "loudspeaker spreading coefficient" ls_{spread} . In the

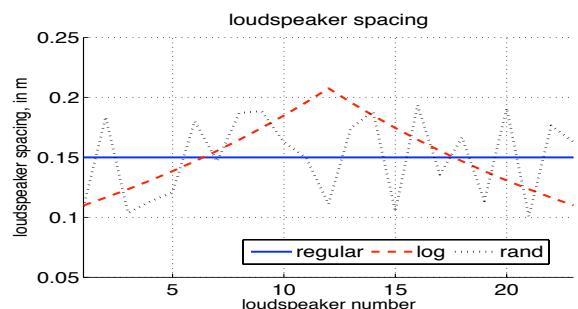


Figure 2: loudspeaker spacing for the three array type.

case of the randomly spaced array, $ls_{\text{spread}}^{rand}$ is simply defined as the ratio between the maximum and the minimum loudspeaker spacing. In the case of "logarithmically" spaced array, ls_{spread}^{log} is defined as the ratio between the spacing of the loudspeakers at the extremities of the array and the spacing of the loudspeakers at the center of the array. a and b are then calculated considering a given value of ls_{spread} and the total length of the array.

In the following, we consider $ls_{\text{spread}}^{rand} = 2$ and $ls_{\text{spread}}^{log} = 0.5$

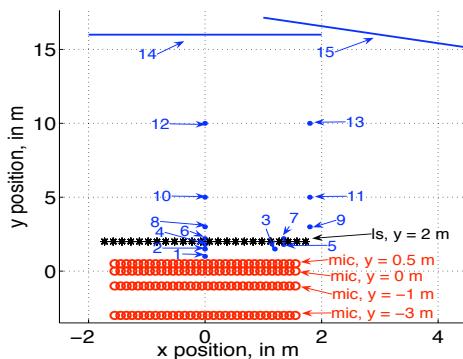


Figure 3: Top view of loudspeakers (black *), microphones (red o), and test sources (blue dots) configuration for regularly spaced loudspeaker.

(smaller spacing of the loudspeakers to the sides). The corresponding loudspeaker spacings are displayed in figure 2.

A test ensemble of 15 omnidirectional virtual sources (cf. figure 3) is composed of 5 centered and off-centered focused sources (sources 1/2/3/4/5), 8 centered and off-centered sources (sources 6/7/8/9/10/11/12), and 2 “plane waves” at 0 and 30 degrees (sources 14/15). The chosen test ensemble represents typical WFS sources reproduced by such a loudspeaker array. Figure 3 also displays measuring positions (microphone positions) at which the quality function Q_Ψ is evaluated for each source and loudspeaker array type.

3.3. Results

Tables 1 and 2 show mean values and standard deviation of MAG_{ERB} and GD_{ERB} calculated for all listening positions and virtual sources for the three loudspeaker array types between 100 and 1000 Hz.

	regular	log	rand
mean (dB)	-1.44	-1.40	-1.43
standard deviation (dB)	2.59	2.57	2.61

Table 1: Mean value and standard deviation of MAG_{ERB} considering all microphone positions and virtual sources between 100 and 1000 Hz.

	regular	log	rand
mean (ms)	0.13	0.13	0.13
standard deviation (ms)	0.88	0.87	0.88

Table 2: Mean value and standard deviation of GD_{ERB} considering all microphone positions and virtual sources between 100 and 1000 Hz.

The reproduction errors at low frequencies are due to known limitations of Wave Field Synthesis rendering (stationary phase approximation limitations, diffraction) that may be reduced using multichannel equalization methods such as described in [5] [7]. It can be seen that the three loudspeaker arrays show very similar performances in terms of both magnitude and group delay deviation. It can be expected that observed differences have no significant perceptual impact.

4. ALIASING FOR WAVE FIELD SYNTHESIS

The spatial sampling of the loudspeaker array limits the reconstruction possibilities of WFS at high frequencies. Contributions of individual loudspeaker do not fuse into a unique wave front as they do at low frequencies [8]. The synthesized sound field thus exhibits complex temporal and frequency characteristics [9] [8]. The spatial aliasing frequency corresponds to the corner frequency above which this phenomenon is noticeable. It is a key parameter for the analysis of the performances of a given loudspeaker array. Most available expressions of the aliasing frequency for WFS are given for infinite arrays of regularly spaced loudspeakers [9] [8]. They suggest that the aliasing frequency is independent of the listening position which is not true for finite length arrays [5].

In this section, alternative formulations of the spatial aliasing frequency are proposed that remain valid both for finite length and irregularly spaced loudspeaker arrays.

4.1. Frequency based evaluation of the aliasing frequency

4.1.1. Proposed criterion

We propose to extract the frequency response of the “aliased contributions” $H_\Psi^{al}(\vec{r}_R, k)$ from the frequency response of the considered array at position \vec{r}_R for the synthesis of source Ψ using:

$$H_\Psi^{al}(\vec{r}_R, k) = H_\Psi(\vec{r}_R, k) - H_\Psi^{noal}(\vec{r}_R, k), \quad (13)$$

where $H_\Psi^{noal}(\vec{r}_R, k)$ is the frequency response of a continuous linear array of same length for the synthesis of source Ψ . The exact response of a continuous array may be estimated as the frequency response of a regularly closely spaced (typically 1 cm) loudspeaker array. It is expected that for such an array aliasing artifacts are observed only above audible frequencies.

The aliasing frequency can thus be defined as the lower frequency for which the level of the aliased contributions exceeds a certain threshold $Tr_{al}^{simFreq}$:

$$f_{al}^{simFreq}(\vec{r}_R, \Psi) = \min_f(|H_\Psi^{al}(\vec{r}_R, k)| > Tr_{al}^{simFreq}(\vec{r}_R, \Psi)) \quad (14)$$

We propose to define this threshold as:

$$Tr_{al}^{simFreq}(\vec{r}_R, \Psi) = \frac{Att_\Psi^{wfs}(\vec{r}_R)}{2}, \quad (15)$$

which corresponds to half of the expected level at low frequencies.

4.1.2. Simulations

$H_\Psi^{al}(\vec{r}_R, k)$ is evaluated for the three loudspeaker array types for a centered omnidirectional source located 3 m behind the loudspeaker array (source 10 in figure 3). Considered listening positions \vec{r}_R are microphone positions at $y = 0$ m in figure 3.

Figures 4, 5, and 6 display the corresponding frequency responses. The frequency based aliasing criterion (cf. equation 14) is displayed on the figures as a magenta dashed-dotted line.

For both regularly spaced and logarithmically spaced loudspeaker arrays (cf. figures 4 and 5), a clear distinction can be observed between a low frequency response and high frequency response. At low frequencies, the level of the response is generally low (≈ -30 dB) whereas it raises quickly at higher frequencies and established a complex response with relatively high average level (≈ 0 dB). The frequency based criterion establishes thus

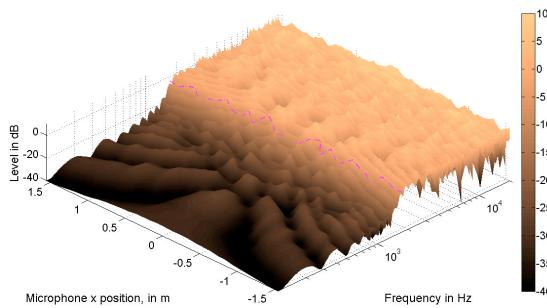


Figure 4: Frequency responses of the aliased field H_{Ψ}^{al} , source 10, regularly spaced array.

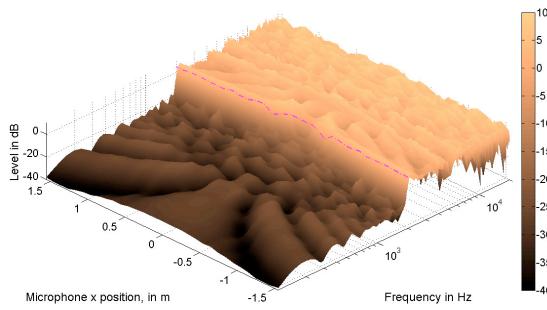


Figure 5: Frequency responses of the aliased field H_{Ψ}^{al} , source 10, logarithmically spaced array.

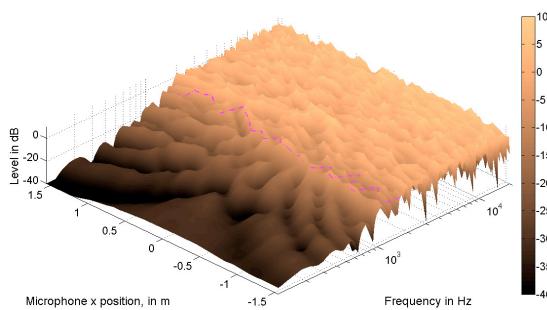


Figure 6: Frequency responses of the aliased field H_{Ψ}^{al} , source 10, randomly spaced array.

a clearly defined aliasing frequency. The same simulations were achieved considering other source/listening positions and have shown similar results.

For randomly spaced loudspeaker arrays, there is no such clear separation between low and high frequency responses. It can be seen that the "aliased field" has significant contributions ($\approx -15/-5$ dB) from frequencies as low as 1000 Hz. The aliasing frequency is thus hardly defined for that kind of loudspeaker array. The proposed sampled version of the WFS filters (cf. equation 8) is probably not completely valid for randomly spaced loudspeaker arrays. An alternative WFS filter definition may provide increased reconstruction performances at

higher frequencies but is out of the scope of this paper.

4.2. Temporal based evaluation of the aliasing frequency

The proposed frequency based criterion provides an accurate definition of the aliasing frequency. However, it requires the simulation of the aliased field response which is a computationally expensive task.

In this part, we propose a computationally efficient evaluation of the aliasing frequency which relies on sampling of the temporal response of the loudspeaker array at a given listening position.

4.2.1. Temporal response of a finite continuous array

In the following, the virtual source Ψ is located in Ω_{Ψ} and the 3 dB per octave filter $f(t)$ is omitted from the WFS filters to clarify the demonstration.

We define $t_{\Psi}(\vec{r}_R, x_L)$ as the arrival time at the listening position R of the contribution radiated by a secondary source at x_L :

$$t_{\Psi}(\vec{r}_R, x_L) = \frac{\Delta r}{c} + \tau_{\Psi}(x_L). \quad (16)$$

The impulse response h_{Ψ}^{wfs} of the continuous linear L for the synthesis of the source Ψ at \vec{r}_R is thus expressed as:

$$h_{\Psi}^{wfs}(\vec{r}_R, t) = \int_{x_A}^{x_B} G_{\Psi}(x_L) \frac{\delta(t - t_{\Psi}(\vec{r}_R, x_L))}{4\pi\Delta r} dx_L, \quad (17)$$

We introduce $t^-(x_L)$ and $t^+(x_L)$,

$$\begin{aligned} t^-(x_L) &= t_{\Psi}(\vec{r}_R, x_L) \quad \forall x_L \in]x_A, x_0], \\ t^+(x_L) &= t_{\Psi}(\vec{r}_R, x_L) \quad \forall x_L \in]x_0, x_B[\end{aligned} \quad (18)$$

where x_0 is the intersection of L and the line joining the source Ψ and the receiving position R (cf. figure 1). Similarly, we define:

$$\begin{aligned} x^-(t^-(x_L)) &= x_L \quad \forall x_L \in]x_A, x_0], \\ x^+(t^+(x_L)) &= x_L \quad \forall x_L \in]x_0, x_B[. \end{aligned} \quad (19)$$

Furthermore, we introduce:

$$\begin{aligned} h_{\Psi}^-(\vec{r}_R, t) &= \int_{x_A}^{x_0} G_{\Psi}(x_L) \frac{\delta(t - t^-(x_L))}{4\pi\Delta r(x_L)} dx_L, \\ h_{\Psi}^+(\vec{r}_R, t) &= \int_{x_0}^{x_B} G_{\Psi}(x_L) \frac{\delta(t - t^+(x_L))}{4\pi\Delta r(x_L)} dx_L. \end{aligned} \quad (20)$$

By definition, the function $t^-(x_L)$ is a strictly increasing function for $x_L < x_0$ and $t^+(x_L)$ is a strictly decreasing function for $x_L > x_0$. The impulse response h_{Ψ}^{wfs} is thus the sum of the impulse responses of the two parts of the loudspeaker array separated by x_0 (h_{Ψ}^- and h_{Ψ}^+).

By substituting $t^-(x_L)$ and $t^+(x_L)$ to x_L into equation 20 and using the fundamental property of the direct distribution:

$$\begin{aligned} h_{\Psi}^-(\vec{r}_R, t) &= -Y(t - t_0) \frac{G_{\Psi}(x^-(t))}{4\pi\Delta r(x^-(t))} \frac{dx^-(t)}{dt} Y(t_A - t), \\ h_{\Psi}^+(\vec{r}_R, t) &= Y(t - t_0) \frac{G_{\Psi}(x^+(t))}{4\pi\Delta r(x^+(t))} \frac{dx^+(t)}{dt} Y(t_B - t), \end{aligned} \quad (21)$$

where Y is the Heavyside function.

4.2.2. Derivation of aliasing criterion

Let's consider an array of N ideal omnidirectional loudspeakers located at $x_n, n = [1 \dots N]$ such that $x_{n+1} > x_n$ and $x_A < x_n < x_B, i = [1 \dots N]$. We define $n_0 = \min_n(x_n > x_0)$. The impulse response $h_\Psi^{samp}(r_R, t)$ of this array for the synthesis of source Ψ can be obtained from WFS filters (cf. equation 8) as:

$$h_\Psi^{samp}(r_R, t) = \sum_{n=1}^{n_0} \frac{|x_{n+1} - x_{n-1}|}{2} G_\Psi(x_n) \frac{\delta(t - t^-(x_n))}{4\pi\Delta r(x_n)} + \sum_{n=n_0}^N \frac{|x_{n+1} - x_{n-1}|}{2} G_\Psi(x_n) \frac{\delta(t - t^+(x_n))}{4\pi\Delta r(x_n)}. \quad (22)$$

Thus, it appears as the sum of time sampled versions of $h_\Psi^-(r_R, t)$ and $h_\Psi^+(r_R, t)$:

$$h_\Psi^{samp}(r_R, t) = h_\Psi^-(r_R, t) \left(\sum_{n=1}^{n_0} \frac{|x_{n+1} - x_{n-1}|}{2} \frac{dt^-(x_n)}{dx^-} \delta(t - t^-(x_n)) \right) + h_\Psi^+(r_R, t) \left(\sum_{n=n_0+1}^N \frac{|x_{n+1} - x_{n-1}|}{2} \frac{dt^+(x_n)}{dx^+} \delta(t - t^+(x_n)) \right). \quad (23)$$

The spatial sampling of the loudspeaker array is thus equivalent to irregular time sampling of both $h_\Psi^+(r_R, t)$ and $h_\Psi^-(r_R, t)$. The minimum Nyquist frequency associated to each of the irregular temporal sampling therefore corresponds to the spatial aliasing frequency evaluated at R .

As for regular sampling, the Nyquist frequency is linked to the sample distribution, and especially to the time difference between successive samples. Two temporal distributions have to be considered: $t^-(x_n)$ for $n \leq n_0$ and $t^+(x_n)$ for $n > n_0$. The arrival time differences $\Delta\tau_R^\Psi(n)$ can be defined as:

$$\begin{cases} \Delta\tau_R^\Psi(n) = t^-(x_{n-1}) - t^-(x_n) & \text{for } n_A < n \leq n_0 \\ \Delta\tau_R^\Psi(n) = t^+(x_{n+1}) - t^+(x_n) & \text{for } n_0 < n < n_B. \end{cases} \quad (24)$$

We propose to define the spatial aliasing frequency f_{al}^{temp} derived from this analysis of the temporal response of the array as:

$$f_{al}^{temp}(r_R, \Psi) = \frac{g_{al}}{\max_{n \in N_{sel}(\Psi, r_R)} |\Delta\tau_R^\Psi(n)|}, \quad (25)$$

where g_{al} is a weighting factor and $N_{sel}(\Psi, r_R)$ is a subset of $n = [1 \dots N]$ defined as:

$$N_{sel}(r_R, \Psi) = \left\{ n, \frac{G_\Psi(x_n)}{4\pi\Delta r(x_n)} > tr_{al} \cdot \max_{i=[1 \dots N]} \left(\frac{G_\Psi(x_i)}{4\pi\Delta r(x_i)} \right) \right\}, \quad (26)$$

where tr_{al} is a threshold value used for the selection of loudspeakers that contribute significantly to the sound field at position R , recalling that $\frac{G_\Psi(x_i)}{4\pi\Delta r(x_i)}$ is the level of the contribution of loudspeaker i at R for the synthesis of Ψ .

Both g_{al} and tr_{al} are free parameters of the proposed calculation method. Optimization is proposed in the following.

4.2.3. Validation

The free parameters of the time domain method have been set so as to minimize the root mean square error of the time based estimation compared to the frequency based estimation of the aliasing frequency. Only regularly and logarithmically spaced loudspeaker arrays were considered. The obtained values are $g_{al} = 0.95$ and

Array type	Mean error	Standard deviation
Regular	-1.92%	7.69%
Logarithmic	1.12%	4.38%
Random	2.92%	22.58%

Table 3: Error of aliasing frequency using time based compared to simulation based estimation for the three loudspeaker array types, considering all sources and microphone positions, cf. figure 3.

$$tr_{al} = -13dB.$$

Table 3 presents mean values and standard deviation of the estimated error of the aliasing frequency using the temporal based criterion compared to the frequency based criterion. It can be seen that for finite length and/or logarithmically spaced loudspeaker arrays, the aliasing frequency can be reliably estimated using temporal based criterion which is computationally more efficient than frequency based criterion.

For the randomly spaced loudspeaker array, both criteria provide rather dissimilar results. However, for this type of array, the aliasing frequency is difficult to define (cf. section 4.1.2).

5. ALIASING FREQUENCY DEPENDENCY ON LOUDSPEAKER SPACING

In this section we compare irregularly spaced loudspeaker arrays with regularly spaced arrays in terms of obtained aliasing frequency. The test parameter is the loudspeaker spreading coefficient that determines the amount of irregularity introduced in the loudspeaker spacing.

5.1. Aliasing for randomly spaced loudspeakers

Figure 7 shows quantiles (0.1, median, 0.9) of the aliasing frequency estimated with the frequency based criterion. The analysis is performed on "random" loudspeaker spacing for different values of ls_{spread}^{rand} . For each defined loudspeaker array all sources and all microphone positions of the test setup (cf. figure 3) are considered for the evaluation. We recall that $ls_{spread}^{rand} = 1$ corresponds to a regularly spaced loudspeaker array.

It can be seen that the aliasing frequency is generally lower for randomly spaced arrays than for regularly spaced arrays. A deeper analysis considering each source and listening position separately did not show any particular improvement. One should consider however that the aliasing frequency is not properly defined for this kind of array.

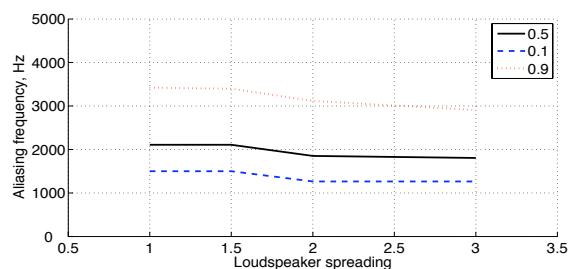


Figure 7: Quantiles of aliasing frequency, randomly spaced arrays, all sources and microphone positions, spreading coefficient dependency.

5.2. Aliasing for logarithmic loudspeaker arrays

For logarithmically spaced loudspeaker arrays, a spreading coefficient below 1 corresponds to a larger spacing to the sides compared to the center, whereas a spreading coefficient above 1 implies a smaller spreading to the sides.

Figure 8 shows quantiles (0.1, median, 0.9) of the aliasing fre-

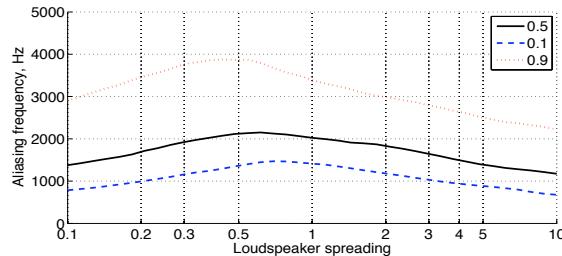


Figure 8: Quantiles of aliasing frequency, logarithmically spaced arrays, all microphone positions, loudspeaker spreading coefficient dependency, all sources.

quency estimated with the time based criterion for different values of ls_{spread}^{log} considering all sources and all microphone positions of the test setup (cf. figure 3). It can be seen that all spreading coefficients above 1 generally decrease the aliasing frequency, whereas spreading coefficients around 0.5 provide a slight increase of both median and 0.9 quantile.

Figures 9 and 10 show respectively quantiles of aliasing frequency considering non-focused sources only (sources 6 to 15 in 3) and focused sources only (sources 1 to 5 in 3). This analysis shows significant increase of the aliasing frequency using a logarithmically spaced loudspeaker array for non-focused sources for a loudspeaker spreading coefficient of 0.5. Most significant increase is for the 0.9 quantile value which raises by more than 20 % compared to regularly spaced loudspeakers.

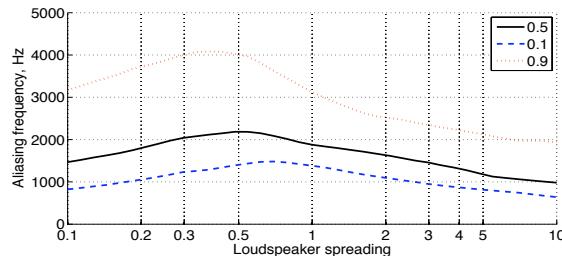


Figure 9: Quantiles of aliasing frequency, logarithmically spaced arrays, all microphone positions, spreading coefficient dependency, non-focused sources only.

However, it can be seen from figure 10 such loudspeaker spreading coefficients lower the aliasing frequency for focused sources.

6. CONCLUSION

In this paper, the potential use of irregularly spaced loudspeaker arrays for WFS has been addressed. Two test arrays have been compared to a regularly spaced loudspeaker array of same length. It has been shown that the three arrays have similar performances

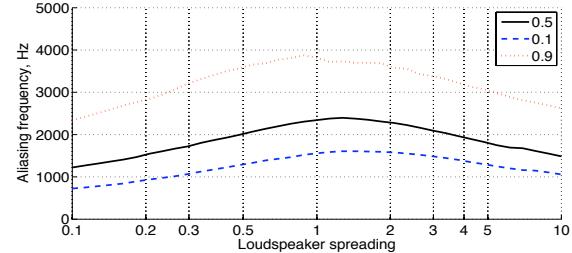


Figure 10: Quantiles of aliasing frequency, logarithmically spaced arrays, all microphone positions, spreading coefficient dependency, focused sources only

at low frequencies. New formulations for aliasing frequency have been introduced. They provide accurate results for finite length arrays with both regular and irregular loudspeaker spacing. It has been shown however that the aliasing frequency is difficult to define for randomly spaced loudspeaker arrays. It was also shown that, for the considered loudspeaker arrays (24 channels, 3.6 m long), dual logarithmic spacing allows for a significant increase in the aliasing frequency considering non focused virtual sources. If both focused and non focused sources need to be rendered on the same array, regular spacing remains most effective.

7. REFERENCES

- [1] A. J. Berkhouit, D. de Vries, and P. Vogel, "Acoustic control by Wave Field Synthesis," *J. Acoust. Soc. Am.*, vol. 93, pp. 2764–2778, 1993.
- [2] M. van der Wal, E. W. Start, and D. de Vries, "Design of logarithmically spaced constant-directivity transducer arrays," *J. Audio Eng. Soc.*, vol. 44, no. 6, pp. 497–507, June 1996.
- [3] A. Laborie, R. Bruno, and S. Montoya, "High spatial resolution multichannel recording," in *116th Conv. Audio Eng. Soc.*, Berlin, Germany, Mar. 2004, poster Z7-3.
- [4] P. Vogel, "Application of Wave Field Synthesis in room acoustics," Ph.D. dissertation, TU Delft, The Netherlands, 1993.
- [5] E. Corteel, "Caractérisation et extensions de la wave field synthesis en conditions réelles d'écoute," Ph.D. dissertation, Paris 6 University, Paris, France, 2004, [Online] <http://mediatheque.ircam.fr/articles/textes/Corteel04a/>.
- [6] B. C. J. Moore and B. R. Glasberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *J. Acoust. Soc. Am.*, vol. 74, no. 3, pp. 750–753, Mar. 1983.
- [7] E. Corteel, U. Horbach, and R. S. Pellegrini, "Multichannel inverse filtering of multiexciter distributed mode loudspeaker for Wave Field Synthesis," in *112th Conv. Audio Eng. Soc.*, Munich, Germany, May 2002, preprint Number 5611.
- [8] R. Nicol, "Restitution sonore spatialisée sur une zone étendue: Application à la téléprésence," Ph.D. dissertation, Université du Maine, Le Mans, France, 1999, [Online] http://gyronymo.free.fr/audio3D/Guests/RozennNicol_PhD.html.
- [9] E. W. Start, "Direct sound enhancement by Wave Field Synthesis," Ph.D. dissertation, TU Delft, The Netherlands, 1997.

DETECTION OF ROOM REFLECTIONS FROM A BINAURAL ROOM IMPULSE RESPONSE

Sampo Vesa, Tapiola Lokki

Telecommunications Software and Multimedia Laboratory
Helsinki University of Technology
sampo.vesa@tml.hut.fi

ABSTRACT

A novel analysis method for binaural room impulse responses (BRIRs) is presented. It is based on the analysis of ear canal signals with continuous wavelet transform (CWT). Then, the cross-wavelet transform (XWT) is used for detection of the direct sound and individual reflections from a BRIR. The new method seems to time-localize the reflections quite accurately. In addition, the proposed analysis method enables detailed study of the frequency content of the early reflections. The algorithm is tested with both measured and modeled impulse responses. A comparison with an FFT-based cross-spectrogram is made. The results show that XWT has potential in audio signal analysis.

1. INTRODUCTION

In many cases detailed time scale analysis of binaural impulse responses is needed. For example, a dense pattern of early reflections is usually associated with good concert hall acoustics and it would be great to be able to study these reflections individually from measured responses. One practical application—in which details of early reflections is needed—is the auralization in slow motion using measured binaural impulse responses [1]. When using simulated (binaural) impulse responses the auralization in slow motion is simple: the speed of sound is slowed down by a certain factor in the auralization. This allows perception of the time and direction of arrival of individual reflections. If measured binaural responses are to be slowed down, the situation becomes more complicated, since individual reflections should be very accurately localized in time in order to isolate them from the original response. This calls for a method that can time-localize the individual reflections as accurately as possible.

The problem studied in this paper is time-localization of early reflections from a binaural impulse response. It is assumed that if a reflection occurs there is correlation in short time window between left and right ear canal signals. Auditorily motivated approaches typically employ filter banks and calculation of cross-correlation between channels. Here, a more signal processing oriented approach is taken. An obvious method would be the frame-wise calculation of Fourier cross-spectrum between the left and right channels. A wavelet-based approach was hypothesized to be a better alternative, because of better time resolution compared to frame-based approaches. Therefore, both frame-based and wavelet-based approaches are evaluated in the study.

An introductory paper on wavelet analysis with applications to time series analysis [2] provided inspiration for using wavelet methods for localizing the reflections. Wavelet analysis, in the form of filter bank decomposition, has been used for approximating room impulse responses in simulations [3]. The continuous

wavelet transform (CWT) has also been applied to audio signal processing previously for noise reduction and signal compression [4], intermodulation effects analysis [5], sound synthesis [6] and sound signal modeling [7]. The CWT has also been used for decomposition of room and loudspeaker impulse responses [8]. In the current work, the continuous, non-orthogonal and complex cross-wavelet transform (XWT) is used, because the interest is in the correlation between two time series, i.e., the left and right ear canal signals of a binaural impulse response. As far as we know, the cross-wavelet transform seems not to be applied to audio signal processing previously.

2. THE CONTINUOUS WAVELET TRANSFORM AND THE CROSS-WAVELET TRANSFORM

The CWT is a method that can be used for time series analysis. It gives a highly redundant time-frequency representation of a time series, being very different from the *discrete wavelet transform* (DWT), which gives a compact representation of the signal and is thus better suited for signal processing [2]. For time series analysis, the CWT is much more suitable. The DWT also has the disadvantage of an aperiodic shift in the time series giving a different wavelet spectrum [2].

The continuous wavelet transform for a signal $x(t)$ is defined as [9, 2, 8]:

$$W_x(t, s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} x(t') \psi^* \left(\frac{t' - t}{s} \right) dt' \quad (1)$$

where the asterisk (*) indicates complex conjugation, t is time (*translation*), s is *scale* (dilation) and $\psi(t)$ is the *wavelet function*.

For a discrete sequence, the continuous wavelet transform is defined as a convolution sum [2]:

$$W_x(n, s) = \frac{1}{\sqrt{s}} \sum_{n'=0}^{N-1} x(n') \psi^* \left(\frac{n' - n}{s} \right) \quad (2)$$

For more efficient computation, the Fourier transform of Eq. (2) is used. Two complex wavelets are used in this study, the first one being the Morlet wavelet, defined as [2]:

$$\psi(t) = \pi^{-1/4} e^{j\omega_0 t} e^{-t^2/2} \quad (3)$$

where t is a dimensionless time parameter and ω_0 is a non-dimensional frequency. Since the CWT can also be seen as a filter bank, ω_0 is also called the *center or oscillating frequency* of the wavelet. A wavelet with a better time resolution and poorer frequency resolution, compared to the Morlet wavelet, is the Paul wavelet [2]:

$$\psi(t) = \frac{2^m j^m m!}{\sqrt{\pi(2m)!}} (1 - jt)^{-(m+1)} \quad (4)$$

where m is the order of the Paul wavelet.

Because the interest in audio signal analysis is on frequency, the scale s (non-dimensional) should be converted into frequency f (in Hz), using the following equation [8]:

$$f = \frac{f_s f_0}{s} \quad (5)$$

where f_s is the sampling frequency (in Hz) and $f_0 = \frac{\omega_0}{2\pi}$ is the non-dimensional wavelet center frequency. It should be noted that Eq. (5) only holds for the Morlet wavelet. When generalizing to all possible analyzing wavelets, f_0 can be seen as a proportionality constant which depends on the particular choice of wavelet base and order/center frequency. The relationships between scale and frequency for different wavelets can be found in [2].

The CWT enables also to study similarities of two signals in the same way as FFT-based cross-spectrum. The equivalent CWT-based tool is the *cross-wavelet transform* (XWT) [10], also known as *cross-wavelet spectrum*:

$$W_{xy}(t, s) = W_x(t, s)W_y^*(t, s) \quad (6)$$

Because the interest in this paper is on time-localizing reflections, we are only concentrated on the power of the XWT, which is used for all plots. The phase of the XWT could possibly be used for calculating the azimuth angles of each individual reflection, but such a study is left for future work.

3. CROSS-WAVELET ANALYSIS OF BRIRS

The cross-wavelet transform gives information on the dependence between two signals as a function of time, similar to cross-correlogram (or the cross-spectrogram, i.e., the short-time cross-spectrum presented as a function of time). In our case these two signals are the left and right ear canal signals of a binaural impulse response. Therefore, the XWT should be useful in localizing individual reflections of a binaural room impulse response, which manifest themselves as correlation between left and right ear signals, the time lag of the correlation maximum being proportional to the azimuth angle of the reflection.

Fig. 1 compares the Fourier cross-spectrogram and cross-wavelet transforms calculated from a BRIR, measured in a small room (the listening room of the Laboratory of Acoustics and Audio Signal Processing at TKK, see Fig. 2). The Fourier spectrogram was calculated using a time-domain window length of 64 samples and a very small hop size of 2 samples. The time-domain windows were zero-padded to yield a 512-point FFT per frame. The cross-wavelet transforms were calculated using the Morlet and Paul wavelets, the scales ranging logarithmically (base 2) from 2 to 512 with 24 scales per octave, resulting in the total number of scales being 193. The frequency ranges covered were 83.4-21300 Hz for the Morlet wavelet and 61.7-15800 Hz for the Paul wavelet.

As can be seen in Fig. 1, the smaller details are given more emphasis in the cross-wavelet spectrograms. The difference in time and frequency resolutions of the Morlet and the Paul wavelets is also very evident. The logarithmic frequency resolution of the wavelet transforms is also clear in the figures. It is also clear how the time resolution of the wavelet transforms is much worse at low than high frequencies.

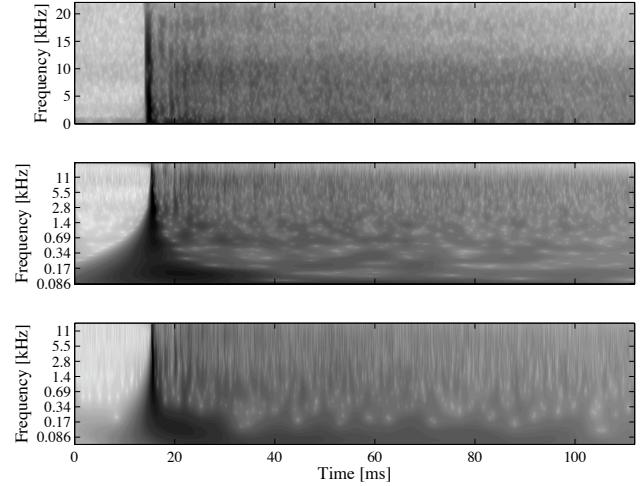


Figure 1: Comparison between the Fourier cross-spectrogram (top panel) and the cross-wavelet spectrogram calculated using the Morlet (middle panel) and the Paul (bottom panel) wavelets.

4. FINE-SCALE TIME-LOCALIZATION OF REFLECTIONS

Our hypothesis is that each reflection manifests itself as local maxima in the XWT on different scales at the time of the reflection, resulting in vertical stripes in the XWT plots (see Fig. 1). In order to use the XWT for time-localizing room reflections, we need to extract information on the time locations of the stripes from the XWT. An obvious way to do this is to “integrate out” the scale axis, i.e., to sum along the scales in the discrete case, yielding a one-dimensional signal, which is a function of time. Locations of the local maxima of this signal should correspond to the individual reflections. The sum across scale j_1 to j_2 is calculated from the cross-wavelet transform $W_{xy}(n, s)$ as:

$$W_{sum}(n) = \sum_{j=j_1}^{j_2} |W_{xy}(n, s_j)|^2 \quad (7)$$

As an alternative, the maximum across scale could be used:

$$W_{max}(n) = \max_{j \in [j_1, j_2]} \{|W_{xy}(n, s_j)|^2\} \quad (8)$$

Since it is expected that sum and/or maximum across scale could have some minor local maxima that do not correspond to reflections, Savitzky-Golay smoothing [11] was applied to them (using the function `sgolayfilt` in MATLAB). The filter order was 15 and the frame length was 23 samples. It turned out, that the maximum across scales was smooth enough as such and thus smoothing was only used for the sum across scale.

4.1. Choosing the set of scales

In contrast to orthogonal wavelet analysis, in non-orthogonal analysis the set of used scales can be chosen arbitrarily. The scales can be specified as fractional powers of two [2]:

$$s_i = s_0 2^{i\delta_i}, \quad i = 0, 1, \dots, I \quad (9)$$

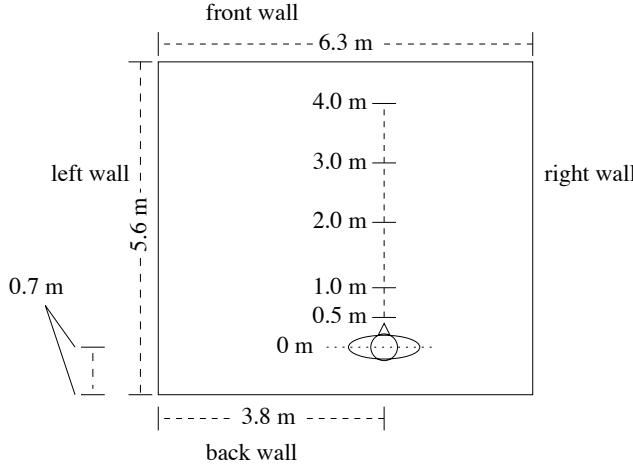


Figure 2: Position of the binaural manikin in the room where the BRIRs were measured. The room height is 3.0 m and the listener and the source height was 1.5 m. The binaural room impulse responses were measured at source-to-receiver distances of 0.5, 1, 2, 3 and 4 meters.

where s_0 is the smallest scale, I is the number of scales and δ_i controls the scale resolution ($D = 1/\delta_i$ gives the number of scales per octave).

5. EXPERIMENTS

As an example of the XWT analysis, both measured and modeled binaural impulse responses are analyzed and studied. First, the XWT is calculated for the entire measured binaural impulse response. This yields a matrix sized the length of the signal (in samples) times the number of scales the XWT was calculated at. The scale configuration is as described in Section 3. Since the largest scales convey little information about the time locations of the reflections, the XWTs were truncated in scales to the range $s \in [2, 64]$, which is 494-15800 Hz for the Paul wavelet and 667-21300 Hz for the Morlet wavelet. The range was chosen by hand so that the sum and maximum across scale seem to give a reasonable amount of detail, i.e., reasonable amount of local maxima. Including the frequencies below ~ 500 Hz results in the sum and maximum across scale being overly smooth, and thus some reflections will not be detected.

For comparison, a standard FFT-based cross-spectrogram (calculated as described in Section 3) is analyzed in the 500-16000 Hz band, which is close to the band covered by the Paul wavelet. After calculating the XWT or FFT-based cross-spectrogram, the sum and maximum across scale/frequency is evaluated using Eqs. (7) and (8). The curve representing the sum across scale is then smoothed as described previously. Finally, all of the local maxima of the resulting curves are located in time.

For a quantitative validation of the time-localization method, the exact locations of the room reflections should be known. For modeled responses, the exact delays for each reflection are known precisely. The situation is more complicated with measured responses, when there is no exact information on the timing of the reflections. However, since the room dimensions are known in this case (see Fig. 2), the image-source method can be used to calcu-

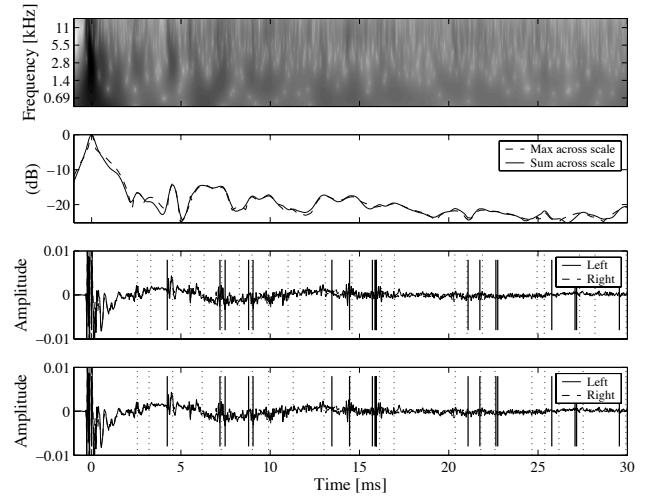


Figure 3: Top panel: the XWT of a binaural impulse response measured in a standard listening room (source-to-receiver distance 0.5 m, azimuth angle 0°). The Paul wavelet was used in the analysis. Upper middle panel: maximum of the XWT across scale, versus time (dashed line) and sum of the XWT across scale, versus time (solid line), presented on a logarithmic scale (base 2). Lower middle panel: the binaural impulse response, with the localized reflections marked by dotted vertical lines (time-localization based on local maxima of the maximum across scale) and the first and second order reflections calculated from a shoebox model marked by solid vertical lines. Bottom panel: same as the lower middle panel, but with time-localization based on local maxima of the sum across scale.

late the early reflections approximately. In this work, only the first and second order reflections were calculated.

5.1. Qualitative evaluation

Figs. 3 and 4 present the XWT of the first 30 ms of a binaural impulse response measured in the standard listening room (refer to Section 3) with $T_{60} \approx 0.3$ s, its maximum and sum across scale, and the localized reflections based on both the maximum and sum across scales. In Fig. 3, the Paul wavelet was the analyzing wavelet, while in Fig. 4, the Morlet wavelet was used. A localized reflection is set to each local maximum of the maximum or sum across scale, and is indicated by dotted vertical lines. The first and second order reflections given by the image-source model [12] are shown by solid vertical lines.

The better time resolution of the Paul wavelet (Fig. 3) is evident when compared to the Morlet wavelet (Fig. 4). The FFT-based analysis of a measured response (Fig. 5) seems to have a time-localization accuracy comparable to the Paul wavelet. By visual inspection it is hard to tell which peaks in the time-domain impulse response are true reflections, but using the Paul wavelet or the FFT cross-spectrogram seems to locate many details from the response. Many of these details are likely caused by reflections. Some of the reflections given by the image-source model also seem to coincide with the localized reflections. The room model agrees well with visual inspection of the time-domain responses in the case of the almost simultaneous floor and ceiling reflections (at

Method	Avg. abs. error [ms]								
	back wall	front wall	left wall	right wall	floor	ceiling	1st ord.	1st & 2nd ord.	
XWT, Paul wavelet (max)	0.24	0.51	0.29	0.23	0.063	0.21	0.26	0.33	
XWT, Paul wavelet (sum)	0.30	0.39	0.36	0.27	0.076	0.29	0.28	0.32	
XWT, Morlet wavelet (max)	0.39	0.52	0.67	0.20	0.12	0.27	0.36	0.63	
XWT, Morlet wavelet (sum)	0.43	0.39	0.33	0.29	0.048	0.24	0.29	0.47	
Fourier cross-spectrogram (max)	0.21	0.32	0.19	0.37	0.13	0.19	0.24	0.25	
Fourier cross-spectrogram (sum)	0.37	0.43	0.32	0.44	0.087	0.16	0.30	0.32	

Table 1: Average absolute errors for the detected reflections using sum and maximum across scale (measured responses).

Method	Average absolute error [ms]
XWT, Paul wavelet (max)	0.10
XWT, Paul wavelet (sum)	0.19
XWT, Morlet wavelet (max)	0.28
XWT, Morlet wavelet (sum)	0.33
Fourier cross-spectrogram (max)	0.29
Fourier cross-spectrogram (sum)	0.35

Table 2: Average absolute errors of reflections up to 30 ms using sum and maximum across scale (artificial response).

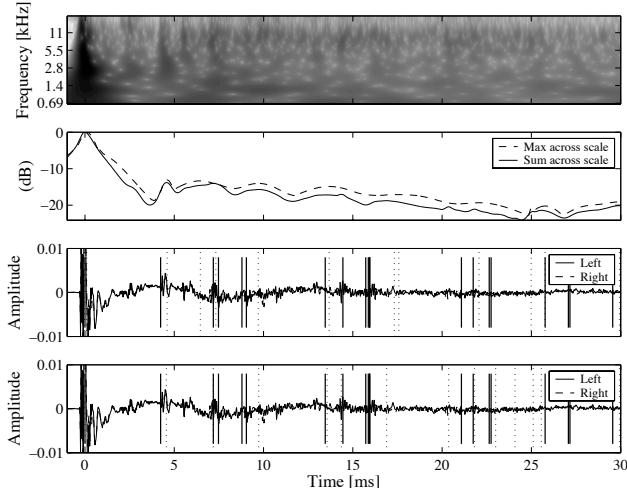


Figure 4: Same as Fig. 3, but the Morlet wavelet was used in the analysis.

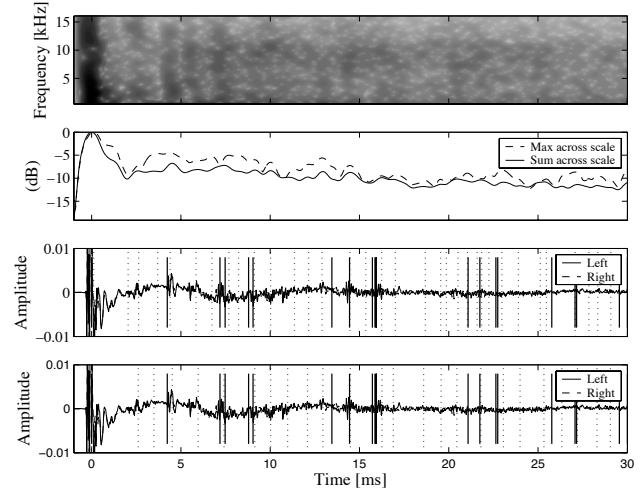


Figure 5: Same as Fig. 3, but a standard FFT-based cross-spectrogram (with a hop size of one sample) was used.

7.2 and 7.5 ms, respectively, according to the model). The analysis methods fuse the two reflections together, except with the combination of the FFT cross-spectrogram the maximum across frequency. The back wall reflection (at 4.2 ms) is correctly localized by both visual inspection and the analysis methods. However, room model may be slightly inaccurate, and therefore the three later reflections coming from the sides and the front wall may be inaccurate. By inspecting the two bottom panels of Fig. 3, it is very hard to tell, which of the “bumps” in the responses are actually due to the three later reflections. Besides the first and second order reflections from the walls, higher-order reflections and reflections from objects in the room appear in the response.

Fig. 6 presents the XWT analysis for an artificial modeled BRIR, using the Paul wavelet. The response is computed with the DIVA software [13] from a shoebox-shaped room. The reflections are modeled up to fifth order and each reflection is processed

with the appropriate head-related transfer functions to get a binaural response. As can be seen in Fig. 6, the reflections contain energy over the whole frequency region and no background noise exists. Such facts make the artificial response easier for the proposed algorithm and the algorithm has localized most of the peaks correctly. This is also verified by comparing the localized reflections to the ground truth (solid vertical lines) in the two bottom panels of Fig. 6.

By looking at the top panels of Figs. 3, 4 and 6 it is seen that the XWT can also be used to analyze the frequency content of the early reflections¹. As an example, the measured response (topmost panels of Figs. 3 and 4) contains a clear reflection (a darker area) right before the 5 ms time stamp. This reflection contains energy

¹Because of its linear frequency resolution, the FFT cross-spectrogram (Fig. 5) is not so well suited for detailed visual analysis of the frequency content, at least at the low frequencies.

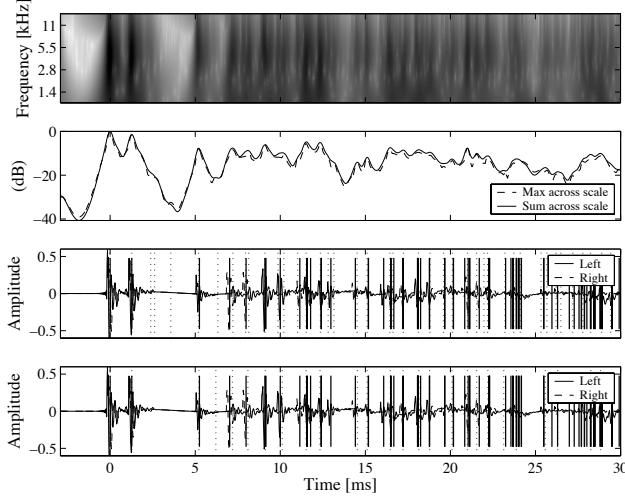


Figure 6: Same as Fig. 3, but a modeled BRIR was analyzed. The vertical solid lines in the two bottom panels indicate the ground truth locations for the reflections.

only at middle frequencies while next reflections a little bit later contain only low frequencies. From the artificial response analysis (Fig. 6) it can be seen that modeled reflections are all wideband reflections. Such analysis might be useful, e.g., in optimizing the loudspeaker placement in the home theaters, where some clear reflections typically occur.

5.2. Quantitative evaluation

Quantitative analysis of the accuracy of the time-localization is presented in Tables 1 and 2, where the average absolute errors of the time difference between the true reflection location and the nearest localized reflection are presented. This was done separately for the first and second order reflections in the case of the real responses, and for all of the reflections falling in the 30 ms time window following the direct sound in the case of the artificial response. For the measured responses (Table 1), the average error is taken as the arithmetic mean over the analysis results of set of binaural room impulse responses measured at source-to-receiver distances of 0.5, 1, 2, 3 and 4 meters (see Fig. 2). The average errors are presented for each of the six first order reflections separately. A total average for each analysis method is also given for the first order reflections and all calculated reflections, i.e., both first and second order reflections. Only the reflections falling within 30 ms of the direct sound are considered. A single artificial response was analyzed and the averages in Table 2 present the total average errors over all the reflections in the 30 ms time window following the direct sound, as calculated for each of the analysis methods.

By looking at Table 1 it seems that on average, the Paul wavelet and the FFT method perform equally well, while the Morlet wavelet performs slightly worse, on a real measured signal. The average absolute errors are always less than 1 ms. On average, the floor reflections are clearly located more accurately than the other reflections. Besides the floor reflection being a very strong one (it is actually superimposed with the ceiling reflection in this case), the theoretically calculated reflection is close to the true reflection.

With the other first order reflections listed in Table 1, as well as the second order reflections, the theoretical time locations might not hold exactly due to inaccuracies in the measured distances (Fig. 2) used to calculate the theoretical locations. Because of this, the results presented in Table 1 give only an approximate sense of how the methods perform in real spaces, and compared to each other. The methods also differ in terms of the density of localized reflections. Some of the localized reflections might not actually correspond to a real reflection, but are just statistical fluctuations. This also affects the results of Table 1 so that a low absolute error may also be due to a localized reflection being close to the theoretical location just by chance.

Table 2 indicates that for the artificial response, the Paul wavelet performs significantly better than the Morlet wavelet or the FFT method. The average error of the Paul wavelet is almost half that of the Morlet wavelet or the FFT. The performance of the wavelet methods is also significantly better than that of with measured responses. With the artificial response, reflections up to fifth order are considered, not just the first and second order reflections. The FFT method seems to be unable to localize each of these reflections individually. However, with the real response (Fig. 5) it seems that the FFT method gives more detected reflections than the wavelet methods. Conclusions from the superiority of the any of the methods can not be therefore drawn based on this.

The maximum across scale/frequency seems to be superior to the sum across scale/frequency for each method listed in Tables 1 and 2. The only exception is with the Morlet wavelet for the measured responses. The superiority of the maximum might be explained so that when a reflection is present in the signal at a certain time point, there is typically a clear peak in the XWT/cross-spectrogram at some frequency at that point. When the maximum across scale/frequency is evaluated as a function of time, a peak in the curve results at the very same time point, and thus the reflection is localized accurately. The sum across scale/frequency averages this peak so that the time accuracy of localization is worse and some narrow-band reflections are easily missed completely. On the contrary, the maximum across scales also gives rise to many more localizations, many of which may be just due to statistical fluctuations. From Figs. 3, 4 and 6 it is evident that there are more localized reflections when using the maximum across scales.

6. CONCLUSIONS

The cross-wavelet transform is applied for time-localizing reflections from binaural room impulse responses. The method performs accurately for artificial responses and finds almost all early reflections. Locations of the first and second order reflections were calculated for the room and a comparison to the reflections found by the proposed method was made. At least the strongest of the six first order reflections were localized quite accurately. However, the performance with real measured responses is hard to evaluate, since the exact locations of all of the reflections are not known. The proposed XWT with Morlet and Paul wavelets was also compared to the conventional FFT cross-spectrum. The Paul wavelet has the best time resolution, and thus seems to be the most accurate one of the tested methods, at least in the case of the artificially generated room response. For the measured response, the FFT cross-spectrogram method gave equally good results.

Future work should concentrate on improving the accuracy of the method. The algorithm should be made robust so that no reflections are missed and no false detections are made either. The

spectral content of the reflections should also be taken into account – there might be need to differentiate between low and high frequency reflections, as well as narrowband and wideband reflections. More advanced algorithms for detecting the reflections could be developed. Worth of investigation is also how the direction, i.e., azimuth angle, of the individual reflections could be calculated. One possibility is the use of the phase of the XWT for azimuth localization.

7. ACKNOWLEDGMENT

A freely available MATLAB toolbox for calculating the cross-wavelet transform and wavelet coherence was used [14]. This work was partly funded by the HeCSE graduate school.

8. REFERENCES

- [1] T. Lokki, “Auralization of simulated impulse responses in slow motion,” in *118th Conv. Audio Eng. Soc.*, Barcelona, Spain, May 2005, paper no. 6500.
- [2] C. Torrence and G. P. Compo, “A practical guide to wavelet analysis,” *Bulletin Am. Meteorological Soc.*, vol. 79, no. 1, pp. 61–78, 1998.
- [3] M. Schönle, N. Fliege, and U. Zölzer, “Parametric approximation of room impulse responses based on wavelet decomposition,” in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, Oct. 1993, pp. 68–71.
- [4] P. J. Wolfe and S. J. Godsill, “Audio signal processing using complex wavelets,” in *114th Conv. Audio Eng. Soc.*, Amsterdam, The Netherlands, Mar. 2003.
- [5] J. R. Beltrán, J. P. de León, and E. Estopiñán, “Intermodulation effects analysis using complex bandpass filterbanks,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, Sept. 2005, pp. 149–154.
- [6] J. R. Beltrán and F. Beltrán, “Additive synthesis based on the continuous wavelet transform: a sinusoidal plus transient mode,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, Sept. 2003.
- [7] P. Guillemain and R. Kronland-Martinet, “Characterization of acoustic signals through continuous linear time-frequency representations,” *Proc. IEEE*, vol. 84, no. 4, pp. 561–585, 1996.
- [8] S. J. Loutridis, “Decomposition of impulse responses using complex wavelets,” *J. Audio Eng. Soc.*, vol. 53, no. 9, pp. 796–811, 2005.
- [9] A. Cohen and J. Kovacevic, “Wavelets: The mathematical background,” *Proc. IEEE*, vol. 84, no. 4, pp. 514–522, 1996.
- [10] A. Grinsted, J. C. Moore, and S. Jevrejeva, “Application of the cross wavelet transform and wavelet coherence to geophysical time series,” *Nonlinear Processes in Geophysics*, vol. 11, pp. 561–566, 2004.
- [11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992, ch. 14.8.
- [12] J. B. Allen and D. A. Berkley, “Image method for efficiently simulating small-room acoustics,” *J. Acoust. Soc. Am.*, vol. 65, no. 4, pp. 943–950, 1979.
- [13] T. Lokki, “Physically-based auralization – design, implementation and evaluation,” Ph.D. dissertation, Helsinki University of Technology, Espoo, Finland, 2002, ISBN 951-22-6157-X.
- [14] A. Grinsted, J. C. Moore, and S. Jevrejeva, “Cross wavelet and wavelet coherence MATLAB toolbox,” 2006, [Online] <http://www.pol.ac.uk/home/research/waveletcoherence/>.

ARCHAEOLOGICAL ACOUSTIC SPACE MEASUREMENT FOR CONVOLUTION REVERBERATION AND AURALIZATION APPLICATIONS

Damian T. Murphy

Intelligent Systems — Audio Lab, Department of Electronics
University of York, Heslington, York, YO10 5DD, UK
dtm3@ohm.york.ac.uk

ABSTRACT

Developments in measuring the acoustic characteristics of concert halls and opera houses are leading to standardized methods of impulse response capture for a wide variety of auralization applications. This work presents results from a recent UK survey of non-traditional performance venues focused in the field of acoustic archaeology. Sites are selected and analyzed based on some feature of interest in terms of their acoustic properties. As well as providing some insight as to the characteristics and construction of these spaces, the resulting database of measurements has a primary use in convolution based reverberation and auralization. A recent sound installation based on one of the selected sites is also presented.

1. INTRODUCTION

Convolution based reverberation and auralization for audio post-production and computer music applications require a large, high quality database of Room Impulse Responses (RIRs). The static nature of RIR based reverb/auralization does not lend itself to real-time editing of the virtual environment in the same manner as more established IIR filter/circulant-network based implementations. Hence, a larger database of virtual spaces is required to attempt to cater for every creative possibility. These RIRs must therefore be obtained through a process of either recording/measurement or modeling. Current research in capturing the acoustic characteristics of concert halls and opera houses is leading to standardized methods for carrying out high quality RIR measurements that are compatible with many different spatial audio auralization and rendering systems.

This paper presents results from a study of four selected archaeological sites in the UK, each demonstrating some feature of interest in terms of their acoustic characteristics. These sites do not necessarily come under the focus of current acoustic survey and measurement work in traditional music performance venues, but rather achieve their reputation and interest for other notable reasons.

The aim of this work therefore lies in a number of different areas. The primary aim is to expand the range of current acoustic measurement surveys to provide an increased palette of virtual spaces, particularly for practitioners in the fields of audio post production and sound design/composition. In a recent survey of concert halls and opera houses the importance of capturing the unique sound of these spaces and preserving them for posterity is highlighted as another valuable aim [1]. A similar study uses the acoustics of early music spaces to inform the design of modern concert halls [2]. This work also leads to the possibility of using archaeological/architectural acoustic analysis and spatial sound for the

interpretation of important historical buildings or heritage sites. For the researcher such analysis may help to give further insight to the purpose of a site, its use or construction. The ability to audition and experience these sites via auralization will also help in the development of more rewarding and informative visitor interactives. This paper is organized as follows. In Section 2 the measurement techniques used for this study based on current best practice are defined, including surround-sound decoding and rendering options. Section 3 examines each of the four sites in turn and considers their relative acoustic characteristics, highlighting pertinent features appropriate for the interested convolution reverb user. Section 4 discusses how these techniques have been used to realize an audio-visual installation artwork and this paper is concluded with a summary of the work to date, and an indication of future directions for this measurement and research programme.

2. RIR MEASUREMENT SYSTEM

2.1. RIR Measurement Guidelines

There exist a number of prior studies that have explored methods for room acoustics measurement and impulse response capture. The CIARM group have written guidelines for measuring the acoustic characteristics of historical opera houses [3], using ISO3382 [4] as a reference. Recommendations include the use of omnidirectional sources exhibiting a wide, flat frequency response, with measurements recorded monaurally, binaurally and/or in B-format. The latter two approaches facilitate spatial analysis of the measured impulse responses. B-format recordings generally employ the use of the Soundfield Microphone, the output of which is four coincident signals corresponding to an omnidirectional sound pressure signal, W, and three figure-of-8 velocity responses, X, Y, and Z, directed as forwards-backwards, left-right and up-down respectively. With such a B-format measurement the W-component can be used for evaluating monaural acoustic parameters, and the additional directional signals to used evaluate spatial characteristics. The B-format signals can also be decoded for a wide variety of multi-channel surround-sound systems.

These guidelines have been developed and refined in more recent acoustic measurement work [2] especially that of Farina *et al.* [1], [5]. Two sound sources are also used, the first a small dodecahedral omnidirectional transducer, combined with a subwoofer, equalized to give a flat response between 80 Hz and 16 kHz. The second is a Genelec S30D, a three-way active multi-amped loudspeaker with AES/EBU digital input. Although it exhibits a more directional characteristic than the omni/subwoofer combination, its frequency response extends from 37 Hz to greater than 20 kHz with only ± 3 dB variation. The directivity pattern is

also more consistent across this frequency range than the omni/sub combination, and avoids the associated errors present at higher frequencies due to the spacing and arrangement of the individual drivers. The microphones used comprise an ORTF cardioid pair, a binaural dummy head and a Soundfield ST-250 all arranged on an automated rotating turntable. The dummy head and the point of intersection of the ORTF pair are arranged at the centre of the turntable's axis of rotation.

2.2. Transducers, Signal Generation and Capture



Figure 1: Measurement microphones mounted on a turntable and S30D sound source transducer.

The main aim this project is for multi-speaker RIR auralization of the studied spaces. Hence, the need for binaural measurement is not of paramount importance although if required at a later stage it is possible to derive a binaural representation from a B-format signal. The chosen microphone combination is both a refinement and simplification of that used in [1] and is shown in Fig. 1. A 4-channel Soundfield SPS422B is positioned on a boom arm, 1m from the centre axis of an automated rotating turntable. A single Neumann KM140 cardioid microphone is situated with the capsule end 10.4 cm from the centre axis, essentially one half of an ORTF pair spaced 17 cm apart at an angle of 110°. Both microphones are set at a height of 1.5 m. A 15 s 22 – 22000 Hz logarithmic sine sweep is used as the excitation signal via a Genelec S30D. This excitation-deconvolution technique has been shown to give better results than previously used methods in terms of signal-to-noise ratio, minimisation of harmonic distortion and not having to rely on repeated measurements and averaging for best results [6]. The rotating turntable is triggered automatically after each excitation sweep and measurements are made at 5° intervals over a complete 360° revolution. Typically, a single set of 72 measurements takes between 25–50 minutes depending on the reverberation time of the space being studied. Post-processing, deconvolution and objective parameter extraction from the resulting RIRs is carried out using Adobe Audition and the Aurora Plug-In Suite [7].

2.3. Decoding and Auralization

The 1 monaural + 4 B-format channels of RIR information can be combined for a wide variety of surround-sound rendering via an appropriate multi-channel audio convolution engine. Those most appropriate for this work are summarized as follows.

2.3.1. Stereo

ORTF stereo presentation for a source at angle θ is possible by selecting RIR pairs at $\theta \pm 55^\circ$. The measurement method employed in this study enables ORTF stereo auralization via only one directional microphone rather than the two used in [1].

2.3.2. Ambisonic Decoding

It is possible to facilitate 2-D or 3-D first-order Ambisonic surround-sound decoding for a mono source to a regularly arranged speaker system of diametrically opposing pairs using the Soundfield B-format RIR channels. W, X, Y give 2-D horizontal decoding with W, X, Y, Z used for full 3-D surround-sound.

2.3.3. B-format derived 5.1 Surround

The B-format responses can also be used to derive discrete 5.1 surround-sound via virtual microphone array modeling. A number of possible microphone arrays have been suggested for recording in ITU 5.1 surround-sound, using combinations of spaced directional (usually cardioid) microphones. It is possible to process and combine the measured B-format RIRs at a specific angle θ to generate any first-order microphone directivity pattern according to (1) and (2) as presented in, for example [8]:

$$V(\alpha, \beta) = \frac{1}{2} [(2 - d) \cdot W + d \cdot (r_x \cdot X + r_y \cdot Y + r_z \cdot Z)] \quad (1)$$

where:

$$\begin{cases} r_x &= \cos(\alpha) \cos(\beta) \\ r_y &= \sin(\alpha) \cos(\beta) \\ r_z &= \sin(\beta) \end{cases} \quad (2)$$

$V(\alpha, \beta)$ is the virtual microphone response, pointing in the direction of α , with elevation β , relative to angle θ around the central axis of measurement. W, X, Y, Z are the B-format RIRs in this case, and $0 \leq d \leq 2$ is the directivity factor where for instance $d = 0$ results in an omnidirectional response, $d = 1$ is a cardioid response and $d = 2$ is a figure-of-8 pattern. Using this method, virtual microphone responses at specific angular positions can be generated. These can then be used to simulate the equivalent discrete 5.1 spaced microphone array recording of a mono source in the measured space, offering an alternative auralization method to standard first order Ambisonic decoding.

2.3.4. High Order Spatial Decoding

Note also that other decoding schemes are also possible with additional post-processing, including higher order Ambisonic B-format according to Poletti's high directivity virtual microphones [9], Wavefield Synthesis [10], Spatial Impulse Response Rendering [11], as well as other hybrid approaches [8].

3. MEASUREMENT SITES

3.1. Acoustic Parameters

With a large database of RIRs available it becomes relatively straightforward to extract and analyze objective acoustic parameters to help quantify the characteristics of the measured spaces. The first parameter considered is ISO3382 T30 relating to the standard interpretation of *Reverberation Time* in octave bands and this is calculated and averaged from W-channel RIRs for angles 0°, 90°,

180° and 270°. The second parameter relates to the *Spatial Impression* of the space and there are a number of objective spatial measures that may be used to help characterize a space in terms of the perception of music heard within its walls, particularly with respect to how sound envelopes and surrounds a listener. *Inter Aural Cross Correlation* (IACC) is often used for binaural sound [12] and ISO3382 defines *Lateral Fraction*, LF [4]. LF can be derived from the W and Y channels of the B-format response, and [1] further defines a modified polar plot for the quantity ‘1-LF’ which shows the angular variation of LF (strictly 1-LF) as it varies with the locus of movement of the Soundfield Microphone in this particular measurement arrangement. The results are directly comparable with standard IACC if it similarly varies with angle, and can be used to give a measure of the diffusivity of the space with high values for 1-LF (hence low values of LF) indicating a highly diffuse soundfield.

3.2. St. Andrew's Church, Lyddington

St Andrew's Church, built in the 14th Century, has one of the finest examples of in-situ acoustic jars (vases or pots) in the UK. These jars were common to European church construction in the late Middle Ages and are said to be based on the ideas of Roman architect Vitruvius, who discussed the use of resonant jars in the design of amphitheatres to provide clarity of voice presentation [13]. They are designed as Helmholtz resonators, giving narrow band energy absorption according to the natural frequency of the jar although there is little conclusive acoustical evidence to show that they behave as designed. Studies suggest that the success or otherwise of these devices depends on the number of jars used and their placement, as well as the characteristics of the building and jars themselves. A ratio of one jar to 120 m³ is hypothesized as being a good example for success [14]. In anechoic and reverberant chambers the absorption effects of such jars are weak and highly selective, although can be significant below 200 Hz. Together with their additional diffusive effects, the jars potentially help to eliminate strong normal modes and hence can be made effective with careful tuning and positioning [15].

St Andrew's Church presents a good example for study with 11 jars placed high in the chancel, 6 in the north wall and 5 in the south, arranged at irregular intervals such that there are no directly opposite pairs. Although the total volume of the church (chancel + nave + aisles) is of the order of 2600 m³, the chancel has a volume of only 700 m³ giving a (Jar:Volume) ratio of 63 m³, well within the ratio suggested in [14]. A detailed consideration of the jars present on this site as presented in [16] reveals they are clearly non-optimal in their construction, exhibiting weak Helmholtz resonance effects with any absorption that might be demonstrable locally not evident in more general source/receiver positions. There are also strong axial modes evident in the chancel space where the jars are situated, with the critical frequency of this part of the building being approximately 170 Hz. The natural jar resonances are between 350–470 Hz and are therefore beyond this region of modal dominance where they might have helped to absorb problematic axial modes between north and south walls as their placement possibly indicates.

3.2.1. Reverberation Time, T30

St. Andrew's is considered as having a “good” acoustic and is widely used by musicians and ensembles as a performance venue.

Fig. 2 presents ISO3382 T30 values, calculated and averaged for the W-channel RIRs over angles 0°, 90°, 180° and 270°. The source was located at the altar steps in the chancel, with the microphone assembly placed in the nave giving a source-receiver distance of 11.5 m.

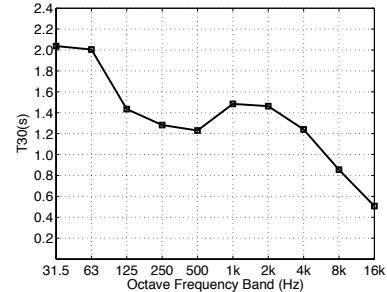


Figure 2: ISO3382 T30 value for St Andrew's Church W-channel RIR averaged over four measurement angles.

Considering the 1 kHz T30 value of 1.5 s as a measure of reverberation time perception note the significant bass rise in the two lowest octave bands, potentially helping to support the bass end of musical material presented in the space and together with the gradual roll off in the high end giving a sense of perceived warmth. The dip in the lower mid-range will help with clarity of speech and single note melody lines.

3.2.2. Spatial Impression, I-LF

Fig. 2 shows the polar plot for 1-LF calculated from the W and Y B-format responses at 10° increments, for a source placed at 0°. This result would indicate a reasonably diffuse space. Note that the 1-LF value is greatest directly to the front and rear demonstrating the coupled reverberant nature of the small chancel and the much larger nave, with most of the diffuse energy in the space arriving at the receiver from the rear (nave).

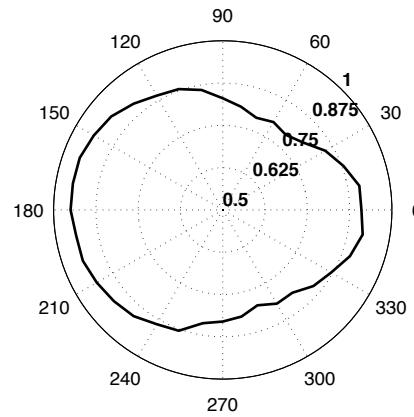


Figure 3: (1-LF) polar plots for a source at 0°.

3.3. Hamilton Mausoleum, Hamilton

Construction on the Hamilton Mausoleum, Hamilton, Scotland, built for the 10th Duke of Hamilton, started in 1842 and lasted

until 1858. It is constructed of marble and sandstone and is surmounted by a dome 36m in height, with two main spaces, a crypt in the lower section, and a chapel that was supposed to be used for worship. However the construction materials, size, shape and dimensions of the latter result in a complex, dense and very long reverberation, and hence render it almost useless for speech presentation. In fact the Guinness Book of World Records claims that the Hamilton Mausoleum has the longest “echo” of any building [17], recorded on 27 May 1994 as taking 15 s for the sound of the reverberation caused by slamming one of the main doors to die away to nothing. The space is now often used by recording musicians for its unique acoustic properties. The interior of Hamilton Mausoleum is approximately octagonal in plan, with a diameter of 18 m. Each side of the octagon is either a plane wall or a further semicircular alcove. The results presented below having the microphone assembly in the centre and the source placed to one side, just outside one of the alcoves, giving a source-receiver distance of 4.8 m.

3.3.1. Reverberation Time, T30

ISO3382 T30 is calculated and averaged for the W-channel RIRs over angles 0°, 90°, 180° and 270° as shown in Fig. 4.

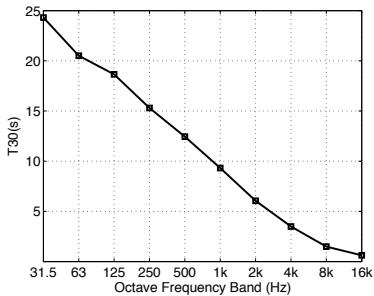


Figure 4: ISO3382 T30 values for Hamilton Mausoleum W-channel RIR averaged over four measurement angles.

The T30 values evident in Fig. 4 are clearly very dramatic, in excess of 20 s in the lower octave bands, falling almost linearly to 0.6 s at 16 kHz. Subjective listening confirms this, with the low frequency components for both the RIR in isolation and RIR/convolved audio lasting for some considerable time. Calculating T30 across the whole spectrum of the RIRs, rather than in octave bands gives a value of 15.0s — exactly that of the previously recorded “echo” duration. However, averaging over octave bands gives a value of 11.2 s, and the 1 kHz T30 value is 9.32 s, and these quantities perhaps give a more realistic measure of the perceived reverberation time.

3.3.2. Spatial Impression, I-LF

Fig. 5 shows the polar plot for 1-LF calculated from the W and Y B-format responses at 10° increments, for a source placed at 0°.

Note that there is a slight reduction in 1-LF at 140° and 240°. This is due to two of the four flat wall sections of the Mausoleum’s octagonal boundary, behind the microphone assembly when it is oriented at 0°. These hard flat surfaces act to generate strong specular reflections towards the centre of the space possibly over a

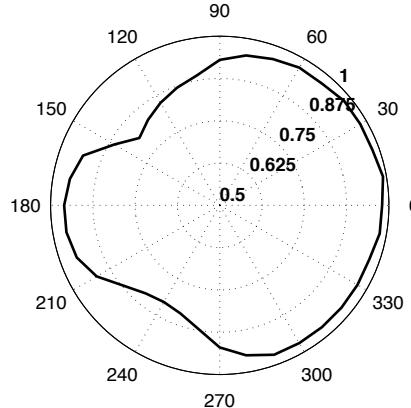


Figure 5: (I-LF) polar plots for a source at 0°.

closed path due to the symmetrical nature of the construction. Potentially this could result in stronger lateral reflections incident on the Soundfield Microphone, hence a reduction in overall diffusion for these regions. Note that relatively the drop in 1-LF is small and the effect is not completely symmetrical as the wall section at 240° has a large rectangular marble sarcophagus placed in front of it.

3.4. Maes Howe, Orkney

Maes-Howe, Orkney, is one of the finest chambered cairns in Europe, dated to 3000BC. Prior work in the acoustics of ancient sites explores how the resonances exhibited therein might have affected regular human ritual and interaction with the space. For instance [18] presents results from six such sites, revealing strong standing wave patterns between 95–120 Hz with minimal azimuthal or vertical variation. It is hypothesized that as these resonances are within the lower male vocal range, they may have been used in ritual to accentuate aspects of the voice. Unlike many similar ancient structures that have been studied to date, Maes Howe lends itself to the presence of strong modal frequencies. It is almost cubic in shape, of dimension 4.6 m, with walls made from large, flat slabs of stone, resulting in smooth reflecting surfaces rather than more commonly found irregular placement of smaller stones.

3.4.1. Reverberation Time, T30

ISO3382 T30 is calculated and averaged for the W-channel RIRs as before, with the source located at the mid-point of the centre wall, and the microphone assembly in the centre of the space giving a source-receiver distance of 2 m. The results are shown in Fig. 6.

Note the rise in T30 for the 63 Hz and 125 Hz bands. Calculating T30 across the whole spectrum, rather than in octave bands gives a value of 0.55s. Averaging over octave bands T30 = 0.57 s, and at 1 kHz T30 = 0.51 s. The rise to almost 0.9 s at 125 Hz is therefore significant and is due to the low frequency modal response, with strong peaks evident at 45, 90, 110, 120, 130 and 145 Hz [16].

3.4.2. Spatial Impression, I-LF

Fig. 7 shows the polar plot for 1-LF calculated from the W and Y B-format responses at 10° increments, for a source placed at 0°.

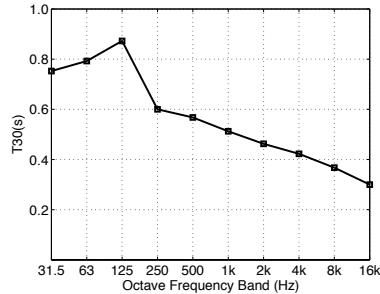


Figure 6: ISO3382 T30 values for W-channel RIR averaged over four measurement angles.

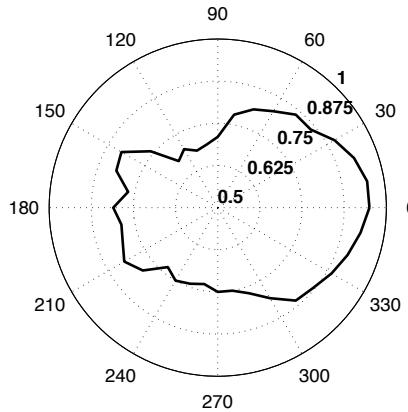


Figure 7: (1-LF) polar plots for a source at 0°.

These results indicate that this is a much less diffuse space compared with the other examples presented, and is to be expected given the small size and regular geometry of the chamber. There is also a general irregularity for 1-LF — the plot is less smooth as it varies with angle — due to dominant early reflections/direct sound present in this small space compared with late reverberation, reducing relative diffusivity according to angle of incidence at the receiver.

3.5. York Minster, York

York Minster is the largest medieval gothic cathedral in the UK and one of the finest in Europe, built between the 12th and 15th centuries on the foundations of the previous Norman church that was in turn constructed on the foundations of the original Roman fortress. It is approximately 160 m long, 76 m wide and 27 m high to the vaulted ceiling, constructed predominantly of stone with extensive, large panels of stained glass windows. Its beautiful acoustic and setting make it a sought after and highly popular music performance venue.

3.5.1. Reverberation Time, T30

ISO3382 T30 is calculated and averaged for the W-channel RIRs as before, with the source located directly under the centre tower, and the microphone assembly in the central nave area giving a source-receiver distance of 23.5 m. The results are shown in Fig. 8.

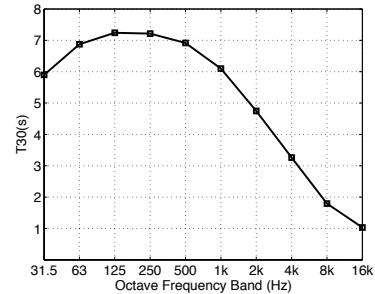


Figure 8: ISO3382 T30 values for W-channel RIR averaged over four measurement angles.

Note that the peak T30 value is 7.25 s in the 250 Hz band. Calculating T30 across the whole spectrum of the RIRs, rather than in octave bands gives a value of 6.4 s. Averaging over octave bands gives a value of 5.1 s, and the 1 kHz T30 value is 6.1 s.

3.5.2. Spatial Impression, I-LF

Fig. 9 shows the polar plot for 1-LF calculated from the W and Y B-format responses at 10° increments, for a source placed at 0°.

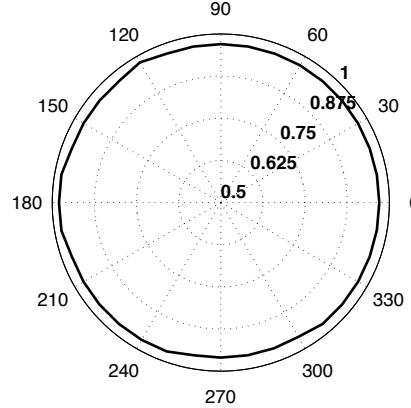


Figure 9: (1-LF) polar plots for a source at 0°.

York Minster is the largest of the presented spaces, with the greatest source-receiver distance (although still typical for music performances held here), and its construction and size result in long T30 values across octave bands. Hence, from Fig. 9 it is not surprising that the Minster demonstrates the highest level of diffusivity of all these results, being almost independent of direction, and hence demonstrating a high level of perceived spatial envelopment.

4. A SENSE OF PLACE

A Sense of Place is an interactive sound/light installation commissioned by the York Renaissance Project [19]. The artwork focuses on three specific aspects of York's 2000 year history interpreted using sound and light. The foundation of the piece is the Minster and what it has represented to York since the site was first used by the Romans. This has been realized using the Minster RIRs to render a

virtual acoustic representation of this complex, diffuse reverberant space. The work is presented in one of the gatehouses on the city walls (Bootham Bar) transforming it from a small, enclosed space into a larger and more dramatic sounding virtual acoustic representation of the Minster. Further details about the project can be found at the accompanying online web resource [20].

5. CONCLUSIONS

This paper has demonstrated how developments in room acoustics measurement for auralization can be applied to the field of acoustic archaeology. RIR analysis can provide an insight to the characteristics and construction of these spaces, and the resulting database of measurements has a primary use in convolution based reverberation/auralization. The spaces examined all demonstrate some specific interest in their acoustics or construction. With the exception of the highly modal acoustics of Maes Howe, the sites demonstrate long T30 values and highly diffuse soundfields possibly limiting musical application of these RIRs to specific genres or post-production situations. Therefore it is important to continue to develop this RIR database, surveying a wider range of sites and extending the areas where these RIRs might be creatively applied. Selected RIRs will be available to the audio/computer music community as part of the UK Spatial Audio Creative Engineering research network (SpACE-Net) website [21] and will go live in September 2006.

6. ACKNOWLEDGEMENTS

This work was by the Arts & Humanities Research Council/Arts Council England, award AN8885/APN16671 and the York Renaissance Project. Recognition also goes to the collaborators in this work: Helen Dorward for her valuable field work; John Oxley, City of York Archaeologist; Mark Hildred of Immersive Media Spaces Ltd. Thanks are extended to the custodians of these sites: the Reverend Jane Baxter, the wardens, and Parochial Church Council at the Benefice of Lyddington, St Andrew; Gillian Urquhart and Alan Jones at Historic Scotland (Maes Howe); Linda Barrett and staff at Low Parks Museum, Hamilton (Hamilton Mausoleum); Dean and Chapter of York Minster and the on-site Minster Police.

7. REFERENCES

- [1] A. Farina and R. Ayalon, "Recording concert hall acoustics for posterity," in *Proc. AES 24th Int. Conf. on Multi-channel Audio*, Banff, Canada, June 26-28 2003, [Online] <http://pcfarina.eng.unipr.it/Public/Papers/185-AES24.PDF>.
- [2] A. Bassuet, "Acoustics of early music spaces from the 11th to 18th century (abstract)," *J. Acoust. Soc. Am.*, vol. 115, no. 5, pt 2, p. 2582, May 2004.
- [3] R. Pompoli and N. Prodi, "Guidelines for acoustical measurements inside historical opera houses: Procedures and validation," Retrieved June 29th, 2006, [Online] <http://acustica.ing.unife.it/ciarm/download.htm>.
- [4] ISO3382, "Acoustics – measurement of reverberation time of rooms with reference to other acoustical parameters," ISO, Tech. Rep., 1997.
- [5] A. Farina and L. Tronchin, "Advanced techniques for measuring and reproducing spatial sound properties of auditoria," in *Int. Symp. Room Acoustics: Design and Science*, Kyoto, Japan, Apr 11-13 2004, [Online] <http://pcfarina.eng.unipr.it/Public/Papers/190-RADS2004.pdf>.
- [6] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept sine technique," in *108th Conv. Audio Eng. Soc.*, Paris, France, Feb. 18-22 2000, preprint No. 5093.
- [7] Aurora, "Plug-ins Home Page v4.0," Retrieved June 29th, 2006, [Online] <http://pcfarina.eng.unipr.it/aurora/home.htm>.
- [8] A. Farina, R. Glasgal, E. Armelloni, and A. Torger, "Am-biophonic principles for the recording and reproduction of surround sound for music," in *Proc. AES 19th Int. Conf. on Surround Sound, Techniques, Technology and Perception*, Schloss Elmau, Germany, June 21-24 2001, pp. 26–46.
- [9] M. A. Poletti, "A unified theory of horizontal holographic sound systems," *J. Audio Eng. Soc.*, vol. 48, no. 12, pp. 1155–1182, 2000.
- [10] E. Hulsebos, D. de Vries, and E. Bourdillat, "Improved microphone array configurations for auralization of sound fields by Wave-Field Synthesis," *J. Audio Eng. Soc.*, vol. 50, no. 10, pp. 779–790, 2002.
- [11] J. Merimaa and V. Pulkki, "Spatial impulse response rendering 1: Analysis and synthesis," *J. Audio Eng. Soc.*, vol. 53, no. 12, Dec. 2005.
- [12] Y. Ando, *Concert hall acoustics*. Berlin: Springer Series in Electrophysics, 1985.
- [13] I. Rowland and E. Howe, T. N., *Vitruvius: Ten Books on Architecture*. Cambridge: Cambridge University Press, 1999.
- [14] V. Desarnaulds, Y. Loerincik, and A. Carvalho, "Efficiency of 13th Century acoustic ceramic pots in two Swiss churches," in *Noise-Con*, Oct 29-31 2001, [Online] <http://paginas.fe.up.pt/~carvalho/nc01.pdf>.
- [15] A. Carvalho, V. Desarnaulds, and Y. Loerincik, "Acoustic behavior of ceramic posts used in middle age worship spaces, a laboratory analysis," in *9th Int. Congress Sound and Vibration*, Jul 8-11 2002, [Online] <http://paginas.fe.up.pt/~carvalho/icsv9.pdf>.
- [16] D. T. Murphy, "Multi-channel impulse response measurement, analysis and rendering in archaeological acoustics," in *119th Conv. Audio Eng. Soc.*, New York, USA, 2005, paper No. 6532.
- [17] Guinness World Records, "Longest lasting echo," 2004, 2005, [Online] http://www.guinnessworldrecords.com/content_pages/record.asp?recordid=47025&Reg=1.
- [18] R. G. Jahn, P. Devereux, and M. Ibison, "Acoustical resonances of assorted ancient structures," *J. Acoust. Soc. Am.*, vol. 99, no. 2, pp. 649–658, 1996.
- [19] Renaissance, "The York Renaissance Project," Retrieved June 29th, 2006, [Online] <http://www.renaissanceyork.org.uk/>.
- [20] D. Murphy, J. Oxley, and M. Hildred, "A Sense of Place," Retrieved June 29th, 2006, [Online] <http://www.boothambar.org.uk/>.
- [21] SpACE-Net, "The spatial audio creative engineering network – SpACE-Net," Retrieved June 29th, 2006, [Online] <http://www.space-net.org.uk/>.

IMPROVED COCKTAIL-PARTY PROCESSING

Alexis Favrot, Markus Erne

Scopein Research
Aarau, Switzerland

postmaster@scopein.ch

Christof Faller

Audiovisual Communications Laboratory, LCAV
Swiss Institute of Technology
Lausanne, Switzerland
christof.faller@epfl.ch

ABSTRACT

The human auditory system is able to focus on one speech signal and ignore other speech signals in an auditory scene where several conversations are taking place. This ability of the human auditory system is referred to as the “cocktail-party effect”.

This property of human hearing is partly made possible by binaural listening. Interaural time differences (ITDs) and interaural level differences (ILDs) between the ear input signals are the two most important binaural cues for localization of sound sources, i.e. the estimation of source azimuth angles.

This paper proposes an implementation of a cocktail-party processor. The proposed cocktail-party processor carries out an auditory scene analysis by estimating the binaural cues corresponding to the directions of the sources. And next, as a function of these cues, suppresses components of signals arriving from non-desired directions, by speech enhancement techniques. The performance of the proposed algorithm is assessed in terms of directionality and speech quality.

The proposed algorithm improves existing cocktail-party processors since it combines low computational complexity and efficient source separation. Moreover the advantage of this cocktail-party processor over conventional beam forming is that it enables a highly directional beam over a wide frequency range by using only two microphones.

1. INTRODUCTION

1.1. Overview

The “cocktail-party effect” is the ability of the human auditory system to select one desired sound from an ambient background of noise, reflections, or other sounds. For instance, at a party, where many talkers are speaking simultaneously, humans may focus their attentions on one voice and ignore other voices and noise which are possibly equally strong in loudness.

The concept of a cocktail-party processor, motivated by simulating electronically the “cocktail-party effect”, has been introduced earlier in [1]. The algorithm simulated neural excitation patterns based on specific physiological assumptions about the auditory system. Next, by a model of central stages of the signal processing in the auditory system, a spatial analysis of the auditory scene was performed in order to predict the azimuth angles of the sound sources. These spatial parameters were then used to control the transfer function of a time-variant filter, removing the components of signal arriving from non-desired directions.

For low computational complexity, the proposed cocktail-party

processor makes simplified physiological assumptions compared to [1]. Using a FFT based time-frequency representation of the ear input signals, the proposed algorithm first estimates the binaural localization cues (ITDs and ILDs) related to the azimuth angles of the sources to be recovered. Next, different speech enhancement techniques are controlled as a function of these binaural cues: in addition to conventional short-time spectral modification, as in [1], the proposed algorithm applies blind source separation in order to improve source separation.

1.2. Mixing model

The goal of the proposed cocktail-party processor is to recover a desired speech signal given two linear mixtures of speech signals, representing the right and left ear input signals, $x_R[n]$ and $x_L[n]$:

$$x_R[n] = \sum_{i=1}^N s_i[n] \quad \text{and} \quad x_L[n] = \sum_{i=1}^N a_i s_i[n - d_i], \quad (1)$$

where s_1, \dots, s_N are the N speech sources, spatially distributed as represented in Figure 1. a_i and d_i are the attenuation coefficient and time delay associated with the path from the i^{th} source to the left ear. The azimuth angle of the i^{th} source is Φ_i . Note that it is assumed that all sources are in different directions¹, and only the direct paths are considered, i.e. we assume anechoic conditions.

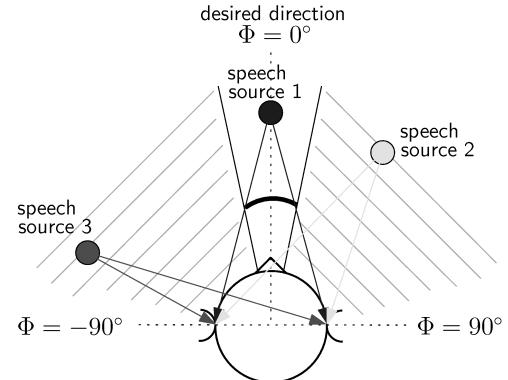


Figure 1: The ear input signals are linear mixtures of the speech signals coming from spatially distributed sound sources.

¹If several sources are in the same direction they are considered as a single source.

Let $s_1[n]$ be the desired speech signal arriving from the direction defined by $\Phi_1 = 0^\circ$. The other speech sources are considered as interfering speech sources. For a low computational complexity, we consider the short-time spectra of the ear input signals, $\mathcal{X}_R[m, k]$ and $\mathcal{X}_L[m, k]$, obtained by a windowed short-time Fourier transform (STFT). m denotes the frequency index, and k the frame number.

2. CONSIDERING BINAURAL CUES

2.1. Definition of binaural cues

Localization of sound is partly made possible by capturing the slight differences between sound signals at the right and left ear entrances. In order to understand how the auditory system estimates the direction of arrival of a sound, we first consider a single sound source [2]. The Ear input signals can be seen as filtered versions of the source signal, as shown in Figure 2(a): the filters used are referred to as head related transfer function (HRTF). But a more simple manner to model the ear input signals is to assume only a difference of path length from the source to both ears, as shown in Figure 2(b). As a result of this path length difference,

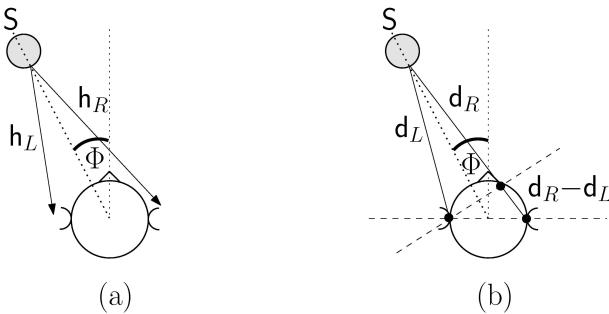


Figure 2: (a): Ear input signals modelled as filtered versions of the source signal, by the HRTFs h_R and h_L . (b): Ear input signals modelled with a difference in length of paths $d_R - d_L$ to both ears. HRTFs and the difference in length of paths are linked to the azimuth angle Φ of the source.

there is a difference in time of arrival of sound, denoted interaural time difference (ITD). Additionally, the shadowing of the head results in an intensity difference to the right and left ear input signals, denoted interaural level difference (ILD). ITD and ILD are the binaural localization cues of the considered sound source. They are directly linked to the azimuth angle Φ of this source.

2.2. Auditory scene analysis

The auditory scene analysis (source localization) is important for ultimately estimating the source signals. The directions of the sources are evaluated based on the ITDs. Next, the corresponding ILDs are computed by considering HRTF data lookup.

Source localization is mainly based on the coherence function between right and left ear input signals:

$$\Gamma_{LR}[m, k] = \frac{\Psi_{LR}[m, k]}{\sqrt{\Psi_{LL}[m, k]\Psi_{RR}[m, k]}}, \quad (2)$$

where m is the frequency index, k is the frame number, and

$$\Psi_{LR}[m, k] = E\{\mathcal{X}_L[m, k]\mathcal{X}_R^*[m, k]\}, \quad (3)$$

where $E\{\dots\}$ stands for the mathematical expectation [3]. In the time domain the coherence function $\Gamma_{LR}[m, k]$ corresponds to the normalized cross-correlation function $\gamma_{LR}[n, k]$ between $x_R[n]$ and $x_L[n]$. $\gamma_{LR}[n, k]$ is evaluated over time lags in the range of $[-1, 1]$ ms, i.e $n/f_s \in [-1, 1]$ ms, where f_s is the sampling rate. If only a single source s_i is emitting sound, the ITD is estimated as the lag of the peak of the normalized cross-correlation function:

$$ITD_i = \arg \max_n \gamma_{LR}[n, k]. \quad (4)$$

In a more complex auditory scene, where a number of sources are emitting simultaneously, we assume that the auto-correlation functions $\gamma_{s_i}[n, k]$ of the source signals $s_i[n]$ do not overlap. And thus the resulting cross-correlation function $\gamma_{LR}[n, k]$ is the sum of auto-correlation functions $\gamma_{s_i}[n, k]$, shifted in time by the corresponding ITD_i . Figure 3 illustrates the peaks detection of the normalized cross-correlation function. A peak corresponds to a source emitting from a direction leading to a time lag, corresponding to the ITD, in the normalized cross-correlation $\gamma_{LR}[n, k]$.

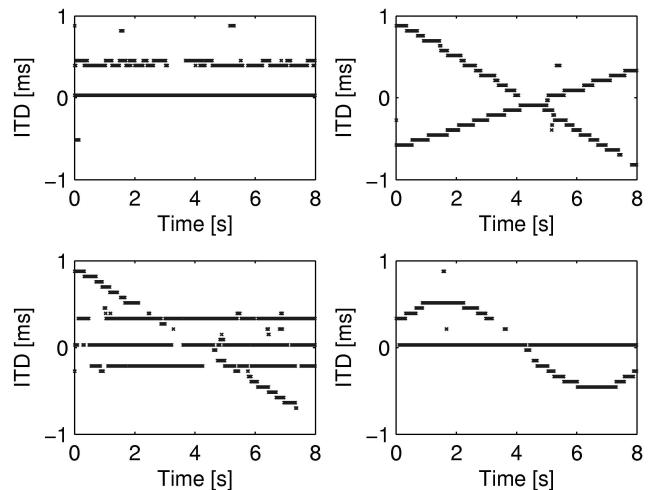


Figure 3: Different auditory scenes analyzed by ITD estimation. Two static sources are emitting simultaneously with two different ITDs (right top corner). Two sources are moving linearly over time (left top corner). Three static sources with an additional source moving linearly (left bottom corner). A static source with a source moving by following a cosine law (right bottom corner).

So far, we have only considered ITD. However, in order to analyze precisely the auditory scene, ILD needs to be taken into account. For each sound source s_i , the missing cue (ILD_i) is evaluated from a head related transfer function (HRTF) data lookup.

ITD and ILD can be described as functions of azimuth angle and frequency: $g_T(m, \Phi)$ and $g_L(m, \Phi)$, respectively. While ITD can be approximatively considered as independent of frequency, ILD is highly frequency dependent and is relevant for spatial perception for frequencies above about 1.5 kHz. However, we only estimate a single full band ILD in order to complete the scene analysis. In this case ITD and ILD are independent of frequency

by considering a weighted sum of the two dimensional functions, $g_T(m, \Phi)$ and $g_L(m, \Phi)$, among frequencies,

$$\begin{aligned} \text{ITD} &= g_T(\Phi) = \sum_m c_T^m g_T(m, \Phi) \\ \text{ILD} &= g_L(\Phi) = \sum_m c_L^m g_L(m, \Phi), \end{aligned} \quad (5)$$

where c_T^m and c_L^m are the frequency dependent scale factors for ITD and ILD, determined by the HRTF CIPIC Database [4].

Now, the azimuth angle Φ_i of each source can be calculated from the ITD_i by using the inverse function $g_T^{-1}(\text{ITD}_i)$. And next from this azimuth angle we can estimate the corresponding ILD_i , as shown in Figure 4.

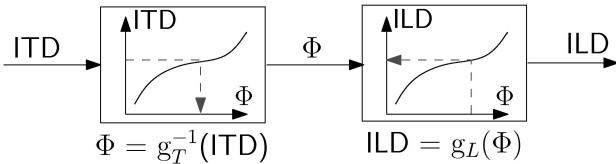


Figure 4: Evaluation of azimuth angle from estimated ITD by an inverse function, followed by the estimation of ILD by the function which directly links azimuth angles to ILD.

The auditory scene analysis yields a single pair of full band binaural cues (ITD_i and ILD_i) for each speech source s_i . These binaural cues are directly linked to the direction of arrival of source s_i .

2.3. Using binaural cues for speech enhancement techniques

In the proposed algorithm, the main application of previously binaural cues estimation is to get source dependent parameters in order to control the used speech enhancement techniques: blind source separation (BSS) and noise-adaptive spectral magnitude expansion (NASME), presented in Sections 3 and 4, respectively.

BBS needs to solve the mixing model in equation (1). The estimated binaural cues are directly linked to the attenuation coefficients a_i and the time delays d_i , defined in equation (1). Indeed, for a source s_i , a positive azimuth angle Φ_i corresponds to a point source situated at the right side with respect to the head of the listener. Also the sound of a source localized on the right side of the head will arrive first at the right ear ($d_i > 0$ ms) and the signal level will be stronger at the right ear ($a_i < 0$ dB). Because of $g_T(\Phi)$ and $g_L(\Phi)$ being monotonically increasing, a positive Φ_i yields positive ITD ($\text{ITD}_i > 0$ ms) and ILD ($\text{ILD}_i > 0$ dB). More generally we can write for the source i ,

$$\begin{aligned} d_i &= \text{ITD}_i & [\text{samples}] \\ 20 \log_{10}(a_i) &= -\text{ILD}_i & [\text{dB}]. \end{aligned} \quad (6)$$

Moreover, NASME requires signal statistics in order to be carried out. The variances of the sources signals are estimated from the power spectra of the input signals. since speech signals are assumed stationary over short time periods between 10 ms and 20 ms. A short-time estimate of the frequency domain cross-correlation between $x_R[n]$ and $x_L[n]$, defined in (3), is obtained by:

$$\Psi_{LR}[m, k] = \alpha \mathcal{X}_L[m, k] \mathcal{X}_R^*[m, k] + (1 - \alpha) \Psi_{LR}[m, k - 1], \quad (7)$$

where the factor α determines the degree of smoothing over time. With an inverse fourier transform, the smoothed time domain cross-correlation $\psi_{LR}[n, k]$ is obtained. As before, under the assumption that the resulting cross-correlation function $\psi_{LR}[n, k]$ is the sum of auto-correlation functions $\psi_{s_i}[n, k]$ of the source signals $s_i[n]$, shifted in time by the corresponding ITD_i , the variance $\sigma_{s_i}^2[k]$ is:

$$\sigma_{s_i}^2[k] = \psi_{LR}[\text{ITD}_i, k]. \quad (8)$$

The variances $\sigma_{x_R}^2[k]$ and $\sigma_{x_L}^2[k]$, of ear input signals $x_R[n]$ and $x_L[n]$, are computed in the same manner by considering the auto-correlation functions $\Psi_{RR}[m, k]$ and $\Psi_{LL}[m, k]$.

As a conclusion, the mixing model defined in equation (1) has been solved. Additionally, the short-time variances of the signals $s_i[n]$, $x_R[n]$ and $x_L[n]$ have been estimated. The resulting parameters, a_i , d_i , $\sigma_{s_i}^2[k]$, $\sigma_{x_R}^2[k]$ and $\sigma_{x_L}^2[k]$, are used to control speech enhancement techniques.

3. BLIND SOURCE SEPARATION

3.1. W-disjoint orthogonality

The first speech enhancement technique to be used is blind source separation (BSS) [5]. The goal of BSS is to recover the original source signals, given linear mixtures of these source signals. The considered linear mixtures are defined in equation (1). By performing a discrete windowed STFT, with a suitable window function $W[n]$, the mixing model can be expressed in the frequency domain as:

$$\begin{aligned} \mathcal{X}_R[m, k] &= \sum_{i=1}^N \mathcal{S}_i[m, k], \\ \mathcal{X}_L[m, k] &= \sum_{i=1}^N a_i \mathcal{S}_i[m, k] e^{-j \frac{2\pi m d_i}{M}}, \end{aligned} \quad (9)$$

where M is the length of the discrete fourier transform (DFT).

In BSS, it is assumed that the spectra, $\mathcal{S}_1[m, k], \dots, \mathcal{S}_N[m, k]$, of the N source signals satisfy the W -disjoint orthogonality condition. W -disjoint orthogonality corresponds to non-overlapping windowed STFT representations of the sources. This condition

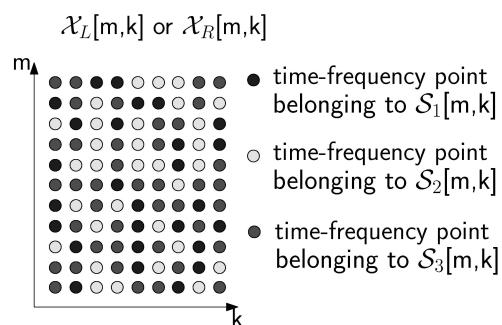


Figure 5: Time-frequency representation of the ear input signals for a scenario as is shown in Figure 1. The spectrogram with two-dimensional time-frequency grid shows the basic of W -disjoint orthogonality assumption that each point of this grid is related to only one of the three sources.

means that at most one source is active at each time-frequency point $[m, k]$. That is, each point of the time-frequency grid represents only one source, as illustrated in Figure 5.

3.2. Time-frequency masks

In order to decide on the pairing between speech sources and time-frequency points, for each source i , the maximum likelihood function is evaluated:

$$L_i[m, k] = \frac{1}{2\pi} e^{-\frac{1}{2(1+a_i^2)} |a_i e^{-j\frac{2\pi m d_i}{M}} \mathcal{X}_L[m, k] - \mathcal{X}_R[m, k]|^2}, \quad (10)$$

where the parameters a_i and d_i have been evaluated estimated by the audio scene analysis in (6). $L_1[m, k]$ is the likelihood that the source s_1 is dominant at time-frequency point $[m, k]$. The points $[m, k]$ of the time-frequency grid, which represent the source s_1 , satisfy:

$$\forall i \neq 1, L_i[m, k] < L_1[m, k]. \quad (11)$$

Then the binary time-frequency mask, used for extracting the contributions of the source s_1 from the ear input spectra, is computed as follows:

$$M_1[m, k] = \begin{cases} 1 & \forall i \neq 1, L_i[m, k] < L_1[m, k] \\ 0 & \text{otherwise}. \end{cases} \quad (12)$$

From this mask, the source s_1 is recovered from the mixtures by:

$$\begin{aligned} S_{1_R}[m, k] &= M_1[m, k] \cdot \mathcal{X}_R[m, k] && \text{for the right ear,} \\ S_{1_L}[m, k] &= M_1[m, k] \cdot \mathcal{X}_L[m, k] && \text{for the left ear;} \end{aligned} \quad (13)$$

and by considering both ears, the spatial distribution of the sources is preserved.

4. NOISE-ADAPTIVE SPECTRAL MAGNITUDE EXPANSION

4.1. Gain filter

The second speech enhancement technique implemented is noise-adaptive spectral magnitude expansion (NASME) [6]. This technique combines both companders and conventional noise reduction techniques such as parametric spectral subtraction. The main idea is to adapt the spectral magnitude expansion as a function of noise level and spectral components. NASME focuses on the suppression of uncorrelated additive background noise,

$$x[n] = s_1[n] + v[n], \quad (14)$$

where $v[n]$ is the noise measured at the ear entrance. Note that $x[n]$ can represent either the right or the left ear input signal, since NASME is performed separately on each channel and thus the parameters a_i and d_i are not considered.

By analogy with parametric spectral subtraction, the estimated desired speech signal magnitude spectrum can be computed with the gain filter H by:

$$\hat{\mathcal{S}}_1[m, k] = H[m, k] \cdot |\mathcal{X}[m, k]| e^{j \arg \mathcal{X}[m, k]} = H[m, k] \cdot \mathcal{X}[m, k]. \quad (15)$$

The phase remains unchanged by this filtering, which has no consequence since the human perception is relatively insensitive to phase corruption.

In NASME, the gain filter, H , is given by:

$$H \left(\frac{|\hat{\mathcal{V}}[m, k]|}{|\mathcal{X}[m, k]|} \right) = \left[A[m, k] \frac{|\hat{\mathcal{V}}[m, k]|}{|\mathcal{X}[m, k]|} \right]^{1-\theta[m, k]}. \quad (16)$$

Moreover $|H|$ is upper-bounded by 1. $A[m, k]$ is the crossover point, used to adapt the gain filter to the estimated noise magnitude spectrum $|\hat{\mathcal{V}}[m, k]|$ and $\theta[m, k]$ controls the expansion as a function of the inverse signal to noise ratio (SNR).

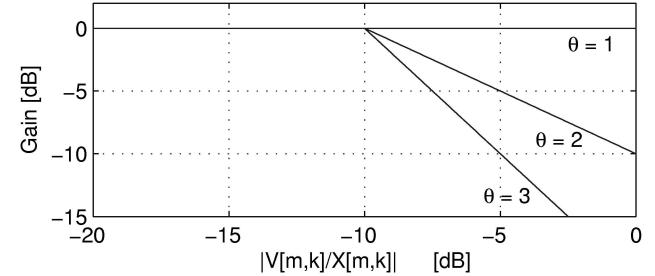


Figure 6: Gain filters $H[m, k]$ for several parameters θ and a constant crossover point $A = 10$ dB are plotted as functions of the inverse signal to noise ratio.

The gain curves, as a function of the inverse SNR, for several expansion powers θ and a constant crossover point $A = 10$ dB, are shown in Figure 6.

4.2. Extension to speech signals

In the proposed cocktail-party processor, NASME is used to enhance a desired speech signal out of the spatial distribution of concurrent speech signals. In this case, the noise signal $v[n]$ is composed of speech signals which are not completely uncorrelated with the desired speech signal and are not stationary:

$$v[n] = \sum_{i=2}^N s_i[n]. \quad (17)$$

But such signals are considered as statistically reasonably independent if they are observed over a sufficient long period of time. And over a sufficient short period of time they can be considered as stationary. By choosing a suitable analysis frame size, we assume that speech signals satisfy statistical independence and stationarity. By doing some approximations, we adapt NASME to mixtures of speech signals, as explained next.

The first approximation is related to the estimated noise spectrum. Indeed the noise spectrum is a combination of the spectra $S_2[m, k], \dots, S_N[m, k]$. And since each of the $N - 1$ noise sources can not a priori be separated, the noise magnitude spectrum can not be directly estimated. But by assuming that $s_1[n]$ and $v[n]$ are uncorrelated, the instantaneous power spectrum of the noise $v[n]$ can be recovered by subtracting an estimate of $|\mathcal{S}_1[m, k]|$ from the estimate $|\hat{\mathcal{X}}[m, k]|$:

$$|\hat{\mathcal{V}}[m, k]|^2 = |\hat{\mathcal{X}}[m, k]|^2 - |\hat{\mathcal{S}}_1[m, k]|^2. \quad (18)$$

The corresponding noise spectral magnitude is,

$$|\hat{\mathcal{V}}[m, k]| = \sqrt{|\hat{\mathcal{V}}[m, k]|^2} = \left[|\hat{\mathcal{X}}[m, k]|^2 - |\hat{\mathcal{S}}_1[m, k]|^2 \right]^{\frac{1}{2}}. \quad (19)$$

A more general form can be derived by introducing the parameters α , β and γ ,

$$|\hat{\mathcal{V}}[m, k]| = \left[|\hat{\mathcal{X}}[m, k]|^\alpha - \gamma |\hat{\mathcal{S}}_1[m, k]|^\alpha \right]^\beta. \quad (20)$$

where α and β are exponents and γ controls the estimation of $|\mathcal{S}_1[m, k]|$ in case it is under or over estimated. The estimated noise magnitude spectrum is calculated only from the spectra of the ear input signals and the desired signal. This method has the advantage that the computation time is reduced, and if the number N of sources becomes large, the computation time stays the same.

As a second approximation, the variances of signals are estimated rather than their entire power spectrum. The magnitude spectrum $|\hat{\mathcal{S}}_1[m, k]|$ is estimated according to:

$$|\hat{\mathcal{S}}_1[m, k]| = \sqrt{\sigma_{s_1}^2[m, k]} \approx \sqrt{|\mathcal{S}_1[m, k]|^2}. \quad (21)$$

The variance of the desired speech signal, $\sigma_{s_1}^2[m, k]$, as well as the variances, $\sigma_{x_R}^2[m, k]$ and $\sigma_{x_L}^2[m, k]$, of the ear input signals have been estimated from the auditory scene analysis.

Then, the noise magnitude spectrum is directly computed from equations (20) and (21):

$$|\hat{\mathcal{V}}[m, k]| = \left[\sqrt{\sigma_x^2[m, k]}^\alpha - \gamma \sqrt{\sigma_{s_1}^2[m, k]}^\alpha \right]^\beta. \quad (22)$$

Finally the gain filter H defined in equation (16) becomes:

$$H_1[m, k] = A[m, k] \left[\frac{|\sqrt{\sigma_x^2[m, k]}^\alpha - \gamma \sqrt{\sigma_{s_1}^2[m, k]}^\alpha|}{\sqrt{\sigma_x^2[m, k]}^\alpha} \right]^\beta \quad , \quad (23)$$

where $A[m, k]$ defines the crossover point, and $\theta[m, k]$ controls the expansion power. Finally, the source s_1 is recovered from the mixtures by:

$$\begin{aligned} S_{1_R}[m, k] &= H_1[m, k].\mathcal{X}_R[m, k] && \text{for the right ear,} \\ S_{1_L}[m, k] &= H_1[m, k].\mathcal{X}_L[m, k] && \text{for the left ear.} \end{aligned} \quad (24)$$

5. THE PROPOSED COCKTAIL-PARTY PROCESSOR

The proposed cocktail-party processor combines both BSS and NASME as illustrated in the block diagram in Figure 7. The first step is concerned with time-frequency transform adapted to speech signals. Then the scene analysis is carried out by estimating the binaural cues (6) related to the directions of the sources to be recovered. The different source dependent parameters, are evaluated using these binaural cues. Next, as the function of these parameters, the speech enhancement techniques, blind source separation and noise-adaptive spectral magnitude expansion, are performed simultaneously. However, the uniform spectral resolution of the STFT is not well adapted to human perception. Therefore, BSS and NASME are carried out within critical bands, which are formed by grouping the STFT coefficients such as each group corresponds to a critical band.

The binary time-frequency mask defined in equation (12) and the gain filter defined in equation (23) are combined together in a combined gain filter $G_1[m, k]$. The combined gain filter which is used to recover speech source s_1 , is given by:

$$G_1[m, k] = M_1[m, k].H_1[m, k]. \quad (25)$$

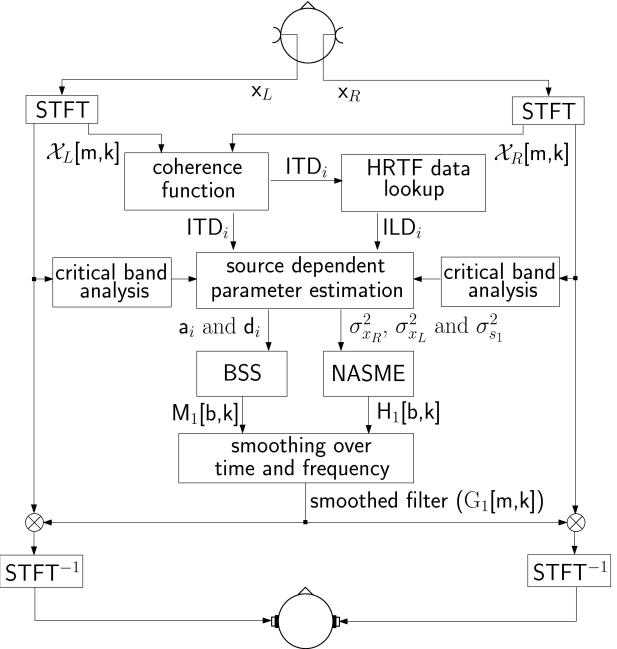


Figure 7: Detailed block diagram of the proposed algorithm for the cocktail-party processor.

In order to reduce artifacts and distortions, the last step is devoted to time and frequency smoothing of the combined gain filter applied to the ear inputs signals, which are converted back into the time domain.

6. PERFORMANCE

6.1. Directionality pattern

The ability of the proposed cocktail-party processor to suppress interfering sources arriving from non-desired directions can be expressed by means of directionality patterns. The desired direction is defined by azimuth angle $\Phi_1 = 0^\circ$. The simulations involve input signals, coming from different directions between -90° and 90° , which have been obtained by convolving white noises with HRTFs. The attenuations of the output signals have been plotted within different critical bands.

The resulting directionality patterns are shown in Figure 8: they are narrow even at low frequencies and their widths are nearly independent of frequency.

The cocktail-party processor enables a highly directive beam over a wide range of frequency with only two microphones placed at the ear entrances. With two microphones, conventional beam former are much more limited in terms of directivity.

6.2. Intelligibility

For performance evaluation, a concurrent speech signal s_2 is added to the desired speech signal s_1 at the mean SNR of 0 dB. In the processed signal, at the output of the cocktail-party processor, the concurrent signal is attenuated by 15 dB and only slight changes compared to the desired signal can be observed by visual inspection of Figure 9.

concurrent source(s)	mean STI	min STI	max STI
1	0.95	0.92	0.98
2	0.89	0.78	0.95
3	0.82	0.73	0.88
4	0.73	0.61	0.80

Table 1: The STI is evaluated under diverse acoustical conditions.

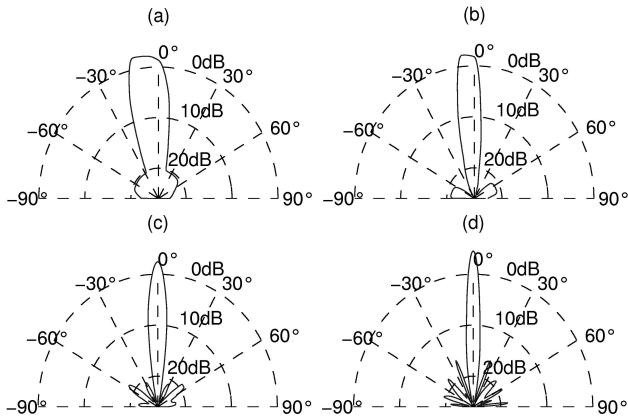


Figure 8: Directionality patterns of the cocktail-party processor within different critical bands. (a): Critical band 125 – 250 Hz. (b): Critical band 620 – 745 Hz. (c): Critical band 1425 – 1800 Hz. (d): Critical band 3535 – 4400 Hz.

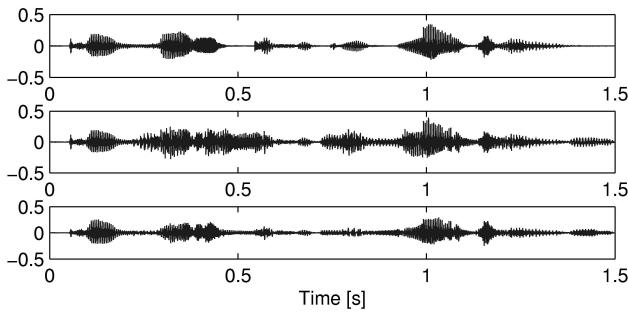


Figure 9: The desired speech signal (top) of a female speaker ($\Phi_1 = 0^\circ$). A concurrent male speech signal ($\Phi_2 = 30^\circ$) is added to the desired speech (middle). The output of the cocktail-party processor (bottom).

The intelligibility of the processed signal is evaluated by calculating the speech transmission index (STI) of the cocktail-party processor, in order to find a good trade-off between the degree of suppression of signal components and resulting distortions. The STI is a single number between 0 (unintelligible) and 1.0 (perfectly intelligible) [7].

The STI, for the proposed cocktail-party processor, is calculated for a set of different HRTFs and under diverse acoustical conditions: with several interfering sources coming from different directions of arrival. The results are presented in Table 1. For only

one concurrent signal the intelligibility remains nearly perfect, but for two concurrent signals the intelligibility starts to be deteriorated. By increasing the number of concurrent signals the intelligibility becomes worse, but remains still excellent (that means larger than 0.75 for the STI scales) with up to three interfering sources.

7. CONCLUSIONS

In this paper, we presented a cocktail-party processor controlled by binaural localization cues and signal statistics. The proposed algorithm improves source separation in existing cocktail-party processors by implementing blind source separation. The good performance of this algorithm has been demonstrated in terms of directionality and intelligibility by using the STI.

The proposed algorithm is expected to be of advantage for many applications such as automatic speech recognition, intelligent hearing aids, or speaker identification. Often, a low computational complexity is needed for real-time application. The proposed algorithm, implemented with a FFT, offers such a computational complexity.

8. ACKNOWLEDGEMENTS

Numerous individuals, at LCAV and Scopein Research, have contributed suggestions, thoughts, references, potential problems, and perspectives that have shaped this work.

9. REFERENCES

- [1] M. Bodden, "Modeling human sound source localization and the cocktail-party-effect," *Acta Acustica* 1, vol. 1, pp. 43–55, February/Apr. 1993.
- [2] J. Blauert, *Spatial hearing: The psychophysics of human sound localization*, revised ed. Cambridge, Massachusetts, USA: The MIT Press, 1997.
- [3] C. Faller, "Parametric coding of spatial audio," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, July 2004, thesis No. 3062, [Online] <http://library.epfl.ch/theses/?nr=3062>.
- [4] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, "The CIPIC HRTF database," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, Oct. 2001, pp. 99–102.
- [5] Ö. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Trans. Sig. Proc.*, vol. 52, no. 7, pp. 1830–1847, July 2004.
- [6] W. Etter and G. S. Moschytz, "Noise reduction by noise-adaptive spectral magnitude expansion," *J. Audio Eng. Soc.*, vol. 42, pp. 341–349, May 1994.
- [7] H. J. M. Steeneken and T. Houtgast, "A review of the MTF concept in room acoustics and its use for estimating speech intelligibility in auditoria," *J. Acoust. Soc. Am.*, vol. 77, no. 3, pp. 1069–1077, Mar.. 1985.

A SOURCE LOCALIZATION/SEPARATION/RESPATIALIZATION SYSTEM BASED ON UNSUPERVISED CLASSIFICATION OF INTERAURAL CUES

Joan Mouba, Sylvain Marchand

SCRIME – LaBRI, University of Bordeaux 1
 351 cours de la Libération, F-33405 Talence cedex, France
 firstname.name@labri.fr

ABSTRACT

In this paper we propose a complete computational system for Auditory Scene Analysis. This time-frequency system localizes, separates, and spatializes an arbitrary number of audio sources given only binaural signals. The localization is based on recent research frameworks, where interaural level and time differences are combined to derive a confident direction of arrival (azimuth) at each frequency bin. Here, the power-weighted histogram constructed in the azimuth space is modeled as a Gaussian Mixture Model, whose parameter structure is revealed through a weighted Expectation Maximization. Afterwards, a bank of Gaussian spatial filters is configured automatically to extract the sources with significant energy accordingly to a posterior probability. In this frequency-domain framework, we also inverse a geometrical and physical head model to derive an algorithm that simulates a source as originating from any azimuth angle.

1. INTRODUCTION

In many applications, it may be desirable to manipulate the individual sound sources that can be heard in a mix. It remains a great challenge to separate the sources in our case where we only have two sensors (human ears) and apply no restriction on the number of sources. In this case, called *degenerated*, most of the techniques for source separation, based on matrix inversion, just fail.

Another approach tries to mimic the human auditory system, based on perception and psychoacoustics: Computational Auditory Scene Analysis techniques (CASA). A recent source separation approach called *Degenerate Unmixing and Estimation Technique* (DUET) was proposed by Rickard *et al.* [1]. This technique relies on binaural cues – the interaural differences in time (ITD) and amplitude (ILD) – which play an important role in the human localization system, since they are both related to the azimuth of the source. However, DUET is mainly restricted to low frequencies, since the phase becomes ambiguous above 1500 Hz. Viste and Evangelista [2] get rid of this ambiguity by minimizing the distance between the ILD and ITD based azimuth estimates, thus obtaining an enhanced azimuth estimate for each time-frequency bin. The power of each bin is then accumulated in an histogram, where we can read the energy as a function of the azimuth.

Furthermore, in order to separate sources, Rickard *et al.* attribute exclusively the energy of a bin to one source. In contrast, Avendano proposes in [3] an adaptive spatial Gaussian mapping to achieve the source separation. However, in his approach there is no true azimuth but only an inter-channel amplitude panning coefficient for each source. Moreover, the inter-channel time / phase is not considered.

Here we introduce a Gaussian Mixture Model (GMM) of the azimuthal histogram, and use a Maximum Likelihood (ML) approach based on a modified Expectation Maximization (EM) [4]

to learn the GMM structure (mix order, weight, azimuthal location and deviation of each source) from the power-weighted histogram. A similar reformulation of EM was proposed in [5] in the case of spectral density functions.

The GMM parameters setup automatically a demixing stage (Figure 1) where bins belonging to each source are statistically selected and the energy of each bin is assigned according to a posterior probability. In contrast to others, we consider the exclusive energy assignment as too destructive.

This paper is organized as follows. In Section 2, we describe the binaural model, the localization approach, and we also detail the time-frequency algorithm proposed to map a source signal to a pair of binaural signals with the expected Direction Of Arrival (DOA). Section 3 introduces a GMM of the energy-weighted histogram and explains the parameters learning with an EM approach. In the same section, we present the probabilistic demixing algorithm. Finally, experiments and results are described in Section 4.

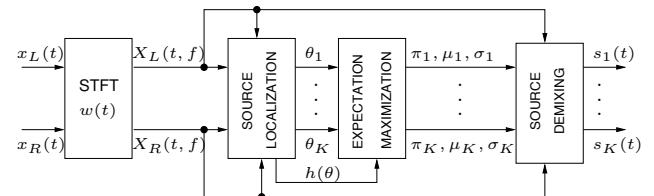


Figure 1: Overview of the proposed CASA-EM system.

2. SINGLE SOURCE

2.1. Model

A (vibrating) sound source radiates spherical acoustic waves, that propagate to the ears through an energy transmission between air particles of the surrounding environment. In this paper, we consider the sound sources as punctual and omni-directional.

In a polar coordinate system (see Figure 2), the source point is localized given its (ρ, θ, ϕ) coordinates, where ρ is the distance between the source and the head center (O), θ is the azimuth angle, and ϕ the elevation angle.

In the present work, the sources are approximately in the same horizontal plane as the ears ($\phi = 0$). This is the case in many musical situations, where both the listener and instrumentalists are standing on the (same) ground. Also, we focus on the DOA and thus neglect the distance between the source and the listener. More precisely, we are in a situation where the instrumentalists are a few meters from the listener. We consider that this distance is small enough to neglect the (frequency-selective) attenuation by the air, though large enough for the acoustic wave to be regarded as planar when reaching the ears. As a consequence, our source localization

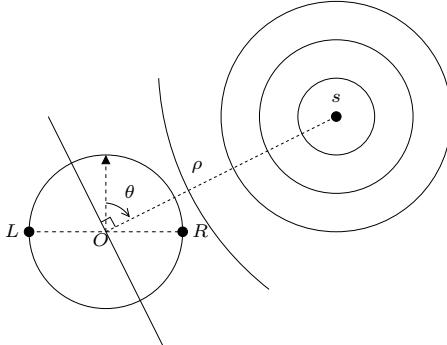


Figure 2: A source s positioned in the horizontal plane at azimuth θ , propagating acoustic waves to the head.

/ separation / respatialization system will depend on the azimuth angle θ only.

Moreover, we consider that we are in outdoors condition. We consider neither any room nor any obstacle. In this free-field case, only the head, the torso, and the outer-ear geometry modify the sound activity content by reflections and shadowing effect.

The source s will reach the left (L) and right (R) ears through different acoustic paths, characterizable with a pair of Head-Related Impulse Responses (HRIR). For a source s located at the azimuth θ , the left (x_L) and right (x_R) signals are given by:

$$x_L = s * \text{HRIR}_L(\theta) \quad (1)$$

$$x_R = s * \text{HRIR}_R(\theta) \quad (2)$$

where $*$ is the convolution among time-domain signals, the HRIR being impulse responses of filters. The HRIRs depend on the morphology of the head of the listener, are different for each ear, and are functions of the location of the source as well as its frequency.

In fact, each HRIR can be regarded as a filter, and defined in the time domain by its impulse response for each azimuth θ . The CIPIC database [6] contains this information for several listeners and different directions of arrival. In our experiments, to be listener-independent, we consider the mean HRIR (MHRIR), that is, for a given position θ , the average (in the time domain) of the HRIRs for this position among the several listeners.

In the case of s being a pure sinusoid, the convolution in preceding equations is replaced by a simple multiplication in the spectral domain. A sound source positioned to the left will reach the left ear sooner than the right one, in the same manner the left level should be higher due to head-shadowing.

More precisely, between the two ears, the amplitude ratio is approximately 10^{Δ_a} and the phase difference is approximatively Δ_ϕ , where Δ_a and Δ_ϕ are given by:

$$\Delta_a = \text{ILD}(\theta, f)/20 \quad (3)$$

$$\Delta_\phi = \text{ITD}(\theta, f) \cdot 2\pi f \quad (4)$$

where the difference in amplitude or interaural level difference (ILD, expressed in decibels – dB) and difference in arrival time or interaural time difference (ITD, expressed in seconds) are the principal spatial cues for the human auditory system localization. Since we restrict our preliminary study to the simple free-field case, the inter-channel coherence [7] will not be considered.

These binaural cues can be related to physical parameters such as the celerity of sound c and the head radius r . From the analysis of the CIPIC database, Viste [2] extends the Woodworth's formula [8] for the ITD and derives a sinusoidal model for the ILD:

$$\text{ILD}(\theta, f) = \alpha_f \sin(\theta) \quad (5)$$

$$\text{ITD}(\theta, f) = \beta_f r (\sin(\theta) + \theta) / c \quad (6)$$

where α_f and β_f are frequency-dependent scaling factors (see [2]), that encapsulate the head / ears morphology. In our experiments, we use the mean of individual scaling factors over the 45 subjects of the CIPIC database. For each subject, we measure the interaural cues from the HRIR and derive the individual scaling factors that best match the model – in the least-square sense – for all azimuths.

In the case of a complex sound, we can decompose it as a sum of sinusoids. We perform a time-frequency decomposition using the Short-Time Fourier Transform (STFT):

$$X(t, f) = \int_{-\infty}^{\infty} x(t + \tau) w(\tau) e^{-j2\pi f \tau} d\tau \quad (7)$$

where x is the temporal signal, w some windowing function, and X is the (short-term) spectrum of x .

In our practical experiments, we use sounds sampled at 44100 Hz. We then perform discrete Fourier transforms of size $N = 2048$ using the Fast Fourier Transform (FFT) algorithm. For w , we use the periodic Hann window. The hop size between two consecutive windows is $H = N/2$ (50% overlap). For each spectrum, we use the zero phase technique: In order to cancel the linear phase of the analysis window, we swap the two halves of the sample buffer prior to any FFT (and after any Inverse FFT).

Then each point of the time-frequency plane can be regarded as the contribution of a single sinusoid, and the Δ_a and Δ_ϕ coefficients can in turn be used, as functions of frequency. In fact, we should have the following relation between the spectra measured at the left and right ears:

$$X_R(t, f) = X_L(t, f) \cdot 10^{\Delta_a} e^{j\Delta_\phi} \quad (8)$$

2.2. Localization

In Auditory Scene Analysis, the ILD and the ITD are the most important cues for source localization. Lord Rayleigh (1907) mentioned in his Duplex Theory that the ILD are more prominent at high frequencies whereas the ITD are crucial at low frequencies. In fact, the human auditory system is well adapted to the natural environment. Indeed, high frequencies are more sensitive to frequency-selective amplitude attenuation (by the air or the head shadowing), but the associated signals exhibit phase ambiguities. In contrast, low frequencies are not ambiguous, but are less sensitive to amplitude attenuation.

Given the short-time spectra of the left and right channels, we can estimate the ILD and ITD for each time-frequency bin with:

$$\text{ILD}(t, f) = 20 \log_{10} \left| \frac{X_R(t, f)}{X_L(t, f)} \right| \quad (9)$$

$$\text{ITD}_p(t, f) = \frac{1}{2\pi f} \left(\angle \frac{X_R(t, f)}{X_L(t, f)} + 2\pi p \right) \quad (10)$$

The coefficient p outlooks that the phase is determined up to a modulo 2π factor. In fact, the phase becomes ambiguous beyond 1500 Hz, according to the Duplex Theory.

Obtaining an estimation of the azimuth based on the ILD information is just a matter of inverting Equation (5):

$$\theta_L(t, f) = \arcsin \left(\frac{\text{ILD}(t, f)}{\alpha_f} \right) \quad (11)$$

Similarly, using the ITD information, to obtain an estimation of the azimuth candidate for each p , we invert Equation (6) by com-

puting:

$$\theta_{T,p}(t, f) = \Pi \left(\frac{c \cdot \text{ITD}_p(t, f)}{r \cdot \beta_f} \right) \quad \text{with}$$

$$\Pi(x) = 0.50018 x + 0.009897 x^3 + 0.00093 x^5 + O(x^5) \quad (12)$$

$\Pi(x)$ is an order-5 polynomial approximation of the inverse of the $\sin(\theta) + \theta$ function. The $\theta_L(t, f)$ estimates are more dispersed, but not ambiguous at any frequency, so they are exploited to find the right modulo coefficient p that unwraps the phase. Then the $\theta_{T,p}(t, f)$ that is nearest to $\theta_L(t, f)$ is validated as the final θ estimation, since it exhibits a smaller deviation:

$$\theta(t, f) = \theta_{T,m}(t, f) \text{ with } m = \operatorname{argmin}_p |\theta_L(t, f) - \theta_{T,p}(t, f)| \quad (13)$$

In theory, in the case of a single source all frequencies should give the same azimuth, exactly corresponding to the source position θ . However, in practice, the presence of noise and estimation errors make things a little more complicated. In fact, as a first approximation, we will consider that the energy of the source is spread following a Gaussian distribution centered at the theoretical value θ . Although the Gaussian nature of the distribution is questionable and should be verified in the near future, we are comforted by the well-known Central Limit Theorem as well as preliminary experiments. In this context, the ideal case is a Gaussian of mean θ and variance 0.

2.3. Spatialization

In order to spatialize a sound source s to an expected position θ (see Figure 3), we first transform its (mono) signal into the time-frequency domain using a windowed FFT. Then the pair of left and right spectra is computed. A first approximation is to set $X_L = S$ (the spectrum of s), and to compute X_R from X_L using Equation (8). This trivially respects the interaural cues, but one of the ears plays a specific role then. Another solution is to divide the spatial cues equally, using the Equations (3), (4), (5), (6), and the following equations:

$$X_L(t, f) = X(t, f) \cdot 10^{-\Delta_\theta/2} e^{-j\Delta_\phi/2} \quad (14)$$

$$X_R(t, f) = X(t, f) \cdot 10^{+\Delta_\theta/2} e^{+j\Delta_\phi/2} \quad (15)$$

where X is the spectrum reaching both ears when the source is played from position $\theta = 0$. More precisely, $X = S \cdot \text{HRTF}(0)$, where HRTF – Head-Related Transfer Function – is the spectral equivalent of the HRIR (S , the spectrum of the source itself, being directly perceived by the left or right ear if θ is $-\pi/2$ or $\pi/2$, respectively). Nevertheless, as a first approximation, we will take $X = S$, thus the interaural cues will be respected, as well as the symmetric role of the ears.

Finally, the signals x_L and x_R are computed from their spectra X_L and X_R using the Inverse FFT, and sent to the left and right ears via headphones. In practice, we use the periodic Hann window with an overlap factor of 25%.

3. SOURCE SEPARATION

3.1. WDO Assumption

To achieve degenerated separation of a arbitrary number of sources given binaural mixtures, we consider any pair of sources $(s_k(t), s_l(t))$ as Windowed-Disjoint Orthogonal (WDO). This means that their short-time spectra do not superpose. This ideal case is given by:

$$\forall k \neq l, \quad S_k(t, f) \cdot S_l(t, f) = 0 \quad (k, l = 1, \dots, K) \quad (16)$$

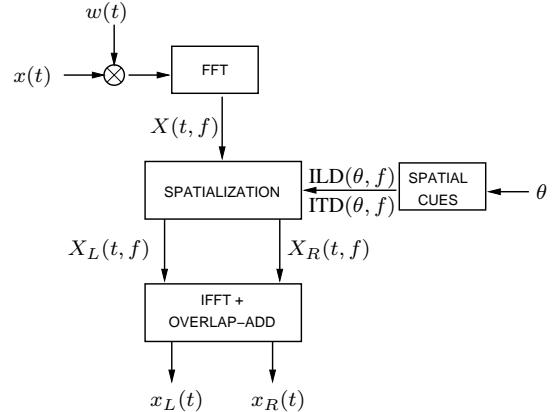


Figure 3: Spatialization of a source x to an azimuth θ .

where K is the number of sources in the mix. This condition is rarely satisfied in music signals, at least exactly. However, experiments carried out in [1] verify that speech signals are approximately WDO.

3.2. Histogram Building

We define a discrete axis for the azimuth values with resolution Δ_θ . Every computed $\theta(t, f)$ is compared to its nearest discrete neighbor multiple of Δ_θ , in order to define a binary mask:

$$M_\theta(t, f) = \begin{cases} 1 & \text{if } |\theta(t, f) - \theta| \leq \Delta_\theta/2 \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

At each time t , we then cumulate the power of the source in an histogram:

$$h(\theta) = \sum_f |M_\theta(t, f) X_L(t, f) X_R(t, f)| \quad (18)$$

Note that, from Equations (14) and (15), the energy we cumulate is $X_L X_R = X^2$, which would be roughly the energy of the source if it were played at $\theta = 0$ (monophonic case where $x_L = x_R$). This makes the histogram unbiased: Rotating the source is equivalent to a shift of the θ axis of the histogram, but then its energy remains unchanged.

3.3. Gaussian Mixture Model

Since the sources are not exactly WDO, for each source we obtain a distribution around the true value. As mentioned in Section 2, we choose to approximate the energy accumulation distribution around each point with a Gaussian distribution. In the case of K sources, we then introduce a model of K Gaussians (K -GMM, order- K Gaussian mixture model):

$$P_K(\theta|\Gamma) = \sum_{k=1}^K \pi_k \phi_k(\theta|\mu_k, \sigma_k^2) \text{ with } \pi_k \geq 0 \text{ and } \sum_{k=1}^K \pi_k = 1 \quad (19)$$

where Γ is a multiset of K triples $(\pi_k, \mu_k, \sigma_k^2)$ that denotes all the parameters of the model; π_k , μ_k , and σ_k^2 indicate respectively the weight, the mean, and the variance of the k -th Gaussian component described mathematically by:

$$\phi_k(\theta|\mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp \left(-\frac{(\theta - \mu_k)^2}{2\sigma_k^2} \right) \quad (20)$$

We are interested in estimating the architecture of the K -GMM, that is the number of sources K and the set of parameters Γ , to be able to setup the separation filtering.

3.3.1. First Estimation

In the histogram, we observe local maxima which number provides an estimation of the number of sources in the mixture. The abscissa of the k -th local maximum reveals the location θ_k of the k -th source. Generally, a finer tuning of the histogram allows a visual separability of very close-lying sources. But we will detect other local maxima around the true maxima. So, we propose to smooth the histogram using a binomial operator \mathcal{B} to fuse the encountered peaks:

$$\mathcal{B}(n) = \frac{1}{2^{D-1} \binom{D-1}{n}} \quad n = 0, \dots, D-1 \quad (21)$$

where D is the dimension of the operator. We recommend to use an odd order bigger than three. The smoothed histogram \tilde{h} is constructed by convolution of the current histogram h with the smoothing kernel \mathcal{B} .

The interferences between sources bring corrupted cues, from which the cumulated energy indicates a source at a position that does not match that of any existing source. When the sources do not coincide, usually the heights of the undesired peaks are very low. Thus, we apply a threshold, such that only the peaks above are considered as real sources. The floor of the threshold is set relatively to a noise level estimate in the histogram.

Informal experiments show that the estimated source number and location are rather good. This gives the model order K and a first estimation of the means of the Gaussians (μ_k in Γ). This estimation can be refined and completed – with the variances σ_k^2 and the weights π_k – for example by the EM algorithm.

3.3.2. Expectation Maximization

Each source in the mix is characterized with a Gaussian representation in the histogram. To discriminate the sources, the weight, mean, and variance of each source are essential for our spatial Gaussian filtering algorithm.

Expectation Maximization (EM) is a popular approach to estimate parameters in mixture densities given a data set x . The idea is to complete the observed data x with a unobserved variable y to form the complete data (x, y) , where y indicates the index of the Gaussian component from which x has been drawn. Here, the role of x is played by the azimuth θ , taking values in the set of all discrete azimuths covered by the histogram. We associate θ with its intensity function $\tilde{h}(\theta)$ (the smoothed histogram). The role of y is played by $k \in \{1, \dots, K\}$, the index of the Gaussian component θ should belong to.

The EM algorithm proceeds iteratively, at each iteration the optimal parameters that increase locally the log-likelihood of the mixture are computed. In other words, we increase the difference in log-likelihood between the current with parameters Γ and the next with parameters Γ' . This log-form difference, noted $Q(\Gamma', \Gamma)$, can be expressed as:

$$\begin{aligned} Q(\Gamma', \Gamma) &= \sum_{\theta} \tilde{h}(\theta) (\mathcal{L}(\theta|\Gamma') - \mathcal{L}(\theta|\Gamma)) \quad \text{with} \\ \mathcal{L}(\theta|\Gamma) &= \log (P_K(\theta|\Gamma)) \end{aligned} \quad (22)$$

We can then reformulate $\mathcal{L}(\theta|\Gamma)$ like this:

$$\begin{aligned} \mathcal{L}(\theta|\Gamma) &= \log \left(\sum_k P_K(\theta, k|\Gamma) \right) \quad \text{with} \\ P_K(\theta, k|\Gamma) &= \pi_k \phi_k(\theta|\mu_k, \sigma_k) \end{aligned} \quad (23)$$

The concavity of the log function allows to lower bound the $Q(\Gamma', \Gamma)$ function using the Jensen's inequality. We can then write:

$$Q(\Gamma', \Gamma) \geq \sum_{\theta} \sum_k \tilde{h}(\theta) P_K(k|\theta, \Gamma) \log \left(\frac{P_K(\theta, k|\Gamma')}{P_K(\theta, k|\Gamma)} \right) \quad (24)$$

where $P_K(k|\theta, \Gamma)$ is the posterior probability, the degree to which we trust that the data was generated by the Gaussian component k given the data; it is estimable with the Bayes rule:

$$P_K(k|\theta, \Gamma) = \frac{P_K(\theta, k|\Gamma)}{P_K(\theta|\Gamma)} \quad (25)$$

The new parameters are then estimated by maximizing the lower bound with respect to Γ :

$$\Gamma' = \operatorname{argmax}_{\gamma} \sum_{\theta} \sum_k \tilde{h}(\theta) P_K(k|\theta, \Gamma) \log (P_K(\theta, k|\gamma)) \quad (26)$$

Increasing this lower bound results automatically in an increase of the log-likelihood, and is mathematically easier. Finally, the maximization of Equation (26) provides the following update relations (to be applied in sequence, because they modify – update – the current value with side-effects, thus the updated value must be considered in the subsequent relations):

$$\pi_k \leftarrow \frac{\sum_{\theta} \tilde{h}(\theta) P_K(k|\theta, \Gamma)}{\sum_{\theta} \tilde{h}(\theta)} \quad (27)$$

$$\mu_k \leftarrow \frac{\sum_{\theta} \tilde{h}(\theta) \theta P_K(k|\theta, \Gamma)}{\sum_{\theta} \tilde{h}(\theta) P_K(k|\theta, \Gamma)} \quad (28)$$

$$\sigma_k^2 \leftarrow \frac{\sum_{\theta} \tilde{h}(\theta) (\theta - \mu_k)^2 P_K(k|\theta, \Gamma)}{\sum_{\theta} \tilde{h}(\theta) P_K(k|\theta, \Gamma)} \quad (29)$$

The performance of the EM depends of the initial parameters. The first estimation parameter should help to get around likelihood local maxima trap. Our EM procedure operates as follows:

1. Initialization step
 - initialize K with the order of the first estimation
 - initialize the weights equally, the means according to the first estimation, and the variances with the data variance (for the initial Gaussians to cover the whole set of data):
 $\pi_k = 1/K$, $\mu_k = \theta_k$, and $\sigma_k^2 = \operatorname{var}(\theta)$
 - set a convergence threshold ϵ
2. Expectation step
 - compute $P_K(k|\theta, \Gamma)$ with Equation (25)
3. Maximization step
 - compute Γ' from Γ with Equations (27), (28), and (29)
 - if $P_K(\theta|\Gamma') - P_K(\theta|\Gamma) > \epsilon$
then $\Gamma \leftarrow \Gamma'$ and go back to the Expectation step
else stop (the EM algorithm has converged).

3.4. Source Filtering Algorithm

In order to recover each source k , we select and regroup the time-frequency bins belonging to the same azimuth θ . We use the parameters issued from the EM-component number k , and the energy of the mixture channels is allocated to the (left and right) source channels according to the posterior probability. More precisely, we define the following mask for each source:

$$M_k(t, f) = P_K(k|\theta(t, f), \Gamma) \quad (30)$$

if $10 \log_{10} |\phi_k(\theta(t, f)|\mu_k, \sigma_k)| > L_{\text{dB}}$, and 0 otherwise. This mask limits the fact that the tail of a Gaussian distribution stretches out to infinity. Below the threshold L_{dB} (expressed in dB, and set to -20 in our experiments), we assume that a source of interest does not contribute anymore. For each source k , the pair of short-term spectra can be reconstructed according to:

$$S_L(t, f) = M_k(t, f) \cdot X_L(t, f) \quad (31)$$

$$S_R(t, f) = M_k(t, f) \cdot X_R(t, f) \quad (32)$$

The time-domain version of each source k is finally obtained through an inverse Fourier transform and an overlap-add procedure (as described in Section 2).

4. SIMULATION RESULTS

We have implemented, using MATLAB, two source separation / localization systems: DUET and the method proposed in Section (3), as well as two source spatialization techniques: MHRIR based on Equations (1) and (2) and the method we proposed in Section 2, called SSPA (see Figure 3).

4.1. Spatialization Results

To verify the effectiveness of the SSPA and MHRIR spatialization systems, we spatialized speech and music sources, then we conducted listening tests. Since determining the absolute location of a source could be a hard task for a listener, we preferred to conduct tests on relative locations. One (mono) source was spatialized to two different locations to create two (stereo) sounds – each sound consisting of a pair of binaural signals. During the audition, listeners had to tell if the second sound was at the left, right, or same position in comparison to the first sound. In a first round, we considered only couples of sounds produced with the SSPA system (parameterized with a sliding Hann window of length 2048 samples and a 25% overlap). The analysis of the results indicates no ambiguity between left and right. However, sources distant from less than 5° were often judged as coming from the same location. For comparison purposes, we also attended cross hearing experiments. The disposition was similar to the previous one, but here one of the source was spatialized using the MHRIR method. First observation, the MHRIR sources clang more naturally. The SSPA signals have more treble than the MHRIR signals. It is also likely that the SSPA still misses some characteristics of the perception. Another point, the MHRIR lateralized the source more accurately to the expected location. For example, a speaker voice spatialized to -80° with the MHRIR was perceived more to the left than with the SSPA procedure. However, the HRIRs are known for only some discrete position (25 azimuths in the case of the CIPIC database), and interpolating them for the other positions is not trivial. SSPA does not suffer from this limitation. Enhancing the quality of SSPA is one of our research directions.

source	theory	1^{st} est.	EM-est.
xylophone	-55°	-42°	-48°
horn	-20°	-13°	-15°
kazoo	30°	29°	25°
electric guitar	65°	64°	61°

Table 1: Locations of four sources estimated with the local maxima search, and with Expectation Maximization.

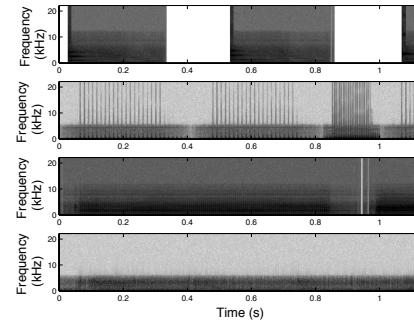


Figure 4: Spectrograms of the sources of Table 1, from top to bottom: xylophone, horn, kazoo, and electric guitar.

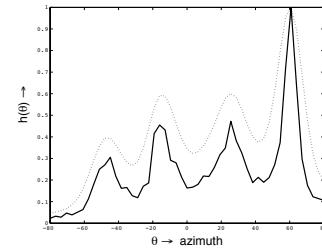


Figure 5: Localization of four sources: histogram (solid line) and smoothed histogram (dashed line).

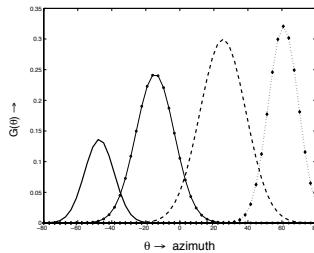


Figure 6: GMM for the histogram of the four-source mix.

4.2. Localization Results

Source localization is a decisive task for source separation based on spatial filtering. First, we created binaural signals with MHRIR, then the individually-created signals are mixed into a single binaural signal. In Figure 6, we show an example for the localization of four instruments in a binaural mixture with the proposed localization method: xylophone at -55° , horn at -20° , kazoo at 30° , and electric guitar at 65° . We note that the procedure performs well for music signals. However, we can hardly compare these results with the output of the DUET method, which relies on the interaural

ral cues but ignore the azimuth. In the smoothed histogram version of the relative noisy histogram (Figure 5), many unwelcome peaks have been dropped, while the correct sources were enhanced. In the two-dimensional histogram (ILD and ITD) obtained with the DUET method, we observed a stockpile of peaks with substantial energy. More precisely, DUET is unable to handle the phase ambiguity (appearing above 1500 Hz) for our test signals sampled at 44100 Hz, with broadband content (Figure 4).

We then proceed to the localization using a direct local maxima search in the smoothed histogram (see Table (1)). It appeared that a size of 65 was sufficient for the histogram. To build this histogram, we used FFTs of 2048 samples with an overlap of 50%. The mix order was accurately identified (4 sources). The kazoo and the electric guitar were precisely localized, only an error of 1° is observable. As a matter of fact, the kazoo has constantly high power and a wide-band spectrum, so its peak is predictably high. But because their spectrograms overlap (see Figure (4)), the sources underly interferences that corrupt the cues. However, we still get an histogram that conveys an outgoing material to segregate the sources. In a next step, we modeled the histogram as a GMM. The normalized estimated Gaussian components are depicted in Figure 6. The individual peaks are incorporated inside a Gaussian distribution. This emphasizes our motivation that a Gaussian mixture is objective for separation purposes. The location estimates of EM are exposed in Table (1), and do not really enhance the first estimation of the Gaussian means. However, the EM algorithm gives us the widths of the Gaussian distributions. Moreover, in the near future we plan to enhance these results by applying the EM algorithm on the raw azimuth estimates $\theta(t, f)$ instead of the data stored in the smoothed histogram. Indeed, the operations consisting in setting the histogram size, building this histogram, then smoothing it, are well-suited for a direct local maxima search in the histogram, but these operations may alter the data themselves.

4.3. Separation Results

We carried out subjective investigations on the time-domain signals and spectrograms. And we also judged the similarity of the sources based on hearing test.

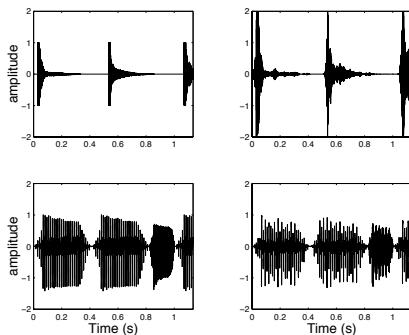


Figure 7: Waveforms of the demixtures (on the right, originals being on the left): xylophone (-55°) (top) and horn (30°) (bottom).

To separate the sources, a spatial filtering identifies and clusters bins attached to the same source. Many methods, like DUET, separate the signals by assigning each of the time-frequency bins to one of the sources exclusively. We assume that several sources can share the power of a bin, and we attribute the energy according to a membership ratio – a posterior probability. The histogram

learning with EM provides a set of parameters for each Gaussian distribution. The parameters are then used to parameterize automatically a set of spatial Gaussian filters. A result of demixing is depicted in Figure (7) for a two-instrument mixture: xylophone at -55° and horn at -20° ; their original spectrograms are shown in Figure 4. In the time domain, the xylophone rhythm is respected, its signal looks amplified and its shape is preserved. Perceptively, the demixed xylophone is very similar to the original one. Also, for the horn, we must tolerate some interference effects, and the spectrograms are partly damaged. A portion of energy was absorbed by an unwanted source generated from interferences. We also conducted tests on speech samples. The reconstruction quality was good. The quality was much better than for long-distance telephone lines. Several listening examples for the spatialization and the separation systems are available online¹.

5. CONCLUSIONS AND FUTURE WORK

The performance tests show that our system reaches promising results for speech and music signals. In our future works we will extend the localization to moving sources, but we await undesirable phase effects in the re-synthesis. In the case of FFT bins shared by several sources, the source separation is an interesting challenge. We also plan to study the brightness of spectra to judge the closeness of a source. Currently, the overall system is being implemented in a real-time environment for many applications and further investigations as part of the InSpect/ReSpect projects at the SCRIME/LaBRI research centers in Bordeaux.

6. REFERENCES

- [1] O. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Trans. Sig. Proc.*, vol. 52, no. 7, pp. 1830–1847, 2004.
- [2] H. Viste, "Binaural localization and separation techniques," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Switzerland, 2004.
- [3] C. Avendano, "Frequency-domain source identification and manipulation in stereo mixes for enhancement, suppression, and re-panning applications," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 2003, pp. 55–58.
- [4] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via EM algorithm," *J. Royal Stat. Soc.*, vol. 39, no. 1, pp. 1–38, 1977.
- [5] K. Hirokazu, N. Takuya, and S. Shigeki, "Separation of harmonic structures based on tied gaussian mixture model and information criterion for concurrent sounds," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)*, Montreal, Canada, 2004, pp. 297–300.
- [6] V. R. Algazi, R. O. Duda, and D. P. Thompson, "The CIPIC HRTF database," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 2001, pp. 99–102.
- [7] C. Faller and J. Merimaa, "Source localization in complex listening situations: Selection of binaural cues based on interaural coherence," *J. Acoust. Soc. Am.*, vol. 116, no. 5, pp. 3075–3089, Nov. 2004.
- [8] R. S. Woodworth, *Experimental Psychology*. New York: Holt, 1954.

¹<http://dept-info.labri.fr/~sm/DAFx06/>

ASSESSING THE QUALITY OF THE EXTRACTION AND TRACKING OF SINUSOIDAL COMPONENTS: TOWARDS AN EVALUATION METHODOLOGY

Mathieu Lagrange, Sylvain Marchand

SCRIME – LaBRI, University of Bordeaux 1
 351 cours de la Libération, F-33405 Talence cedex, France
 firstname.name@labri.fr

ABSTRACT

In this paper, we introduce two original evaluation methods in the context of sinusoidal modeling. The first one assesses the quality of the extraction of sinusoidal components from short-time signals, whereas the second one focuses on the quality of the tracking of these sinusoidal components over time.

Each proposed method intends to use a unique cost function that globally reflects the performance of the tested algorithm in a realistic framework. Clearly defined evaluation protocols are then proposed with several test cases to evaluate most of the desired properties of extractors or trackers of sinusoidal components.

This paper is a first proposal to be used as a starting point in a sinusoidal analysis / synthesis contest to be held at DAFX'07.

1. INTRODUCTION

The sinusoidal model is widely known, and its usability and versatility do not have to be proved. Many tools [1, 2, 3, 4] have been proposed to analyze and synthesize speech and musical sounds using this model for various applications from sound modifications [5, 6] to audio coding [7, 8].

The analysis / synthesis chain is generally divided into three main stages, as shown in Figure 1. Despite the numerous approaches that have been proposed in the literature to implement each of the three stages, the issue of evaluating and comparing the performance of these proposals is still an open question, at least for the first two stages. However, we think that there is a strong need to be able to assess the performance of each of these two stages in case of general audio processing uses.

Ideally, one would like general purpose evaluation methods with the following properties:

1. each part of the analysis chain is evaluated without the influence of the others;
2. the test cases should be realistic: the number of sinusoids is unknown and their parameters can be modulated.

In an effort to address these issues, we propose in this paper two original methods: the first one assesses the quality of sinusoidal components extractors, whereas the second one assesses the quality of trackers of sinusoidal components.

The sinusoidal model and the complete analysis / synthesis chain is presented in Section 2 where the output of each stage is formalized in terms of the set theory. Existing methodologies that evaluate the performance of sinusoidal components extractors are then reviewed and discussed in Section 3. We then show that the issue of quality measurement can be considered as a set comparison problem. From these remarks, an original evaluation metric is

proposed. This metric is then considered in various test cases that build a complete evaluation framework of sinusoidal components extractors. A similar approach (review of existing methodologies, new evaluation method, and associated protocol) is done in Section 4 for the trackers of these sinusoidal components.

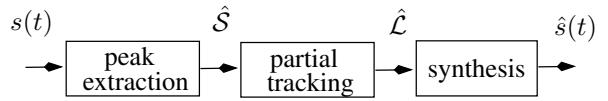


Figure 1: *The sinusoidal analysis / synthesis chain. The peak extractor extracts sinusoidal components from short-time signals. These components belong to the \hat{S} set. Components of successive frames are then considered by some tracking algorithm to estimate the evolutions of the parameters of sinusoidal components (the $\hat{\mathcal{L}}$ set) over time.*

2. SINUSOIDAL MODELING

Additive synthesis is the original spectrum modeling technique. It is rooted in Fourier's theorem, which states that any periodic function can be modeled as a sum of sinusoids at various amplitudes and harmonic frequencies. For stationary pseudo-periodic sounds, these amplitudes and frequencies continuously evolve slowly with time, controlling a set of pseudo-sinusoidal oscillators commonly called *partials*. The audio signal s can be calculated from the additive parameters using Equations 1 and 2, where N is the number of partials and the functions f_n , a_n , and ϕ_n are the instantaneous frequency, amplitude, and phase of the n -th partial, respectively. The N pairs (f_n, a_n) are the parameters of the additive model and represent points in the frequency-amplitude plane at time t . This representation is used in many analysis / synthesis programs such as AudioSculpt [1], Lemur [2], SMS [3], or InSpect [4].

$$s(t) = \sum_{n=1}^N a_n(t) \cos(\phi_n(t)) \quad (1)$$

$$\phi_n(t) = \phi_n(0) + 2\pi \int_0^t f_n(u) du \quad (2)$$

As presented in the introduction, the sinusoidal analysis / synthesis chain is generally divided into three main stages, as shown in Figure 1.

2.1. The Extraction of Peaks

The first stage, called “peak extraction”, is intended to determine the number of sinusoids from a short-time signal and to estimate the parameters of each sinusoid within the interval of observation. The term *peak* is due to the fact that the power spectrum of a sinusoid mainly consists of a prominent peak (local maximum).

At each frame, some sinusoidal components are detected and their parameters are estimated. In this paper, a peak p_i^k (peak number i of frame number k) is defined by four parameters: the frequency f_i^k (considered as constant during the observation interval), amplitude a_i^k , phase ϕ_i^k , and confidence c_i^k :

$$p_i^k = (f_i^k, a_i^k, \phi_i^k, c_i^k) \quad (3)$$

where f_i^k , a_i^k , and c_i^k are normalized parameters between 0 and 1. The c_i^k parameter denotes the degree of confidence given by the extraction algorithm. A small c_i^k indicates that the peak p_i^k should not be trusted (e.g. may belong to some noisy component).

The peaks of the same frame may be clustered in a frame set and the resulting sets may in turn be grouped to build the (multi) set of peaks \mathcal{S} :

$$\mathcal{S} = \bigcup_k \mathcal{S}_k \quad \text{where} \quad \mathcal{S}_k = \bigcup_i \{p_i^k\} \quad (4)$$

A review of the methods that permit such extraction from audio signals is out of the scope of this paper, see [9, 10] for reviews and references about methods based on the Fourier spectrum and see for example [11] concerning high-resolution estimation methods. In this paper, we aim at finding a way of *evaluating* such methods.

2.2. The Tracking of Partials

The second stage is known as “partial tracking”. This part identifies continuities between peaks of consecutive frames and therefore determines the continuous evolutions of sinusoids over time. The term *partial* is due to the fact that many partials are usually needed at the same time to model one musical or voice tone.

Formally, a partial is a vector of peaks (of confidence 1) with successive time indices:

$$P_n(m) = \{F_k(m), A_k(m), \Phi_k(m)\}, \forall m \in [b_n, \dots, b_n + l_n - 1] \quad (5)$$

where P_n is the partial number n , of length l_n , and that appeared (was born) at frame index b_n ; $P_n(m)$ is the peak of time index m of this partial. Following these notations, a peak that does not belong to any partial is noted p and noted P otherwise. For example, $p_i^m = P_n(m)$ means that the i -th peak of frame m has been assigned to the partial number n .

The set of partials that represents the entire sound is defined as:

$$\mathcal{L} = \bigcup_{n=1}^N P_n \quad (6)$$

2.3. Synthesis

The last stage synthesizes back the output audio signal using the estimated sinusoidal parameters. The continuous evolutions of the sinusoidal parameters of Equations 1 and 2 are computed from the

parameters of the partials using interpolation methods [12, 13] and efficient synthesizers are then used to obtain the output signal $\hat{s}(t)$.

The performance of different methods [14, 15] that implement this last stage can be evaluated and compared using some clearly defined protocols. To do so, the synthesized sound $\hat{s}(t)$ is compared to the original one $s(t)$ using some objective or subjective criteria.

The Reconstruction Signal-to-Noise Ratio (R-SNR) can be considered as an objective criterion:

$$\text{R-SNR} = 10 \log 10 \left(\sum_{l=0}^{L-1} (s(l) - \hat{s}(l))^2 / \sum_{l=0}^{L-1} s^2(l) \right) \quad (7)$$

where s is the original (short-time) signal and \hat{s} the synthesized one, both signals consisting of L samples. This objective criterion can be easily computed but has the disadvantage of being sometimes perceptively irrelevant. Alternatively, some expert listeners may be asked to judge the quality of \hat{s} versus s using software tools such as [16].

Approximately the same protocol can be considered to compare the performance of the other stages of the chain. If we want to compare several extractors (*resp.* trackers), a given tracker (*resp.* extractor) and a synthesis module are chosen and a switch is done between the several peak extractors (*resp.* partial trackers). However, the influence of the chosen tracker (*resp.* extractor) and synthesis modules may favor a specific extractor (*resp.* tracker) and these biases cannot be quantified easily.

Alternatively, we introduce in this paper flexible evaluation methods that do not compare time-signals and thus do not need implementations of the other stages to compare a specific stage of the chain.

3. PEAK EXTRACTION

Extractors are generally designed to detect the right number of sinusoids and to estimate the parameters of each sinusoid precisely.

3.1. Existing Evaluation Methods

Therefore, existing evaluation methods usually decorrelate the parameter estimation issue from the detection one and evaluate them independently.

3.1.1. Frequency Estimation

As far as the estimation of parameters is concerned, we generally focus on the frequency one since the others (amplitude and phase) can be found by Maximum Likelihood (ML) methods once the frequency is estimated.

Let us consider a (real) sinusoid x (of amplitude 1) in a Gaussian noise y (of variance σ^2), both (short-time) signals consisting of L samples:

$$x(l) = \sin(j\omega l + \Phi) \quad (8)$$

$$y(l) = 10^{-\text{SNR}/20} / \sqrt{2} \cdot z(l) \quad (9)$$

where ω is the frequency (in radians per sample) and z is a Gaussian noise of variance 1. The variance of the signal part x is $1/2$, and the variance of the noise part y is $\text{var}(y) = \sigma^2 = 10^{-\text{D-SNR}/10} / 2$. The analyzed signal is $s = x + y$.

For the case of the estimation of the frequency ω of a sinusoid in noise, the lower Cramér-Rao Bound (CRB) [17] is:

$$\text{var}(\hat{\omega}) \geq \frac{24 \sigma^2}{a^2 L(L^2 - 1)} = \frac{12}{L(L^2 - 1)} 10^{-\text{D-SNR}/10} \quad (10)$$

where a is the amplitude of the sinusoid (here $a = 1$), and the Degradation-SNR is given by the following equation:

$$\text{D-SNR} = 10 \log 10 \left(\sum_{l=0}^{L-1} y^2(l) / \sum_{l=0}^{L-1} x^2(l) \right) \quad (11)$$

We can easily show that, in the log scales, the CRB in function of the SNR is a line, see Figure 2.

Therefore, the variance of the error versus the SNR gives an indication of the precision of the tested estimator, *i.e.* the closer to the CRB, the better. This framework has also been used to evaluate the frequency resolution of estimators in [10], by considering two sinusoids instead of only one. However, such tests are often done using complex exponentials instead of real sinusoids.

The clear mathematical formulation of this evaluation method is convenient for mathematical derivations of theoretical performance estimation bounds. However, this unrealistic test case has the major disadvantage of being over-simplified and may therefore lead to over-specialized estimators whose applicability to musical and speech signal analysis remains to be proved.

3.1.2. Detection

Keiler *et al.* use in [9] an algorithm similar to the one proposed in this paper, where extracted peaks are compared to the original ones and the following quantities are computed: the mean number of detected peaks per frame, the mean number of peaks per frame that are in the reference signal but not detected by the algorithm, and the mean number of peaks that are detected by the algorithm but not present in the reference signal.

Hainsworth in [10] also uses two quantities: the number of correctly identified sinusoids and the number of falsely detected sinusoids. However, this “correctness” is not clearly defined.

3.2. Proposed Evaluation Methodology

As remarked by Hainsworth in [10], even if the problems of parameter estimation and component detection can be separated, they are in practice inextricably linked. We therefore propose in this section a method to evaluate globally the performance of a peak extractor without considering the other stages of the analysis / synthesis chain.

Given a set of peaks S_k of cardinal $\#S_k = N_k$, the original (short-time) sound is synthesized and possibly degraded. The tested extractor is then used to extract another set of sinusoids \hat{S}_k with cardinal M_k . The frame index k will be removed in the following for the sake of clarity. The closer this set is to the original, the better the extractor is.

The issue here is then to be able to compare these two sets. Following the remarks of Section 3.1, we consider only the frequency and confidence parameters for the evaluation.

3.2.1. Unitary Case

Let us consider a simple case, where only one peak is in the original set and the estimated one: $N = M = 1$. To compute the

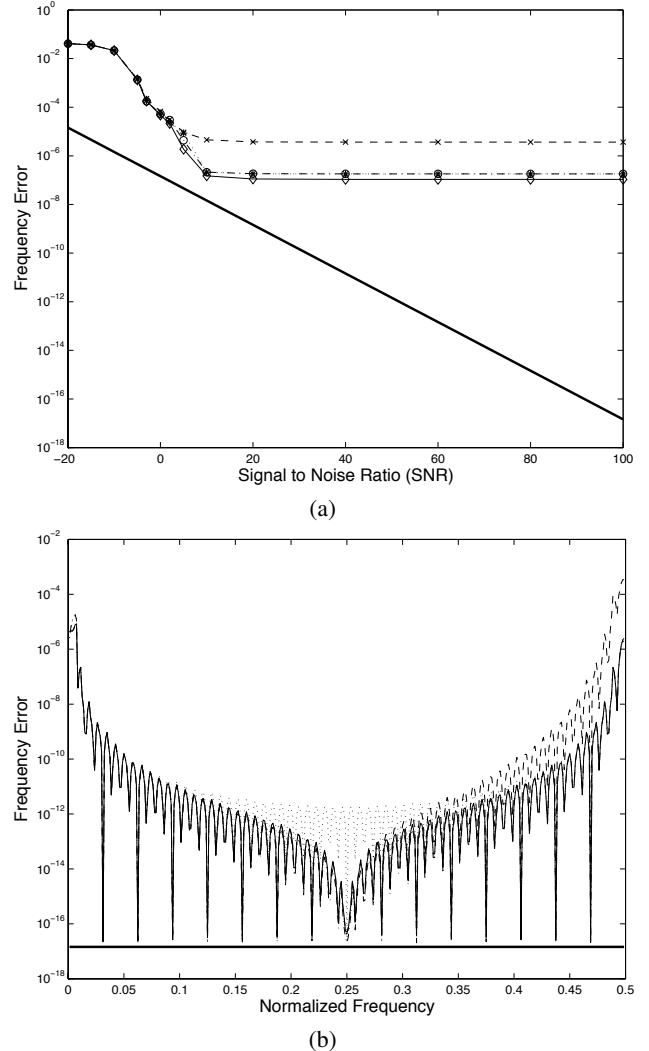


Figure 2: (a) Performance comparison of several estimators from [18] for the analysis of a real sinusoid with frequency lying in the whole (normalized) frequency range $[0, 0.5]$: the reassignment method (dotted line with $*$), the phase-vocoder estimator (dash-dotted line with o), the derivative estimator (dashed line with x), and the trigonometric estimator (solid line with \diamond). The CRB is plotted with a double solid line. (b) Performance of the tested estimators at $\text{SNR}=100$ dB versus the frequency of the analyzed sinusoid (symbols are not plotted here for the sake of clarity).

imprecision cost between a peak of the original set p_i and a peak in the estimated set \hat{p}_j , we use only the frequency parameter. Since we consider only real signals, the normalized frequencies are between 0 and 0.5. A normalization factor of 2 is then used to obtain a cost function between 0 and 1. More precisely:

$$c(p_i, \hat{p}_j) = (2(f_i - \hat{f}_j))^2 \quad (12)$$

3.2.2. General Case

Let us now consider the general case where N is set arbitrarily. Since all the sinusoids of the original set exist, they have full con-

fidence, *i.e.* $c_i = 1$ for all peaks $p_i \in \mathcal{S}$. Ideally, the estimated set $\hat{\mathcal{S}}$ should be of the same cardinality as \mathcal{S} and the frequencies of each peak of $\hat{\mathcal{S}}$ should be close to the one of a peak in \mathcal{S} . By applying Algorithm 1, we iteratively seek for correspondences between peaks of the two sets by decreasing confidence order of estimated peaks until no peak remains in one of the two sets.

Algorithm 1 considered to seek for correspondence between peaks of the two sets \mathcal{S} and $\hat{\mathcal{S}}$.

```

 $c_I \leftarrow 0$ 
 $\mathcal{S}_r \leftarrow \mathcal{S}$ 
 $\hat{\mathcal{S}}_r \leftarrow \hat{\mathcal{S}}$ 
while  $\mathcal{S}_r \neq \emptyset$  and  $\hat{\mathcal{S}}_r \neq \emptyset$  do
    take  $p_j \in \hat{\mathcal{S}}_r$  such that  $c_j = \max_{p_k \in \hat{\mathcal{S}}_r} c_k$ 
    find  $p_i \in \mathcal{S}_r$  such that  $|f_i - f_j| = \min_{k \neq j} |f_k - f_j|$ 
     $c_I \leftarrow c_I + c(p_i, p_j)$ 
     $\mathcal{S}_r \leftarrow \mathcal{S}_r - \{p_i\}$ 
     $\hat{\mathcal{S}}_r \leftarrow \hat{\mathcal{S}}_r - \{p_j\}$ 
end while

```

Once the algorithm has been processed, the imprecision cost c_I is obtained. If the remaining set $\hat{\mathcal{S}}_r \neq \emptyset$, then some peaks have been over estimated. On contrary, if the remaining set $\mathcal{S}_r \neq \emptyset$, then some peaks have been under estimated. The over and under estimation costs are defined then by:

$$c_O = \sum_{j=1}^{\#\hat{\mathcal{S}}_r} c_j \quad \text{and} \quad c_U = \sum_{i=1}^{\#\mathcal{S}_r} c_i = \#\mathcal{S}_r \quad (13)$$

We propose that these three costs can be summed and normalized to obtain an overall cost that reflects both the issue of finding the correct number of peaks and of estimating their parameters precisely:

$$C = (c_O + c_U + c_I)/N \quad (14)$$

By using the cost function C , we consider that missing a sinusoid is equivalent to completely misestimating its frequency. In the same time, over estimating the number of sinusoids is a handicap only if the extractor had great confidence for the over detected peaks.

3.3. Test Cases

Now, the cost C defined in Equation 14 is used to assess the performance of a given extractor in a realistic framework. Two tests cases are considered to evaluate the precision and the resolution of the tested extractor, while the third one is intended for an overall evaluation by simulating a realistic case.

Precision. In this test, we consider only one sinusoid embedded in a Gaussian noise. Several runs with randomized frequencies and phases are operated and the mean cost versus the D-SNR is plotted such as in Figure 2. A perfect extractor will lead to performances equal to the CRB.

Resolution. To assess the resolution capabilities of a given extractor, we consider a test with two sinusoids with the same amplitude. Several runs with randomized phase and frequencies are operated and the mean cost versus the absolute difference between the frequencies of the two sinusoids will be plotted.

Overall Quality Assessment. In this test, we consider several sinusoids. Several runs with randomized frequencies and phases are

operated and the mean cost versus the number of sinusoids will be plotted. For this test, the amplitudes may be set with the same value. This would be more correct, since it is much more difficult to detect a sinusoid of low amplitude than one with high amplitude and this is not reflected by the proposed cost C of Equation 14. However, tested estimators may take that into account to artificially achieve good evaluation. We then consider that the amplitude should also be randomized but within a reasonable range. For example, the threshold of hearing may be considered as a lower bound.

4. PARTIAL TRACKING

Trackers of partials are designed to identify continuities between peaks of (generally) successive frames to build a continuous representation of the evolutions of the sinusoidal parameters over time.

4.1. Existing Evaluation Methods

The first algorithms were proposed by McAulay and Quatieri in the speech processing field [12] and by Serra and Smith in the musical sound processing field [19]. They are now references and many implementations of these algorithms are available.

Several other methods have been proposed then [20, 21, 22, 23, 24, 25, 26] to achieve better identifications of the continuities between peaks either by extending these first algorithms or by considering a totally different framework.

Once proposed, most of the algorithms are validated by visual examination of the output of the trackers, *i.e.* the evolutions of the frequency or the amplitude of the partials are plotted over time. This may be used to show specific properties of a given algorithm but not to compare this algorithm to existing ones.

Alternatively, we have proposed and used a methodology of evaluation in [24, 25]. A single synthetic sinusoidal source x is corrupted by the addition of noise or another sinusoidal source y , leading to the signal s . The performance of the trusted tracker is evaluated by considering the ratio between the R-SNR (see Equation 7) versus the D-SNR (see Equation 11). The results for three trackers in two test cases are plotted in Figure 3. This methodology is useful to compare different trackers in a more systematic manner. However, since only SNRs are considered, it requires the implementation of the other stages of the chain, which could result in biased evaluations.

A first attempt to get rid of the influence of the other stages while assessing the performance of a tracker is proposed in [27]. By doing so, we have to compare the output of the tracker – set $\hat{\mathcal{L}}$ – to the original set \mathcal{L} .

In [26], Satar-Boroujeni *et al.* uses two factors defined as follows to compare the performance of their trackers with another one proposed in the literature:

$$R_d = n_d/n_e \quad \text{and} \quad R_f = n_f/n_e \quad (15)$$

where n_d is the number of detected tracks, n_f is the number of false tracks, and n_e is the number of expected tracks.

However, the direct comparison between the two sets of partials is a complicated task due to the high dimensionality of the elements of each set (a partial has a time location, and several parameters that evolve during this activity time slot), so that a comparison of the two sets by considering their cardinals appears as not sufficient. This issue will be addressed by the introduction of an algorithm to estimate how different two sets of partials are.

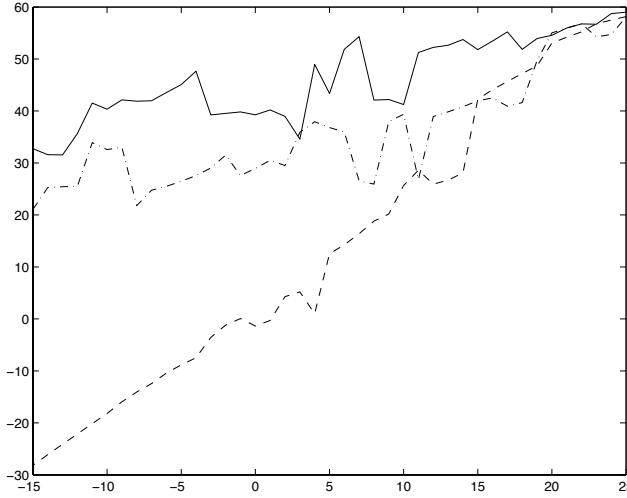


Figure 3: Evaluation of the management of crossing partials for three trackers from [24]: the High-Frequency Content (HF) algorithm (solid line), the Linear-Prediction (LP) algorithm (dashed line), and the McAulay-Quatieri (MAQ) one (dash-dotted line). The R-SNR is plotted in function of the D-SNR.

4.2. Evaluation Methodology

A set of partials \mathcal{L} is considered as a reference. This set of partials \mathcal{L} of cardinal $N = \#\mathcal{L}$ is converted into a set of peaks \mathcal{S} . After being degraded by several types of degradation studied in 4.3, this set is used as the input of the tested tracker to give another set of partials $\hat{\mathcal{L}}$ of cardinal $M = \#\hat{\mathcal{L}}$.

The issue is then to evaluate the quality of the tested tracker by comparing the two sets $\hat{\mathcal{L}}$ and \mathcal{L} .

4.2.1. Unitary Case

Let us first consider a simple case, where only one partial is in the original set and the estimated one: $N = M = 1$.

To evaluate how different two partials P_i and P_j are, we consider only the frequencies and amplitudes parameters. These parameters are supposed to be null if the partial is not active at a given frame:

$$A_i(k) = F_i(k) = 0 \text{ if } k < b_i \text{ or } k \geq b_i + l_i \quad (16)$$

(b_i being the birth index of partial i , and l_i its length).

Let us consider P_i as the reference partial. Considering that $b_i \leq b_j$, we define four frame indices as follows:

$$\begin{aligned} n_1 &= b_i \\ n_2 &= \min(b_j, b_i + l_i) \\ n_3 &= \min(\max(b_j, b_i + l_i), b_j + l_j) \\ n_4 &= \max(b_i + l_i, b_j + l_j) \end{aligned}$$

We assume that a missing part of the original partial has a cost equivalent to the cumulated amplitudes of the original partial. We also consider that artificially prolongating a partial in the resulting representation has a cost equivalent to the cumulative amplitude of the prolonged part.

The common part between the two partials (between frames n_2 and n_3) is then considered. We assume that the imprecision cost between a peak of the original partial p_i^k and a peak of the estimated partial \hat{p}_j^k is the product of the differences between their frequencies and amplitudes.

The cost function between the two partials P_i and P_j is then:

$$\begin{aligned} c(P_i, P_j) &= \sum_{k=n_2}^{n_3} 2 |F_i(k) - F_j(k)| \cdot |A_i(k) - A_j(k)| \\ &+ \sum_{k=n_1}^{n_2} \max(A_i(k), A_j(k)) + \sum_{k=n_3}^{n_4} \max(A_i(k), A_j(k)) \quad (17) \end{aligned}$$

Since we consider only real signals, the normalized frequencies are between 0 and 0.5. A normalization factor of 2 is then used to obtain a cost function between 0 and 1 (as in Equation 12).

4.2.2. General Case

Let us now consider the general case where N is set arbitrarily. Ideally, the estimated set $\hat{\mathcal{L}}$ should be of the same cardinality ($M = N$) and each partial of the estimated set should be close to one partial of \mathcal{L} . By applying Algorithm 2, we iteratively seek for correspondences between partials of the two sets until no partial remains in one of the two sets. This search is done by decreasing overall amplitude order of the estimated partials, the overall amplitude of a partial P_k being defined as:

$$A_k = \sum_{i=b_k}^{b_k+l_k-1} A_k(i) \quad (18)$$

Algorithm 2 considered to seek for correspondences between partials of two sets \mathcal{L} and $\hat{\mathcal{L}}$.

```

 $c_I \leftarrow 0$ 
 $\mathcal{L}_r \leftarrow \mathcal{L}$ 
 $\hat{\mathcal{L}}_r \leftarrow \hat{\mathcal{L}}$ 
while  $\mathcal{L}_r \neq \emptyset$  and  $\hat{\mathcal{L}}_r \neq \emptyset$  do
    take  $P_j \in \hat{\mathcal{L}}_r$  such that  $A_j = \max_{P_k \in \hat{\mathcal{L}}_r} A_k$ 
    find  $P_i \in \mathcal{L}_r$  such that  $c(P_i, P_j) = \min_{P_k \in \mathcal{L}_r} c(P_k, P_j)$ 
     $c_I \leftarrow c_I + c(P_i, P_j)$ 
     $\mathcal{L}_r \leftarrow \mathcal{L}_r - \{P_i\}$ 
     $\hat{\mathcal{L}}_r \leftarrow \hat{\mathcal{L}}_r - \{P_j\}$ 
end while

```

Once the algorithm has been processed, the imprecision cost c_I is obtained. If the remaining set $\hat{\mathcal{L}}_r \neq \emptyset$, then some partials have been over estimated. On the contrary, if the remaining set $\mathcal{L}_r \neq \emptyset$, then some partials have been under estimated. The over and under estimation costs are then defined by:

$$c_O = \sum_{j=1}^{\#\hat{\mathcal{L}}_r} A_j \quad \text{and} \quad c_U = \sum_{i=1}^{\#\mathcal{L}_r} A_i \quad (19)$$

We assume that these three costs can be summed and normalized by the number of partials of the original set \mathcal{L} to obtain an overall cost that reflects the quality of the tracking:

$$C = (c_O + c_U + c_I)/N \quad (20)$$

4.3. Test Cases

The cost C defined in Equation 20 can be used to assess the performance of a given tracker. As described in the beginning of Section 4.2, a given set of partials \mathcal{L} – considered as a reference – is converted back to a set of peaks \mathcal{S} . This set is possibly degraded to simulate degradations that trackers have to face when tracking partials within audio sounds.

4.3.1. Simulating Degradations

When considering a monophonic source embedded in noise or a polyphonic sound with many sources, the \mathcal{S} set extracted by the first stage of the analysis / synthesis chain is usually degraded. In case of noise addition, the number of peaks is over estimated. In case of partials with close or crossing frequencies, some peaks are missing. In both cases, the parameters of the peaks are also not estimated precisely.

Therefore the three types of degradations proposed below – if combined – will produce a degradation close to the one obtained by the addition of a noise signal to the input signal $s(t)$ before processing a peak extractor.

We assume that these degradations may be considered independently in three different test cases.

Adding Peaks. In this test case, some peaks are added to \mathcal{S} before the tracking. The frequencies and the phases of these peaks are randomly chosen. The amplitude of each peak is also chosen randomly, and the addition of peaks is stopped when the sum of their amplitudes has reached a given amount. The evaluation is done by considering the overall cost C versus the ratio between the summed amplitudes of the added peaks and the summed amplitudes of the peaks of \mathcal{S} .

Removing Peaks. In this test case, we randomly remove a given number of peaks from \mathcal{S} . The evaluation is done by considering the overall cost C versus the ratio between the number of removed peaks and the number of peaks in \mathcal{S} .

Degrading the Parameters of Partial. In this test case, the amplitude and frequency parameters of the peaks are modified by the addition of randomly chosen values. For the frequency, this value is chosen between $-\Delta_f$ and Δ_f . The evaluation is then done by considering the overall cost C versus the Δ_f threshold.

4.3.2. Polyphonic Case

In this last test case, we simulate a polyphonic context by adding several monophonic sound sources together within a unique set of partials before converting it to the peak representation \mathcal{S} .

However, this test is obviously over-simplified since the resulting representation is of far better quality than if the peak set was obtained by analyzing a polyphonic sound with a peak extractor.

5. CONCLUSION

In this paper, we have introduced two original evaluation methods that intend to ease the comparison of peak extractors and partial trackers in a realistic framework, and independently to any specific application. Clearly defined protocols are also proposed to evaluate most of the desired properties of peak extractors and partial trackers with several test cases that simulate typical problems while considering the sinusoidal model in audio processing applications. Each protocol allows to evaluate individually each stage of the sinusoidal analysis / synthesis chain.

Although the capacities of extractors to detect the sinusoids and to estimate the parameters of these sinusoids are generally evaluated independently, they are in practice inextricably linked. The proposed method for peak extractors uses a unique cost function that globally reflects the performance of the tested method.

The protocol for partial trackers considers a cost function that evaluates how different the extracted set of partials is from the original one. The definition of this cost function is a difficult problem, since the comparison of two sets of partials is not clearly defined yet. The empirical cost function proposed in this paper constitutes a first attempt to address this issue. However, a more theoretically motivated comparison method should be considered as a future research subject.

This paper is a first proposal to be used as a starting point in a sinusoidal analysis / synthesis contest to be held at DAFx'07. Participants will have the opportunity to submit extractors, trackers, as well as evaluation methodologies and protocols. The resulting survey will be automatically generated and published. This survey should be of great interest for people involved in sinusoidal modeling.

6. REFERENCES

- [1] Ircam, *AudioSculpt User's Manual*, 2nd ed., Ircam, Paris, 1996.
- [2] K. Fitz and L. Haken, "Sinusoidal modeling and manipulation using Lemur," *Computer Music J.*, vol. 20, no. 4, pp. 44–59, 1996.
- [3] X. Serra, *Musical Signal Processing*, ser. Studies on New Music Research. C. Roads, S. T. Pope, A. Piccialli, G. De Poli (eds.), Swets & Zeitlinger, Lisse, the Netherlands, 1997, ch. Musical sound modeling with sinusoids plus noise, pp. 91–122.
- [4] S. Marchand and R. Strandh, "InSpect and ReSpect: spectral modeling, analysis and real-time synthesis software tools for researchers and composers," in *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, 1999, pp. 341–344.
- [5] R. J. McAulay and T. F. Quatieri, "Shape invariant time-scale and pitch modification of speech," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 40, no. 3, pp. 497–510, 1992.
- [6] S. N. Levine, T. S. Verma, and J. O. Smith, "Multiresolution sinusoidal modeling for wideband audio with modifications," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'98)*, Seattle, USA, vol. 6, 1998, pp. 3585–3588.
- [7] H. Purnhagen, N. Meine, and B. Edler, "Sinusoidal coding using loudness-based component selection," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'02)*, Orlando, USA, vol. 2, 2002, pp. 1817–1820.
- [8] G. Garcia and J. Pampin, "Data compression of sinusoidal modeling parameters based on psychoacoustic masking," in *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, 1999, pp. 40–43.
- [9] F. Keiler and S. Marchand, "Survey on extraction of sinusoids in stationary sounds," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 51–58.
- [10] S. W. Hainsworth and M. D. Macleod, "On sinusoidal parameter estimation," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003, pp. 151–156.

- [11] R. Badeau, B. David, and G. Richard, "High resolution spectral analysis of mixtures of complex exponentials modulated by polynomials," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 54, no. 4, pp. 1341–1350, 2006.
- [12] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [13] L. Girin, S. Marchand, J. di Martino, A. Röbel, and G. Peeters, "Comparing the order of a polynomial phase model for the synthesis of quasi-harmonic audio signals," in *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 2003, pp. 193–196.
- [14] A. Freed, X. Rodet, and P. Depalle, "Performance, synthesis and control of additive synthesis on a desktop computer using FFT^{-1} ," in *Proc. Int. Comp. Music Conf. (ICMC'93)*, Tokyo, Japan, 1993, pp. 98–101.
- [15] N. Meine and H. Purnhagen, "Fast sinusoid synthesis for MPEG-4 HILN parametric audio decoding," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 105–110.
- [16] CRC, "CRC-Seaq," [Online] <http://www.crc.ca/fr/html/aas/home/products/products>, 2000.
- [17] D. C. Rife and R. R. Boorstyn, "Single-tone parameter estimation from discrete-time observations," *IEEE Trans. Information Theory*, vol. IT-20, pp. 591–598, 1974.
- [18] M. Lagrange and S. Marchand, "Improving sinusoidal frequency estimation using a trigonometrical approach," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005, pp. 59–64.
- [19] J. O. Smith and X. Serra, "PARSHL: an analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation," in *Proc. Int. Comp. Music Conf. (ICMC'87)*, Champaign-Urbana, USA, 1987, pp. 290–297.
- [20] P. Depalle, G. Garcia, and X. Rodet, "Tracking of partials for additive sound synthesis using Hidden Markov Models," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'93)*, Minneapolis, USA, 1993, pp. 225–228.
- [21] A. Sterian and G. H. Wakefield, "A model-based approach to partial tracking for musical transcription," in *Proc. SPIE Int. Symp. Optical Science, Eng., and Instrumentation, San Diego, California*, San Diego, 1998.
- [22] H. Purnhagen, "Parameter estimation and tracking for time-varying sinusoids," in *Proc. 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA)*, Leuven, Belgium, 2002.
- [23] M. Lagrange, S. Marchand, and J.-B. Rault, "Partial tracking based on future trajectories exploration," in *116th Conv. Audio Eng. Soc.*, Berlin, Germany, 2004, preprint 6046 (10 pages).
- [24] ———, "Using linear prediction to enhance the tracking of partials," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)*, Montreal, Canada, vol. 4, 2004, pp. iv/241 – iv/244.
- [25] ———, "Tracking partials for the sinusoidal modeling of polyphonic sounds," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'05)*, Philadelphia, USA, vol. 3, 2005, pp. iii/229 – iii/232.
- [26] H. Satar-Boroujeni and B. Shafai, "A robust algorithm for partial tracking of music signals," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005, pp. 202–207.
- [27] M. Lagrange, "Sinusoidal modeling of polyphonic sounds," Ph.D. dissertation, University of Bordeaux 1, France, 2004, in French.

SINUSOIDAL EXTRACTION USING AN EFFICIENT IMPLEMENTATION OF A MULTI-RESOLUTION FFT

Karin Dressler

Fraunhofer Institute for Digital Media Technology
Langewiesener Str. 22, 98693 Ilmenau, Germany
dresslkn@idmt.fraunhofer.de

ABSTRACT

This paper provides a detailed description of the spectral analysis front-end of a melody extraction algorithm. Our particular approach aims at extracting the sinusoidal components from the audio signal. It includes a novel technique for the efficient computation of STFT spectra in different time-frequency resolutions. Furthermore, we exploit the application of local sinusoidality criteria, in order to detect stable sinusoids in individual FFT frames. The evaluation results show that a multi resolution analysis improves the sinusoidal extraction in polyphonic audio.

1. INTRODUCTION

The presented approaches to sinusoidal identification are part of a melody extraction algorithm, which aims at the transcription of the predominant voice out of polyphonic real-world music. The most relevant melody information can be found in the sinusoidal components of the audio signal, thus it is very desirable to divide the signal into a deterministic signal component plus noise [1, 2]. Only partials which originate from periodic sound are processed further. This way we particularly reduce the computational cost of our pitch estimation method, which strongly depends on the number of peaks to be analysed.

Often, sinusoidal extraction methods, which are designed to work with monophonic audio or artificial test signals, fail to produce satisfactory results with polyphonic audio signals. The presented approaches are adapted to the specific demands of this challenging task and – consequently – will also be validated against the melody extraction results.

Another problem we address, is the choice of the proper time-frequency resolution of the spectral analysis method. If the spectral representation exhibits constant frequency resolution, almost no change in frequency is observed for the low fundamental of an frequency-modulated tone, but vivid dynamics are noted for its higher harmonics. To cover fast signal changes, we have to increase the analysis bandwidth, but at the same time we have to maintain an adequate discrimination of concurrent sounds. A solution to this conflict lies in analysis methods which provide a more or less logarithmic frequency scale.

Unfortunately such techniques are often computationally expensive, so the Fast Fourier Transform remains the tool of choice in time-critical applications. In need of good frequency resolution, long FFT windows have to be applied – accepting a distorted spectrum for faster changing signal components. In practice, local sinusoidality criteria for the detection of sinusoids often fail in high frequency bands, because they depend on the shape of the spectral window function. The same is true for the estimation of the instantaneous frequencies and magnitudes. Masri presented a

method for the identification of non-stationary sinusoids for well defined types of spectral distortion, but in real-world signals the frequency modulation will not follow such idealised trajectories [3].

A good compromise is a multi-resolution analysis based on the FFT algorithm. A prominent example is the application of a multirate filter bank in combination with the FFT used by Goto [4]. Another straight forward idea is the calculation of the FFT with different window lengths, resulting in different time-frequency resolutions. Essentially the multi-resolution FFT (MR FFT) is an efficient implementation of this idea.

2. MULTI-RESOLUTION FFT

Given a sequence of data samples $x[n]$ the Short Time Fourier Transform is defined as $X_l[k]$:

$$X_l[k] = \sum_{n=0}^{M-1} x[n + lL] \cdot w_N^{-kn}, \quad (1)$$

$l = 0, 1, \dots$ and $k = 0, 1, \dots, N - 1$

where

N	is the number of STFT points
L	is the time advance of the data frame (hop-size)
M	is the size of the data frame
l	is the number of the data frame
k	is the frequency bin number
w_N	is the N^{th} primitive complex root of unity

The values of N , L and M are the control parameters of the STFT, which determine certain characteristics of the spectrogram representation: the spacing of the discrete time-frequency grid of the spectrogram depends on the sampling rate f_s , the number of STFT points N and on the time advance of the data window L , which is also called hop-size. The grid spacing is determined by $\Delta f_{\text{grid}} = \frac{f_s}{N}$ and $\Delta t_{\text{grid}} = \frac{L}{f_s}$.

The grid spacing is not necessarily the time and frequency resolution we obtain from the spectrogram. The frequency resolution (the ability to distinguish two closely spaced frequencies from the original input signal) and also the time resolution is determined by the sampling rate, the size of the data window M and also by the shape of the window function, which will not be under consideration here. The resolution is given by $\Delta f = \frac{f_s}{M}$ and $\Delta t = \frac{M}{f_s}$.

If we use zero-padding, the frequency resolution is smaller than the spacing between the frequency bins, because the used data frame is smaller than the number of STFT points ($M < N$). If we use overlapping data frames ($L < M$), the time resolution does not increase, but nevertheless more STFT frames are included on the time axis. The additional information is obtained by interpolation.

The MR FFT determines the different resolutions by changing the data frame size M only. The hop-size L , the number of STFT points N and – in consequence – the spacing of the time-frequency grid remain unchanged among the different spectrograms.

The basic idea of the MR FFT is derived from the fact, that the summation operation is associative, thus we are allowed to split summations into simpler sums. In a reformulation of equation (1) (for clarity with $M = N$), we split the original sum with length N into N/L sums of length L (hop-size). Hereafter we can again sum the partial sums, and the result is of course the same:

$$\begin{aligned} X_l[k] &= \sum_{n=0}^{N-1} x[n + lL] \cdot w_N^{-kn} \\ &= \sum_{c=0}^{\frac{N}{L}-1} \sum_{n=cL}^{(c+1)L-1} x[n + lL] \cdot w_N^{-kn}. \end{aligned} \quad (2)$$

The inner sum in equation (2) can be expressed as a (time-shifted) zero-padded STFT of the data sequence $x_c[n]$:

$$X_c[k] = \sum_{n=0}^{N-1} x_c[n] \cdot w_N^{-kn}, \quad k = 0, 1, \dots, N-1, \quad (3)$$

with

$$x_c[n] = \begin{cases} x[n + lL], & \text{for } cL \leq n < (c+1)L; \\ 0, & \text{elsewhere;} \end{cases}$$

where c is a circular counter related to the data frame number l by $c = l \bmod \frac{N}{L}$.

This transform can be computed by an FFT algorithm. The resulting complex Fourier coefficients are stored in a circular buffer of the dimension $[N/L, k_{\max}]$, since one elementary transform is used in N/L calls of the MR FFT method and only up to k_{\max} frequency bins may be of interest for the subsequent analysis.

The FFT spectra $X_c[k]$ form the basis of the MR FFT: all different resolutions can be calculated as a summation of the elementary transforms. Summing up to N/L neighboring elementary transforms increases the frequency resolution from f_s/L to f_s/N with the increasing number of summands r . In order to comply with the condition for windowing in the frequency domain (see section 2.1), the number of summands r is restricted to certain values, because the fraction $N/M = N/(rL)$ has to be an integer value. For example, if $N = 2048$ and $L = 256$, the sum of $r = 1, 2, 4, 8$ elementary transforms is possible – resulting in four different spectrogram resolutions with $M = 256, 512, 1024, 2048$.

While the magnitudes of the summed spectrograms are immediately valid, the phase of the complex Fourier coefficients has to be corrected in order to make windowing in the frequency domain possible. The phase error is due to the time-shift of the data which introduces a phase shift in the frequency domain according to the shifting theorem of the DFT:

$$x[n + L] \mapsto \overline{X[k]} \cdot w_N^{kL}. \quad (4)$$

The angle of the phase shift is dependent on the frequency of the designated frequency bin k and the circular counter c , which is related to the data frame number l as defined in (3). Fortunately this effect can be cancelled by multiplying the phase-shifted spectrum $X_r^*[k]$ with a twiddle factor as follows:

$$X_r[k] = X_r^*[k] \cdot w_N^{-k c_{\min,r} L}, \quad r = 1, 2, 4, \dots, N/L, \quad (5)$$

where r is the number of summed elementary transforms $X_c[k]$ and $c_{\min,r}$ is the circular counter index of the smallest frame number $l_{\min,r}$ of the summed elementary transforms:

$$c_{\min,r} = l_{\min,r} \bmod \frac{N}{L}. \quad (6)$$

2.1. Windowing in the frequency domain

It is obvious that we cannot use time domain windowing with the MR FFT. But rather than applying the window in the time domain, we always have the option to perform frequency domain windowing, because the transform of a product is equivalent to the convolution of the two corresponding transforms. Admittedly, convolution is a time consuming operation, and it is only an alternative if the discrete spectrum of the window function is a short sequence of convolution coefficients. Fortunately some common windows have this desired property. The temporal weightings of interest have the general form:

$$\begin{aligned} h[n] &= \sum_{m=0}^{M/2} (-1)^m a_m \cos \left[\frac{2\pi}{M} mn \right] \\ &= a_0 - a_1 \cos \left(\frac{2\pi}{M} n \right) + a_2 \cos \left(\frac{2\pi}{M} 2n \right) - \\ &\quad a_3 \cos \left(\frac{2\pi}{M} 3n \right) + \dots, \quad n = 0, 1, \dots, M-1, \end{aligned} \quad (7)$$

and

$$\sum_{m=0}^{M/2} a_m = 1,$$

where M is the size of the data window and a_m are real constants [5]. Since the most important windows of this form have $a_m \neq 0$ only for small m , equation (7) is reduced to a few terms.

For any K nonzero coefficients a_m , the continuous spectral window function $H(\omega)$ consists of a summation of $2K-1$ weighted Dirichlet kernels:

$$H(\omega) = \sum_{m=0}^{M/2} (-1)^m \frac{a_m}{2} \left[D\left(\omega - \frac{2\pi}{M} m\right) + D\left(\omega + \frac{2\pi}{M} m\right) \right], \quad (8)$$

where $D(\omega)$ is the Dirichlet kernel as given in

$$D(\omega) = \left(+j \frac{\omega}{2} \right) \frac{\sin(\frac{M}{2}\omega)}{\sin(\frac{1}{2}\omega)}. \quad (9)$$

The Dirichlet kernel is implicitly available through the STFT with a rectangular window. So for the discrete case and if $M = N$ equation (8) simplifies as indicated in:

$$\begin{aligned} X[k]|_{\text{win}} &= \sum_{m=0}^{M/2} (-1)^m \frac{a_m}{2} (X[k-m] + X[k+m]), \\ k &= 0, 1, \dots, N-1. \end{aligned} \quad (10)$$

That is the reason why these windows are especially useful for frequency domain windowing, because they can be described by a short ($2K-1$) sequence of convolution coefficients. For example the Hann and Hamming windows possess only three nonzero coefficients, the Blackman window five:

- Hann: $a_0 = 0.5, a_1 = 0.5$
- Hamming: $a_0 = 0.53836, a_1 = 0.46164$
- Blackman: $a_0 = 0.42, a_1 = 0.5, a_2 = 0.08$

Other windows of this form with very good sidelobe behavior are described in [5].

Yet, we have to pay attention to the fact that equation (10) only holds for the special case $M = N$. In order to apply frequency windowing to the transform of zero-padded data we have to refine:

$$X[k]_{\text{win}} = \sum_{m=0}^{M/2} (-1)^m \frac{a_m}{2} \left(X\left[k - m \frac{N}{M}\right] + X\left[k + m \frac{N}{M}\right] \right) \quad k = 0, 1, \dots, N-1. \quad (11)$$

The term $m \frac{N}{M}$ in (11) must be an integer, because we only know the Fourier coefficients $X[k]$ at discrete bin locations k . Hence the number of possible spectrogram resolutions is reduced to a subset by the condition $(\frac{N}{M} = \frac{N}{rL}) \in \mathbb{N}$, where r is the number of summed elementary transforms.

2.2. Algorithmic Complexity

The algorithmic complexity of an N -point FFT is usually quantified as $C_{\text{FFT}} = O(N \log N)$, yet, if we use overlapping FFT windows only hopsize L new data samples are processed within one FFT frame.

While the computation of the MR FFT elementary transforms $X_c[k]$ has also complexity C_{FFT} per L samples, the additional effort depends on the required frequency range of the distinct spectrogram resolutions. If we compute all MR FFT spectrograms up to the highest frequency bin k_{\max} , we have to perform $(\frac{N}{L} - 1) k_{\max}$ complex additions (due to the summation of the elementary transforms), plus k_{\max} complex multiplications for every spectrogram resolution. Furthermore, we require additional storage for the intermediate results and the twiddle factors. If we compute the multiple resolutions (including zero padding) using the standard FFT, the computational cost is C_{FFT} multiplied with the number of resolutions. If a Hann window is applied to a real data sequence of length N in the time domain, N real multiplies have to be performed. In the frequency domain, the computational complexity can be identified as $6k_{\max}$ real multiplications and $3k_{\max}$ complex additions.

2.3. Implementation

Our approach to sinusoidal extraction has been successfully implemented in a melody extraction algorithm. For audio data sampled at $f_s = 44.1\text{kHz}$, we employ a Multiresolution FFT with $N = 2048$ and $L = 256$, resulting in four distinct spectrogram resolutions. While the best time resolution of 5.8 ms is obtained with the elementary transform ($M = 256$), the highest frequency resolution is achieved by the summation of all elementary transforms ($M = 2048$) and amounts to 21.5 Hz. The spectrogram with the most accurate frequency representation is used in the low frequency region, or to be exact, in the first six critical bands of the Bark scale. Accordingly, every other resolution covers five critical bands up to the maximum frequency $f_{\max} = 5000$ Hz, i.e. $k_{\max} = 232$.

3. SINUSOIDAL IDENTIFICATION

Since sinusoidal components of the audio signal contain the most relevant information about the melody, a sinusoids plus noise analysis is performed on the spectral data. The underlying idea of this technique is, that an audio signal can be divided into stable partials originating from periodic sound and a noise component [1]. Only partials which (probably) result from a deterministic signal are used for further melody analysis; stochastic components are neglected.

The most common criterion for detecting a sinusoid is the spectral peak. While peak picking is very robust against noise and distortion and also works in dense spectra, it produces a high number of false positives due to spurious peaks. That is why additional criteria are employed; for example the continuity of sinusoidal trajectories over time, or the concordance of the extracted sinusoidal components with harmonic patterns [1].

Since the explicit identification of continuous sinusoidal trajectories is not a precondition for our frame-wise pitch estimation method, we aim to identify sinusoids by distinct spectral features in one frame alone.

3.1. Estimation of Instantaneous Frequency and Magnitude

There are many methods for the estimation of the instantaneous frequency (IF) and magnitude from Fourier coefficients. Keiler and Marchand compared some of the most popular ones in [6]. They found that methods which are in some way based on the phase information of the FFT give the best results regarding frequency resolution. This property is extremely important for the analysis of polyphonic music.

We apply the well-known phase vocoder method proposed by Flanagan and Golden for the IF extraction and compute the instantaneous frequency $f_i[k]$ from the phase difference $\Delta\phi[k]$ of successive phase spectra as follows [7]:

$$f_i[k] = (k + \kappa[k]) \frac{f_s}{N}, \quad (12)$$

with:

$$\kappa[k] = \frac{N}{2\pi L} \text{princarg} \left[\phi_l[k] - \phi_{l-1}[k] - \frac{2\pi L}{N} k \right],$$

where *princarg* is the principal argument function mapping the phase to the $\pm\pi$ range. The bin offset κ denotes the deviation of the partial's IF from the bin frequency expressed in the unit bin. If the estimated bin offset of a peak is less than $\pm 1/2$, we can say that the instantaneous frequency of the peak corresponds to the bin frequency. In order to estimate valid IF over a range of frequency bins with the phase vocoder method, the use of overlapping STFT windows (or zero-padding) is required, because otherwise the phase difference between frames might exceed 2π . The maximum bin offset which can be computed with this method is $\frac{N}{2L}$.

The instantaneous magnitude of the sinusoidal peak is estimated from the local maximum $|X[k]|$ and its bin offset $\kappa[k_{\text{peak}}]$ as follows:

$$A_{\text{peak}} = \frac{1}{2} \frac{|X[k]|}{W_{\text{Hann}} \left(\frac{N}{M} \kappa[k_{\text{peak}}] \right)}, \quad (13)$$

where W_{Hann} is the Hann window kernel:

$$W_{\text{Hann}}(\kappa) = \frac{1}{2} \frac{\sin \left(\frac{M}{N} \pi \kappa \right)}{1 - \left(\frac{M}{N} \kappa \right)^2}.$$

3.2. Bin Offset Criterion

Charpentier proposed a sinusoidality criterion for speech processing in [8], which is derived from local characteristics of the phase spectrum, or more precisely, the instantaneous frequencies of neighboring frequency bins. By using a local criterion we avoid including restrictive assumptions about the harmonic structure of the signal and leave the observation of the temporal continuity to a processing level subsequent to the pitch estimation.

Using the sinusoidality criterion suggested by Charpentier we can verify a given spectral peak¹ at bin k by the two conditions:

$$\kappa[k] \leq \frac{1}{2} \quad (14)$$

and:

$$(f_i[k] \approx f_i[k-1]) \wedge (f_i[k] \approx f_i[k+1]). \quad (15)$$

Equation (15) may be expressed in terms of the bin offset κ :

$$\kappa[k] \approx \kappa[k \pm 1] \pm 1. \quad (16)$$

For sinusoidal peaks with a stationary frequency and amplitude these conditions hold, but for the most noisy peaks they do not. Since we cannot expect that all sinusoids are ideally stationary, we allow some error in the actual implementation considering peaks with:

$$\kappa[k] < 0.7 \quad (17)$$

and:

$$|\kappa[k] - \kappa[k \pm 1] \mp 1| < 0.4 \quad (18)$$

as sinusoidal.

3.3. Weighted Bin Offset Criterion

While Charpentier's criterion works very well with monophonic audio, we face a more challenging situation within polyphonic audio, where we find a higher number of concurrent harmonics and additional noise through percussive instruments. Effectively, the phase spectrum is more distorted and the calculated IF are often not very reliable. This is especially true for Fourier coefficients with a weaker magnitude – often close to the minima of the spectral window function. That is why we have to relax the bin offset criterion further. Hence, the estimated frequency error (the difference between instantaneous frequencies of two frequency bins) is weighted according to the instantaneous magnitude A_{peak} of the sinusoid and the respective magnitude of the neighboring Fourier coefficients $|X[k \pm 1]|$:

$$|\kappa[k] - \kappa[k \pm 1] \mp 1| < 0.4 \frac{A_{\text{peak}}}{|X[k \pm 1]|}. \quad (19)$$

Furthermore, the IF of the sinusoidal peak may deviate more from the corresponding bin frequency:

$$\kappa[k] < 0.7(r + 1), \quad (20)$$

where r is the MR FFT resolution parameter.

¹Within a specific MR FFT resolution a spectral peak can only be verified if the neighboring bins are of the same resolution. Hence the different resolutions must overlap by one bin.

3.4. Masking Criterion

Unlike the before-mentioned criteria the masking criterion is a method to exclude non audible peaks – sinusoidal or not – from further processing. We use a very simplified implementation of simultaneous and temporary masking, which by far does not reach the complexity of models used in modern lossy audio coders, as for example the AAC codec² [9].

Simultaneous masking is a property of the human auditory system where certain masked sounds are not audible in the presence of concurrent masker sounds. The spread masking across critical bands is very basically modeled as triangular spreading function $SF(z)$ with slopes of +25 and -10 dB on the normalised Bark scale:

$$SF(z) = \begin{cases} 10^{25z/20}, & \text{for } z \leq 0 \\ 10^{-10z/20}, & \text{for } z > 0 \end{cases} \quad (21)$$

The resulting approximate spread spectrum function $SSF[i]$ is computed with a resolution of 1/3 Bark, so that the critical band position index i corresponds to the Bark value z with $z = i/3$.

Temporal masking is the characteristic of the auditory system where the masker sound makes inaudible other sounds which are present immediately preceding or following the stimulus. We only take into account the much more pronounced effect of forward masking (masking that obscures a sound immediately following the masker) which is computed as given in:

$$TS[i] = 0.4 TS_a[i] + 0.6 TS_b[i], \quad (22)$$

with:

$$\begin{aligned} TS_a[i] &\leftarrow t_a TS_a[i] + (1 - t_a) SSF[i] \quad \text{and} \\ TS_b[i] &\leftarrow t_b TS_b[i] + (1 - t_b) SSF[i]. \end{aligned}$$

The parameters t_a and t_b are time constants which determine the exponential growth and decay of $TS_a[i]$ and $TS_b[i]$:

$$t_a = 0.5^{\Delta t_{\text{grid}}/5ms} \quad \text{and} \quad t_b = 0.5^{\Delta t_{\text{grid}}/70ms}.$$

Although the resulting masking threshold depends on the tonality of the masker, our approach uses a constant masking threshold $M[i]$ which is 15 dB below the maximum between the values of $SSF[i]$ and $TS[i]$:

$$M[i] = \max(SSF[i], TS[i]) \cdot 10^{15/20}. \quad (23)$$

4. EVALUATION

4.1. Multiresolution FFT

Figure 1 shows a comparison of the analysis results obtained with either the MR FFT or the FFT with constant frequency resolution. Spectrogram (a) shows the FFT together with simple peak picking. The high number of spurious peaks is obvious. The number of peaks is gradually reduced in spectrogram (b), which illustrates the spectral peaks obtained by the MR FFT analysis. The decreasing number of peaks for the high frequency regions is due to a masking effect, which can be explained by the wider main lobe of the spectral window function with decreasing frequency resolution. As a consequence we will of course loose selectivity in the

²The proposed masking model is optimized for sinusoidal peak identification and efficiency. It is not useful for encoding audio.

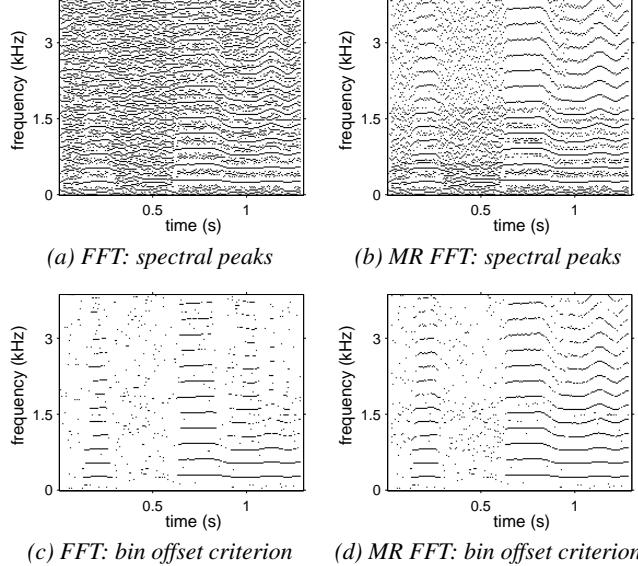


Figure 1: Comparison of spectral analysis using either FFT or MR FFT.

higher frequency regions and concurrent signal components may not be resolved. Besides the fact that the human auditory system also has a limited, frequency dependent, resolution, the MR FFT offers additional advantages.

This is especially true for the instantaneous frequency estimation. If the signal's frequency is changing in time an accurate measurement of frequency should be as local as possible [10]. This implies that the analysis window size should be as local as possible. There is always a trade-off between this claim and the wish to discriminate concurrent signal components. In polyphonic music, we find a mixture of voices in the low and middle frequencies, while the harmonics of the leading voice dominate the higher spectral bands [4]. Thus a good frequency resolution is required mostly in the low frequency regions, where the harmonics exhibit a quasi stationary frequency compared with the FFT filter bandwidth. With increasing harmonic number frequency modulation of the partials becomes more evident, so for higher harmonics the stationarity criterion is often violated. The MR FFT analysis offers the possibility to adapt the frequency resolution accordingly and considerably improves the IF estimation for the higher frequency regions.

Another advantage of the MR FFT lies in the improvement of the sinusoidal detection. Whenever a sinusoid is not stationary within one FFT frame, the corresponding spectral peak becomes distorted. Since both the Charpentier bin offset criterion and the weighted bin offset criterion require a more or less undistorted phase spectrum, such sinusoidal peaks will not be identified. This effect may be observed in spectrogram (c), which shows the application of the Charpentier bin offset criterion on the ordinary FFT spectrum. Indeed most of the unwanted spectral peaks disappear in the lower frequency regions, but at the same time we observe the deletion of high harmonics with a rapidly changing frequency. Finally spectrogram (d) shows the sinusoidal identification based on the MR FFT analysis, where high harmonics are correctly identified as deterministic partials. We see that the MR FFT accounts

for a significant improvement of the sinusoidal detection.

4.2. Sinusoidal Identification

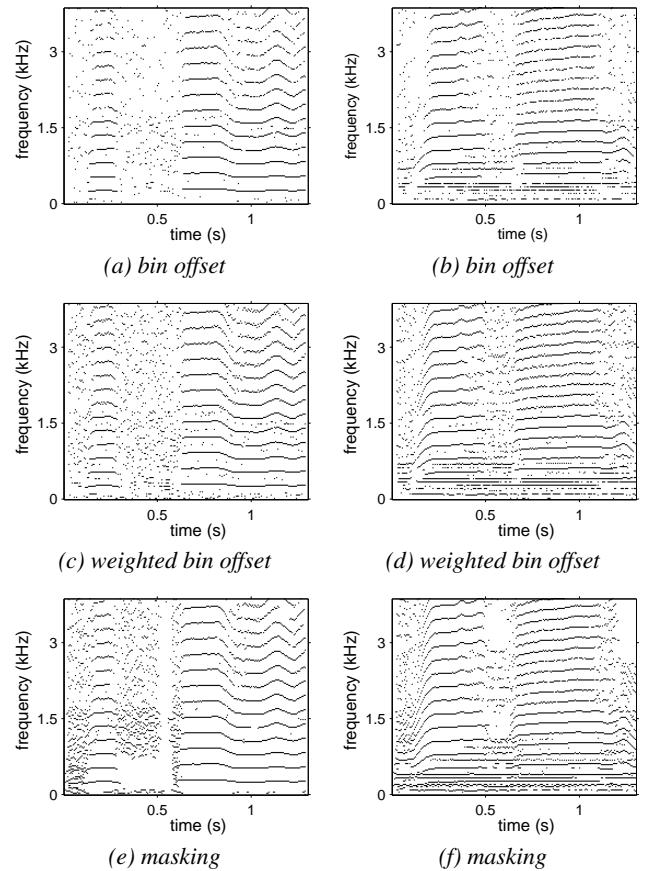


Figure 2: Analysis results for the distinct sinusoidality criteria using the MR FFT. Monophonic audio: figures (a), (c) and (e). Polyphonic audio: figures (b), (d) and (f).

Figure 2 allows us to compare qualitatively the analysis results of the proposed criteria. All examples have been computed using the MR FFT. The images on the left hand side show the results for monophonic audio (the sung word 'history'). We see that all methods perform well on the monophonic example. However, the bin offset criterion inspired by Charpentier's method causes the lowest number of false positives (peaks which have been erroneously identified as sinusoidal). At the same time it maintains most of the valid sinusoidal peaks. Clearly, this criterion is very suitable for monophonic analysis – for example speech processing.

Yet, the task of melody extraction naturally implies the analysis of polyphonic audio. The sinusoidal detection should not only be quiet robust against noise and distortion, it should also give adequate results for dense spectra consisting of many simultaneous sounds, which for example can be found in many pieces of rock music.

We have chosen a short excerpt from the file pop1.wav, which is included in the ISMIR2004 melody contest test set³, to illustrate

³The ISMIR 2004 melody test set with reference transcriptions is avail-

Voicing Detection	Voicing False Pos.	Raw Pitch Accuracy	Raw Chroma Accuracy	Overall Accuracy
81.8%	17.3%	68.1%	71.4%	71.4%

Table 1: MIREX 2005 evaluation results.

the performance of the distinct sinusoidality criteria with polyphonic music. This piece of music exhibits a rather moderate degree of polyphony, featuring a predominant singing voice and a light accompaniment without percussion. Nevertheless, it reveals the main drawback of the two phase dependent criteria: if two sinusoids are very close to each other in frequency, the frequency responses interfere and the sinusoidal peak will not be identified due to a distorted phase. Even if the frequency estimate of the actual peak remains reliable, the estimated bin frequencies (or bin offsets) of the neighboring bins are often more affected. This is due to their weaker magnitude and of course they may be closer to the disturbing sinusoidal.

This effect can clearly be noted in spectrogram (b), where the harmonics in the lower frequency regions almost disappear. The weighted bin offset criterion allows a bigger offset error for weaker magnitudes – as we can see in spectrogram (d) the sinusoidal detection is improved noticeable. This criterion was utilized in our submission to the MIREX2005 Melody Contest (see section 4.3). It proved suitable for the extraction of the predominant voice from musical audio with a moderate accompaniment, since the melody line still can be reconstructed from the higher harmonics. However, the bass line as well as other accompanying instruments often cannot be extracted reliably, even if they can be easily discriminated by human listeners.

Finally spectrogram (f) displays the most general solution to sinusoidal identification, which is based on psychoacoustic masking principles. In comparison with the bin offset criteria the masking criterion produces a higher number of false positives. However, this criterion guarantees a robust identification of all kinds of music and nonetheless significantly reduces the number of spectral peaks.

4.3. MIREX 2005 Audio Melody Extraction Contest

The aim of the MIREX Audio Melody Contest is to evaluate different approaches to extracting the main melody from polyphonic audio⁴. The MIREX 2005 dataset contains 25 phrase excerpts of 10-40 seconds length from different genres.

Our submission to the MIREX Audio Melody Contest used the MR FFT analysis as described in section 2 together with the weighted bin offset criterion introduced in section 3.3. Reaching 71.4% our algorithm has performed best on Overall Accuracy with a significant difference to other submissions. Of course the result should not be attributed to the spectral analysis front-end alone, but at least we can say that an FFT-based analysis does not contradict with a good melody extraction performance.

During the MIREX evaluation the execution time for the entire melody analysis has been measured as approximately 20 times

able at <http://www.iua.upf.es/mtg/ismir2004/contest/melodyContest/FullSet.zip>

⁴A detailed description of the MIREX 2005 evaluation procedure and the results can be found online at <http://www.music-ir.org/evaluation/mirex-results/audio-melody/index.html>

faster than real-time on an Intel® Pentium® 4 3.0 GHz CPU system with 3 GB RAM – the fastest runtime among all ten submissions. Despite this encouraging result, we want to emphasise that the MIREX evaluation did not attach importance to the execution time, and the measures should be recognised as rough indicators for algorithm efficiency, if at all.

5. SUMMARY

The MR FFT has proved to be useful for the efficient analysis of polyphonic audio – especially, if good frequency resolution as well as good time resolution is needed. The multi-resolution approach significantly improves the IF estimation and the detection of sinusoidal components by simple local sinusoidality criteria. As an alternative to the phase dependent criteria we proposed a masking criterion which is based on a very simple psychoacoustic model. This criterion has proved robust in all kinds of polyphonic music.

The presented spectral analysis front-end successfully deals with the dynamics of the singing voice or a lead instrument, but of course it can be applied to various problems in audio analysis.

6. REFERENCES

- [1] X. Serra, “A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition,” Ph.D. dissertation, Stanford University, USA, 1989.
- [2] A. Röbel, M. Zivanovic, and X. Rodet, “Signal decomposition by means of classification of spectral peaks,” in *Proc. Int. Comp. Music Conf. (ICMC’04)*, Miami, USA, 2004, pp. 446–449.
- [3] P. Masri and A. Bateman, “Identification of nonstationary audio signals using the FFT, with application to analysis-based synthesis of sound,” in *Proc. IEE Colloquium Audio Eng., Digest No. 1995/96*, 1995, pp. 11/1–11/6.
- [4] M. Goto, “A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals,” *Speech Communication*, vol. 43, no. 4, pp. 311–329, 2004.
- [5] A. H. Nuttall, “Some windows with very good sidelobe behavior,” *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. ASSP-29, no. 1, pp. 84–91, 1981.
- [6] F. Keiler and S. Marchand, “Survey on extraction of sinusoids in stationary sounds,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 51–58.
- [7] J. L. Flanagan and R. M. Golden, “Phase vocoder,” *Bell System Technical Journal*, vol. 45, pp. 1493–1509, 1966.
- [8] F. J. Charpentier, “Pitch detection using the short-term phase spectrum,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP’86)*, Tokyo, Japan, 1986, pp. 113–116.
- [9] ANSI, *Information technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced Audio Coding (AAC)*, INCITS/ISO/IEC 13818-7, 2003.
- [10] M. S. Puckette and J. C. Brown, “Accuracy of frequency estimates using the phase vocoder,” *IEEE Trans. Speech and Audio Proc.*, vol. 6, no. 2, pp. 166–176, 1989.

HIGH ACCURACY FRAME-BY-FRAME NON-STATIONARY SINUSOIDAL MODELLING

Jeremy J. Wells, Damian T. Murphy

Audio Lab, Intelligent Systems Group, Department of Electronics
 University of York, YO10 5DD, UK
 {jjw100|dtm3}@ohm.york.ac.uk

ABSTRACT

This paper describes techniques for obtaining high accuracy estimates, including those of non-stationarity, of parameters for sinusoidal modelling using a single frame of analysis data. In this case the data used is generated from the time and frequency reassigned short-time Fourier transform (STFT). Such a system offers the potential for quasi real-time (frame-by-frame) spectral modelling of audio signals.

1. INTRODUCTION

Spectral modelling (SM) for transformation of musical signals is a well established area of digital audio effects [1]. Whereas the STFT represents a signal as grains with stationary magnitude and phase, which overlap in time and frequency, spectral models attempt to infer more intuitive and flexible representations of sound from such data. Whilst the original signal cannot be perfectly reconstructed from such models they are generally more amenable to feature extraction and perceptually meaningful transformations such as pitch shifting and hybridisation (cross-synthesis). The spectral modelling synthesis system (SMS) of Serra represents signals as the combination of sinusoids with slowly varying amplitude and frequency and filtered noise [2]. Other systems have extended the component set to include transients [1].

Whilst SM systems exist that can perform transformations and resynthesis from model data in real-time, the possibility of *generating* model data in real-time has received little attention. Since the complete analysis-modification-resynthesis cycle cannot be currently implemented in real-time audio processors that use SM, such a modelling paradigm is unavailable in the traditional real-time studio effects unit. A real-time/streaming system has recently been described but there is more than a single frame's delay between input and output while a minimum number of 'track points' are acquired [3]. There have been investigations into both single-frame sinusoidal discrimination and non-stationarity but this has been applied to the improvement of offline analysis. The system described in this paper produces non-stationary 'sinusoidal plus residual' model data on a frame-by-frame basis. Once a single frame of data has been acquired, sinusoids and non-sinusoids can be separated and the sinusoids described and synthesized with non-stationary amplitude and frequency (*i.e.* these parameters can change on a sample by sample basis).

Since this paper builds on existing work relating to single-frame non-stationary modelling of sinusoids an overview of this work is given in section 2. The limitations of existing non-stationary analysis and an improved system, which is adapted to function with reassigned STFT data and reduce parameter interdependence, are discussed in section 3. Section 4 describes how estimates of intra-frame amplitude and frequency change obtained using the

methods described in the previous section can improve estimates of the mean amplitude and frequency. Section 5 presents a frame-by-frame spectral modelling system which uses the analysis techniques described in the previous sections.

2. EXISTING METHODS FOR PARAMETER ESTIMATION

The non-stationary sinusoids discussed in this paper are assumed to be of the form:

$$s(t) = A(t) \sin \left(\int_{\tau=0}^{\tau=t} 2\pi f(\tau) d\tau + \phi \right) \quad (1)$$

where, for a single frame, $A(t)$ is an exponential function describing the amplitude trajectory and $f(t)$ is a linear function describing the frequency trajectory. ϕ is the phase of the sinusoid at the start of the frame. Therefore in this model the amplitude is piecewise exponential, the frequency is piecewise linear and the phase is piecewise quadratic. The 'piecewise' nature of these trajectories is inherent in the frame-by frame approach and existing cubic phase modelling techniques require more than a single frame of data to have been acquired [4].

Many methods exist for the estimation of the mean instantaneous frequency of components in the Fourier domain. These include measuring the phase difference between successive frames, interpolation of the magnitude spectrum and time-frequency reassignment. Reassignment is used in the system described here since estimates are obtained from a single analysis frame and it provides better estimates than other single-frame methods [5]. Time-frequency reassignment estimates the deviation of component energy from the centre of an analysis bin (frequency) and analysis frame (time) by taking two additional DFTs per frame. The first DFT uses a time ramped version of the original window function, the second uses a frequency ramped (time domain first order difference) version of the window. The estimate of frequency deviation from the centre of an analysis bin is given by:

$$f_{\text{deviation}} = -B \Im \left(\frac{DFT_{\text{frequency ramped window}}}{DFT_{\text{original window}}} \right) \quad (2)$$

where B is the width of a single analysis bin and \Im denotes the imaginary part of a complex number. The estimate of time deviation (in seconds) from the centre of an analysis frame is given by:

$$t_{\text{deviation}} = -\frac{1}{F_S} \Re \left(\frac{DFT_{\text{time ramped window}}}{DFT_{\text{original window}}} \right) \quad (3)$$

where F_S is the sampling rate and \Re denotes the real part of a complex number [6].

Systems for the single frame estimation of the parameters of non-stationary sinusoids have recently been proposed. These include the use of direct analytical methods for Gaussian windows and Fresnel integral approximations or empirical adaptation of the Gaussian methods for other window types [7, 8, 9]. The technique adopted and adapted here is phase distortion analysis (PDA) [10]. This method uses phase differences either side of a zero-padded spectral peak to provide a measure of intra-frame linear frequency change and exponential amplitude change within a single frame. The relationship between these measures and the actual amplitude change (dB per frame) and frequency change (bins per frame) is dependent upon the window type and is empirically determined. This is formally described by:

$$\Delta A_p = g(\phi_{p+1} - \phi_{p-1}) \quad (4)$$

$$\Delta f_p = h(\phi_{p+1} + \phi_{p-1}) \quad (5)$$

where p is the index of a magnitude spectrum peak, ϕ is phase, ΔA and Δf are the intra-frame amplitude and frequency change and ($g(x)$ and $h(x)$) are the functions relating the phase difference to the intra-frame parameter changes. Amplitude and frequency non-stationarity produces changes in the window shape in the Fourier domain. Therefore, if these non-stationarities can be estimated then errors in the estimation of amplitude can be corrected and the quality of the model data improved [11].

3. REASSIGNMENT DISTORSION ANALYSIS

In this section we describe the adaptation of PDA to reassignment data, referred to here as reassignment distortion analysis (RDA). PDA uses phase deviations either side of the magnitude peaks in the DFT spectrum. For reassignment these deviations are embedded in the corresponding frequency and time offset estimates, given by Equation (2) and Equation (3). PDA effectively models the phase either side of a magnitude peak as a first order polynomial:

$$y = mx + c \quad (6)$$

where y represents the phase value, x the bin number, c the value from which the intra-frame frequency change Δf is derived, and m the value from which the intra-frame amplitude change ΔA is derived. PDA uses the difference in phase between the peak bin and those either side of the peak giving two data points. RDA directly uses time reassignment offset data across a peak giving three data points. Since three data points are available they can be modelled using a second-order polynomial which better represents the underlying shape of the phase spectrum. The non-stationary measures are therefore given by:

$$y = px^2 + mx + c \quad (7)$$

For RDA the relationship between this polynomial and the non-stationary measures is reversed: c is the value from which ΔA is derived and m is the value from which Δf is derived, p is not used. Figures 1 and 2 show the relationship between the RDA measure m and various combinations of values of ΔA and Δf for non-stationary Hann-windowed sinusoids using first- and second-order polynomials obtained from an 8192 point FFT of a 1025 sample frame. Where the frequency is decreasing the sign of m changes but its magnitude is the same.

Figures 1 and 2 show that, as for PDA, there is a limited range of Δf values for which m is monotonically increasing. This is

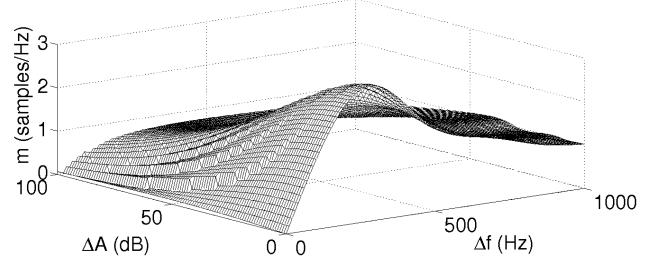


Figure 1: m versus ΔA and Δf , first order polynomial.

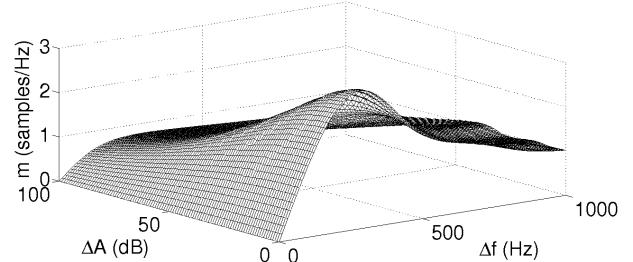


Figure 2: m versus ΔA and Δf , second order polynomial.

related to the length of the input frame and for a 1025 sample frame at 44.1 kHz this range of values is 0 to approximately 260 Hz/frame. Secondly, it can also be seen that a second order polynomial provides a smoother relationship between the polynomial coefficients and the non-stationary measures. Thirdly, these measures are not independent of each other as has been assumed in previous applications of PDA for non-stationary analysis [10, 11]. An intuitive explanation for the latter observation is that amplitude change effectively changes the shape of the analysis window and hence alters this relationship, since the relationship between m and Δf is dependent upon the type of window used.

Figure 3 shows the relationship between the RDA measure c and various combinations of values of ΔA and Δf . As for Δf and m , if the amplitude is decreasing then the sign of c changes but its magnitude is the same. From this figure it can be seen that c is monotonically increasing for all values of ΔA , although the relationship is not linear, as has been assumed in previous work, if the range of amplitude change values is wide enough to account for full onset or offset of a component within a single frame (≈ 96 dB in a 16 bit system).

Large values of Δf do not cause a significant change in the effective window shape and so the influence of this parameter over the relationship between c and ΔA is not as great as the opposite situation depicted in Figure 2.

The data presented in Figures 2 and 3 indicates that if highly non-stationary sinusoids are to be accurately quantified the assumption of independence for the two RDA measures is no longer valid. The relationship between Δf and m is affected by ΔA and the relationship between ΔA and c is affected, to a lesser extent, by Δf . We deal with this by using iterative 2D table look-up. Two modestly sized (100 by 100 element) arrays are filled with the data obtained for Figures 2 and 3. Small arrays and linear interpolation can be used since the functions they represent are smooth. The range of values for Δf is chosen to be that over which m is monotonically increasing and the range of values for ΔA is chosen as

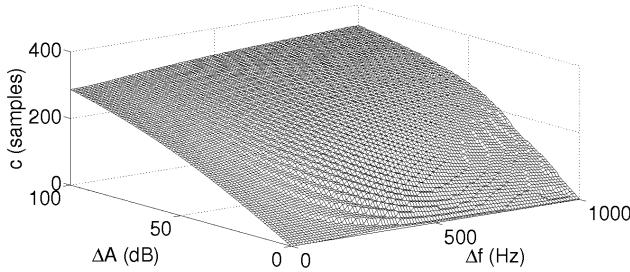


Figure 3: c versus ΔA and Δf , second order polynomial.

the largest range of values that can be represented in a linear 16 bit system. These arrays are then used to look up values of ΔA and Δf using the values of m and c and the current estimates of ΔA and Δf (which are assumed to be zero if no estimate is yet available, *i.e.* we are at the first iteration). The steps of the algorithm are as follows:

1. Obtain values for m and c by fitting a second order polynomial to the time reassignment offset data for the spectral component of interest.
2. Estimate ΔA from the amplitude change array, assuming that Δf is 0 Hz since changes in Δf have a smaller effect on c than those in ΔA have on m .
3. Estimate Δf from the frequency change array, assuming that ΔA is the value estimated in the previous step.
4. Estimate ΔA from the amplitude array, assuming that the value of Δf is that obtained in the previous step.
5. Repeat steps 3 and 4 until the algorithm is terminated (see below).

The termination point may be determined by the processing power available (particularly in a real-time context), the required accuracy of the estimates or the number of iterations before the final estimates of Δf and ΔA are no longer improved by repeated steps but begin to oscillate either side of their correct values. Increasing the number of iterations beyond 3 and taking the resulting single estimate does not improve the accuracy of the method. However, a small improvement over the accuracy obtained with 3 steps can be achieved by taking the mean of the estimates after 3 and 4 steps.

Figure 4 shows the percentage error in the estimation of Δf where $\Delta A = 90$ dB for existing applications of PDA and that for the iterative 2D interpolated RDA method described here. Whereas the estimation error for the former method rises from 58% to almost 80% as Δf increases from 0 to 260 Hz, the error for the new method is close to zero between 0 and 150 Hz, only rising above 10% for 220 Hz or greater with a maximum error of 20% at 260 Hz.

Figure 5 shows the percentage error in the estimation of ΔA where $\Delta f = 250$ Hz for the two methods. A logarithmic scale has been used for the error since there is such a large difference between the two methods. Even for such a large change in frequency within a single bin the error for the iterative method is never greater than 1%.

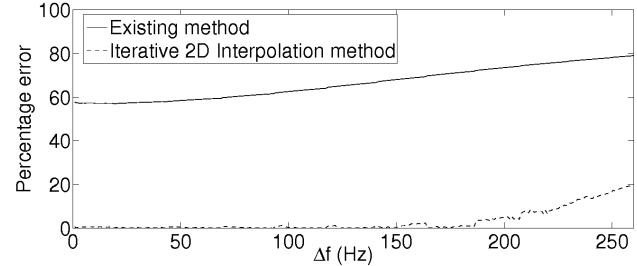


Figure 4: Δf estimation error for $\Delta A = 90$ dB.

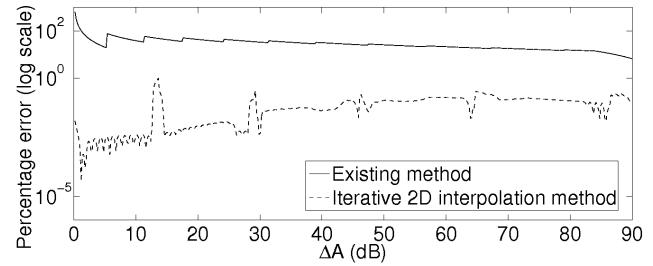


Figure 5: ΔA estimation error for $\Delta f = 250$ Hz.

4. FREQUENCY AND AMPLITUDE ESTIMATION

Like most DFT based frequency estimators, reassignment gives an estimate of the mean instantaneous frequency of a component across an entire analysis frame. This mean is amplitude weighted which allows the use of a windowing function to bias the estimate towards the instantaneous frequencies which occur nearest the centre of the frame. Where the amplitude of the sinusoid is constant throughout the frame no other biasing will occur. However where there are amplitude and instantaneous frequency non-stationarities this will affect the mean frequency estimate. We refer to this estimate as \bar{f}_{amp} , the amplitude-weighted mean instantaneous frequency. To fully separate the amplitude and frequency functions knowledge of the non amplitude-weighted mean instantaneous frequency, \bar{f} , is required. Previous non-stationary sinusoidal models simply use \bar{f}_{amp} but in the presence of large amplitude and frequency changes, such as at the onset and offset of sounds, large errors in the frequency estimation will result. Here we propose a method to correct this bias using the estimates of \bar{f}_{amp} , obtained directly from frequency reassignment, and ΔA and Δf obtained using the RDA method described in the previous section.

Taking the continuous case of a non-stationary sinusoid, as in Equation (1), with the parameters f , ΔA and Δf , where ΔA is given in dB and is assumed to be exponential and Δf is assumed to be linear, the sinusoid has the following amplitude and frequency functions:

$$f(t) = \bar{f} + \frac{\Delta f t}{2} \quad (8)$$

$$A(t) = 10^{at} \bar{A} \quad (9)$$

where

$$a = \frac{\Delta A}{40} \quad (10)$$

and t is in the range -1 to 1 (chosen to simplify the following integration). For a Hann window the amplitude weighted mean

instantaneous frequency is given by:

$$\bar{f}_{\text{amp}} = \frac{\frac{1}{2} \int_{-1}^1 10^{at} \bar{A} \left(\bar{f} + \frac{\Delta \bar{f}_t}{2} \right) \left(\frac{1}{2} + \frac{1}{2} \cos(\pi t) \right) dt}{\frac{1}{2} \int_{-1}^1 10^{at} \bar{A} \left(\frac{1}{2} + \frac{1}{2} \cos(\pi t) \right) dt} \quad (11)$$

The \bar{A} in the numerator and the denominator of (11) cancel out so solving the integrals and rearranging to find \bar{f} gives:

$$\bar{f} = \bar{f}_{\text{amp}} - \frac{\frac{\Delta f}{\kappa^2} [a(\kappa-1) + \frac{1}{a}(\kappa+1)]}{K} - \frac{\left[\frac{\Delta f}{(\pi^2 + \kappa^2)^2} \right] [\frac{1}{a} (\pi^2 - \pi^2 \kappa - \kappa^2 - \kappa^3) - a (\pi^2 + \pi^2 \kappa - \kappa^2 + \kappa^3)]}{K} \quad (12)$$

where

$$\kappa = \ln(a) \quad (13)$$

and

$$K = \left(a - \frac{1}{a} \right) \left(\frac{1}{\kappa} - \frac{\kappa}{\pi^2 + \kappa^2} \right) \quad (14)$$

Using this equation to improve the estimate of \bar{f} gives a significant improvement in the model accuracy for highly non-stationary components. This is illustrated by comparing figures (6) and (7) which show the error in estimating \bar{f} for different values of ΔA and Δf with and without this bias correction. For both figures estimates of ΔA and Δf obtained using the methods described in section 3, rather than the actual values used to synthesize the sinusoids, were used in the bias correction. Again, the data was derived from an 8 times zero-padded FFT of a 1025 sample frame.

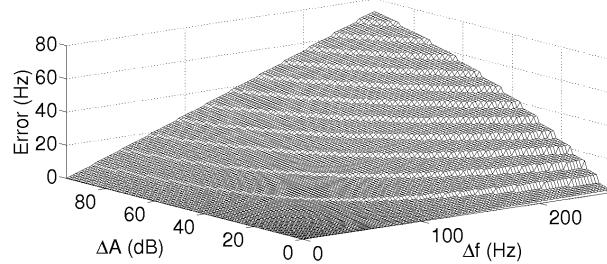


Figure 6: Amplitude biased frequency estimation error.

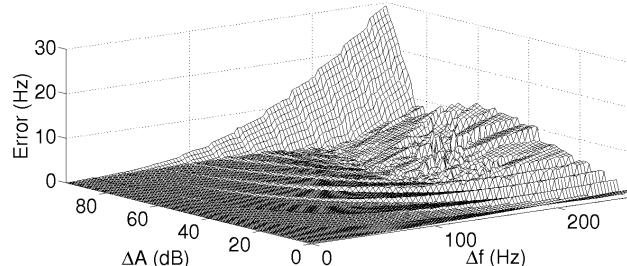


Figure 7: Frequency estimation error after correction.

Mean amplitude estimation is also affected by non-stationarity: frequency change within a frame causes greater spreading of signal energy across bins around a peak, which lowers the magnitude of the peak bin, and amplitude change causes the signal to

be more localised in time thus widening the main lobe. In addition to this the peak magnitude varies with the difference between \bar{f}_{amp} and the actual centre frequency of the bin in which the peak occurs. For stationary sinusoids, knowledge of the magnitude of the window function in the frequency domain allows amplitude estimation errors caused by the deviation from the centre of the analysis bin to be corrected [12]. Since no analytical solution for a Hann windowed sinusoid with non-stationary frequency is known it has been proposed to calculate the magnitude spectrum of the window for each component via FFT. From this an ‘amplitude factor’ is derived which is multiplied by the initial estimate of \bar{A} (the magnitude of the peak bin) [10]. As previously discussed, such an approach is likely to be prohibitively expensive in a real-time context.

Two new approaches are presented in this paper which do not require additional FFTs to be calculated: estimation of the amplitude correction factor by 2D array look-up (as described in the previous section for estimating ΔA and Δf) and modelling of the relationship between the amplitude correction factor, ΔA and Δf with two polynomials. For the first approach a 100 by 100 element array and linear interpolation is used (as described in section 3). The values for the array are calculated by inverting the normalised magnitude values obtained for sinusoids with \bar{f} which coincides with the centre of an analysis bin for the same range of values of ΔA and Δf . The second approach models the inverted normalised magnitude values with quartic polynomials. The non-stationary amplitude correction factor, α , is then estimated by:

$$\alpha = g(\Delta A) \cdot h(\Delta A) \quad (15)$$

where $g(x)$ and $h(x)$ are the quartic functions. Figures 8 and 9 show the data obtained and the best least squares fit provided by the quartic functions. For both figures an 8 times zero-padded FFT of a 1025 sample frame was used. For Figure 8, $\Delta A = 0$ dB, and for Figure 9, $\Delta f = 0$ Hz.

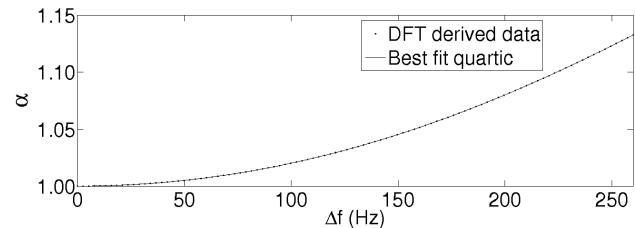


Figure 8: Amplitude correction factor versus Δf .

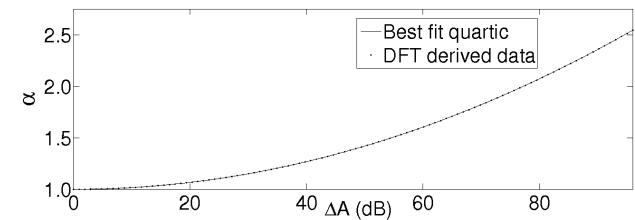


Figure 9: Amplitude correction factor versus ΔA .

Figures 10 and 11 show the percentage error in amplitude estimation for non-stationary sinusoids, whose \bar{f} is at the centre of an

analysis bin, for the 2D lookup and polynomial fit methods respectively. Again, for both figures estimates of ΔA and Δf obtained using the RDA method rather than the actual values used to synthesize the sinusoids, were used in the bias correction. The error without any correction is greater than 75% for $\Delta A = 96$ dB and $\Delta f = 260$ Hz and both methods offer a significant improvement over this. The array lookup performs best out of the two methods, indicating that the effects of amplitude and frequency stationarity are not entirely independent of each other. The first method is the one used in our system but the second may be useful in a system where memory is scarce or where memory lookup is a relatively expensive operation.

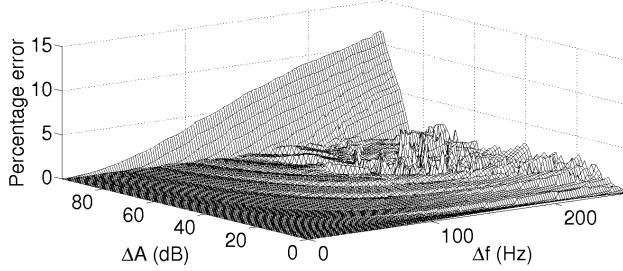


Figure 10: \bar{A} estimation error using 2D array lookup.

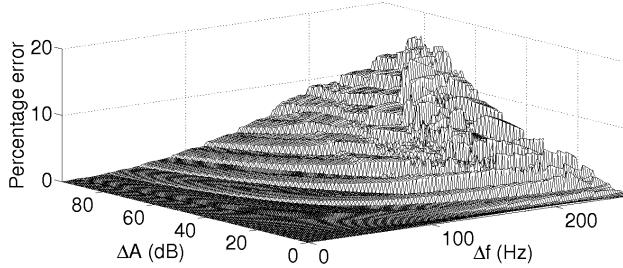


Figure 11: \bar{A} estimation error using polynomial product.

Once the corrected amplitude estimate has been obtained, the deviation of \bar{f}_{amp} from the centre of the analysis bin can be used to produce a further correction as is performed for stationary sinusoids with knowledge of the window shape [12]. However for 8 \times zero-padding, as used here, the amplitude estimation difference between a frequency offset of zero and the maximum of half an analysis bin is negligible (< 0.03 dB) and so this step can be omitted.

5. FRAME-BY-FRAME SPECTRAL MODELLING

The prime motivation for the research presented in this paper is the development of a real-time spectral modelling system. Our definition of real-time in this context is:

1. Quasi instantaneous: as close to instantaneous as is allowed by the frame size of the algorithm.
2. Frame-by-frame: this is implied in point 1. Only the current and/or previous frames may be used, waiting for future frames is not permitted.

3. Real-time execution: the execution time of the algorithm must be shorter than the time taken to acquire/replay the data it analyses/produces.

We have developed a frame-by-frame spectral modelling system which uses sinusoids to model the deterministic part of monophonic signals and a bank of parametric equalisers applied to a noise source to model the residual. Complex wavelet analysis is used to determine the centre frequency, bandwidth and gain of the equalisers. Since both signal types are synthesized in the time domain the model can be interacted with for sound transformation on a sample-by-sample basis. The separation of sinusoidal and residual parts of the signal is performed by measuring the goodness of fit of time reassignment offset data around magnitude peaks in the spectrum to the second order polynomial used to produce the RDA data. The benefit of the high accuracy description of sinusoids presented here is that they can tracked more accurately within a single frame, and across frames, in a real-time system. More accurate modelling of amplitude and frequency trajectories within each frame minimises discontinuities in amplitude and frequency between frames. This is shown for frequency in Figure 12 which shows the partial tracks generated by the system for a synthetic harmonic signal with fast and deep vibrato. This signal is chosen to demonstrate that this method produces accurate frequency tracks even in the presence of highly non-stationary components. In this example the analysis frame length is 513 samples with a window overlap of 2, therefore the synthesis frame length is 256. The system is able to produce such partial tracking with no knowledge of previous or subsequent analysis frames.

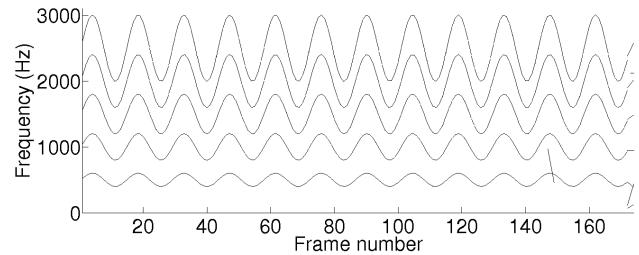


Figure 12: Partial tracks of harmonic sound with vibrato.

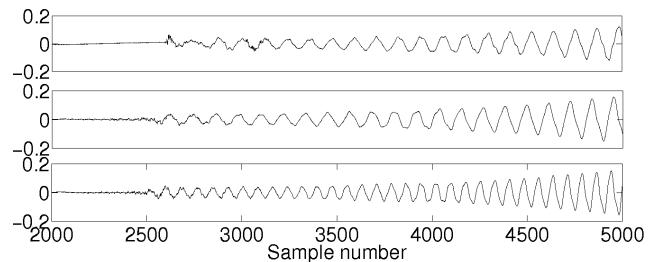


Figure 13: Flute onset: original (top), resynthesized (middle) and pitch shifted up by a perfect fifth (bottom).

Figure 13 shows time domain waveforms of the onset of a flute note, its unmodified resynthesized version and the resynthesis shifted in pitch by a perfect fifth (ratio of 3:2). In this example the analysis frame length is 1025 with an overlap of 2, giving a

synthesis frame size of 512. Even with a relatively long analysis frame the temporal envelope of the signal is largely retained, although the onset is not quite as sharp as in the original signal. With the signal modelled in such a way pitch shifting is a trivial operation: the values for \bar{f} are simply multiplied by the pitch ratio prior to resynthesis. For the residual the centre frequencies of the parametric equalisers are also scaled by the same amount. Of course, the spectral envelope can be preserved by frequency-domain interpolation of amplitudes, if required. The system has been implemented in Matlab as a combination of ‘m’ and MEX files. The sinusoidal analysis, modelling, discrimination and synthesis is executed in faster-than-real-time for all input sound types and the combined sinusoidal and residual modelling takes less than twice real-time when run on a modest general purpose PC. This suggests that a real-time spectral modelling system based on these methods, written entirely in a low level language and/or running on specialised hardware, can be realised.

6. CONCLUSION

A frame-by-frame sinusoidal analysis system which offers high accuracy estimates of the intra-frame change of amplitude and frequency using time reassignment data has been presented. These estimates can, in turn, be used to reduce errors in the estimation of the means of the amplitude and frequency functions. The high accuracy sinusoidal model that these techniques yield can be implemented with much smaller discontinuities in amplitude and frequency trajectories across frames than would otherwise be possible in such a frame-by-frame system. A detailed description and assessment of the sinusoidal discrimination and residual modelling methods used will be the subject of future papers. Further work will investigate how the bin phase affects estimation of the parameters considered in this paper and whether higher-order polynomial modelling of the time reassignment data could improve parameter estimates.

7. REFERENCES

- [1] X. Serra, “Spectral modeling synthesis: Past and present, keynote in Proc. Int. Conf. on Digital Audio Effects (DAFx-03), London, UK. [online] <http://www.iua.upf.es/~xserra/presentaciones/Spectral-Modeling-Synthesis-Past-and-Present.pdf>, 2003,” sep 2003.
- [2] ———, “A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition,” Ph.D. dissertation, Stanford University, USA, 1989.
- [3] V. Lazzarini, J. Timoney, and T. Lysaght, “Alternative analysis-synthesis approaches for timescale, frequency and other transformations of musical signals,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005, pp. 18–23.
- [4] R. J. McAulay and T. F. Quatieri, “Speech analysis/synthesis based on a sinusoidal representation,” *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [5] F. Keiler and S. Marchand, “Survey on extraction of sinusoids in stationary sounds,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 51–58.
- [6] F. Auger and P. Flandrin, “Improving the readability of time-frequency and time-scale representations by the reassignment method,” *IEEE Trans. Sig. Proc.*, vol. 43, no. 5, pp. 1068–1089, 1995.
- [7] G. Peeters and X. Rodet, “SINOLA: A new analysis/synthesis method using spectrum peak shape distortion, phase and reassigned spectrum,” in *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, 1999, pp. 153–156.
- [8] A. S. Master, “Nonstationary sinusoidal model frequency parameter estimation via fresnel integral analysis,” Master’s thesis, Stanford University, USA, 2002.
- [9] M. Abe and J. Smith, “AM/FM rate estimation for time-varying sinusoidal modeling,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'05)*, Philadelphia, USA, 2005.
- [10] P. Masri, “Computer modelling of sound for transformation and synthesis of musical signals,” Ph.D. dissertation, University of Bristol, UK, 1996.
- [11] M. Lagrange, S. Marchand, and J.-B. Rault, “Sinusoidal parameter extraction and component selection in a non-stationary model,” in *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, 2002, pp. 59–64.
- [12] M. Desainte-Catherine and S. Marchand, “High precision Fourier analysis of sounds using signal derivatives,” *J. Audio Eng. Soc.*, vol. 48, pp. 654–667, 2000.

GRANULAR RESYNTHESIS FOR SOUND UNMIXING

Gunnar Eisenberg, Thomas Sikora

Communication Systems Group

Technical University of Berlin

{eisenberg|sikora}@nue.tu-berlin.de

ABSTRACT

In modern music genres like Pop, Rap, Hip-Hop or Techno many songs are built in a way that a pool of small musical pieces, so called loops, are used as building blocks. These loops are usually one, two or four bars long and build the accompaniment for the lead melody or singing voice.

Very often the accompanying loops can be heard solo in a song at least once. This can be used as *a-priori* knowledge for removing these loops from the mixture. In this paper an algorithm based on granular resynthesis and spectral subtraction is presented which makes use of this a-priori knowledge. The algorithm uses two different synthesis strategies and is capable of removing known loops from mixtures even if the loop signal contained in the mixture signal is slightly different from the solo loop signal.

1. INTRODUCTION

In the field of musical studio productions and remix applications there exists a high demand for unmixing the single tracks of musical pieces from each other and extracting the used loops, if there are any. An example of a typical track scenario is given in Figure 1. For this scenario extracting the loops would result the loops A, B and C.

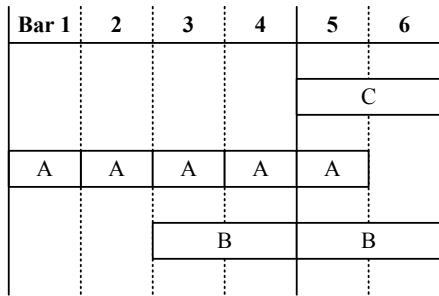


Figure 1: Typical simplified track scenario for modern music genres.

Besides the techniques presented for blind source separation so far, this paper presents a straightforward approach for unmixing songs which are built up on a loop base like it is described above. If a loop appears solo in the musical piece its characteristics can be used as *a-priori* knowledge to remove this solo part from the following parts resulting in a residual part which is often a single loop itself. From the scenario shown in figure 1, loop A could be retrieved directly from bar one or bar two. Loop B could be unmixed by removing two bars of loop A from the song's bars three and four. Loop C could be retrieved by removing the newly

unmixed loop B and one bar of loop A from the song's bars five and six.

A naïve implementation for this approach would be to subtract an accompaniment-loop which has been played solo before directly from the mix using the time signals. For this implementation to be successful it would be necessary that the accompaniment-loop is being looped in a sample-identical way. Further the boundaries, especially the starting-sample of the accompaniment-loop would have to be set in a sample-accurate way. Additionally the gain of the loop would have to remain absolutely constant.

These limitations cannot be held for professionally edited studio productions. In professional studio productions subsequently added effects like reverb, equalizers, dynamics, and mastering result in phase and amplitude distortions. Thus the assumption of sample accurate looping does not hold.

Further, many productions do not loop small wave-files directly, which would be classical looping, but repeat only the notes for the loop continuously. Normally these loops are programmed in a MIDI sequencer which sends control information to a synthesizer. The synthesizers itself often produce sounds which are not identical on the sample base. This applies especially to analogue-synthesizers which are being recorded using A/D-converters afterwards. Even digital synthesizers can have this behavior to make the sound livelier.

The mentioned drawbacks which occur for a subtraction using the time signals do not hold for the spectral properties of the tracks to be unmixed. This fact is exploited by the presented algorithm. The method is based on a granular resynthesis process together with spectral subtraction. Although the algorithm does not operate directly in the time domain the nomenclature of this case is used for this paper. This means that the music signal from which the known loop is extracted is called mix, the known loop itself is called subtrahend and the result of this unmixing is called residual. As it is explained in the following in more detail, the spectral properties of the resynthesis signal are adapted using the information which is gained by a granular analysis of the mix signal as well as the subtrahend signal to be removed from the mix signal.

2. PREVIOUS WORK

The presented algorithm faces the problem of monophonic sound unmixing with the technique of granular resynthesis. There are several publications on both of the named topics.

Casey has presented a method on independent subspace analysis (ISA) for separating individual audio sources from a single-channel mixture [1]. The method is based on the independent component analysis (ICA) but can operate in scenarios where there are less mixture observations than sources [2].

Virtanen has presented a data-adaptive sound source separation system, which is able to extract meaningful sources from polyphonic real-world music signals [3]. He also has developed an algorithm which transforms each source into a parameterized version which is expressed as a convolution between a time-frequency magnitude spectrogram and an onset vector [4].

Smaragdis has developed a method for the extraction of multiple sound sources from monophonic inputs which is an extension to the non-negative matrix factorization (NNMF). It is capable of identifying components with temporal structure and extracting multiple sound objects from a single channel auditory scene [5].

Since all algorithms developed for mono source separation fail in certain situations the topic has not been solved by now and further research needs to be performed.

The theory of granular synthesis was initially invented by Gabor who proposed that any sound could be decomposed into small acoustical grains [6, 7, 8]. Since then very much research on this topic has been performed under different names. Today aliases of the term "grain" are: "acoustic quantum", "gaborets", "gabor atom", and "wavelet" to name only a few. A good overview is given by Roads who has explained that the potential of granular representations has yet to be fully explored [9]. Further use cases and methods of granular synthesis have been named by Zölzer *et al.* [10].

New approaches to signal analysis by Mallat show several techniques that analytically combine granular synthesis with the broad category of wavelet or atomic decompositions [11].

3. ALGORITHM

3.1. Preliminaries

Songs to be unmixed using the described algorithm need to fulfill the following requirements:

- The tracks to be removed need to consist of mainly loops.
- The loop to be removed from a mix must appear solo at least once throughout the song.
- The loops of one track must not change their spectral characters heavily from one loop cycle to the next one. This means that for example deep filter sweeps should not occur.

These assumptions easily hold for many songs from modern music genres like Pop, Rap, Hip-Hop or Techno. Furthermore these assumptions directly model almost every modern musical studio production where different tracks and loops are composed in a sequencer or tracker program. Although the presented algorithm only performs the unmixing itself and no automatically loop boundary detection, aligning the song in a sequencer or tracker program will deliver these loop boundaries indirectly.

3.2. Granular Analysis

Let $x(n)$ be the time signal of the mix-loop and $s(n)$ be the time signal of the subrahend-loop. With equation (1) and (2) from both signals grains are extracted with their anchors having a distance of the hopsize N . The grains are windowed out of the signals using a Hanning window $w(n)$ of length W .

The grains themselves are denoted with $x_k(n)$ and $s_k(n)$ with k being the grain index. After extracting the grains they are analyzed using a FFT of size L , yielding in the spectral blocks $X_k(l)$

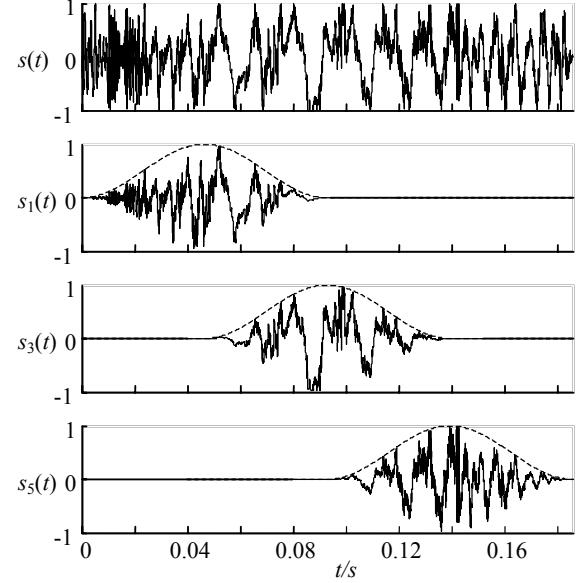


Figure 2: Extraction process of three analysis grains.

and $S_k(l)$.

$$x_k(n) = x(n + kN) \cdot w(n) \quad (1)$$

$$s_k(n) = s(n + kN) \cdot w(n) \quad (2)$$

$$x_k(n) \xrightarrow{\text{FFT}} X_k(l) \quad (3)$$

$$s_k(n) \xrightarrow{\text{FFT}} S_k(l) \quad (4)$$

For further processing the spectral blocks $X_k(l)$ and $S_k(l)$ are split into magnitude blocks $\hat{X}_k(l)$, $\hat{S}_k(l)$ and phase blocks $\hat{X}_k(l)$, $\hat{S}_k(l)$. Figure 2 shows an example-extraction of three grains from the subrahend signal. For audio signals with a sample rate of 44.1 kHz the following parameters have been determined heuristically to produce good analysis results:

- hopsize $N = 1024$ taps (23, 22 ms),
- window length $W = 4096$ taps (92, 88 ms),
- FFT size $L = 4096$ taps (92, 88 ms).

3.3. Basic Grain Synthesis

The information gained during the analysis process is now used for setting up the resynthesis grains. The following equations compute the magnitude $\hat{R}_k(l)$ and $R_k(l)$ of each residual grain:

$$\hat{R}_k(l) = \hat{X}_k(l) - \hat{S}_k(l), \quad (5)$$

$$\hat{R}_k(l) = \hat{X}_k(l). \quad (6)$$

By these two steps each residual grain's spectral information is now synthesized to:

$$R_k(l) = \hat{R}_k(l) \cdot \exp\left(j\hat{R}_k(l)\right) \quad (7)$$

From this residual grain spectrum the grain itself can easily be computed by an IFFT. It can then be used for the actual resynthesis in the time domain which is performed by summing up all synthesized residual grains.

The drawback with this basic grain synthesis process is that the spectral character of neighboring grains could vary quite heavily. This results in artifacts which make the grain-frequency *i.e.* the frequency with which the grains are placed according to the hopsize, quite audible. Further the subtrahend-loop must match the mix-loop very well. Often this cannot easily be provided since in almost every song there are slight variations from one loop to the next one.

The advanced grain synthesis technique fixes this problem by using more information of neighboring analysis grains to build the spectrum for the resynthesis grains.

3.4. Advanced Grain Synthesis

The advanced grain synthesis technique takes the spectral properties of M neighboring analysis grains with a certain amount into account. Therefore equation (5) is replaced by

$$\hat{R}_k(l) = \hat{X}_k(l) - \sum_{m=-(M-1)/2}^{(M-1)/2} \hat{S}_{k+m}(l) \cdot g(m). \quad (8)$$

The factors $g(m)$ which are called shadow factors have the shape of a window-function which is shown in Figure 3.

For audio signals with a sample rate of 44.1 kHz the shadow factors' total window size should be around 8192 taps (185, 76 ms), to achieve the best resynthesis quality. For a hopsize N of 1024 taps this means that seven analysis grains are taken into account for equation (8). This results in values for the shadow coefficients which are shown in Figure 3.

The rest of the residual grains' computation does not change, compared to the basic method. This means that equations (6) and (7) are also valid here.

Although the advanced grain synthesis is more robust against spectral changes of the analyzed material it can add an amount of blur to the synthesized grains. This especially occurs in percussive sounds with dominant transients. In these cases the Basic Grain Synthesis is preferred.

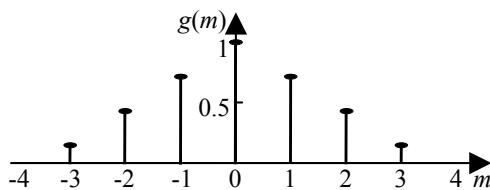


Figure 3: A typical set of shadow factors.

3.5. Granular Resynthesis

Every grain $R_k(l)$ is transformed from the spectral domain back into the time domain by using an IFFT.

$$R_k(l) \xrightarrow{\text{IFFT}} \tilde{r}_k(n), \quad (9)$$

Since we have manipulated the amplitudes of the contained sinewaves it is not guaranteed that the grain still has the shape

No.	Artist	Song
1	Beastie Boys	Hey Ladies
2	Depeche Mode	ASN - Voices
3	Depeche Mode	Any Second Now (RMX)
4	Electro Nation	Woman Machine
5	Led Zeppelin	Stairway to Heaven

Table 1: Demo songs used for evaluation.

of the Hanning window which we used for the initial extraction process. To avoid block artifacts which may occur when the grains are summed together another less invasive window $v(n)$ is applied to each grain. This window is mainly a rectangular window which fades in and out on the first and last 10% of the window size. This window should have the same length as the Hanning window which was used for the analysis grain extraction.

$$r_k(n) = \tilde{r}_k(n) \cdot v(n). \quad (10)$$

The actual resynthesis of the output signal $r(n)$ is performed by summing up all edge-corrected grains $r_k(n)$ at their correct time position:

$$r(n) = \sum_k r_k(n - kN). \quad (11)$$

4. RESULTS

The algorithm performs a psycho-acoustically motivated sound separation based on spectral properties. This makes it almost impossible to evaluate the algorithms quality in a purely deterministic way. Therefore the algorithm's quality was evaluated by performing an expert listening test based on a well known mean opinion score (MOS) criterion [12]. The five songs depicted in figure 4 were used as examples. The songs come from the genres Rap (No. 1), Pop (No. 2, No. 3) and Techno (No. 4). To show the limits of the algorithm also one Ballad was processed (No. 5) which does not hold the preliminaries named in section 3.1. The sound demos together with more examples can be found at our institute's website¹.

The MOS listening test has been performed with 25 musicians. For each song they were presented the mix signal, the subtrahend signal and the residual signal. For the unmixing the parameter set proposed in section 3 was used. These settings focus more on a good separation's quality than on little artifacts. This is important since these two quality aspects somewhat contradict each other.

The listeners had to judge two aspects of the sounds. The first aspect was how much from the subtrahend remained in the residual after unmixing, the second aspect was the presence of artifacts in the residual *i.e.* noise, crackles, fading and musical tones.

For both tests a MOS ranging from one to five was used. The separation's quality was scaled from one meaning "Unsatisfactory (Bad)" to five meaning "Excellent". The artifact impairment was scaled from one meaning "Very Annoying (Objectionable)" to five meaning "Imperceptible".

The results of the MOS test are depicted in figure 4. The overall MOS is 4.12 for the separation's quality, which is better than

¹www.nue.tu-berlin.de/wer/eisenberg/unmixing/

good, and 3.38 for the artifact impairment. This means that artifacts are perceptible and slightly annoying. The separation's quality is better than "Good" for all songs except No. 5. For No. 3 and No. 4 which contain dominant synthetic sounds it is almost "Excellent".

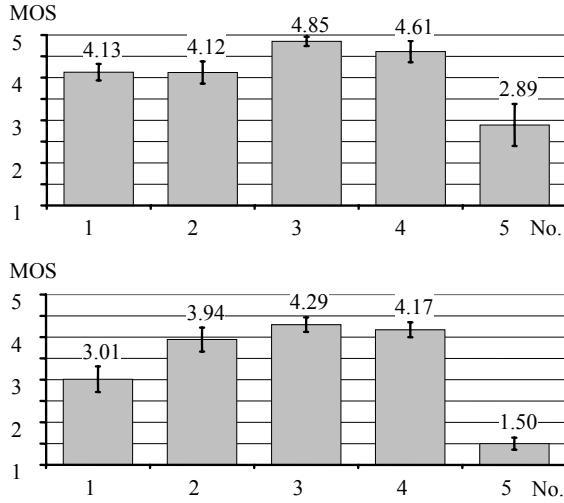


Figure 4: Mean values of the MOS test for the separation's quality (top) and artifact impairment (bottom).

The artifact impairment depends on the nature of the processed material. For songs which sound very synthetic like No. 3 and No. 4 the sound artifacts occurring in the residual signal are almost imperceptible. If the residual signal is a singing voice like in No. 1 and No. 2 the artifacts are just perceptible and slightly annoying. If there are little differences between the notes played in the subtrahend and in the mix, like in the bass line of No. 2, these cannot be removed completely and remain present in the residual, although this is hardly audible.

Song No. 5 shows the limitations of the presented algorithm by using a hand played loop as subtrahend which does not hold the limitations mentioned in section 3.1. The song's single notes vary heavily in time and have different spectral characteristics. Furthermore the residual signal is a singing voice which reduces the listener's tolerance for artifacts. After unmixing the residual signal contains annoying artifacts and the unmixing has almost a fair overall quality.

5. FUTURE WORK

Besides the expert listening sessions performed for the presented paper, a MOS test with more examples and more listeners is pre-

pared to illustrate the performance of the method.

The songs which the algorithm can unmix contain a high amount of self-similarity by default. The next step for the presented system will be to evaluate this self similarity and use it to perform automatic boundary detection for loops. This would result in less manual work for the user of the system since the system would present reasonable suggestions.

6. REFERENCES

- [1] M. A. Casey, "Separation of mixed audio sources by independent subspace analysis," in *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, 2000, pp. 154–161.
- [2] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. Wiley, 2001.
- [3] T. Virtanen, "Sound source separation using sparse coding with temporal continuity objective," in *Proc. Int. Comp. Music Conf. (ICMC'03)*, Singapore, 2003, pp. 231–234.
- [4] ———, "Separation of sound sources by convulsive sparse coding," in *ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing (SAPA)*, 2004, paper 55.
- [5] P. Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *Int. Cong. Independent Component Analysis and Blind Signal Separation (ICA)*, 2004, pp. 494–499.
- [6] D. Gabor, "Theory of communications," *J. Inst. Elect. Eng.*, vol. 93, no. III, pp. 429–457, 1946.
- [7] ———, "Acoustical quanta and the theory of hearing," *Nature*, vol. 159, no. 4044, pp. 591–594, 1947.
- [8] ———, "Lectures on communication theory," Technical Report 238, Research Laboratory of Electronics, Cambridge, Massachusetts, Massachusetts Institute of Technology, 1952, 238.
- [9] C. Roads, *Microsound*. Cambridge, Massachusetts: MIT Press, 2002.
- [10] P. Dutilleux, G. D. Poli, and U. Zölzer, "Time-segment processing," in *DAFX – Digital Audio Effects*, 2002, pp. 201–236.
- [11] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego: Academic Press, 1998.
- [12] N. S. Jayant and P. Noll, *Digital Coding of Waveforms – Principles and Applications to Speech and Video*. New Jersey: Prentice-Hall, 1984.

EXTRACTION AND REMOVAL OF PERCUSSIVE SOUNDS FROM MUSICAL RECORDINGS

John Usher

Sound Recording Area
McGill University, Montreal, Canada
jusher@po-box.mcgill.ca

ABSTRACT

Automated removal and extraction (isolation) of percussive sounds embedded in an audio signal is useful for a variety of applications such as speech enhancement and for music processing effects. A novel method is presented to accomplish both extraction and removal of beats, using an adaptive filter based on the LMS algorithm. Empirical evaluation is undertaken using computer generated music with a mix of natural voice and repeating drum, and shows that the efficacy of the system is robust to different sound processing techniques such as non-linear distortion and tempo jitter.

1. INTRODUCTION

Extraction and removal of percussive sounds from musical recordings are two different things; the former requires that only the beats be left after the audio signal processing, and that the user of such a “beat-extractor” wishes the extracted beats to be undistorted relative to those beats used to create the original musical piece. Beat removal, on the other hand, implies that the user is not concerned whether the beats themselves are recoverable from the audio mix. In the latter case, it is the non-percussive sounds which are of interest. The motivations for either objective are varied. An example is as an effect for reproduction of pre-recorded music, such as live DJ performances where a percussive beat from one musical piece can be mixed with the non-beat component of a second piece. Another use is for audio engineering to “fix” a recording which has already been mixed to two-channels, yet the engineer wishes to keep only the drum part of the mix, or everything *BUT* the drum part. A third use of such a beat extracting or removal device is for music production using samples from prerecorded music; a very common phenomenon in modern popular music.

2. SYSTEM OVERVIEW

The system architecture for the new beat extractor/ remover is described in figure 1. The musical input signal is first processed by a rhythmic feature analyser. The function of this device is to extract timing data about percussive events in the musical piece, classifying the sound into onsets and time intervals. Analysis of beat for music with strong percussive content relies on finding the local maximum in the power spectra envelope [1, 2], and can be used with measures of either auto-correlation [1] or outputs from resonant filters [3] to determine inter onset interval. A hierarchical representation of rhythmic patterns can be extracted to give the basic pulse-rate (the *tactus*) and higher structural levels such as bar-length [4]. However, the exploratory discussion of the system

presented here is concerned only with identification of the basic (foot-tapping) pulse, which in the foregoing experiment was identified manually from a single channel of computer-generated music.

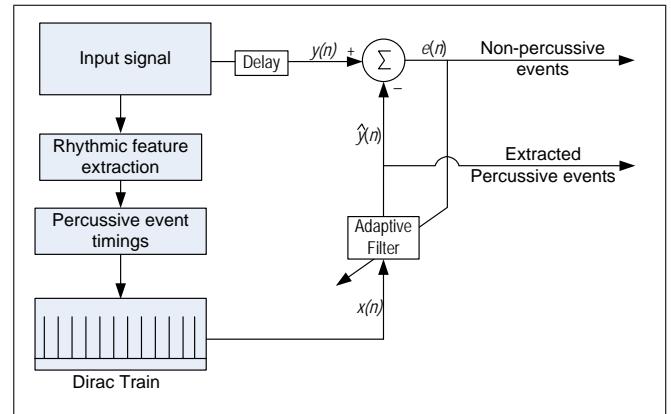


Figure 1: System overview.

Once the basic pulse has been identified, a scaled dirac train is created which is defined as a single sample value of unity at the start of each beat. The delay (τ) on the input signal (i.e. to create signal $y(n)$) ensures that the onset time is not critical; the onset estimate could be up to τ samples earlier than the actual onset time: the adaptive filter can compensate for such timing discrepancies with non-minimum phase filter coefficients. Such inaccuracy may be caused by a noisy energy-envelope threshold-based system, or a “true-event” time which occurs between samples. Effect of such timing jitter in the estimation of onset time on system performance is investigated later.

The basic goal of the system is to filter the dirac train so as to minimize the difference signal (measured as the mean-square energy difference) between it and the input audio signal; hence the adaptive filter coefficients are updated according to the Least Means Square (LMS) algorithm [5]. The premise of this is that the percussive musical sounds which occur at the dirac event times are caused by the same musical source. This is valid for computer-generated music such as techno, though to ensure that the timings for the dirac train correspond to percussive events from the same instrument an analysis of the local percussive event must be undertaken; feature extraction methods for accomplishing this, such as using harmonic analysis, are suggested in [6]. A major assumption is that time-varying non-percussive events (such as voice) will be

different on each identified transient, so the filter will not adapt to model these and the filtered dirac train (signal $\hat{y}(n)$) will resemble a train of just the percussive sounds.

The convolution of the dirac train signal $u(n)$ with the M -length adaptive filter \mathbf{h} gives signal $\hat{y}(n)$:

$$\begin{aligned}\hat{y}(n) &= \sum_{k=0}^{M-1} x(n-k)h_k \\ &= \mathbf{x}^T(n)\mathbf{h}.\end{aligned}\quad (1)$$

Where:

$$\begin{aligned}\mathbf{x}(n) &= [x(n), x(n-1), \dots, x(n-M+1)]^T, \\ \mathbf{h} &= [h_0, h_1, \dots, h_{M-1}]^T.\end{aligned}$$

It is this filtered dirac signal $\hat{y}(n)$ which approximates the beat occurring at the same time as the dirac impulse; hence this signal can be considered the “extracted” percussive events.

The delayed input audio signal $y(n)$ is then subtracted from the filtered dirac train $\hat{y}(n)$ to give the error signal $e(n)$:

$$e(n) = y(n) - \hat{y}(n). \quad (2)$$

If the adaptive filter coefficients match the actual percussive event at the time of the dirac pulse, then the percussive event would be totally canceled from the original input signal. Hence, under such optimal filter conditions the error signal can be considered to have the percussive events removed (i.e. the percussive events which have the same onset time as the dirac pulse).

The adaptive filter is adjusted over time so as to decrease the error signal level. This goal is formally expressed as a “performance index” or “cost” scaler J , where for a given filter vector \mathbf{h} :

$$J(\mathbf{h}) = E \{ e^2(n) \}, \quad (3)$$

and $E \{ \cdot \}$ is the statistical expectation operator. The requirement for the algorithm is to determine the operating conditions for which J attains its minimum value. This state of the adaptive filter is called the “optimal state” [5].

When a filter is in the optimal state, the rate of change in the error signal level (i.e. J) with respect to the filter coefficients \mathbf{h} will be minimal. This rate of change (or gradient operator) is an M -length vector ∇ , and applying it to the cost function J gives:

$$\nabla J(\mathbf{h}) = \frac{\partial J(\mathbf{h})}{\partial \mathbf{h}(n)}. \quad (4)$$

The right-hand-side of the last equations are expanded using partial derivatives in terms of the error signal $e(n)$ from equation (3):

$$\frac{\partial J(\mathbf{h})}{\partial \mathbf{h}(n)} = 2E \left\{ \frac{\partial e(n)}{\partial \mathbf{h}(n)} e(n) \right\}. \quad (5)$$

Updating the filter vector \mathbf{h} from time sample $(n-1)$ to time (n) is done by multiplying the negative of the gradient operator by a constant scaler and the filter update (i.e. the steepest descent gradient algorithm) is:

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \frac{\alpha}{\delta + \mathbf{x}^T(n)\mathbf{x}(n)} \mathbf{x}(n)e(n) \quad (6)$$

with

$$0 < \alpha < 2.$$

δ is a regularization constant to ensure against computational errors when the power estimate of the input signal is too low (this update version is called the Normalized LMS algorithm [5]).

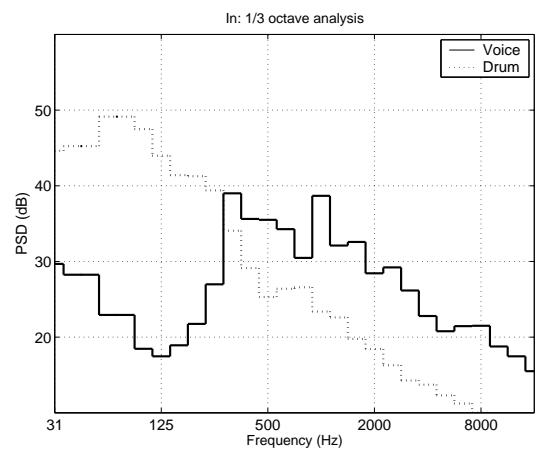
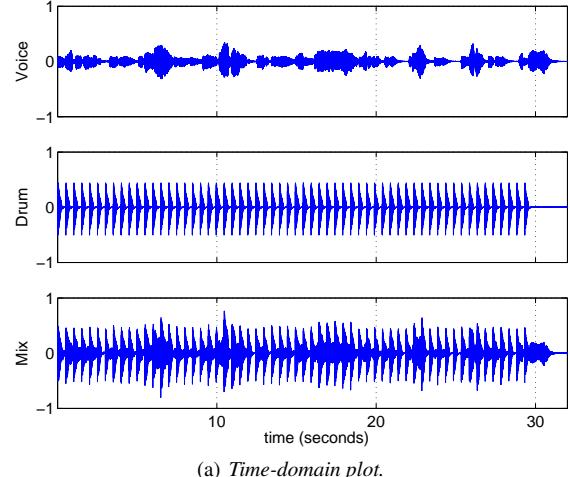


Figure 2: Details of input test stimulus. The mono mix used in the simulations was created by the summation of a voice and a drum track. The drum beat was created electronically from a single sample of a kick drum repeated at regular intervals.

Besides the massive increase in computational efficiency of implementing the filter-update and signal filtering in the frequency domain (requiring 5 FFT's per iteration; i.e. for every M input samples), the performance of the frequency-domain and time-domain NLMS algorithm are equivalent [7]. The overlap-save technique was used (as described in [7]) with an overlap factor of two (performance was not significantly affected by an increase in overlap). In the filter update, the time-domain constraint (to ensure against “wrap-around” errors when M is less than the length of the actual impulse response) was affected so as to weight later coefficients less than early ones; a modification known as the “exponential step” (ES) algorithm [8]. This ensures an exponential decay of the extracted beat. Furthermore, a parallel multi-filter approach was implemented whereby three simultaneous filters ran

with different update (α) parameters [9]; this allowed for fast initial convergence (subjectively, performance stabilized after 1 or 2 iterations) and robustness to sudden changes in envelope from new sounds.

3. ELECTRONIC VALIDATION

3.1. Test stimuli

For this exploratory investigation of the proposed system, a simple test stimulus was created by the summation of a sung voice and kick-drum train, as shown in figure 2. The drum sound was a typical decaying techno bass-drum sample (14000 samples long), and was repeated at intervals of 21900 samples (i.e. 121 beats per minute). The voice tempo was synchronized with the drum. As can be seen from the spectral analysis, the two musical instruments overlap in frequency (within 5-10 dB) for nearly two octaves centred about 500 Hz.

To investigate the effect of non-linear distortion on the system response, two common audio-processing techniques were applied to the mono mix; compression and reverberation. The dynamic compression algorithm applied an increasing gain to low-level sounds with a 2:1 ratio (quite an extreme case of compression), with a 20 ms attack time and 40 ms release time. Reverberation was artificially simulated using a commercially available processor with a reverberation time of 2.6 seconds. The RMS-averaged energy for the processed stimuli were matched with the original mono mix. It was expected that any non-linear distortion would reduce the efficacy of the system, creating a mismatch between the adaptive filter and the time-variant percussive sound (i.e. the optimal filter condition would also be time variant).

Furthermore, to simulate the effect of inaccuracies in the pulse event timing analysis system, the true beat-onset time (which was, of course, known *a priori* when the beat-track was created) was randomized by adding a gaussian-shaped noise process with a mean of zero and a variance of 0-100 samples (a kind of timing jitter).

3.2. System output response

As can be seen from the time-domain signal outputs in figure 3, the extracted drum beat and voice signal were very similar to the original input signals. Subjectively, the extracted bass-drum was distortion free after a single iteration. However, the extracted voice signal (or rather, the input signal with the drum-beat removed) had noticeable distortion artifacts. This was partly due to the exponential window smoothing which forced the adaptive filter coefficients to decay to zero faster than the actual decay of the drum beat (compare the drum envelope in the lower two plots of figure 3). Distortion was also noticeable at repeating intervals related to the block size length caused by high energy narrow-band resonances in the voice. Further work is needed to “fine-tune” the algorithm so that the power-estimate analysis used in the filter update has a memory which can account for such sudden resonances.

Considering an optimal solution set of filter coefficients \mathbf{h} and a current set of filter coefficients $\hat{\mathbf{h}}$ then the magnitude of the difference or mismatch between the two can be expressed as a simple dimensionless quantity ξ called the *misalignment* [10]:

$$\xi = \frac{\|\mathbf{h} - \hat{\mathbf{h}}\|}{\|\mathbf{h}\|}, \quad (7)$$

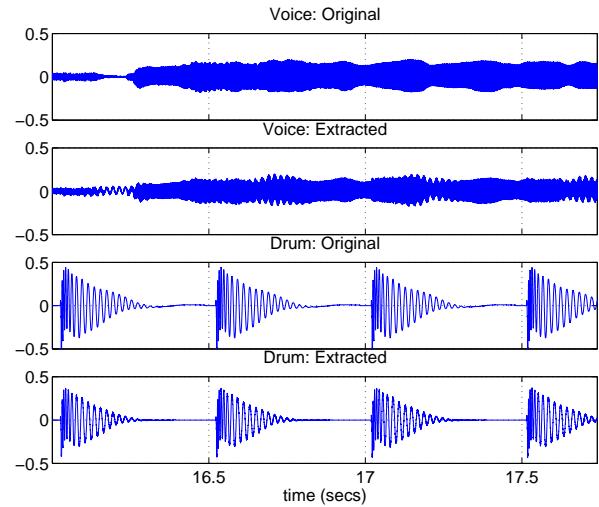


Figure 3: 1.8 second snippet of original voice and drum input signals and extracted output signals.

where $\|\cdot\|$ denotes the two-norm of a vector. In the new system, \mathbf{h} is really the actual percussive event (i.e. the optimal solution) and $\hat{\mathbf{h}}$ is the adaptive filter coefficients (i.e. the approximated beat). In this study, the optimal solution is known *a priori*- they are the individual beats used to create the drum channel.

As can be seen in figure 4, the non-linear processing of the original mono mix signal reduced the system performance (as measured in terms of misalignment). This was expected, though it should be noted that the subjective degradation in performance was not noticeable for the case when artificial reverb was added, and that for the processing involving compression the extracted drum channel was also subjectively undistorted.

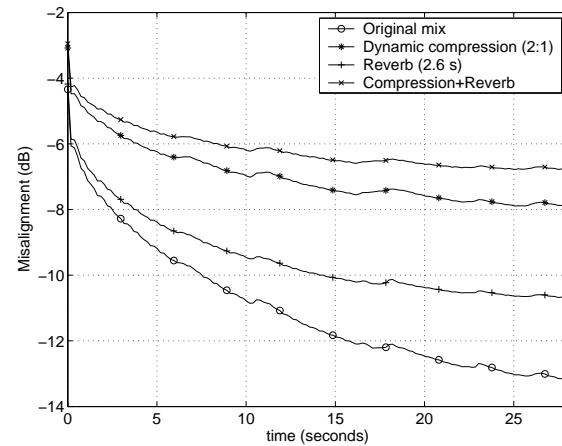


Figure 4: Misalignment for different non-linear audio processing techniques applied to the original mono mix.

Figure 5 shows the degree of robustness of the system to a jitter in the estimated beat-onset detector. For a given variance σ , 16%

of the event timings (i.e. the sample time of a given dirac pulse) will be later or earlier than the true event time by σ samples. Subjectively, the extracted drum and voice signals were undistorted if this variance was below 50 samples. This is still a relatively low tolerance (a 1/16th note inter-event for a 120 BPM beat is 1300 samples), and to accomplish such a necessary beat-tracking accuracy for live (rather than computer-generated) music would be difficult.

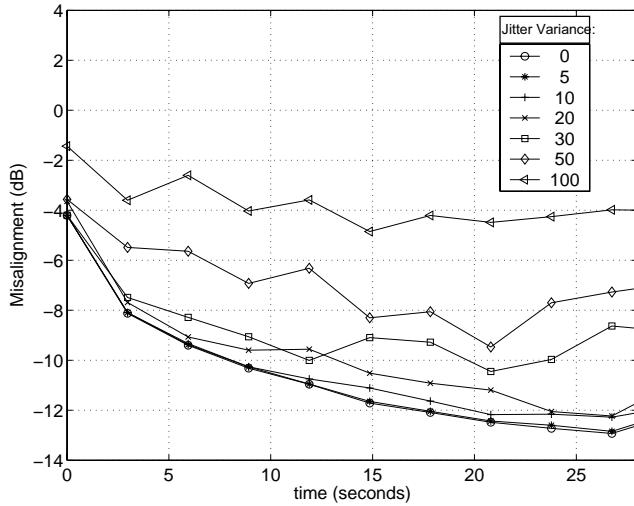


Figure 5: Effect of event timing jitter on misalignment. Different curves show magnitude of jitter: the variance (in samples) for a normal distribution centred about the actual event timing.

4. CONCLUSION

A novel application of adaptive filters for the purpose of extraction and removal of percussive sounds from music recordings has been presented. The proposed system relies on a rhythmic analysis device which extracts event timings for the percussive sounds and creates a dirac train with the pulse located at the percussive event onset time. The dirac train is then filtered with an adaptive filter updated according to the NLMS algorithm and the filtered signal approximates the percussive event. The system was tested with a computer-generated musical signal created from a repeating bass-drum and a sung voice. Subjective audition and electronic

measurement of the extracted drum signal shows very promising effectiveness. However, distortion artifacts are larger in the non-percussive audio output channel which suggests the system is better suited for beat extraction rather than beat removal.

5. REFERENCES

- [1] M. Goto, "An audio-based real-time beat tracking system for music with or without drum sounds," *J. New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.
- [2] C. Duxbury, M. Davies, and M. Sandler, "Separation of transient information in musical audio using multiresolution analysis techniques," in *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, 2001.
- [3] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals," *J. Acoust. Soc. Am.*, vol. 103, no. 1, pp. 588–601, 1998.
- [4] C. Uhle and J. Herre, "Estimation of tempo, micro time and time signature from percussive music," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003.
- [5] S. Haykin, *Adaptive Filter Theory*, 4th ed. Englewood Cliffs, N.J.: Prentice Hall, 2001.
- [6] E. D. Scheirer, "Music-listening systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2000.
- [7] P. C. W. Sommen, P. J. Van Gerwen, H. J. Kotmans, and A. J. E. M. Janssen, "Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function," *IEEE Trans. Circuits and Systems*, vol. 34, no. 7, pp. 788–798, 1987.
- [8] S. Makino and Y. Kaneda, "Acoustic echo canceller algorithm based on the variation characteristics of a room impulse response," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, 1990, pp. 1133–1136.
- [9] J. Usher, J. Cooperstock, and W. Woszczyk, "A multi-filter approach to acoustic echo cancellation for teleconferencing," in *Proc. 147th Meeting Acoust. Soc. Am.*, New York, USA, 2004.
- [10] J. Benesty, T. Gänslер, and P. Eneroth, "Multi-channel sound, acoustic echo cancellation, and multi-channel time-domain adaptive filtering," in *Acoustic Signal Processing for Telecommunication*, S. L. Gay and J. Benesty, Eds. Kluwer Academic Publishers, 2000, ch. 6, pp. 101–120.

REPRESENTATIONS OF AUDIO SIGNALS IN OVERCOMPLETE DICTIONARIES: WHAT IS THE LINK BETWEEN REDUNDANCY FACTOR AND CODING PROPERTIES?

Emmanuel Ravelli, Laurent Daudet

Laboratoire d'Acoustique Musicale
Université Pierre et Marie Curie (Paris 6)
11 rue de Lourmel, 75015 Paris, France
{ravelli|daudet}@lam.jussieu.fr

ABSTRACT

This paper addresses the link between the size of the dictionary in overcomplete decompositions of signals and the rate-distortion properties when such decompositions are used for audio coding. We have performed several experiments with sets of nested dictionaries showing that very redundant shift-invariant and multi-scale dictionaries have a clear benefit at low bit-rates ; however for very low distortion a lot of atoms have to be encoded, in these cases orthogonal transforms such as the MDCT give better results.

1. INTRODUCTION

In the past few years, research has been very active in the area of sparse signal representation in overcomplete dictionaries [1, 2, 3]. These techniques have been found to give successful results for a lot of signal processing problems, including source separation, denoising, detection, classification and compression.

Coding is one of the great application of interest, in particular very low bit-rate image coding (see for ex. the work from EPFL signal processing institute [4]). However, comparable work for the compression of audio signals is quite limited. Matching-pursuit [1] based techniques are used for sinusoidal modeling of audio signals and parametric audio coding in e.g. [5, 6], but the analysis-synthesis is performed on a frame-by-frame basis and thus does not exploit the long-term structure of audio signals. Other works, including [7] consider an audio signal as a whole and represent it using an overcomplete dictionary composed by cosine and wavelet functions, exploiting the structure of the coefficients; however, the described hybrid audio coder is rudimentary, and relies only on a weakly redundant dictionary.

In this paper, we also perform a global analysis of the signal, using a matching-pursuit based algorithm, and study the performance of a coder based on such sparse representation techniques. Particularly, we investigate the link between the degree of redundancy of the dictionary and the Rate-Distortion (R-D) performance of the coder. This is done by using nested dictionaries, with different nesting strategies, and studying the obtained rate-distortion curves on various artificial and natural signals.

The remaining of the paper is as follows: in section 2, we present sparse approximation techniques and how they relate to coding; in section 3, we describe our audio coder; and finally in section 4, we present experiments and results.

2. AUDIO CODING AND SPARSE REPRESENTATIONS

A signal $x \in \mathbb{R}^N$ is decomposed as a weighted sum of functions $g_\gamma \in \mathbb{R}^N$ which form the set of functions $\mathcal{D} = \{g_\gamma, \gamma \in \Gamma\}$.

$$x = \sum_{\gamma \in \Gamma} \alpha_\gamma g_\gamma \quad (1)$$

The ensemble \mathcal{D} is called a dictionary and the functions g_γ are called atoms. The representation is exact-sparse (resp. approximate-sparse) if a large number of the coefficients α_γ are zeros (resp. approximately zeros) i.e. the energy of the signal is concentrated on a small number of coefficients.

In the case when \mathcal{D} has the same dimension as the signal and the functions g_γ form an orthogonal base of \mathbb{R}^N , the decomposition is unique and equivalent to an orthogonal transform. In state-of-the-art audio coders (e.g. AAC, [8]), an orthogonal transform based on local cosine functions is used, the Modified Discrete Cosine Transform (MDCT). The atoms corresponding to the MDCT transform of a signal of length $N = PL$ and a frame size of $2L$, are defined as:

$$x_{k,p}(n) = w(n-pL) \cos\left[\frac{\pi}{L}\left(n-pL + \frac{L+1}{2}\right)\left(k + \frac{1}{2}\right)\right] \quad (2)$$

with $n = 0, \dots, N - 1$, $k = 0, \dots, L - 1$ and $p = 0, \dots, P - 1$. w is a window which is complementary in energy i.e. verifies:

$$w^2(n) + w^2(n + L) = 1, \quad n = 0, \dots, L - 1 \quad (3)$$

However, when the dimension of \mathcal{D} is superior to the dimension of the signal, i.e. when the dictionary is overcomplete, the decomposition is not unique anymore. Hence, one can choose amongst all these decompositions one which is optimal or nearly-optimal with respect to some pre-defined criteria. Several algorithms with different complexities have been proposed in the literature to find such decompositions (see e.g. [1, 2, 3]). We use in our case the matching pursuit algorithm [1], which is a fast sub-optimal iterative algorithm. At each iteration, Matching Pursuit chooses the atom in the dictionary most correlated with the signal, subtracts it, and iterates until some stopping condition is met.

A drawback of the MDCT for the representation of audio is that it does not carry explicitly phase information. Alternatively, one can use the Modulated Complex Lapped Transform (MCLT [9]), which is a complex extension of the MDCT that has the property of phase-invariance [10], at the cost of a $2\times$ increase in the number of (real) coefficients (i.e. a $2\times$ overcompleteness). In this case, both atoms and coefficients are complex ; consequently the standard matching pursuit cannot be used; instead, we use a two-dimensional matching pursuit using the projection of the signal in

the subspace composed by the complex atoms and their conjugates as described in [11].

There are two ways to further increase the redundancy of our dictionary. First, we use a generalized MDCT/MCLT transform where the resolution in frequency is increased. Second, we also use overcomplete dictionaries composed by an union of several transforms with different frame sizes. In a given experiment, in order to ensure a meaningful comparison on the influence of the redundancy factor on the R-D performance, we ensure that the dictionaries are always nested, i.e. lowest-redundant dictionaries are always included in highest-redundant ones.

3. OVERVIEW OF OUR AUDIO CODER

3.1. Analysis

We have used a fast implementation of the matching pursuit algorithm : the Matching Pursuit ToolKit (MPTK [12]). Thirteen dictionaries are tested: standard MDCT with a frame size of 2048 samples, standard MCLT with a frame size of 2048, four dictionaries composed by a generalized MCLT with a frame size of 2048 and a FFT size of respectively 4096, 6144 and 8192 frequency bins; seven dictionaries composed by a union of respectively 2,...,8 standard MDCT with frame sizes 128, 256, 512, 1024, 2048, 4096, 8192, 16384 samples.

3.2. Quantization and entropy coding

For real coefficients (MDCT), the DPCM-based quantization scheme as described in [13] is used. For complex coefficients (MCLT), an extended version of this scheme which quantizes the phase with Unrestricted Polar Quantization (UPQ, [14]) is used.

Adaptive arithmetic coding [15] is used to encode the output of the quantizers and the indexes of the coefficients in the dictionary.

4. EXPERIMENTAL RESULTS

We compare the Rate-Distortion (R-D) curves for different sets of dictionaries (for each figure, Rate is in Kbps and Distortion in dB). First we compare the standard MDCT with the standard MCLT; then, we study the influence of the frequency resolution for a generalized MCLT; and finally, several dictionaries using a concatenation of MDCT with different scales are tested. The signals used for the experiments are: a white noise; a synthetic signal of bell composed by a sum of damped sinusoids; and a real signal of a pop music recording (from MPEG SQAM test database).

4.1. MDCT vs. MCLT

The first idea is to use the MCLT which can be seen as a $2 \times$ overcomplete dictionary. This redundant dictionary needs fewer atoms than the orthogonal MDCT to reach the same target SNR; and thus it needs fewer bits to code the indexes and the norm of the coefficients. However, with the MCLT, there is an additional number of bits needed to code the phase information where the MDCT needs only one bit for the sign of each coefficient. Consequently, the additional number of bits needed by the phase counterbalances the bits gained by the smaller number of atoms; and thus, depending on the number of atoms selected by the coder (and thus the target bit rate), the MCLT gives better or worse distortion-rate performance than the MDCT.

Figures 1 to 3 compare the R-D curves for the MDCT and the MCLT obtained with the three test signals. As expected, the

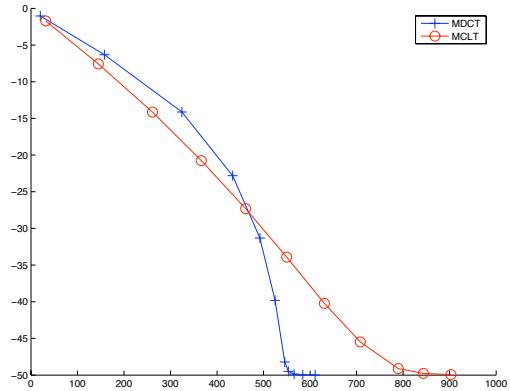


Figure 1: R-D curve for white noise: MDCT vs MCLT (*x*: rate in Kbps; *y*: distortion in dB).

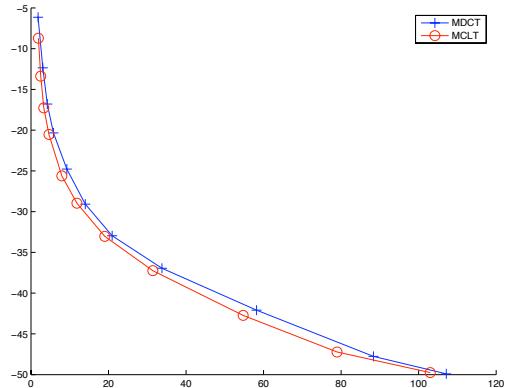


Figure 2: R-D curve for synthetic bell signal: MDCT vs. MCLT (*x*: rate in Kbps; *y*: distortion in dB).

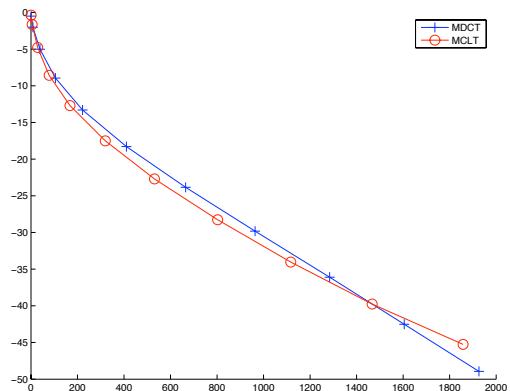


Figure 3: R-D curve for pop music: MDCT vs. MCLT (*x*: rate in Kbps; *y*: distortion in dB).

MDCT performs better at high rates whereas the MCLT gives better performance at low rates (this crossover would appear on fig. 2 at a highest rate than displayed; this is due to the fact that this signal is very tonal by nature).

4.2. Influence of frequency oversampling for the MCLT

The second idea is to increase the redundancy of the MCLT dictionary by increasing the size of the FFT i.e. increasing the precision for the frequency localization of the atoms. In the following figures, the standard MCLT uses the same size for the FFT and the frame, 2048 samples. For the generalized MCLT, the FFT is zero-padded using respectively 4096, 6144 and 8192 samples resulting a degree of redundancy of respectively 4, 6, 8.

By doing that, we hope to better estimate the sinusoidal components of the signal. As an audio signal can be modeled simply by a sum of sinusoids, such dictionaries should give better rate-distortion performance. However, the following figures show that the gain due to the better representation is lost due to the increased number of bits needed to code the indexes. And thus, for any rates, a simple MCLT performs better than the generalized MCLT.

4.3. Influence of the scale for the MDCT

The last idea is to use an overcomplete dictionary composed by a concatenation of several MDCT with different scales. The size of the frames are 128, 256, 512, 1024, 2048, 4096, 8192, 16384 samples. The nested dictionaries are obtained by adding one scale alternatively higher and lower, i.e.

$$\mathcal{D}_1 = \{mdct(2048)\},$$

$$\mathcal{D}_2 = \{mdct(1024)\} \cup \{mdct(2048)\},$$

$$\mathcal{D}_3 = \{mdct(1024)\} \cup \{mdct(2048)\} \cup \{mdct(4096)\}, \text{ etc.}$$

Figs. 7 to 9 show the R-D curves obtained with the three test signals. These show that at high rate MDCT gives better rate-distortion performance whereas at low rate the most redundant dictionary performs better. However, it also shows there is no compromise between an orthogonal transform and a highly redundant dictionary at mid rate.

5. CONCLUSIONS

This study attempts at clarifying the role of overcompleteness in sparse representations of audio signals, in the framework of audio coding. We have shown that, at high bitrates, the orthogonal MDCT is always better in terms of rate-distortion. However, at low bitrates, introducing redundancy improves the R-D performance. Indeed, one can see parametric coders (or more generally sinusoidal models) as very redundant systems with strong signal priors. Furthermore, our findings suggest that increasing the number of scales in the dictionary is a more efficient strategy than increasing the precision of the frequency (e.g. with zero-padding).

In order to broaden these conclusions, these preliminary tests have now to be conducted on a much wider variety of sounds: although general trends are usually similar, details of the R-D behavior is very signal-dependent. Furthermore, it would be desirable to use perceptual measures of distortion instead of the simple quadratic error. Indeed, at crossover points when the MDCT wins over redundant transforms the SNR usually reaches around 30-40 dB, which is the same order of magnitude as the SNR of MDCT-based coders operating at “transparent” qualities. Through such studies, our ultimate goal is to offer a single paradigm for audio coding, encompassing transform and parametric coding.

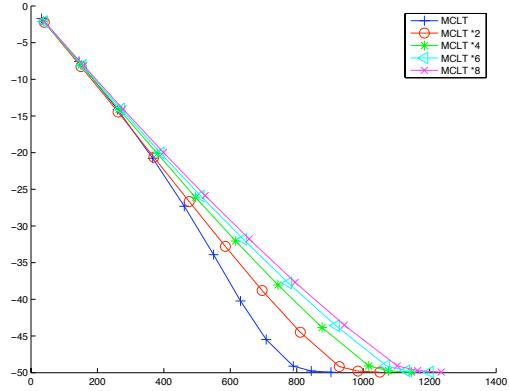


Figure 4: R-D curve for white noise: influence of the frequency resolution for the MCLT (x: rate in Kbps; y: distortion in dB).

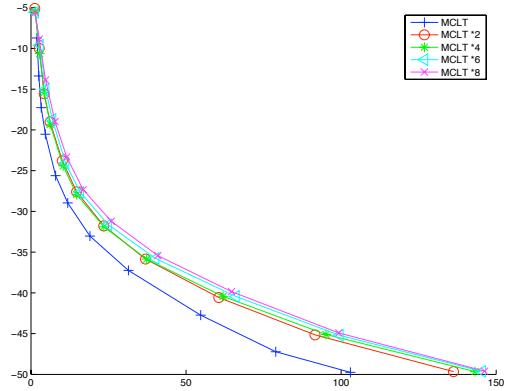


Figure 5: R-D curve for synthetic bell signal: influence of the frequency resolution for the MCLT (x: rate in Kbps; y: distortion in dB).

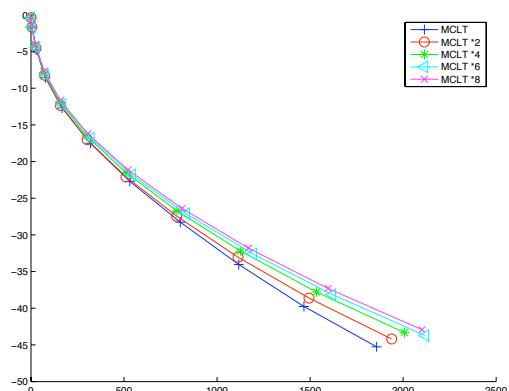


Figure 6: R-D curve for pop music: influence of the frequency resolution for the MCLT (x: rate in Kbps; y: distortion in dB).

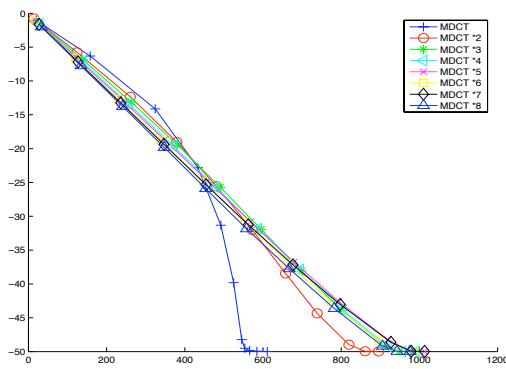


Figure 7: R-D curve for white noise: influence of the scale for the MDCT (x : rate in Kbps; y : distortion in dB).

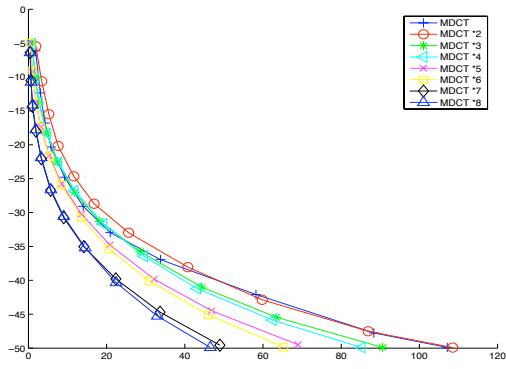


Figure 8: R-D curve for synthetic bell signal: influence of the scale for the MDCT (x : rate in Kbps; y : distortion in dB).

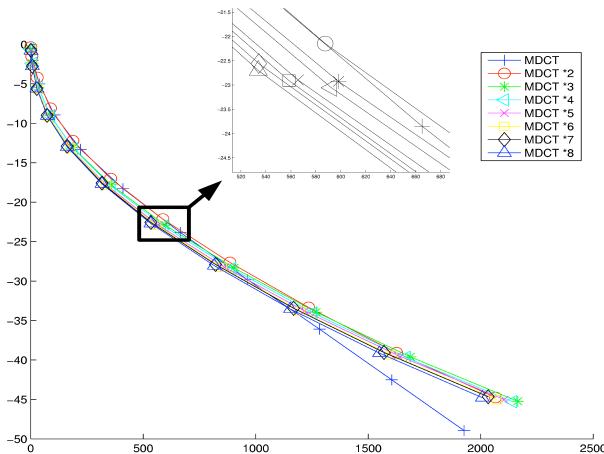


Figure 9: R-D curve for pop music: influence of the scale for the MDCT (x : rate in Kbps; y : distortion in dB).

6. ACKNOWLEDGEMENTS

The authors wish to thank the METISS team at IRISA for their support in the use and development of the MPTK project.

7. REFERENCES

- [1] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Sig. Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.
- [2] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1999.
- [3] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Information Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.
- [4] R. Ventura i Figueras, P. Vandergheynst, and P. Frossard, "Low rate and flexible image coding with redundant representations," *IEEE Trans. Image Processing*, vol. 15, no. 3, pp. 726–739, Mar. 2006.
- [5] T. Verma, "A perceptually based audio signal model with application to scalable audio compression," Ph.D. dissertation, Stanford University, 2000.
- [6] R. Vafin, "Towards flexible coding," Ph.D. dissertation, KTH Stockholm, 2004.
- [7] L. Daudet, S. Molla, and B. Torrésani, "Towards a hybrid audio coder," in *Proc. Third Int. Conf. on Wavelet Analysis and Applications*, 2004, pp. 13–24.
- [8] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa, "Iso/iec mpeg-2 advanced audio coding," in *101st Conv. Audio Eng. Soc.*, Los Angeles, USA, 1996, pp. 789–814.
- [9] H. Malvar, "A modulated complex lapped transform and its applications to audio processing," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'99)*, Phoenix, USA, vol. 3, 1999, pp. 1421–1424.
- [10] M. Davies and L. Daudet, "Sparse audio representations using the MC LT," *Signal Processing*, vol. 86, no. 3, 2006.
- [11] M. Goodwin, "Matching pursuit with damped sinusoids," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'97)*, Munich, Germany, vol. 3, 1997, pp. 2037–2040.
- [12] S. Krstulovic and R. Gribonval, "GNU/GPL C++ Software, the matching pursuit toolkit," Retrieved June 29th, 2006, [Online] <http://gforge.inria.fr/projects/mptk/>.
- [13] P. Frossard, P. Vandergheynst, R. Ventura i Figueras, and M. Kunt, "A posteriori quantization of progressive matching pursuit streams," *IEEE Trans. Sig. Proc.*, vol. 52, no. 2, pp. 525–535, Feb. 2004.
- [14] S. G. Wilson, "Magnitude/phase quantization of independent Gaussian variates," *IEEE Trans. Communication*, vol. COM-28, no. 11, pp. 1924–1929, 1980.
- [15] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Trans. Inf. Syst.*, vol. 16, no. 3, pp. 256–294, 1998.

A SPATIAL INTERFACE FOR AUDIO AND MUSIC PRODUCTION

Mike Wozniewski, Zack Settel

La Société Des Arts Technologiques
Montréal, Québec, Canada
{mikewoz|zack}@sat.qc.ca

Jeremy R. Cooperstock

McGill University Centre for Intelligent Machines
Montréal, Québec, Canada
jer@cim.mcgill.ca

ABSTRACT

In an effort to find a better suited interface for musical performance, a novel approach has been discovered and developed. At the heart of this approach is the concept of physical interaction with sound in space, where sound processing occurs at various 3-D locations and sending sound signals from one area to another is based on physical models of sound propagation. The control is based on a gestural vocabulary that is familiar to users, involving natural spatial interaction such as translating, rotating, and pointing in 3-D. This research presents a framework to deal with real-time control of 3-D audio, and describes how to construct audio scenes to accomplish various musical tasks. The generality and effectiveness of this approach has enabled us to re-implement several conventional applications, with the benefit of a substantially more powerful interface, and has further led to the conceptualization of several novel applications.

1. INTRODUCTION

Even in mainstream applications, where MIDI keyboards provide control of digital synthesis and effects can be added in real-time, the interface between performer and computer remains a limiting factor. In experimental applications, where a performer playing on stage may want to accomplish more and more with the aid of computer technology, conventional audio techniques fail. Though the equipment (audio mixers, synthesizers, effects units) can perform the required signal processing, interaction with it is very poorly adapted to the performing musician who wishes to control multiple effects and processors while playing other instruments. Throwing additional knobs, pedals, buttons, or sliders at this problem only worsens the situation because the task is at variance with the interface. In our approach, we abandon conventional audio control devices, and focus rather on the natural gestures and movements a performer might make.

We seek to capitalize on the great understanding of physical phenomena that humans acquire when dealing with the natural environment. For example, kinesthetic feedback from muscles will inform a person about the locations of their limbs, and the innate understanding of sound propagation allows them to localize sounds and infer various spatial characteristics of the world. We propose an paradigm for interaction with sound that utilizes this understanding. In previous work [1], we have described a framework for immersive spatial audio performance, where a user's bodies are modelled within a virtual 3-D world, and the propagation of audio is computed based on acoustic physical modeling. This framework is among only a few (e.g. [2, 3]) that have explored virtual environments from the perspective of music or digital signal processing (DSP). Ultimately we have created a tool with which artists can create interactive virtual scenes that are controlled by

regular spatial activity such as moving, turning, pointing, or grabbing. Thus, for example, a listener's experience of music in this environment will change as he/she moves or turns.

It is also possible to augment the perceptually accurate model of sound propagation to achieve results that are interesting for artistic or musical purposes. For example, effects of Distance or Doppler shift can be diminished or exaggerated and sound can be radiated or captured with a very narrow focus so that users can choose exactly how sound travels within their scenes.

Users can thus construct a *musical situation* (e.g. performing, listening, both) that takes the form of realistic scene, with similar geometry and behaviour to that of the real world. The ability to control signal processing with natural body motion and modified physical models obviously results in new metaphors for DSP design and the potential for novel sonic applications. In the following discussion, we will review our framework and discuss how to construct various audio scenes. Several sample applications are provided that illustrate the novel uses of this paradigm.

2. THE SPATIAL AUDIO FRAMEWORK

At the heart of our research is something we call the *soundEngine*, which handles the computation of virtual audio. This is a 3-D engine built using OpenSceneGraph to manage the arrangement of objects in space, and efficiently control various geometric transformations performed on the objects. PureData is used as the front-end interface and a method to manage signal processing and audio hardware. The engine allows for the dynamic creation, modification, and control of custom-made virtual audio scenes.

An audio scene for that matter, is composed of sound processing entities called *soundNodes*, which exist at some 3-D location and have various parameters to aid in DSP computation. A *soundConnection* is made between pairs of soundNodes, defining how sound propagates through the space between them. These soundNodes and soundConnections can respectively be thought of as the nodes and edges of a *DSP graph*, which is a convenient way to visualize the processing chain. We will expand on these topics in the following sections, though readers requiring further details can refer to our previous work [1].

2.1. The soundNode

The soundNode is the fundamental building block of a virtual audio scene, and can be either a *source* (which emits sound), a *sink* (which absorbs sound), or *both* of these at the same time. The case where a soundNode represents both is particularly important in the context of musical interaction since this is how signal processing is realized.

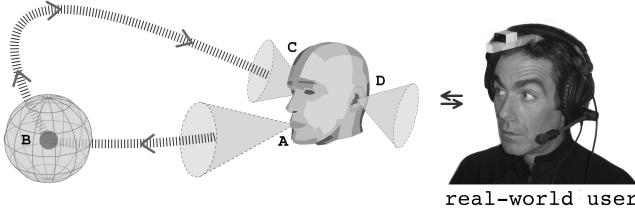


Figure 1: Example of how to model a simple scene with soundNodes.

This type of signal processing is illustrated in Figure 1. A source node *A* emits sound into the virtual world. Node *B* absorbs some of this sound, applies DSP, and emits the signal back into the scene, thus acting as both a sink and source. Nodes *C* and *D* are both simple sink nodes which collect the resulting signal and pass it to loudspeakers in the real world.

We note that source nodes in our representation are quite similar to those found in many other sound spatialization systems. They typically represent input from microphones, sound files, or sound synthesis. The concept of sink nodes is however a little more unique. Most interactive systems only render the audio scene for one listener at one position in space, called the “sweet spot”. There has been some recent work to synthesize sound at arbitrary spatial positions using virtual microphones [4, 5], yet this tends to focus on resynthesis of pre-recorded materials and not real-time performance. In our representation, a listener is just a particular type of sink node (or group of sinks) at a given location in space, where the audio absorbed is written to hardware buffers. Sink nodes are however used for many other purposes as well. Most importantly, they allow for the spatially localized DSP processing described above, but in addition, they allow for the simultaneous rendering of an audio scene at various locations. This can be used to produce surround recordings, or for multiple listeners who wish to occupy the environment at the same time.

Each soundNode entity also has other interesting parameters that users might wish to control. It can be translated/rotated in space, and the directivity (or roll-off) can be adjusted in real time. The roll-off is a parametric function that describes the attenuation that should be applied to a signal at different angles of propagation. Arbitrary functions can be defined, including those that are commonly found in acoustics and sound recording equipment (e.g., cardioids). However, the ability to focus sound with a tighter or wider directivity can be far more interesting than those common cases. Such ability provides the user with highly accurate control over a particular sound signal, and where it can travel. In the end, users can create focused “beams” of sound that they can aim at various soundNodes, which become virtual effects units that process audio at some location in space. A user can thus mix among them by focusing sound in different directions.

2.2. The soundConnection

In our representation, each pair of soundNodes that can exchange audio, must have a *soundConnection* established between them. This is different to many conventional systems, where every sound source is rendered in the same fashion. The reason for this is that we are viewing the propagation of audio as a type of signal bus, similar perhaps to a patch cord that one would use when con-

nnecting audio equipment to accomplish various DSP tasks. Yet in addition to the logical connection made between the nodes, the *soundConnection* also describes *how* the sound should travel through the intermediate space. For example, whether it should be absorbed/scattered by air, delayed in time, have its intensity decay with distance, be filtered based on angular incidence, or obey Doppler shift.

Once a connection is established, a user can specify the intensity and character of each of these propagation features, allowing for various interesting musical effects. For example, a user preparing a scene rich with harmonies might like to prevent frequency shifting and preserve the tonal content of the music. This can easily be accomplished by removing the effects of distance decay and Doppler from a connection, which effectively “teleports” a sound signal between two distant nodes.

One important result of the *soundConnection* paradigm is that mixing is not controlled with sliders and knobs, but is instead determined by the spatial position and orientation of the soundNodes involved. Users are free to move and manipulate these in real-time using a variety of sensors and tracking mechanisms. When the nodes are far apart, it means that the signal will be attenuated more and that there will be a greater delay. By bringing nodes closer together, we will increase the gain just like a slider on a mixing board would do, but with a much more natural conceptual model. Likewise, by orienting a node directly towards a source, we would imagine that the full spectrum of sound gets propagated. Then as the source rotates, pointing further away from the source, the incidence filter will attenuate higher frequencies, thus simulating the diffraction of sound waves.

2.3. Audio Scene Description

The *soundConnections* allow a user to organize soundNodes into groups which accomplish a specific DSP task. These groups can be visualized as *DSP graphs*, where soundNodes correspond to the nodes of a graph and *soundConnections* form the edges. These graphs are directed, always starting from a source node and terminating at one or more sink nodes. They are also potentially cyclic, meaning that recursive filtering can easily be developed.

We note that a single soundNode only has the ability to provide monophonic signal processing. Polyphony is achieved by interconnecting several soundNodes, and copying signals between them. Mixing and real-time control over the DSP graph is accomplished by adjusting the parameters of all the various *soundConnections* and soundNodes involved. This may seem like a difficult task as graphs get larger and larger, yet recall that parameter adjustment is accomplished by regular spatial activity such as rotating and translating. Furthermore, we employ another graph structure to help propagate spatial updates to many nodes at once.

In addition to organizing soundNodes in a DSP graph, those same nodes are also held in a *scene graph*, which is a concept borrowed from the field of 3-D graphics. A scene graph is a tree-structured graph where spatial transformations applied to a node are automatically propagated to all children. Thus geometric relationships between soundNodes can easily be described. For example, if we wanted several nodes to move together then we would attach them all to some parent node that would act as a frame of reference. Whenever the parent is moved, all the children would move accordingly. If the parent rotates, then the children also rotate, but relative to the coordinate system of the parent.

The use of scene graphs can be illustrated with Figure 1 that

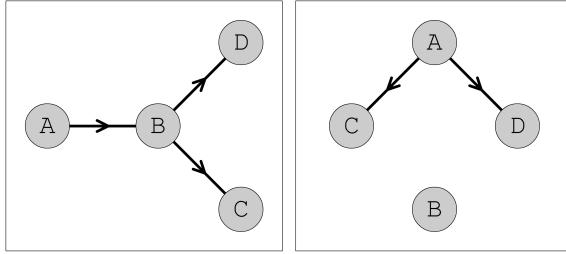


Figure 2: The DSP graph (left) and scene graph (right) associated with Figure 1.

we saw earlier. We see that nodes *A*, *C*, and *D* all share a common geometric reference. If the user's head moves or rotates, then all three of these nodes should move and rotate relative to the central head position. Figure 2 shows both the DSP graph and scene graph associated with the simple example presented in Figure 1. If the user rotates his head to speak in a different direction, the location of the sink nodes representing his ears will rotate accordingly. However, node *B* is not affected since it has no geometric relation to the user's head. Readers should understand that scene graphs have no bearing on the DSP chain. They are simply structures that organize the geometry of the scene.

3. APPLICATIONS

Our ability to simulate not only sound sources, but sound sinks, opens the door to a host of novel applications for directing and manipulating sound in 3-D. The flexibility of our propagation model provides us with a point of departure away from perceptually accurate models of 3-D audio. We may exaggerate or diminish certain acoustic properties to create novel experiences of sound / music for practical or artistic purposes. Of course, our approach lends well to creating applications that involve 3-D content. However, an obvious appeal of working with our framework is that some tasks that were difficult or impossible to do in a conventional audio production environment become easily accomplished in ours, allowing us to revisit conventional applications and to discover new ones. Below we describe several applications that have been built with our framework, and the new paradigms of interaction with sound that we have discovered along the way.

3.1. Active Listening

Since our framework intrinsically provides a model for "hearing", possibilities for *listening* are quite abundant. In the real world, sounds arrive at our ears from all directions, and it is up to the human perceptual system to differentiate and localize them. We are extremely good at this task, even being able to isolate a single conversation in a room full of people. However, the fact that our soundEngine framework allows us to bend the laws of physics means that we can narrowly "focus" hearing in a particular direction when needed: any sound that is not directly ahead would be severely attenuated. Additionally, the user could diminish the effect of distance attenuation so that distant sounds are heard more clearly. This ability to finely direct one's listening can lead to various novel applications, which we refer to these as *Active Listening* applications since they are based on the notion of user-directed listening. It is interesting to note that, given a high level of active

listening in a musically rich field of sound sources, notions of authorship and creation can enter into the picture when considering the listener's particular experience of the given soundscape: who is the composer?

Our active listening application provides a user with the ability to navigate within a space containing source-type soundNodes, each one radiating one track of a multi-track recording of an African percussion ensemble. As the listener moves and turns in the space, his/her musical experience constantly changes based on the proximity and orientation to the various soundNodes. By "rolling" his/her head the listening focus may be narrowed or widened, thus adding an additional element of listening control. Given the polyrhythmic nature of the music, the range of potential listener experiences is quite large. When listeners' sessions are recorded, great differences can be noted among the recordings, pointing to a potential authorship tool for creating remixes.

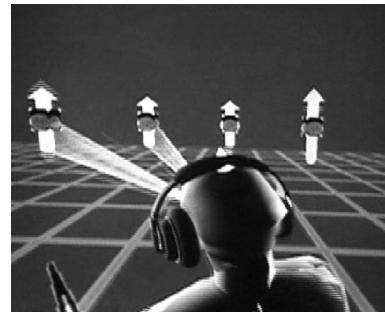


Figure 3: A screenshot from the Active Listening application. A graphical avatar moves and rotates according to a sensor worn by the user, thus controlling two tightly-focused sound sinks which can be aimed at various sources in 3-D space. The arrows attached to each loop indicate sound level, something like a VU meter.

3.2. Multichannel Audio Displays & Mix-downs

The ability to locate listeners in organized "fields" of sound sources lends extremely well to applications for audio mixing (mix-down). Unlike traditional interfaces for audio mixing, which usually involve panning sounds in 2D with some type of joystick interface, our interface intrinsically provides for mix-down using 3D spatial arrangement. In this case, the listener can be thought of as a *virtual microphone*, capturing sound sources in its proximity, and spooling them to disk. Thus, the quality of the mix is achieved by the organization of source sounds (tracks), and the placement of virtual microphones among them.

The particular format of the mix (5.1, stereo, etc.) is determined by the number of virtual microphones. For example, a 5.1 mix-down uses an array of six virtual microphones that collect the audio scene at some location in space. These microphones (sink soundNodes) are oriented to capture sound in their respective directions (e.g. the centre microphone would have 0° rotation, the right microphone would have 30°, the right rear would have 110°, etc.). It is worth noting that unlike conventional surround mixing environments, which localize sound in a horizontal plane (pantophonic), our environment's virtual microphones filter sounds based on their angle of incidence, offering fully 3D (periphonic) audio reproduction. Additionally, a given virtual microphone array can be itinerant, and move dynamically through the "recording space"

during mix-down, thus opening up additional artistic possibilities to the mixing engineer.

3.3. 3D Audio Dipping

The technique of creating music using samples and loops of existing material has become quite popular. It usually involves gear such as turntables, samplers, and sequencers that are triggered using keys, knobs, or sliders. This control paradigm is quite effective, allowing a single performer to produce very intricate musical material. The user simply ensures that all of the devices are synchronized (perhaps by some external clock signal or by simply ensuring that all material has the same tempo). Then using sliders for example, he or she can fade in one or more of the signals.

This style of control can also apply to material generated by computer music processes or synthesizers. Wessel & Wright [6] have defined this technique as “Dipping”, where the musical processes are silent by default, but the performer can “dip” in with some controller to make the processes heard.

Similar to Active Listening, the 3D Dipping application is particularly successful deployed in our framework. We start by placing source soundNodes around the user, ideally so that each node is equidistant. The sound contained in the nodes could be a number of phase-locked loops to ensure synchronization. The performer then focuses a sink soundNode in the direction of the sound he/she wishes to dip into (capture). For example, a 6 DOF sensor, mapped to the position and orientation of a sink node could be coupled to each of the performers hands. Using an additional parameter such as the roll (or twist) of the hand, the user could also tighten/spread the focus so that fewer/more nodes can be dipped into.

Additionally, by performing an editing gesture (via glove controller, modal button, etc.), the performer could relocate and swap soundNodes during performance, thus extending musical range. Such ability could prove quite useful, since the performer could group certain nodes together in space so that they are always activated together. Nodes could also be moved closer or further away which effectively controls the gain, since their emitted sound decays with distance.

3.4. Spatial Routing to DSP Effects

Effects routing is well-known in live electro-acoustic performance circles, and particularly well-suited to re-implementation in our framework. Unlike the above applications, this one focuses on the performer as a modeled human source node surrounded by many sink nodes present in the scene that collect sound from him/her. To make them easy to target, sink soundNodes can be placed in a configuration which corresponds to locations of real objects in the performance space, such as a corner, clock, pillar, person etc.. Thus, we may equip a saxophone with an orientation/position sensor and a wireless microphone. We then define a certain number of sink nodes, each with a particular effects processing unit, such as reverb, delay, flanging, ring modulation etc.. When the sax player points the instrument in a certain direction, its sound is proportionally radiated to one or more effects nodes in space according to the propagation model between them. By “rolling” the horn, its radiation is altered, fanning out to reach more effects nodes, or the inverse. Of particular interest to performers of live electronics, *dangerous dsp*, such as recursive flangers, can be included among the effects nodes, since feedback can only occur when the input is aimed directly at it. In fact, the incredibly high degree of

control that the performer has in sending his/her signal to a node allows for riding an extremely fine edge between stable and unstable DSP. Live sound technicians are particularly appreciative of this capability.

4. CONCLUSIONS

The organization of DSP processors in 3-D space and using physically modelled sound propagation as a signal bus results in a new approach to musical applications. We have described a framework for controlling 3-D audio by creating scenes composed of soundNodes and soundConnections. These scenes can further be analyzed in terms of their DSP graphs and scene graphs, which provide convenient tools for conceptualizing and manipulating parameters. We have shown how several conventional and new applications can be conceived using this paradigm, and how novel approaches can be taken to provide existing tasks with more natural control. In the end, we believe that the conceptual model provided to the user will lighten the cognitive load required to deal with complicated musical situations. The fact that we build on the existing knowledge that humans have of spatial relationships gives us reason to believe that this interaction paradigm has great potential.

5. ACKNOWLEDGEMENTS

The authors wish to acknowledge the generous support of NSERC and Canada Council for the Arts, which have funded the research and artistic development described in this paper through their New Media Initiative. In addition, the authors are most grateful for the resources provided to the project by La Société Des Arts Technologiques and to the Banff Centre for the Arts for offering a productive and stimulating environment in which several of the applications described here were prototyped.

6. REFERENCES

- [1] M. Wozniewski, Z. Settel, and J. Cooperstock, “A framework for immersive spatial audio performance,” in *Int. Conf. New Interf. for Musical Expr. (NIME’06)*, Paris, France, 2006, pp. 11–16.
- [2] J. Pressing, “Some perspectives on performed sound and music in virtual environments,” *Presence*, vol. 6, no. 4, pp. 482–503, 1997.
- [3] T. Maki-Patola, J. Laitinen, A. Kanerva, and T. Takala, “Experiments with virtual reality instruments,” in *Int. Conf. New Interf. for Musical Expr. (NIME’05)*, Vancouver, Canada, 2005, pp. 11–16.
- [4] J. Braasch, “A loudspeaker-based 3d sound projection using virtual microphone control (ViMiC),” in *118th Conv. Audio Eng. Soc.*, Barcelona, Spain, 2005, pp. 968–979.
- [5] A. Mouchtaris, S. S. Narayanan, and C. Kyriakakis, “Virtual microphones for multichannel audio resynthesis,” *EURASIP J. on Applied Sig. Proc.*, vol. 2003, pp. 968–979, 2003.
- [6] D. Wessel and M. Wright, “Problems and prospects for intimate musical control of computers,” *Computer Music J.*, vol. 26, no. 3, pp. 11–22, 2002. [Online]. Available: citeseer.ist.psu.edu/wessel00problems.html

STREAMING FREQUENCY-DOMAIN DAFX IN CSOUND 5

Victor Lazzarini

Music Technology Laboratory
National University of Ireland, Maynooth
victor.lazzarini@nuim.ie

Joe Timoney, Tom Lysaght

National University of Ireland, Maynooth
Department of Computer Science
{Joseph.Timoney|Tom.Lysaght}@cs.nuim.ie

ABSTRACT

This article discusses the implementation of frequency domain digital audio effects using the Csound 5 music programming language, with its streaming frequency-domain signal (fsig) framework. Introduced to Csound 4.13, by Richard Dobson, it was further extended by Victor Lazzarini in version 5. The latest release of Csound incorporates a variety of new opcodes for different types of spectral manipulations. This article introduces the fsig framework and the analysis and resynthesis unit generators. It describes in detail the different types of spectral DAFX made possible by these new opcodes.

1. INTRODUCTION

Csound 5 [1], released in February 2006, is a completely re-coded version of the popular MUSIC *N*-derived music programming language [2]. Among the many new features, it provides completely new host and module APIs, for embedding and extending the system, several new frontends and scripting language support, as well as numerous new opcodes, bringing the total number of unit generators to over 1000. These include a set of opcodes designed to work with spectral signals, defined by the Csound orchestra language type *fsig*.

This signal type [3] was introduced by Richard Dobson to Csound 4.13, together with a few basic opcodes. It provides a framework for spectral processing, which was further extended by Victor Lazzarini (in Csound 5) to support partial track signals. Previously, spectral processing in Csound was limited to transformation and resynthesis of spectral datafiles. With the fsig framework, a Csound instrument can manipulate any input signal in the frequency domain. The opcodes process streaming spectral signal, which is generated by an analysis (or a data file reader) opcode. The frequency-domain signal can be resynthesised using inverse-DFT overlap-add or additive synthesis.

2. THE FSIG FRAMEWORK

Streaming frequency-domain signals are defined by the Csound fsig type. Such signals are processed at a rate that is dependent on the size of the DFT analysis frame and the number of overlapping frames (or the hopsize), effectively the rate of generation of new spectral frames. The ‘perform’-method of a spectral processing opcode is called every control period, but it only outputs a new frame if there is a new frame at its input. The fsig framework provides support for such checks. Consequently, the fsig rate is independent of the control rate and the processing is more efficient than the original datafile-based opcodes, which were tied to the orchestra control rate. Fsig are self-describing, so, unlike

time-domain audio and control signals, are furnished with the extra information about their feature, including: DFT length, number of overlaps (*N/hopsize*), window size, window type and data format.

2.1. Data Formats

The actual format of the spectral data can vary, currently three types are being used: PVS_AMP_FREQ, amplitude and frequency pairs as produced by the phase vocoder and IFD; PVS_AMP_PHASE, amplitude and phase (polar DFT) data; and PVS_TRACKS, partial tracks of amplitude, frequency, phase and track ID [4]. Of these, the first two will have a fixed size, namely the DFT size plus two extra values (holding the positive spectrum plus the Nyquist frequency, generated by the DFT of a real signal), or $N/2 + 1$ bins. These two formats will be henceforth referred to as ‘bin-frame’ data.

The PVS_TRACKS signal actually holds frames of variable size, but ultimately having a maximum number of tracks equivalent to the number of analysis bins. The partial tracks will contain four items each; however not all partial track-processing opcodes will require all of them (the phase can be sometimes omitted). Crucial to its operation is the track ID information, as it is used to match tracks at consecutive frames. It is possible to introduce other types of binframe spectral signals, for instance, data in rectangular (real, imaginary) format. Nevertheless, the musical generality of the PVS_AMP_FREQ has so far fulfilled the needs of most bin-frame processing applications.

3. SPECTRAL ANALYSIS

Streaming spectral data can be generated by three opcodes: the original pvsanal and pvsfread opcodes (written by Richard Dobson) plus the pvsifd, introduced in Csound 5. These unit generators produced data in bin-frame format, which can be further transformed into partial tracks, by the partials opcode.

3.1. Phase Vocoder

The pvsanal opcode, as well as by the pvanal Csound utility, which generates PVOCEX files for the pvsfread, perform phase vocoder analysis. They are loosely modelled on the original CARL phase vocoder [5]. The pvsanal opcode does this operation in a streaming fashion, generating a new frame every hopsize input samples. The following example takes the channel 1 input signal to Csound and produces a fsig with a framesize of 1024 (513 bins), updated every 256 samples, using a 1024-sample hanning window.

```
asig  inch      1
fs1   pvsanal  asig , 1024, 256, 1024, 1
```

4. INSTANTANEOUS FREQUENCY DISTRIBUTION

The pvsifd opcode implements the instantaneous frequency distribution analysis [6]. It actually generates two fsigs, one containing a PVS_AMP_FREQ signal, similar to the pvsanal output, and another containing a PVS_AMP_PHASE signal. This pair of fsigs can then be used for a full partial track analysis. The following example is equivalent to the previous one, except that the opcode generate two fsigs as output and uses an analysis window that is always the same size as the DFT.

```
asig    inch     1
fs1 , fs2  pvsifd  asig , 1024, 256, 1
```

4.1. Partial track analysis

The partials opcode takes in two fsigs, with PVS_AMP_FREQ and PVS_AMP_PHASE and does a partial track analysis, generating a PVS_TRACKS signal containing a variable number of partial tracks. Each track will model one partial of the input signal, with amplitude, frequency, phase and partial ID data. It is possible to use only a single PVS_AMP_FREQ input, in which case the phase information will be omitted from the analysis output. In fact, the majority of the track processing opcodes do not require phase information. It is possible to feed partial track analysis with the output of a pvsanal or pvsfread opcode. The first example outputs a complete (amplitude, frequency and phase) partial track signal, with an analysis threshold of 0.003 (0.3%), a minimum number of 1 time point for each track and a maximum gap of 3 time-points for track continuity. It limits the output to a maximum of 500 tracks:

```
asig    inch     1
fs1 , fs2  pvsifd  asig , 1024, 256, 1
ftrk    partials  fs1 , fs2 , 0.003 , 1,3,500
```

However, as previously discussed, it is also possible to feed a amplitude/frequency-only input to partial track analysis, resulting in tracks with no phase information. The following example takes its input from a PVOCEX analysis file, using the file reader opcode:

```
ktim  line      0, idur , idur
fs1   pvsfread  ktim , "input.pvx"
ftrk  partials  fs1 , fs1 , 0.003,1 ,3,500
```

5. SPECTRAL PROCESSING

In this section, we will look at the different types of transformations, loosely classified in amplitude, frequency and combination effects, as discussed in [7].

5.1. Amplitude transformations

The basic type of amplitude effects are filter-like processes, which will alter the amplitude functions, but leave the frequency (and phase) unaltered. Richard Dobson provided to Csound 4.13, a pvsmask opcode, which uses a function table of $N/2$ length as an amplitude response curve. The opcode multiplies each bin amplitude by a function table value, indexed by the bin number, effectively filtering the signal. In Csound 5, the trfilter opcode operates in a similar fashion, but process partial tracks instead of

bin frames, thus the length of the function table is not required to be fixed to any particular size. Time-varying filter effects can be implemented using a table writing opcode, as show in the example below, which implements a comb filter-like effect:

```
aphs phasor 1 /* table writing index */
/* sinusoid signal to be fed into the
table , kpks is number of spec peaks */
afil oscili 2, kpks/2, 1
/* table writing (table size=44100) */
tabw abs(afil), aphs, 3, 1
ffil trfilter ftrk , 1, 3 /* filtering */
```

In the example above, if the signal kpks is modulated, then a flanger-type effect will result. Time-varying filtering can also be implemented by two other Csound 5 opcodes, pvsfilter and pvsarp. The latter provides a spectral ‘arpeggiation’ effect by zeroing some bin amplitudes and boosting others. The former takes two bin-frame fsigs and uses one of them as an amplitude response, multiplying the two amplitude functions together. This opcode can be also used in some cross-synthesis effects as well as filtering. For instance, a very narrow band-pass filter can be created by using the spectra of a sinusoid and the spectra of an arbitrary source as inputs:

```
asig inch     1 /* input */
asin oscili 1, kcf, 1 /* sinusoid */
/* input spectral signal */
fs1 pvsanal asig , 1024, 256, 1024, 1
/* sinus spectral signal */
fs1 pvsanal asin , 1024, 256, 1024, 1
/* filtering */
ffil pvsfilter asig , asin , 1
```

Also in the category of amplitude transformations we have the mask-based effects, such as noise cancellation that can be performed by the pvstencil opcode. This takes an input signal and compares it, bin by bin with an amplitude response mask in a function table, performing amplitude scaling based on this comparison. For a denoiser type effect, it is possible to construct such a mask table from a PVOCEX file (using GEN43) and apply it to an input signal using this opcode:

```
asig diskin2 "input.wav", 1,0,1
ifn ftgen 1,0, isiz/2,-43, "noise.pvx"
fsig pvsanal asig , isiz , isiz/4,isiz ,iw
fclean pvstencil fsig , kattn , klvl , ifn
```

In the example above, the amount of amplitude change is controlled by kattn, and the noise threshold can also be adjusted by the klvl variable.

5.2. Frequency transformations

The basic frequency effects implemented for streaming spectral signals are frequency scaling and frequency shifting. These are available for both bin-frame (pvscale and pvshift) and partial track signals (trscale and trshift). Frequency scaling of binframe signals is described by the following expression:

$$f_{\text{out}}[np] = f_{\text{in}}[n]p \quad (1)$$

where f_{in} and f_{out} are the input and output bin frequencies, respectively, n is the bin index and p is the scaling interval. A simple harmoniser example is shown below:

```

asig in
/* spectral processing, iscl is the
harmoniser interval */
fsig pvsanal asig, isiz, isiz/4, isiz, iw
ftps pvscale fsig, iscl, ikeepform, igain
atps pvsynth ftps
/* there is a N-sample delay between input
and output */
adp delayr .1
adel deltapn ifftsize
delayw asig
out atps+adel

```

In order to compensate for the N-sample delay between the input of the spectral process and its output, a short delay line is used, so the original and transposed signal can be time-aligned. The bin-frame frequency scaler and shifter opcodes have two special modes of operation that will attempt to preserve formants, for vocal applications. If the `ikeepform` variable is 1 or 2, one of these modes will be used. The method of formant preservation used here is perhaps not as accurate as the one described in [8], but many times more efficient and yielding good results. A comparison of the output and input of `pvscale` using formant preservation with pitch shifting demonstrates that original formants are fairly well preserved in the frequency-scaled spectrum (fig. 1).

Frequency shifting adds a value to all frequencies in the input spectra, as defined by the following expression for bin-frame signals:

$$f_{\text{out}} \left[n + \frac{s}{bw} \right] = f_{\text{in}}[n] + s \quad (2)$$

where n is the bin index, s the frequency shift amount in Hz and bw is the bin bandwidth in Hz. This has the effect of destroying any harmonic relationships that might exist in an input sound. With partial track processing, it is possible to split the tracks into one or more frequency regions and apply such transformation to those regions, altering the timbre of an instrument, but not completely destroying its pitch impression. This is demonstrated in the example below:

```

/* split tracks at 1500 Hz */
ftrkd, ftrku trsplit ftrk, 1500
/* shift upper frequencies by 150 Hz */
fshft trshift ftrku, 150
/* combine split tracks */
ftmix trmix ftrkd, fshft

```

It is important to note that frequency scaling, as well as shifting, is slightly simpler with partial tracks, if compared to bin-frame signals. In fact, it only requires the scaling or shifting of the partial frequencies, with no need for the bin-reallocation implied in eqns. (1) and (2).

5.3. Cross-synthesis and other effects

A number of cross-synthesis effects are possible, from morphing by interpolation of bin values (`pvscross` by Richard Dobson), to channel vocoder-like amplitude substitution (`pvsvoc`) and partial track cross-synthesis (`trcross`). A special signal combination effect is also implemented by the `pvsdemix` opcode, which is loosely based on the reverse-panning ADReSS algorithm. This opcode takes two signals, the left and right channels of a stereo mix and separates the instruments in the mix according to their panning position. The example below demonstrates its use. In it, the `kpos`

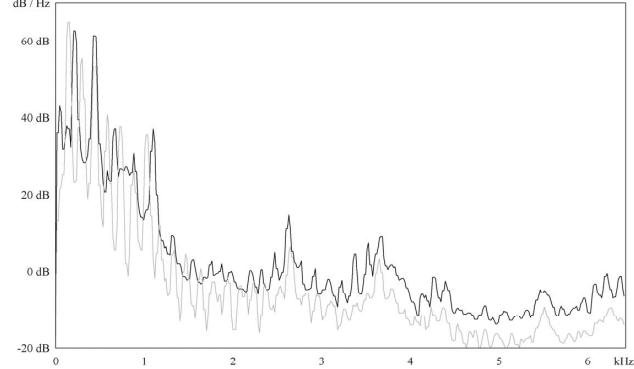


Figure 1: Comparison between original (grey) and transposed (black) spectra. A good amount of formant preservation is clearly seen in the picture.

variable controls the pan position for signal extraction (-1 to 1) and `kwidth` controls the ‘search width’ around that position. The `demix` operation works on the basis of 100 discrete pan positions on each side of the stereo image.

For partial track signals, there are a number of specialised opcodes that will manipulate and transform track data. As shown in a previous example, it is possible to split and mix tracks (`trsplit` and `trmix`), as well as isolate the highest and lowest-frequency tracks (`trhighest` and `trlowest`) and obtain their current frequency and amplitude values. It is also possible to realise some special effects, such as residual extraction, by combining original and track-resynthesis signals, in similar process to the one described in [9].

Another effect that involves the transformation of both frequency and amplitude is that of spectral blurring [10], based on time-averaging the amplitude and frequency spectral functions. The effect is implemented by the `pvsblur` opcode, which takes in a ‘blur time’ parameter, defining the averaging period.

6. RESYNTHESIS

6.1. Overlap-add

This is generally the most efficient way of resynthesising bin-frame amplitude-frequency data. It is performed by `pvsynth` opcode. This takes the amplitude and frequency pairs, integrates the frequencies to obtain the current phases, converts to rectangular data and applies an inverse DFT. The resulting time-domain signal block is then overlap-added to the correct time-aligned position at the output. Partial tracks cannot be fed directly to the overlap-add resynthesis, but can be converted into bin-frame data. This conversion is performed by the `binit` opcode, which generates a frame of amplitude and frequency bins based on the track data input.

6.2. Additive synthesis

Additive synthesis can be applied to bin-frame data or partial tracks, but, generally speaking, it is more suited to the latter. Richard Dobson contributed an additive resynthesis opcode, `pvsadd`, to the original set of `fsig` opcodes, which is reasonably

fast, but producing a medium to low quality (due to the lack of interpolation) resynthesis of bin-frame data.

Partial track additive synthesis can, however, be more efficient and offers better quality. There are three additive opcodes in Csound 5 for track data: using linear (`tradsyn`) and cubic phase interpolation (`sinsyn` and `resyn`). Of the three, `tradsyn` is the most efficient and flexible, as it depends only on amplitude and frequency track data. The cubic-phase opcodes will have better fidelity in signal reconstruction, but will be slower, and in the case of `sinsyn`, will not allow any type of frequency (or timescale) transformations of the original analysis data.

7. CONCLUSION AND FUTURE PROSPECTS

The fsig framework in Csound5 and its spectral opcodes provide a comprehensive, flexible and intuitive way to build frequency-domain effects computer instruments. It is expected that new opcodes will be added to the exiting set. Also, it is important to mention the work on a sliding DFT analysis/resynthesis method by ffitch and Dobson [11], which will eventually be incorporated into the system.

8. ACKNOWLEDGEMENTS

The authors would like to acknowledge the work of Richard Dobson in the development of the fsig framework and opcodes for Csound 4.13. It is also important to mention the contribution of John ffitch and Istvan Varga, among others, to the development of the Csound 5 infra-structure.

9. REFERENCES

- [1] J. ffitch, "On the design of Csound 5," in *Proc. 3rd Linux Audio Conf.*, 2005, pp. 37–42.
- [2] V. Lazzarini, "Extensions to the Csound language: from user-defined to plugin opcodes and beyond," in *Proc. of the 3rd Linux Audio Conf.*, 2005, pp. 13–20.
- [3] V. Lazzarini, J. Timoney, and T. Lysagh, "Alternative analysis-synthesis approaches for timescale, frequency and other transformations of musical signals," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005, pp. 18–23.
- [4] V. Lazzarini, J. Timoney, and T. Lysagh, "Time-stretching using the instantaneous frequency distribution and partial tracking," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 14–27.
- [5] V. Verfaillie and P. Depalle, "Adaptive effects based on STFT, using a source-filter model," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 296–301.
- [6] X. Rodet and A. Röbel, "Real time signal transposition with envelope preservation in the phase vocoder," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 672–675.
- [7] J. ffitch, R. Bradford, and R. Dobson, "Sliding is smoother than jumping," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 287–290.
- [8] M. Dolson, "The phase vocoder: a tutorial," *Computer Music J.*, vol. 10, no. 4, pp. 14–27, 1986.
- [9] B. Vercoe, *Csound: A Manual of the Audio Processing System*. MIT Media Lab, 1986.
- [10] T. Wishart, *Audible Design*. Orpheus the Pantomime, York, 1996.
- [11] X. Serra, *Musical Signal Processing*. G. D. Poli and A. Piciallli and S. T. Pope and C. Roads Eds. Swets & Zeitlinger, 1996, ch. Musical sound modeling with sinusoids plus noise, pp. 91–122.

REAL-TIME CORPUS-BASED CONCATENATIVE SYNTHESIS WITH CATART

Diemo Schwarz, Gr  gory Beller, Bruno Verbrugghe, Sam Britton

Ircam – Centre Pompidou, Paris, France

{schwarz|beller|verbrugg|britton}@ircam.fr

ABSTRACT

The concatenative real-time sound synthesis system *CataRT* plays grains from a large corpus of segmented and descriptor-analysed sounds according to proximity to a target position in the descriptor space. This can be seen as a content-based extension to granular synthesis providing direct access to specific sound characteristics. *CataRT* is implemented in *Max/MSP* using the FTM library and an SQL database. Segmentation and MPEG-7 descriptors are loaded from SDIF files or generated on-the-fly. *CataRT* allows to explore the corpus interactively or via a target sequencer, to resynthesise an audio file or live input with the source sounds, or to experiment with expressive speech synthesis and gestural control.

1. INTRODUCTION

Corpus-based concatenative synthesis methods are attracting more and more interest in the musical sound synthesis and content-based processing communities. They use a large database of source sounds, segmented into *units*, and a *unit selection* algorithm that finds the sequence of units that match best the sound or phrase to be synthesised, called the *target*.

The selection is performed according to the *descriptors* of the units, which are characteristics extracted from the source sounds, or higher level descriptors attributed to them. The selected units can then be transformed to fully match the target specification, and are concatenated.

These methods allow various applications, such as high level instrument synthesis, resynthesis of audio, also called *mosaicing*, texture and ambience synthesis, artistic speech synthesis, and interactive explorative synthesis in different variants, which is the main application of the *CataRT* synthesis system.

Explorative real-time synthesis from heterogeneous sound databases allows to exploit the richness of detail of recorded sound while retaining efficient control of the acoustic result by using perceptually meaningful descriptors to specify a target in the multi-dimensional descriptor space. If the selection happens in real-time, this allows to browse and explore a corpus of sounds interactively.

*CataRT*¹ is a collection of patches for *Max/MSP* using the FTM, *Gabor*, and MnM extensions². It is released as free open source software under the GNU general public license (GPL)

2. RELATED WORK

CataRT is based on the previous developments *Caterpillar* for non-real-time data-driven concatenative musical sound synthesis [1, 2], and *Talkapillar* for artistic text-to-speech synthesis, hybrid concatenation between music and speech, and descriptor-driven or context-sensitive voice effects [3]. In these systems, the unit selection algorithm is by Viterbi path-search, which finds the globally optimal sequence of database units that best match the given target

units using two cost functions: The *target cost* expresses the similarity of a target unit to the database units by weighted Euclidean distance, including a context around the target. The *concatenation cost* predicts the quality of the join of two database units by join-point continuity of selected descriptors.

Many other approaches to concatenative corpus-based synthesis appeared over the last few years. They are compared and classified in [4].

2.1. Real-Time Concatenative Synthesis

Contrary to the systems above, real-time concatenative synthesis systems can not provide a globally optimal selection, because the target is not known entirely in advance. Also, concatenation quality is rarely included in the selection algorithm. They fall into two groups: The first group matches low- or high-level descriptors of the corpus units to target descriptors (*MoSievius* [5], *Synful* [6], *Ringomatic* [7]). The second group performs a spectral match based on FFT-frames where the target is given by live audio input. (*Input Driven Resynthesis* [8], *Scrambled Hackz³*), or on spectro-temporal “sound lexemes” (*SoundSpotter* [9]).

2.2. Granular Synthesis

One source of inspiration of the present work is granular synthesis, which takes short snippets (*grains*) out of a sound file, at an arbitrary rate. These grains are played back with a possibly changed pitch, envelope, and volume. The position and length of the snippets are controlled interactively, allowing to scan through the soundfile, in any speed. Granular synthesis is rudimentarily corpus-based, considering that there is no analysis, the unit size is determined arbitrarily, and the selection is limited to choosing the position in one single sound file. However, its concept of exploring a sound interactively, when combined with a pre-analysis of the data and thus enriched by a targeted selection, results in a precise control over the output sound characteristics, as realised in *CataRT*.

3. MODEL

CataRT's model is a multidimensional space of descriptors, populated by the sound units. The user controls a target point in a lower-dimensional projection of that space with a selection radius around it, and the selection algorithm selects the units closest to the target or within the radius. The actual triggering of the unit is independent of the selection and can happen at any rate.

1. <http://concatenative.net>

2. <http://www.ircam.fr/ftm>

3. <http://www.popmodernism.org/scrambledhackz/>

3.1. Analysis

To get data into the corpus, either all data (audio, segment markers, raw descriptors) are loaded from preanalysed files, or the descriptors are analysed in *CataRT* and an additional onset detection stage segments the sound files. At this point, all analysis takes place inside *CataRT*, which means that we could just as well use real-time audio input that is segmented and analysed on the fly, to feed the corpus. The audio could come, for example, from a musician on stage, the last several minutes of whose playing constitutes the corpus from which a laptop improviser selects units.

Markers generated externally can be loaded from SDIF or ASCII files; internal segmentation calculation is either by arbitrary grain segmentation, by split according to silence (given a threshold), by high-frequency content, or by transient analysis. There is also a mode where imported sound files are taken as a whole, which is appropriate for sets of drum and percussion sounds.

Descriptors are either imported from precalculated MPEG-7 low-level descriptor files, or calculated in the patch. Details for the 230 imported MPEG-7 signal, perceptual, spectral, and harmonic descriptors can be found in [2], following the definitions from [10].

The descriptors calculated in the patch in batch mode, i.e. faster than real-time, thanks to *Gabor*'s event-based signal frame processing, are the fundamental frequency, aperiodicity, and loudness found by the *yin* algorithm [11], and a number of spectral descriptors from [12]: spectral centroid, sharpness, spectral flatness, high and mid frequency energy, high frequency content, first order autocorrelation coefficient (that expresses spectral tilt), and energy.

Note that also descriptors describing the units' segments themselves, such as the unit's unique id, the start and end time, its duration, and the soundfile it came from, are stored. It is convenient to duplicate this information as descriptors to make it available for selection.

The time-varying raw descriptors at FFT-frame rate have to be condensed to a fixed number of scalar values to characterise a unit. These *characteristic values* [2] express the general evolution over time of a descriptor with its mean value, variance, slope, curve, min, max, and range.

3.2. Data

Data is kept in FTM data structures such as tables, dictionaries, matrices. The unit descriptor data is kept in one big matrix with one column per descriptor and one unit per row. For persistent storage of corpora, a layer around the relational database management system *SQLite* keeps track of soundfiles, segments, and unit descriptor data. The Sound Description Interchange Format (SDIF)⁴ is used for well-defined interchange of data with external descriptor analysis and segmentation programs.

3.3. Selection

Because of the real-time orientation of *CataRT*, we cannot use the globally optimal path-search style unit selection based on a Viterbi algorithm as in *Caterpillar*, neither do we consider concatenation quality, for the moment. Instead, the selection is based on finding the units closest to the current position x in the descriptor space, in a geometric sense, i.e. on appropriately scaled dimensions: A straightforward way of achieving this is to calculate the Mahalanobis distance d between x and all units, i.e. the distance normalised by the standard deviation of each chosen descriptor over

the corpus. Either the unit with minimal d is selected, or one randomly chosen from the set of units with $d < r^2$, when a selection radius r is specified.

To improve the efficiency of selection, the units in the descriptor space are indexed by an optimised multi-dimensional k -nearest neighbour index. The algorithm described in [13] constructs a search tree by splitting up the descriptor space along the hyperplane perpendicular to the principal component vector, and thus achieving a maximal separation of units. This is then repeated for each sub-space until only a few units are left in each leaf node of the resulting tree. The k -nearest neighbour search can then, at each step down the tree, eliminate approximately half of the units, by just one distance calculation with the subspace boundary.

3.4. Synthesis

CataRT's synthesis applies a choosable short fade-in and fade-out to the sound data of a selected unit, which is then pasted into the output delay-line buffer, possibly with a random delay. Other manipulations similar to a granular synthesis engine can be applied: the copied length of the sound data can be arbitrarily changed (de facto falsifying the selection criteria) to achieve granular-style effects or clouds of overlapping grains. Also, changes in pitch by resampling and loudness changes are possible. Note that, because the actual pitch and loudness values of a unit are known in its descriptors, it is possible to specify precise pitch and loudness values that are to be met by the transformation.

However, granular synthesis is only one possible realisation of synthesis. Other components might store the sound data in spectral or additive sinusoidal representations for easier transformation and concatenation.

3.5. Interface

Because displaying and navigating in a high-dimensional space is not practical, the descriptor space is reduced to a 2-dimensional projection according to two selectable descriptors.

CataRT's user interface follows the model-view-controller (MVC) design principle. The view (figure 1) plots a 2-dimensional projection of the units in the descriptor space plus a 3rd descriptor being expressed on a colour scale. Note that that the display is dynamic, i.e. multiple views can be instantiated that can connect to the same data component, or one view can be switched between several data instances, i.e. corpora.

In these displays, the mouse serves to move the target point in the descriptor space. Additional control possibilities are MIDI input from fader boxes to set more than two descriptor target values and limit a selectable descriptor range, and advanced input devices for gestural control (see section 4.2).

Independent of the current position, several sources for triggering playing of the currently closest unit exist: an obvious but quite interesting mode plays a unit whenever the mouse moves. De-facto, the friction of the mouse provides an appropriate force-feedback, so that this mode is called *bow*. To avoid the strident repetitions of units, the mode *fence* plays a unit whenever a different unit becomes the closest one (named in homage to clattering a stick along a garden fence). The *beat* mode triggers units regularly via a metronome, and the *chain* mode triggers a new unit whenever the previous unit has finished playing.

4. <http://www.ircam.fr/sdif>

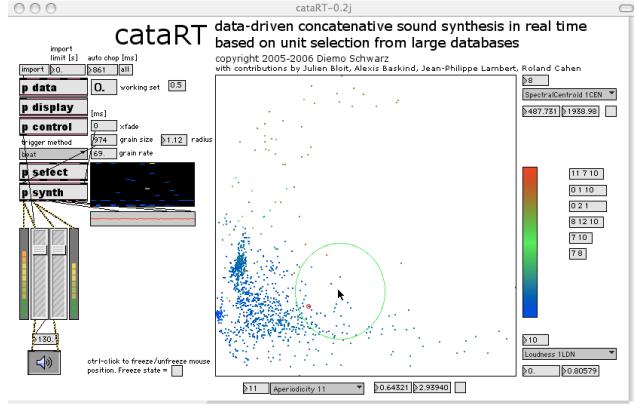


Figure 1: Screen shot of the explorative synthesis interface.

CataRT incorporates a basic loop-sequencer that allows to automate the target descriptor control (see figure 2). Also the evolution of the weight for a descriptor can be sequenced (second curve in figure 2), such that at the desired times, the target descriptor value is enforced, while at others the selection is less dependent on this descriptor.

4. APPLICATIONS

4.1. Explorative Granular Synthesis

The principal application of *CataRT* is the interactive explorative synthesis from a sound corpus, based on musically meaningful descriptors. Here, granular synthesis is extended by a targeted selection according to the content of the sound base. One could see this as abolishing the temporal dimension of a sound file, and navigating through it based on content alone.

Usually, the units group around several clusters. With corpora with mixed sources, such as environmental noises, voice, and synthetic sounds, interesting overlaps in the descriptor space occur and can be exploited. Figure 1 shows the *CataRT* main patch with an example of a corpus to explore.

4.2. Gesture-Controlled Synthesis

The present model of controlling the exploration of the sound space by mouse position is of course very basic and limited. That's why we're working on the integration of a general mapping layer for gestural control into *CataRT*. This allows more expressive musical interaction and tapping into more than just two dimensions by analysis of higher level gestural parameters from more advanced input devices such as graphics tablets. Clustering, rotation and warping of the descriptor space (by multi-dimensional scaling or magnifying-glass type transformations) maximises the efficiency of the interaction, leading to greater expressivity. Sub-spaces of units can be selected by navigation or according to the temporal evolution of the control gesture (attack, sustain, release phases can have their own sub-space to select units from). Note that each sub-space can have its own projection to the most pertinent descriptors therein.

4.3. Audio-Controlled Synthesis

As all the analysis can take place in *CataRT* itself, it is possible to analyse an incoming audio signal for descriptors and use these to control the synthesis, effectively resynthesising a signal with sounds from the database. The target descriptor evolution can also be derived from a sound file, by analysing and segmenting it and playing its controlling descriptors from the sequencer.

CataRT is used as a compositional and orchestration tool in the context of a piece for banjo and electronics by Sam Britton. The work takes large databases of recorded instrumental improvisations and uses concatenative synthesis to re-sequence and orchestrate these sequences. In this context, the concatenation process acts as a kind of oral catalyst, experimentally re-combining events into harmonic, melodic and timbral structures, simultaneously proposing novel combinations and evolutions of the source material.

4.4. Data-Driven Drumbox

A slightly more rigid variation of the *CataRT* sequencer splits the target descriptor sequence into a fixed number of beats, where, for each beat, one sound class can be chosen. The selection within each soundclass, however, is governed by a freely editable descriptor curve, or real-time control.

4.5. Expressive Speech Synthesis

Research in expressive speech synthesis [14] takes advantage of real-time content-based manipulation and synthesis. Special descriptors are used for the speech units, e.g. acoustic descriptors like formant frequencies and pitch and symbolic descriptors such as the phonemes and the grammar. Here, a sentence is set up as a target sequence of successive segments of speech sounds and their descriptors. We can then change descriptor values of the target sequence to catch other sound units (speech or other) from the database. Figure 2 shows the pitch and the phonemic transcription of a sequence of units corresponding to the utterance “au revoir”. It can be seen that we have graphically modified pitch around the temporal index shown by the cursor. All descriptors can be modified in such an easy way to change locally the selection of a unit within a sequence. Text can also be rewritten using the keyboard as in the example where we have delayed the last “R”.

There are two different ways to define temporal boundaries of the speech segments: The first is by segmental units like diphones, phones or semiphones as presented in figure 2, and leads to a small number of units per sentence which allows to use a many files to provide classical TTS with the capability to draw the prosody. The second is by pitch synchronous segmentation in periods of the speech signal. A large number of units per sentence are created which permits a fine grained selection. In this case, modifications of the descriptor sequence lead to a progressive morphing between spoken and singing synthesis, for instance. It is also possible to add jitter (perturbation of the pitch) to provide expressive speech synthesis.

5. FUTURE WORK

Except the obvious work on adding more descriptors and improving the visualisation, interesting questions of control and interaction are raised by the present work.

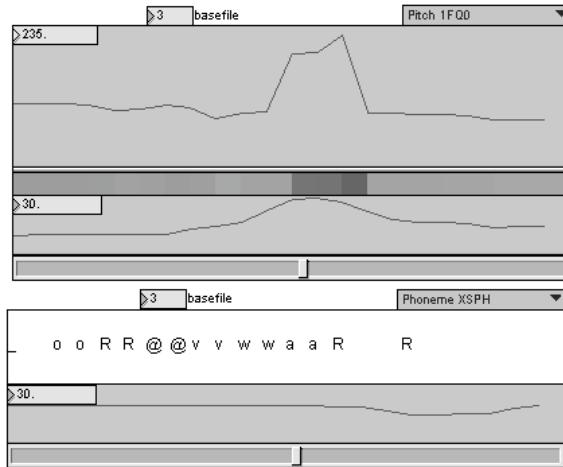


Figure 2: The CataRT descriptor sequencer.

The used corpora are in general unevenly distributed over the descriptor space. Many units are concentrated in clusters, whereas large parts of the space are relatively empty. This is first a problem of interaction and visualisation, which should allow zooming into a cluster to navigate through the fine differences within. However, the model of navigating through the descriptor space could be refined by a notion of subspaces with links to other subspaces. Note that, within different clusters, possibly different descriptors express best the intra-cluster variation.

CataRT should take care of concatenation, at least in a limited way, by considering the transition from the previously selected unit to the next one. The concatenation cost could be given by descriptor continuity constraints, spectral distance measures, or by a precalculated distance matrix, which would also allow distances to be applied to symbolic descriptors such as phoneme class. The concatenation distance could be derived from an analysis of the corpus.

It should be possible to exploit the data in the corpus to analyse the natural behaviour of an underlying instrument or sound generation process. By modeling the probabilities to go from one cluster of units to the next, we would favour the typical articulations of the corpus, or, the synthesis left running freely would generate a sequence of units that recreates the texture of the source sounds.

To make more existing sound collections available to *CataRT*, an interface to the *Caterpillar* database, to the *freesound* repository, and other sound databases is planned. The *freesound* project,⁵ is a collaboratively built up online database of samples under licensing terms less restrictive than the standard copyright, as provided by the *Creative Commons* family of licenses. A transparent net access from *CataRT* to this sound database, with its 170 unit descriptors already calculated, would give us an endless supply of fresh sound material.

6. CONCLUSION

We presented the *CataRT* system that implements a new model of interactive exploration of, and navigation through a sound corpus. The concatenative synthesis approach is a natural extension of granular synthesis, augmented by content-based selection and control, but keeping the richness of the source sounds. By its real-time analysis capabilities, the system can also perform context-based effects. The applications are only beginning to fathom all

the possibilities this model allows for interaction modes and visualisation. Because of their object-oriented software architecture, the *CataRT* components can serve as a toolbox to build individual applications.

7. REFERENCES

- [1] D. Schwarz, "A system for data-driven concatenative sound synthesis," in *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-00)*, Verona, Italy, 2000, pp. 97–102.
- [2] ———, "Data-driven concatenative sound synthesis," Thèse de doctorat, Université Paris 6 – Pierre et Marie Curie, Paris, 2004. [Online]. Available: <http://mediatheque.ircam.fr/articles/textes/Schwarz04a>
- [3] G. Beller, D. Schwarz, T. Hueber, and X. Rodet, "A hybrid concatenative synthesis system on the intersection of music and speech," in *Journées d'Informatique Musicale (JIM)*, MSH Paris Nord, St. Denis, France, June 2005, pp. 41–45.
- [4] D. Schwarz, "Concatenative sound synthesis: The early years," *J. New Music Research*, vol. 35, no. 1, pp. 3–22, Sept. 2006, special Issue on Audio Mosaicing.
- [5] A. Lazier and P. Cook, "MOSIEVIUS: Feature driven interactive audio mosaicing," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003, pp. 312–317.
- [6] E. Lindemann, "Musical synthesizer capable of expressive phrasing," US Patent 6,316,710, Nov. 2001.
- [7] J.-J. Aucouturier and F. Pachet, "Ringomatic: A real-time interactive drummer using constraint-satisfaction and drum sound descriptors," in *Proc. Int. Conf. Music Information Retrieval (ISMIR'05)*, London, UK, 2005, pp. 412–419.
- [8] M. Puckette, "Low-dimensional parameter mapping using spectral envelopes," in *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, 2004, pp. 406–408.
- [9] M. Casey, "Acoustic lexemes for organizing internet audio," *Contemporary Music Review*, vol. 24, no. 6, pp. 489–508, 2005.
- [10] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," Ircam – Centre Pompidou, Tech. Rep., 2004. [Online]. Available: http://www.ircam.fr/anasyneeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf
- [11] A. de Cheveigné and N. Henrich, "Fundamental frequency estimation of musical sounds," *J. Acoust. Soc. Am.*, vol. 111, no. 4, pp. 1917–1930, Apr. 2002.
- [12] J. Bloit, "Analyse temps réel de la voix pour le contrôle de synthèse audio," Master-2/SAR ATIAM, UPMC (Paris 6), Paris, 2005.
- [13] W. D'haes, D. van Dyck, and X. Rodet, "PCA-based branch and bound search algorithms for computing k nearest neighbors," *Pattern Recognition Letters*, vol. 24, no. 9–10, pp. 1437–1451, 2003.
- [14] G. Beller, T. Hueber, D. Schwarz, and X. Rodet, "Speech rates in french expressive speech," in *Speech Prosody*, Dresden, Germany, 2006, pp. 672–675. [Online]. Available: http://recherche.ircam.fr/equipes/analyse-synthese/beller/doc/pdf/Beller_hueber_schwarz_rodet_SP_2006_Speech_Rates_expressive_french.pdf

5. <http://iuafreesound.upf.es>

MULTICHANNEL SIGNAL REPRESENTATION IN PWGLSYNTH

Mikael Laurson, Vesa Norilo

Centre of Music and Technology
Sibelius Academy, Helsinki, Finland
{laurson|vnorilo}@siba.fi

ABSTRACT

This paper gives an overview of one of the most important features in our synthesis language called PWGLSynth. We will concentrate on how to represent visually multichannel signals in a synthesis patch. PWGLSynth synthesis boxes support vectored inputs and outputs. This scheme is useful as it allows to construct compound entities which are used often in sound synthesis such as banks, parallel structures, serial structures, etc. PWGLSynth provides a rich set of tools that allow to manipulate vectors. For instance vectors can mixed, modulated, merged, or split into sub-vectors.

1. INTRODUCTION

In sound synthesis it is essential to be able to represent complex flow of sound signals in a compact and comprehensible manner. One important abstraction mechanism is to combine several parallel signals into one multichannel signal that can be manipulated by the user as one entity. Thus it should be possible to create, initialize, delete, connect and disconnect a multichannel entity with one simple operation. Furthermore these operations should have a simple and uniform syntax that would allow to build complex synthesis patches that are both easy to understand and to maintain.

Although sound synthesis has already a fairly long history, synthesis environments that have appropriate multichannel abstraction mechanisms have only recently been appearing. Early textual implementations in the Music-N tradition have mostly dealt with mono signals. Some exceptional modules that have multichannel properties, such as a panning module, were typically found only at the output of an instrument definition. The lack of appropriate abstraction mechanisms obviously restricts severely the user in more complex cases. These restrictions are perhaps even more pronounced in visual synthesis environments resulting in patches that tend to become crowded and confusing when dealing with multichannel cases.

During the 90s the textual synthesis language SuperCollider [1] introduced a powerful "Multi channel expansion" property, which allows to use multichannel operations in a systematic way. This scheme resulted in an expressive system where patch definitions can be extremely compact and flexible.

This paper gives an introduction to our synthesis environment PWGLSynth [2][3] with an emphasis on the multichannel representation of signals. PWGLSynth is an integral part of our visual Lisp-based programming environment called PWGL [4] which is specialized in computer assisted composition, analysis and sound synthesis. Although our system has been influenced by SuperCollider our approach is quite different as we aim to represent patch definitions visually. The aim is to develop a system where by introducing appropriate abstraction mechanisms, such as multichannel

representation of signals, visual definitions can become as economic and compact as it is the case in some of the state-of-the-art textual synthesis languages.

The rest of this paper is organized as follows. First we introduce the main concepts in our visual system and give some examples on how to define patches that deal both with 'traditional' mono signals and vectored signals. Next we go over to a more detailed discussion and introduce some important vector manipulation modules. The final section shows how multichannel operations are used in conjunction with our copy-synth-patch scheme that has been used extensively in our work related with physics-based instrument models [3].

2. BASIC VISUAL ENTITIES

A PWGLSynth patch is a graph structure consisting of boxes and connections. Boxes, in turn, can be categorized in two main box types from a synthesis point of view. The first box type consists of ordinary PWGL boxes. These boxes can be found at the leaves of a synthesis patch and they are typically evaluated once before the synthesis patch is run. A special case of this category are sliders which can dynamically change the current value while the synthesis is running. The second box type consists of boxes, marked with an 'S', that represent synthesis boxes that are used for the actual sample calculation. 'S' boxes support vectored inputs and outputs. Mono signals are only a special case where the vector length is equal to 1. A synthesis patch always contains a special 'S' box, called 'synth-box', at the root of the graph, which represents the output of the sample calculation. This output can either be sent to audio converters in real-time, or the output can be written to a file.

Figure 1 shows a simple sine wave oscillator with vibrato control. The patch contains four sliders for real-time control and four 'S' boxes. This basic example operates only with mono signals.

The next patch example is more complex and it demonstrates how multichannel signals are represented in our system. The patch is based on vectored 'S' boxes (Figure 2). The patch gives also a visual clue that helps to distinguish between mono signals (vector length is equal to 1) and vectored signals. The connections between vectored boxes are drawn using a thicker line width and a stipple pattern that contains holes.

The vector *length* is specified by the inputs at the leaves (i.e. inputs which are not connected to a 'S' box) of the patch. These inputs can be Lisp expressions (typically lists) or slider-banks which allow a separate real-time control of each individual vector element. If the lengths at the inputs differ, then the shortest vector determines the current vector length. In Figure 2 the vector length is equal to 4, because the inputs of the 'sine-vector', 'impulse-vector', and 'reson-vector' contain lists or sliders with 4 elements.

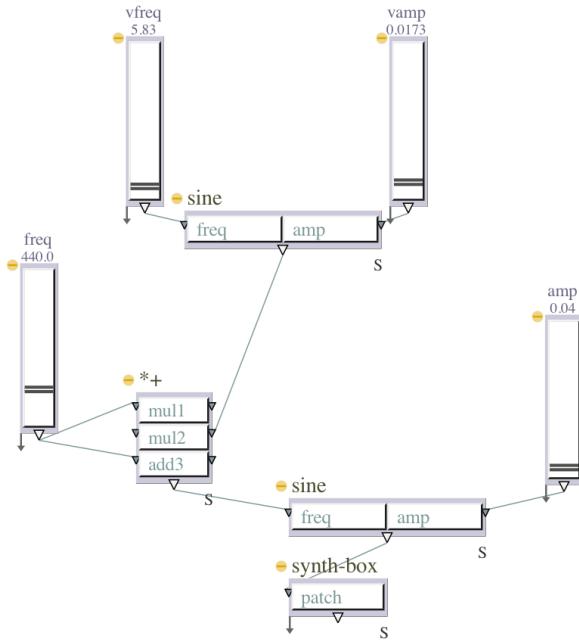


Figure 1: A simple mono signal patch with some real-time sliders.

Thus the patch in Figure 2 can be interpreted as follows. A bank of 4 resonators (“reson-vector”) is excited with a bank of 4 impulse generators (“impulse-vector”). The frequency input of the ‘impulse-vector’ box is of special interest as it allows to control in real-time the frequency of each impulse generator individually. The other inputs of the ‘reson-vector’ box (i.e. frequencies, amplitudes, and bandwidths) are given as static Lisp lists, each containing 4 elements. The output of the ‘reson-vector’ box is connected to a ‘accum’ box that accepts as input any vectored signal and mixes it to a mono signal (thus the final output is a mono signal).

The patch example in Figure 3 demonstrates how the PWGL environment can be used to calculate input values for vectored ‘S’ boxes. The two first inputs, ‘low’ and ‘high’, of the ‘randi-vector’ box contain special PWGL shorthand expressions, ‘(14*(0.995))’ and ‘(14*(1.005))’, for generating lists (here we get 2 lists of 14 elements consisting of the values 0.995 and 1.005). The third input of the ‘randi-vector’ box, called ‘freq’, is connected to an ‘interpolation’ box, that returns a list of 14 values (the result of interpolating values from 5.0 to 20.0). Thus the vectored output of the ‘randi-vector’ has 14 elements which are fed to the first input of a ‘mul-vector’ box. The second input of the latter box is connected to a standard PWGL ‘value-box’ that returns a list of 14 frequency values (only the first values are visible in the figure). The output of the ‘mul-vector’ box consists of 14 frequency values where each value is individually modulated by an interpolating random number generator. This output is connected to the ‘freq’ input of a ‘reson-bank’ module. Like ‘reson-vector’ given in the previous example, ‘reson-bank’ is a bank of resonators. The first input is different, however, as it accepts only a mono signal (here a simple impulse) instead of a vectored input. The other inputs of the

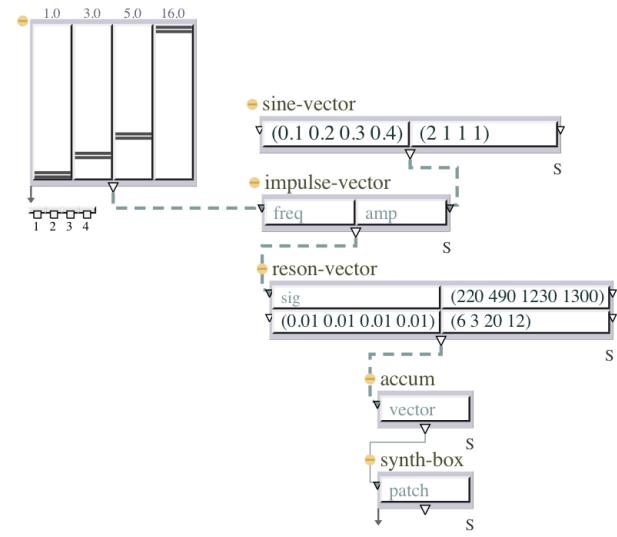


Figure 2: A bank of 4 resonators that are excited by a bank of impulse generators.

‘reson-bank’ module, amplitudes and bandwidths, are lists of 14 values (again only the beginning values are visible). Finally, the output of the ‘reson-bank’ box is mixed to a mono signal with a ‘accum’ box, exactly as was done in the previous example.

3. VECTOR MANIPULATION

In this section we discuss some of the synthesis modules that allow to manipulate vectors. In the simplest case *arithmetic* operations can be applied either to two vectors, or to a mono signal and to a vector. Figure 3 gave already an example of this kind of a box, where the ‘mul-vector’ box multiplied its input vectors resulting in a vector of modulated signals. Similar boxes, called ‘add-vector’ (see Figure 5), ‘sub-vector’, and ‘div-vector’, allow to add, subtract and divide input vectors.

Vectors can be *combined* to a single vector using a box called ‘combiner’, that can have an arbitrary number of inputs. A typical application of this box is when the user wants to combine several mono boxes so that the resulting vector can be fed to a vectored box. Thus in Figure 4 a ‘combiner’ box combines two mono random number generators, and the resulting vector is fed to an ‘impulse-vector’ box. This vector is in turn mixed to a mono signal resulting in a stream of 7 impulses per second where every seventh impulse is strongly accented (see the first ‘freq’ input of the ‘impulse-vector’ box containing the list ‘(1 7)').

Our next example shows how vectors can be *split* into sub-vectors by using the ‘indexor’ box (Figure 5). This box has 3 inputs, ‘vector’, ‘index’, and ‘len’. The starting point is a bank of 29 resonators. The vectored output is split into two sub-vectors so that vector elements 0-15 (the indexing starts from 0) form the first sub-vector (see the ‘indexor’ box to the left), and the remaining elements form the other sub-vector (the ‘indexor’ box to the right). After this both sub-vectors are mixed to 2 mono signals, which are in turn fed to 2 spatialization boxes, called ‘vbap2d’. A ‘vbap2d’ box implements a 2-dimensional version of the Vec-

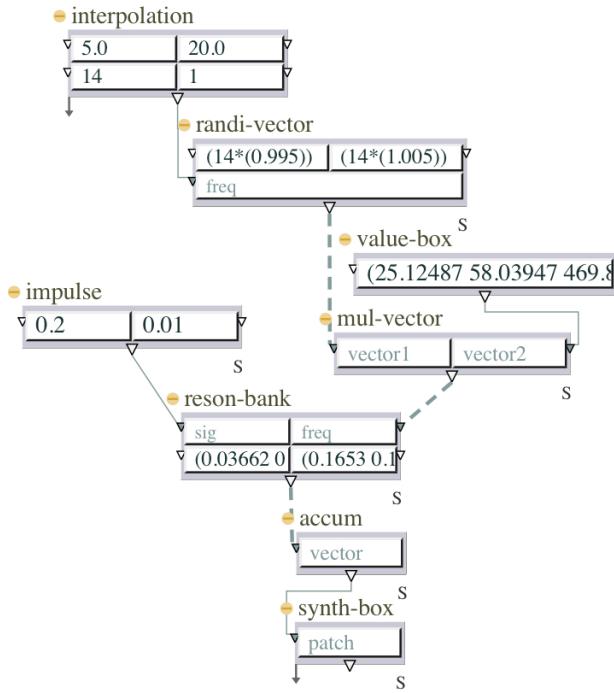


Figure 3: A bank of 14 resonators where the frequencies are modulated by a bank of interpolated random number generators.

tor Base Amplitude Panning (VBAP) algorithm [5]. Here the first input is panned in a 2-dimensional space according to the second input, called ‘azim’. The third input defines the speaker configuration to be used by the system. This input has 4 speaker positions and thus the patch results is a 4-channel output.

4. VECTORED SIGNALS COMBINED WITH THE COPY-SYNTH-PATHC SCHEME

In our final section we discuss some more complex cases that combine vectored signals with a copying scheme for patches [3]. One problem with the system that was presented in the previous sections is that typically one needs box versions for both mono and vectored cases (e.g. ‘sine’ and ‘sine-vector’). Next we introduce a scheme that allows to combine any collection of mono or vectored boxes. This collection can be copied and the output of the result will be one or several vectored signals. We use for this purpose a special box called ‘copy-synth-pathc’ with 2 required inputs, ‘count’ and ‘patch’. A third, optional, input can be given for a name string. A ‘copy-synth-pathc’ box duplicates a patch connected to the ‘patch’ input count times. The output of the box is a vector having the length which is determined by the count input (internally the system uses the ‘combiner’ box discussed above to achieve this result).

Figure 6 gives a simplified overview of a guitar model that copies a ‘string’ abstraction 6 times. Note that the upper part of the figure shows only a part of the content of the ‘string’ abstraction. The output of the ‘copy-synth-pathc’ box is a 6-element vector that is mixed to a mono signal by the ‘accum’ box.

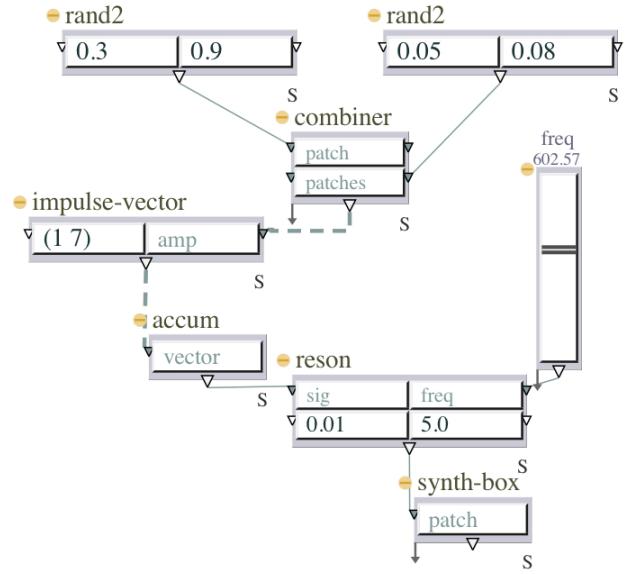


Figure 4: Combining 2 impulse streams to excite a single resonator.

To demonstrate some of the possibilities when combining the vectored-based approach with the copy-synth-pathc scheme we add to the output of the string abstraction a ‘vbap2d’ box with a 4-channel speaker configuration (see the upper part of Figure 7). Note that as the output of the abstraction has changed from a mono signal to a vectored one with 4 channels, the output of the ‘copy-synth-pathc’ box is now a vector of 24 elements (6 strings * 4 channels). In order to get a 4-channel output we mix the 24 element vector to a 4-element vector using a special vector manipulation box called ‘accum-vector’. This box is a vectored version of the ‘accum’ mixer box that has been used in the previous examples. The difference is that ‘accum-vector’ translates an input vector internally to a matrix, where the number of columns is determined by the second input, called ‘len’ (in our case equal to 4). The number of rows is gained by dividing the number of elements of the input vector by the ‘len’ input (in our case 24/4 = 6). After this the box mixes all columns resulting in the final output vector.

5. CONCLUSIONS

This paper gave an overview of some multichannel capabilities of our visual synthesis environment. The system can be used to effectively to realize complex DSP oriented tasks such effects, reverbs and filter banks. The scheme has also proven to be useful when using it in conjunction with our copy-patch-scheme to realize instrument models.

A PWGL beta version that includes our synthesis environment can be loaded from the following homepage:
<http://www.siba.fi/PWGL>

6. ACKNOWLEDGEMENTS

The work of Mikael Laurson and Vesa Norilo has been supported by the Academy of Finland (SA 105557).

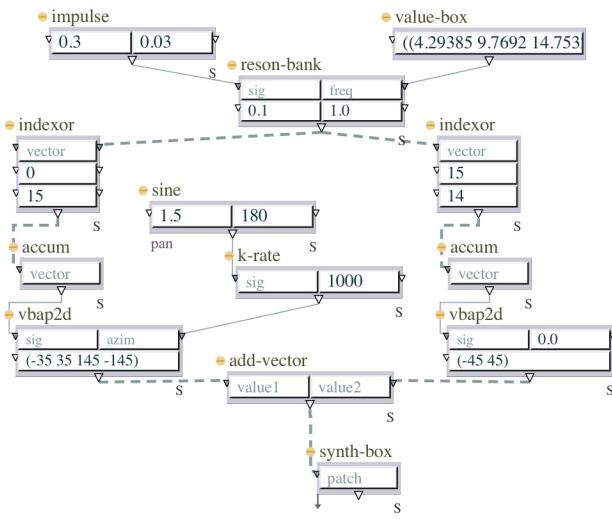


Figure 5: A bank of 29 resonators where the output vector is split into two sub-vectors that are panned individually.

7. REFERENCES

- [1] J. McCartney, "Continued evolution of the Super-Collider real time environment," in *Proc. Int. Comp. Music Conf. (ICMC'98)*, Ann Arbor, USA, 1998, pp. 133–136.
- [2] M. Laurson and V. Norilo, "Recent developments in PWSynth," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003, pp. 69–72.
- [3] M. Laurson, V. Norilo, and M. Kuuskankare, "PWGLSynth: A visual synthesis language for virtual instrument design and control," *Computer Music J.*, vol. 29, no. 3, pp. 29–41, Fall 2005.
- [4] M. Laurson and M. Kuuskankare, "PWGL: A novel visual language based on common Lisp, CLOS and OpenGL," in *Proc. Int. Comp. Music Conf. (ICMC'02)*, Gothenburg, Sweden, 2002, pp. 142–145.
- [5] V. Pulkki, "Virtual source positioning using vector base amplitude panning," *J. Audio Eng. Soc.*, vol. 45, no. 6, pp. 456–466, June 1997.

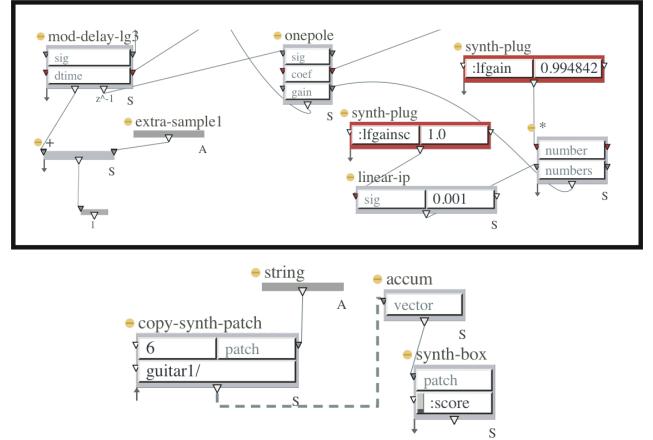


Figure 6: A guitar model consisting of 6 strings. Top: part of the 'string' abstraction. Bottom: a 'copy-synth-patch' box duplicates the 'string' abstraction 6 times.

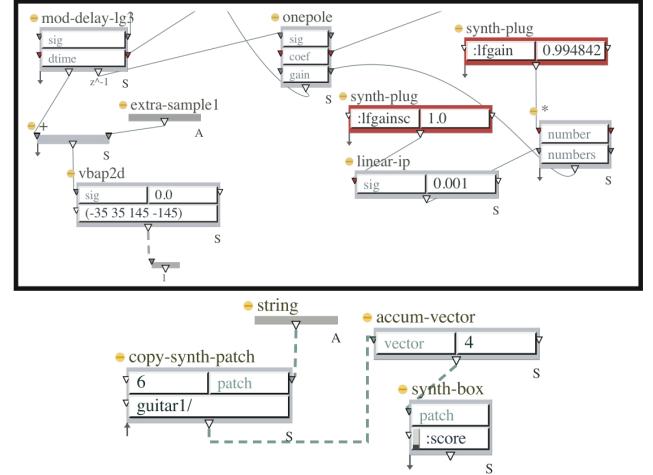


Figure 7: A modified 4-channel guitar model that allows to pan each individual string separately.

USING FAUST FOR FPGA PROGRAMMING

Robert Trausmuth, Christian Dusek

Dept. of Computer Engineering
University of Applied Science
Wiener Neustadt, Austria
trausmuth@fhwn.ac.at

ABSTRACT

In this paper we show the possibility of using FAUST (a programming language for function based block oriented programming) to create a fast audio processor in a single chip FPGA environment. The produced VHDL code is embedded in the on-chip processor system and utilizes the FPGA fabric for parallel processing.

For the purpose of implementing and testing the code a complete System-On-Chip framework has been created. We use a Digilent board with a XILINX Virtex 2 Pro FPGA. The chip has a PowerPC 405 core and the framework uses the on chip peripheral bus to interface the core.

The content of this paper presents a proof-of-concept implementation using a simple two pole IIR filter. The produced code is working, although more work has to be done for implementing complex arithmetic operations support.

1. INTRODUCTION

Modern FPGAs (Field Programmable Gate Arrays) allow the system engineer to design full systems in one chip by defining the desired logical functionality of the chip using a hardware description language. The inherent parallelism of the FPGA logic can be of advantage once it comes to heavy calculation tasks or to problems which use many parallel processes. Our motivation for this project was to create a system on chip design tool which can be used for audio signal processing. Although there are commercial products available, this implementation is part of an open source project and thus can be used and developed further by interested partners.

To ease the programming task we chose the Functional AUdio STream programming language FAUST, developed at GRAME Centre National de Creation Musicale, Lyon, France. This language is based on a block-diagram algebra [1]. All program elements are described as building blocks, the final system consists of a hierarchy of those blocks. Each block has a defined interface (input and output). Once the desired signal processing algorithm is programmed using the FAUST syntax it gets compiled into VHDL logic. VHDL stands for Very high speed integrated circuit Hadware Description Language, a programming language to develop on-chip logic designs. These designs are the input to the chip synthesis tool.

The FAUST compiler originally creates C++ code which can be put into a framework and later be used as plugin for a series of wellknown (software) audio processing programs. Our addition to the system is the possibility to generate on-chip logic to do the task and use the processor on chip for slow control and user interface. With this we are close to the current developments of

Yann Orlarey

GRAME
centre national de creation musicale
Lyon, France
orlarey@grame.fr

hardware/software co-design tools (for a prominent example see [2]).

The choice of the target platform was quite easy because of the experiences with the POWERWAVE synthesizer [3]. The board uses an AC'97 compatible audio codec with 48 kHz sampling rate and 20 bits resolution. The XILINX Virtex 2 Pro has two PowerPC 405 processors on chip running at 300 MHz. One of those is used for implementing the slow control and user interface. The VHDL building blocks are embedded in an OPB module. The On-chip Peripheral Bus is an IBM standard bus intended to connect hardware extensions to the on-chip processor. This bus runs at 100 MHz. The framework implements several dual port RAMs for exchanging parameter data with the audio processor, granting parallel access for parallel running blocks.

2. FAUST EXTENSIONS

The FAUST compiler produces optimized C++ code. The optimization tries to find out what has to be computed and omits all unnecessary parts. For the purpose of FPGA programming we need VHDL modules. Since the implementation of logic on FPGAs has different needs, there are some alterations and extensions needed for the original FAUST compiler. The compiler works in several stages shown in figure 1.

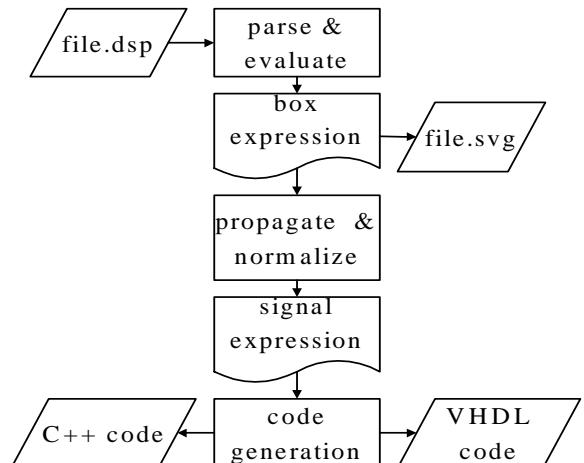


Figure 1: The stages of the FAUST compiler.

The role of the compiler is to translate from FAUST to C++ or from FAUST to VHDL, while preserving the mathematical semantic. To do that, the compiler must first discover the mathematical

semantic of a FAUST program. This is the **propagation** phase of the compiler. It translates box expressions into normalized signal expressions that express the semantic of the box expressions. In theory (and in practice to some extent) two very different FAUST programs which actually compute the same thing from a mathematical point of view should result in the same signal expression after the propagation phase.

The **code generation** phase operates on signal expressions, not on box expressions. The current C++ generator translates these signal expressions into equivalent C++ code. It tries to generate the most efficient C++ code while preserving the mathematical semantic. The VHDL generator should do the same: it should operate on signal expressions and generate the most efficient VHDL code while preserving the semantic of the computation.

One of our main goals is keeping the VHDL code generation totally transparent to the end user. The final version of the compiler should work on any FAUST program (almost) without alterations on the code itself. So any considerations concerning the extensions of the FAUST compiler have to take into account this required transparency.

2.1. Fixed point number representation

The C/C++ code uses floating point number representation. Although VHDL supports floating point numbers, this floating point capability is not easily synthesizable to the chip logic. Therefore we need to implement the capability of fixed point number calculation, which implies a few extensions concerning the VHDL code generator. Fixed point numbers have a total width and a defined binary point. Each fixed point building block produces an output which can be of the same width or even larger. Due to signal propagation times on the chip, there are some limitations concerning the total width of adders and multipliers which also have to be taken into account.

Usually there is a need for truncating the big fixed point numbers after a calculation block. This will be done by a special primitive which incorporates stochastic rounding [4]. This well known method is used in DSPs and helps reduce the quantization error which might be a problem when several stages are calculated one after the other.

To grant transparency to the user all necessary parameters like bit width of the signals, the number of memory and MAC blocks and the VHDL default blocks are predefined by the compiler. However, there is the possibility to include a definition file into the FAUST code to overwrite those default settings.

2.2. Calculation costs

VHDL blocks are usually implemented in a synchronous manner, which implies that it takes at least one clock cycle to evaluate each block. Some blocks may have longer delays, especially if they consist of several other blocks. Since we have a defined timeframe for the signal calculation - roughly 2000 clock cycles in our case - the design has to be checked against this boundary condition.

Each design block is afflicted with the clock cycles it takes to evaluate the block logic. If several processes run in parallel and meet at some point, the design has to assure that the following block gets the input values it expects. After the FAUST code has been processed, the internal signal model is evaluated and a Gantt chart is created to find the critical path through the model. This path determines the total signal propagation time of the VHDL

unit. At the merge points of the FAUST model the implementation model has to assure the synchrony of the temporary calculation results.

2.3. VHDL base blocks

The VHDL result is based on VHDL base building blocks, which have predefined calculation costs. Every junction adds to the total calculation cost. The base cost is part of the VHDL model library. If user specific VHDL blocks are added (like external C functions can be added at the moment), this cost value has to be stated in the VHDL model. The FAUST compiler creates a timing report showing all signal propagation times and the defined fixed point interfaces.

The VHDL top level interface block is called "process" and contains the interface to the framework. All other VHDL modules are logically contained in this toplevel module. Additionally, a C++ header file is produced containing the synonyms and addresses for all the registers of the design. This header file allows the interface implementation running on the PPC to access the proper memory locations for the parameters.

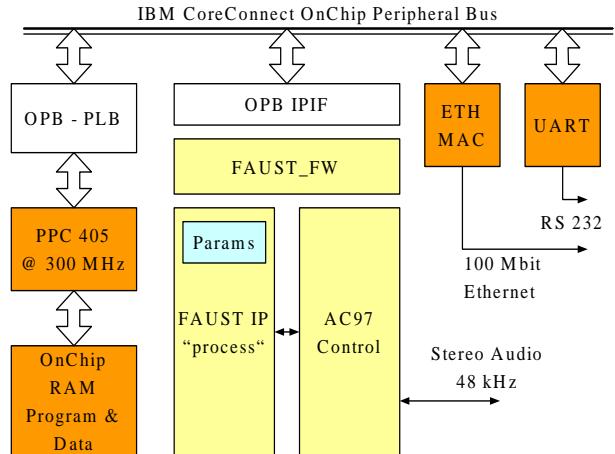


Figure 2: Block diagram of the FAUST VHDL framework.

The number of dual-port RAM blocks actually defines the maximum number of parallel data and address busses which can be used by the calculation cells. The number of parallel calculation cells (each one comparable to a small ALU with a command set optimized for DSP operations, see section 2.4) can also be specified.

Special care has to be taken concerning the width of the data bus as well as the duration and space cost of different block implementations. Although the bus width is set to 20 bits by default, this bus width can be changed. If

the 20 by 20 bit multiplier is too large to fit into the FPGA, another version (20 by 4 bits) is provided, which takes 5 clock cycles to calculate the result but only 1/4th of the FPGA fabric. Divisions on the FPGA logic are very costly and therefore not part of this implementation. Divisions by constants can be implemented using multiplications and reciprocal values.

A typical calculation step takes at least 6 clock cycles. The necessary operations are executed in a pipelined mode: the first 4 clock cycles are generally needed to fill the local registers with the

ALU command	mathematics	execution time
madd R1 R2	$A = A + R1 * R2$	1 (+ 4)
msub R1 R2	$A = A - R1 * R2$	1 (+ 4)
madd4 R1 R2	$A = A + R1 * R2$	5 (+ 4)
msub4 R1 R2	$A = A - R1 * R2$	5 (+ 4)
rounda	$A = \text{rnd}(A)$	1
clra	$A = 0$	1
mova R1	$R1 = A$	1
sra R1	$A = A \gg R1$	1 (+ 3)
srl R1	$A = A \gg R1$	1 (+ 3)
sla R1	$A = A \ll R1$	1 (+ 3)
sll R1	$A = A \ll R1$	1 (+ 3)
and R1	$A = A \& R1$	1 (+ 3)
or R1	$A = A R1$	1 (+ 3)
xor R1	$A = A ^ R1$	1 (+ 3)
not R1	$A = \sim A$	1

Table 1: ALU command set

needed values, although register reuse and optimization can drop those steps. Then the ALU block takes at least one cycle to do the real calculations. After this another cycle may be necessary to store the ALU accumulator back to a local register. The ALU commands are explained in the next section.

2.4. ALU command set

All the FAUST calculations are finally done by a special ALU block which supports a few calculation operations. Since memory operations are handled by special bus controller processes, the ALU only needs the calculation commands shown in table 1. Logical decisions which are needed for branching during the execution of the program are not part of the language and therefore not part of the calculation units.

Execution times for the ALU commands are given in clock cycles, the additional values show the clock cycles used for loading the needed values into local registers, if necessary. The special multiplication used by the *madd4* and *msub4* commands uses less logic cells on the FPGA but takes longer to calculate. The use of these commands can be defined by a global setting in the FAUST program.

2.5. Code generation strategy

The FAUST compiler parses the process model and produces a tree of signal expressions optimized to the mathematical semantic. The VHDL code generator first takes these signal expressions and translates them into a Gantt chart. This chart reflects all memory and calculation tasks in a linear manner in the beginning.

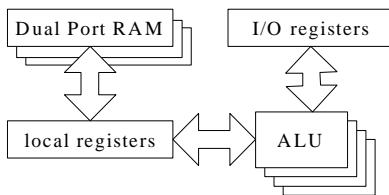


Figure 3: VHDL architecture block diagram.

The optimizer knows how many memory busses and ALU units are available and compacts the linear Gantt chart into a compact execution plan similar to the one shown in table 2. The optimizer determines the final number of local registers.

Each memory bus and ALU cell is then coded into one separate VHDL state machine, all being part of the FAUST IP "process" module. Figure 3 shows the block diagram of the synthesized architecture.

Each calculation is triggered by the AC97 module after receiving the next audio sample. This is the only synchronisation point in the design. After the start pulse all state machines run in parallel as designed by the optimizer. The optimizer has the responsibility to take care about execution times and synchronisation points throughout the VHDL design.

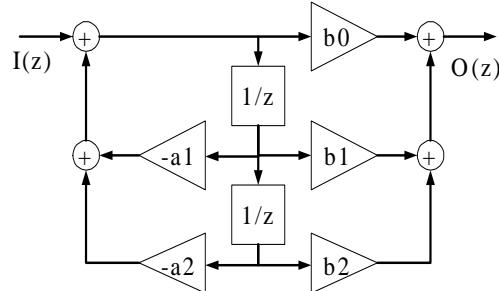


Figure 4: block diagram of the 2 pole IIR filter used as reference design

3. THE FRAMEWORK

The host for the FAUST VHDL module is a XILINX Virtex 2 Pro 30 with roughly 30.000 logic cells. The two PowerPC 405 CPUs use several chip busses to connect to the FPGA logic. Figure 2 shows the block diagram of the FPGA framework.

The VHDL module FAUST_FW is connected to the OPB bus. On the controller runs a small C program, so the user can talk to the system using ethernet or – in our first implementation – the serial RS 232 line. The OPB module provides the interface for the FAUST IP "process" module and also takes care of the audio codec. The AC'97 codec is initialized by the FPGA logic. Each sample is passed to the FAUST IP module using a simple handshake protocol. The FAUST module works on the audio data and provides the result for the next codec cycle.

There is an alternative operating mode (if we want to use block processing) in which 512 samples are stored in memory before the processing is called. In this case block operations can be used to calculate FFT or other block based algorithms. The block is shifted by 64 samples between two calls so the values keep overlapping. In this mode the recalculation is triggered every 64 samples, thus leaving 130000 cycles (or 1,3 ms) for one calculation run. The FFT algorithm as well as a few other block based signal algorithms will be one of the next extensions to the VHDL primitive library.

Parameters are stored in a dual port RAM section on chip. The FAUST module can access the parameters as needed and operates asynchronously to the PPC 405 control program. Several memory blocks can be used in parallel. In the default application, up to 4 busses can be used, each memory block containing 2 kbytes in 32 bit data width (512 parameter words). The FAUST optimizer

takes care of parallel execution of the calculations. The master chip clock runs at 100 MHz, which gives about 2000 cycles per audio sample (still leaving 83 cycles for protocol handling between the process IP and the FAUST_FW IP).

4. THE IMPLEMENTATION

To show the implementation of the VHDL code generator we will use a simple example, the two pole IIR filter. This is used only as reference design in this paper, a detailed discussion can be found in [5]. The filter uses five parameters b0, b1, b2, a1 and a2 as well as two memory blocks. Figure 4 shows the block diagram.

The FAUST code to create this filter looks like this:

```
a1 = 1.73;
a2 = 1;
b0 = 1.25;
b1 = 1.73;
b2 = 1;

filter(b0,b1,b2,a1,a2) = + ~ conv2 : conv3
with
{
    conv2(x) = 0 - a1*x - a2*x';
    conv3(x) = b0*x + b1*x' + b2*x'';
};

process = filter(b0,b1,b2,a1,a2);
```

This code is parsed and normalized by the FAUST compiler to give a very short representation in C++. Basically, a few copy operations and the two additions in the main blocks conv2 and conv3 are left.

```
virtual void compute (int count,
    float** input, float** output)
{
    float* input0 __attribute__ ((aligned(16)));
    float* output0 __attribute__ ((aligned(16)));
    input0 = input[0];
    output0 = output[0];
    for (int i=0; i<count; i++)
    {
        float T0 = M0;
        M0 = R0_0;
        R0_0 = (input0[i] - (T0 +
            (1.730000f * R0_0)));
        float T1 = M1;
        M1 = R0_0;
        float T2 = M2;
        M2 = T1;
        output0[i] = (T2 + ((1.730000f * T1) +
            (1.250000f * R0_0)));
    }
}
```

The translation into assembly code results in 54 CPU clock cycles for computing one filter cycle on a standard CPU (not on a DSP). The VHDL code makes use of parallel register loading and calculating and the calculations are done one per clock cycle. The total calculation time of the VHDL module is 11 clock cycles.

Putting the resulting VHDL code here would exceed the limits of this paper. However, table 2 displays the idea of the parallel processing used in the FPGA VHDL code.

The simple VHDL implementation uses one 42 bit MAC unit, one memory/bus controller and four 20 bit registers. The calcula-

Reg0	Reg1	Reg2	Reg3	MAC
	load z^{-2}	load z^{-1}	load a_2 load a_1 load b_2	0 +input $-a_2 z^{-2}$ $-a_1 z^{-1}$ round
copy MAC	load b_1	store as z^{-2}	load b_0	0 $+b_2 z^{-2}$ $+b_1 z^{-1}$ $+b_0 * Reg0$ round

Table 2: execution plan for the IIR VHDL implementation

tion and the memory access are done in parallel, thus saving time. Each row corresponds to one clock cycle, the result of each operation is available after one cycle. This result is comparable to an implementation on a modern DSP. However, when it comes to multiple filters in parallel, the FPGA gains advantage due to the parallel capabilities.

If we put three IIR filters in parallel and use three MAC units and also three bus/memory units, the whole calculation will be done in 15 clock cycles.

5. CONCLUSIONS

The FAUST to VHDL compiler works for very simple problems where only basic mathematic functions are needed. If we want to use trigonometric functions, additional building blocks have to be provided in VHDL.

Complex mathematical calculations like the FFT algorithm can be implemented in VHDL as IP blocks and therefore improve the data throughput of the design. These IP blocks as well as the trigonometric function blocks will be provided in the future as external library blocks.

Since this is a proof-of-concept we are very positive that future work will produce a general purpose VHDL code generator.

6. REFERENCES

- [1] Y. Orlarey, D. Fober, and S. Letz, "Syntactical and semantical aspects of Faust," *Soft Computing*, vol. 8, no. 9, pp. 623–632, Sep. 2004.
- [2] D. Andrews, W. Peck, J. Agron, K. Preston, E. Komp, M. Finley, and R. Sass, "hthreads: A hardware/software co-designed multithreaded RTOS kernel," in *10th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Catania, Italy, Sep. 2005.
- [3] R. Trausmuth and A. Huovilainen, "POWERWAVE – a high performance single chip interpolating wavetable synthesizer," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, Sep. 2005.
- [4] C. Maxfield, *How Computers Do Math*. John Wiley & Sons, 2005.
- [5] U. Zölzer, *DAFX – Digital Audio Effects*. John Wiley & Sons, 2002.

PARAMETRIC CODING OF STEREO AUDIO BASED ON PRINCIPAL COMPONENT ANALYSIS

Manuel Briand, David Virette

Speech and Sound Technologies
and Processing (SSTP) Laboratory
France Telecom R&D, Lannion, France
{manuel.briand | david.virette }
@orange-ft.com

Nadine Martin

Image and Signal processing Laboratory (LIS)
CNRS UMR 5083
INP-Grenoble, France
nadine.martin@lis.inpg.fr

ABSTRACT

Low bit rate parametric coding of multichannel audio is mainly based on Binaural Cue Coding (BCC). Another multichannel audio processing method called upmix can also be used to deliver multichannel audio, typically 5.1 signals, at low data rates. More precisely, we focus on existing upmix method based on Principal Component Analysis (PCA). This PCA-based upmix method aims at blindly create a realistic multichannel output signal while BCC scheme aims at perceptually restore the original multichannel audio signal. PCA-based upmix method and BCC scheme both use spatial parameters extracted from stereo channels to generate auditory events with correct spatial attributes *i.e.* sound sources positions and spatial impression. In this paper, we expose a multichannel audio model based on PCA which allows a parametric representation of multichannel audio. Considering stereo audio, signals resulting from PCA can be represented as a principal component, corresponding to directional sources, and one remaining signal, corresponding to ambience signals, which are both related to original input with PCA transformation parameters. We apply the analysis results to propose a new parametric coding method of stereo audio based on subband PCA processing. The quantization of spatial and energetic parameters is presented and then associated with a state-of-the-art monophonic coder in order to derive subjective listening test results.

1. INTRODUCTION

Multichannel audio has been established in the consumer environment through the success of DVD-Video players for home theater systems. Moreover, the streaming technology over IP used as a broadcast service is requesting multichannel audio at low data rates. Therefore, multichannel audio coding and processing methods have been investigated by many researchers during last decade.

The first method is denoted as multichannel audio coding. Matrix surround coding schemes and parametric audio coding schemes are the two main multichannel audio coding techniques currently used. Matrix surround coding scheme such as Dolby Pro Logic [1] consists in matrixing the channels of the original multichannel signal in order to reduce the number of signals to be transmitted. Nevertheless, this multichannel audio coding method cannot deliver high quality (close to transparency) at low data rates. That is made possible with low bit rate parametric audio coding mainly based on Binaural Cue Coding (BCC) [2]. This coding scheme represents multichannel audio signals by one or several downmixed audio channels plus spatial cues extracted from the

original channels. The spatial cues refer to the auditory localization cues defined as interaural time and level differences (ITD and ILD) extracted from input channel pairs in a subband domain and then denoted as inter-channel time and level differences (ICLD and ICTD). BCC uses filterbanks with subbands of bandwidths equal to two times the equivalent rectangular bandwidth (ERB) defined in [3]. Moreover, the inter-channel coherence (ICC) is also extracted in order to recreate the diffuseness of the original multichannel input. Indeed, the multichannel audio synthesis at the decoder side is based on the ICC parameter which yields a coherence synthesis relying on late reverberation. The downmixed audio channel (in case of mono downmix) is decoded and then filtered by late reverberation filters which deliver several decorrelated audio channels (see [2] for more details). Then, these signals are combined according to spatial cues (ICTD, ICLD and ICC) such that the ICC cues between the output subbands approximate those of the original audio signal. Then, BCC scheme achieves a drastically data rate reduction by transmitting a perceptually encoded downmixed signal plus quantized spatial cues. Moreover, BCC scheme achieves a better audio quality and perceived spatial image than matrix surround coding scheme (see subjective tests results in [2]). From a spatial attribute point of view, BCC synthesis restricted to ICLD and ICTD achieves the desired sources positions and coloration effects caused by early reflections but suffers from auditory spatial image width reduction. Indeed, spatial impression is related to the nature of reflections that occur following the direct sound. Then, BCC synthesis based on late reverberation mimicks different reverberation times and then achieves a better spatial impression closer to the original multichannel input.

A second multichannel audio processing method, called upmix, classically converts the existing stereo audio contents into five-channel audio compatible with home theater systems. So, the decoding process of BCC has the common intention with upmix method which is to deliver multichannel audio signal – considering upmix stereo input and BCC stereo downmix. *A priori*, more information is available for BCC scheme *i.e.* spatial parameters, than for blind upmix method. However, upmix method uses the spatial characteristics and the coherence of the stereo signal to synthesize a multichannel audio signal, typically 5.1 signal with rear channels considered as ambience channels – defined as diffuse surround sounds – and a center front channel corresponding to the sources panned across the original stereo channels. More precisely, we focus on existing PCA-based upmix method described in [4, 5]. The first step of the upmix algorithm in [4] consists in a Principal Component Analysis (PCA) of the stereo signal. A

subband processing of this upmix method has been recently proposed in [5]. The PCA is equivalent to a rotation of the stereo signal coordinate system and results in one principal component signal and one remaining signal. The principal component signal corresponds to the dominant sources present in the original stereo. Then, the center channel results from the weighting of this principal component by a coefficient derived from the rotation angle of the coordinate system. The rear channels result from the weighting of the remaining signal by a coefficient derived from the correlation coefficient of the stereo signal.

Starting from PCA approach, we propose a general model that may be applied both to parametric representation of multichannel audio signals and upmix methods. Moreover, we apply the analysis results to propose a new parametric audio coding method based on frequency subband PCA processing. This paper is organized as follows. A general model of multichannel audio signals is presented in section 2. Then, subband PCA of stereo audio is considered and results in an energy compaction and a parametric representation related to original stereo with PCA transformation parameters. Finally, a new parametric coding method of stereo signals is presented in section 3. The encoding stage focuses on spatial and energy parameter extraction and quantization while the decoding stage addresses the parametric synthesis of ambience signal in order to achieve accurate inverse PCA. Subjective listening test results are presented in section 4 in order to evaluate the performance of the parametric stereo coding method.

2. PARAMETRIC REPRESENTATION OF MULTICHANNEL AUDIO BASED ON PCA

Multichannel audio signals either originate from studio (artificial) produced signals or from live (natural) recorded signals [6]. Live recording involves many different setups and microphones types which determine the amount of interferences and reverberation on each channel. Decorrelated reverberant channels yield a sensation of realistic ambience. Ambience has a complex audio content, perceptually perceived in background and very heterogeneous. Ambience could be defined as "the sound of the place in which sources are". Such an audio content includes acoustic effects of reverberant volumes and reflective features plus the background *i.e.* the acoustic accumulation of many small sources that are not the identified sources of interest; for example, audience noise. In studio recording, sound sources (instruments) are individually recorded and then processed. The processing of recorded sound sources consists in applying panning functions to the sound sources and then mixing them with synthetic reverberation. In order to increase the perception of spaciousness, weakly correlated reverberation impulse responses are used.

Under these assumptions, we define a general model for multichannel audio signals constituted of M channels. We assume the presence of D directional sources that can be easily localized. These directional sources are distributed over the M channels by means of panning laws. Moreover, we consider M ambience signals *i.e.* one by channel, which are defined as secondary sources and decorrelated reverberant components of directional sources *i.e.* room effect. Naturally, these ambience signals are decorrelated from one channel to another and weakly correlated with directional sources. The model of each channel is defined as the sum of directional sources, weighted according to their spatial perceived positions, and one ambience signal. Signal channels following this instantaneous mixture model are strongly correlated with the pres-

ence of directional sources among several channels. So, the time domain multichannel signal $\mathbf{C}_M = (C_1, \dots, C_M)^T$ (T denotes matrix transposition) can be written as:

$$C_m(t) = \sum_{d=1}^D [g_{m,d}(t).S_d(t)] + A(t) \quad (1)$$

where $m \in [1, \dots, M]$ and $g_{m,d}(t)$ are the panning functions (gains) applied to the directional sources $\mathbf{S}_D = (S_1, \dots, S_D)^T$ of the m^{th} channel. Finally, room effects on directional sources are distributed over all channels because these decorrelated components belong to the ambience signals $\mathbf{A}_M = (A_1, \dots, A_M)^T$.

In a general audio coding context, the separation of dominant sources (even mixed with part of the overall ambience) from background ambiances could be seen as a pre-processing step achieved with PCA before applying a dedicated coding scheme. Indeed, the transmission of encoded dominant sources already provides a basic audio scene of the original input. If the coding method foresees the generation of an ambience signal, then, the decoding process can achieve better spatial impression.

PCA also known as Karhunen-Loëve Transform (KLT) has been used by numerous researchers for miscellaneous applications. D.T. Yang, C. Kyriakakis and C.-C. Jay Kuo use KLT as an inter-channel redundancy removal in a multichannel audio coding context in [7]. Indeed, PCA/KLT achieves an energy concentration over PCA outputs according to the distribution of eigenvalues. Moreover, PCA is theoretically considered as the optimal decorrelation method *i.e.* the covariance matrix of PCA output signals is diagonal. KLT is only optimal in the case of stationary signals, which is an admitted assumption for signal processing with small block length (typically 20 ms for audio/speech codecs). The authors in [7] propose a modified Advanced Audio Coding encoder which performs better coding gain using KLT applied to the multichannel input.

From the audio model defined by equation (1), our intention is to characterize the PCA outputs and then achieve a parametric coding of this compact representation of multichannel audio.

Eigenvalues distribution of such multichannel audio signals precisely indicates how the original audio content is distributed over the PCA outputs. An analysis of estimated eigenvalues from a synthetic stereo signal made up of real recordings has been achieved in [8]. This analysis has shown that the highest eigenvalue power corresponds to the power of the dominant directional source. The lowest eigenvalue power corresponds to the power of the secondary directional sources plus the power of the overall ambience. Moreover, a comparison of time-domain estimated eigenvalues and frequency subband estimated eigenvalues has shown that subband analysis results in a better discrimination of directional sources. Indeed, some directional sources considered as secondary sources with the time domain analysis can be considered as dominant directional sources with the subband analysis when these sources have different frequency support. Finally, with a subband analysis, the lowest eigenvalue has a power much closer to the original ambience signal mean power and respectively, the highest eigenvalue has a power closer to the sum of directional source powers.

This energy concentration over eigenvalues can also be computed directly from the output signals resulting from PCA. These signals result from a rotation of the stereo channels (see [8] for more details) and are then denoted as principal component signal (*PC*) related to the highest eigenvalue and ambience signal

(A) related to the lowest eigenvalue. Time-domain PCA processing and frequency subband (following ERB scale) PCA processing are presented in [8]. Then, a relevant measurement of the energy compaction into PC is achieved by estimating the Principal Component to Ambience energy Ratio

$PCAR[n] = 10 \log_{10} (\sum PC[n]^2 / \sum A[n]^2)$ in dB, estimated from the N signal samples resulting from time-domain or subband rotation(s) of the stereo analyzed block n . For homogeneity, the $PCAR$ is estimated from PC and A signal blocks which have the same length (N) than the processed stereo channel blocks. Then, the $PCAR$ is function of the length of the stereo processed blocks. The Figure 1 is addressing a comparison of the average $PCAR$ over analyzed blocks of length $N = 1024, 2048$ and 4096 samples. These energy ratios have been estimated over a stereo signal corpus.

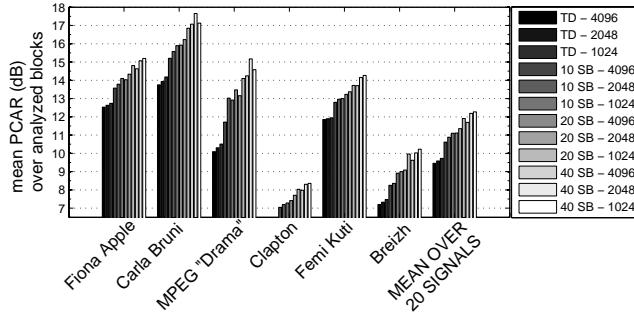


Figure 1: From time-domain (TD) and frequency subband (SB) PCA outputs, $PCAR$ are estimated with the following block processing lengths: 1024, 2048 and 4096 samples. Mean $PCAR$ over all blocks from 6 stereo signals and mean $PCAR$ over the stereo signal corpus are presented.

Moreover, $PCAR$ estimated from subband rotated signals is function of both time and spectral resolutions. In fact, the frequency resolution is determined by the frequency scale used for subband separation and the number of frequency subbands. $PCAR$ estimated from time-domain PCA output signals are also compared to $PCAR$ estimated from subband PCA output signals (see Figure 1) with $N_b = 10, 20$ and 40 subbands of bandwidth following the ERB scale. The energy compaction into the principal component is higher with a subband analysis comparing to time-domain analysis (see Figure 1, mean difference about 2 dB) and naturally increasing with the number of subbands (maximum difference about 5 dB for the MPEG "Drama" sample).

3. PARAMETRIC CODING OF STEREO AUDIO

Starting from the analysis made in section 2, it is possible to encode a stereo signal with frequency subband PCA pre-processing. PCA is used as a power concentration processing such as the Mid-/Side coding scheme which encodes the sum and difference signals of stereo channels. The principal component and transformation parameters should be encoded and transmitted in order to recover the main information of the original input. To obtain good performances in the inverse transformation process, the ambience component should be transmitted with a bit rate compatible with the desired audio quality level. Even if PCA features, such as power concentration and full decorrelation are required for optimal bite rate reduction, traditional encoding of the transformed signals does

not yield significant coding gain (see [9]). In order to provide a low bit rate audio coding method compatible with bit rate constrained applications, a parametric coding of the ambience signal is achieved.

3.1. Parameters based on PCA transformation and ambience energy

We consider band limited signals in the frequency domain. The frequency transform applied to stereo channels is the short time Fourier transform (STFT). The parameters of the STFT used are a sine window of length equal to $N = 4096$ samples, the transform size is also equal to $K = 4096$ (no zero-padding) and the frame overlap is 50%. Then, a $N_b = 20$ subband rectangular frequency windowing, following the ERB scale, is applied to the channel (C) complex spectra $F_C[n, k] = |F_C[n, k]|e^{j\Phi_C[n, k]}$, with frequency index k such as $k \leq f_s/2$, where f_s is the sampling frequency. This process results in N_b frequency subband spectra. The frequency bins of subband b for each frame n belong to the interval $(k_b, \dots, k_{b+1}-1)$, then $F_C^b[n, k] = F_C[n, k_b], \dots, F_C[n, k_{b+1}-1]$.

3.1.1. Subband parameter extraction

The first parameter which needs to be transmitted to the decoder is related to the PCA transformation. As described in [8], PCA of stereo channels is obtained by rotating the stereo subband channels over the stereo covariance eigenvector basis. This subband rotation is performed for each frame n and subband b according to the following expression of the estimated rotation angle $\theta[n, b] \in [0; \pi/2]$

$$\begin{aligned} \theta[n, b] &= \frac{1}{2} \tan^{-1} \left[\frac{2 \times |R_{12}[n, b]|}{R_{11}[n, b] - R_{22}[n, b]} \right] \\ &+ \begin{cases} 0, & \text{if } R_{11}[n, b] - R_{22}[n, b] \geq 0 \\ \frac{\pi}{2}, & \text{else} \end{cases} \quad (2) \end{aligned}$$

where the stereo subband covariance matrix \mathbf{R} is estimated from the subband spectra $F_L[n, k]$ and $F_R[n, k]$ of the windowed and centred signals $\bar{L}[n] = L[n] - E[L[n]]$ and respectively for $\bar{R}[n]$. The channel auto-covariance (R_{11} and R_{22}) correspond to the mean power spectral density of the subband spectra and the channel cross-covariance (R_{12}) is estimated from the cross-spectrum of the stereo channels

$$\mathbf{R} = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}, \text{with}$$

$$\left\{ \begin{array}{l} R_{11}[n, b] = \frac{2}{NK} \sum_{k=k_b}^{k_{b+1}-1} |F_L[n, k]|^2 \\ R_{12}[n, b] = \frac{2}{NK} \Re \left(\sum_{k=k_b}^{k_{b+1}-1} F_L[n, k] \cdot F_R^*[n, k] \right) \\ R_{22}[n, b] = \frac{2}{NK} \sum_{k=k_b}^{k_{b+1}-1} |F_R[n, k]|^2 \end{array} \right. \quad (3)$$

A robust and effective estimation of the rotation angle needed to achieve PCA/KLT is expressed by the equation (3) which permits an estimation only based on the covariance matrix elements. Moreover, the physical meaning of this rotation angle can be interpreted as the perceived position of the dominant directional source

localized in the stereo image $[-30, 30]^\circ$. In fact, this rotation angle is considered as a panning angle in the PCA-based upmix method to compute panning matrix which can generate realistic multichannel signals from PCA outputs. These subband rotation angles are then used to rotate the subband input stereo data blocks. Subband PCA processing results in two frequency components for each subband: the principal component $F_{PC}^b[n, k]$ and the ambience component $F_A^b[n, k]$.

Then, these subband signals resulting from PCA are analyzed and energetic parameters are extracted in order to be able to generate the ambience signal at the decoder side. Actually, the ambience synthesis should be achieved only from the decoded principal component and these energetic parameters. The parametric coding method does not consider the phase of the ambience signal. According to the subjective comparison test achieved in [8], the only perceptually relevant parameter of the ambience signal which needs to be transmitted is related to the ambience signal energy. Therefore, $PCAR$ is estimated from subband spectra from each analyzed block and each subband according to following expression

$$PCAR[n, b] = 10 \log_{10} \left(\frac{\sum_{k=k_b}^{k_{b+1}-1} |F_{PC}[n, k]|^2}{\sum_{k=k_b}^{k_{b+1}-1} |F_A[n, k]|^2} \right) \quad (4)$$

Finally, the PCA-based coding scheme applied to stereo signal is presented on Figure 2. The time domain signal PC is generated with the (overlap-add) sum of all synthesized block signals obtained with inverse short time Fourier transform ($STFT^{-1}$) of the sum of all frequency subband components. After subband PCA processing, the extracted parameters are quantized and perceptual monophonic coding of the principal component signal is achieved.

3.1.2. Subband parameter quantization

Our intention is to realize a low complexity uniform quantization of the subband parameters and then estimate the mean bit rate with Huffman coding. A training basis has been processed in order to estimate the dynamics and the precision needed for the quantization of subband parameters. Subband rotation angles and $PCAR$ have been extracted from a training basis made up of miscellaneous stereo recordings with a total duration of about one hour. The stereo channel blocks have been processed by the frequency transform described at the beginning of section 3.1. Parameters have been estimated according to equations (2) and (4).

Concerning the estimated rotation angle varying in the interval $[0; 90]^\circ$, we propose the quantization of the mean value of subband rotation angles for each analyzed block n , $\bar{\theta}$, defined as

$$\bar{\theta}[n] = \frac{1}{N_b} \sum_{b=1}^{N_b} \theta[n, b] \quad (5)$$

Mean rotation angle transmission assures a basic spatial reconstruction at low data rate with a reduced spatial image width. Then, with an additional data rate corresponding to the subband rotation angles, the original spatial image width can be recovered. Actually, we propose the quantization of differential subband rotation angles $\theta_D[n, b] = \theta[n, b] - \theta[n, b - 1]$, if $b > 1$. Differential subband rotation angles θ_D have pickier distribution with dynamics (in degrees) decreasing along frequencies. Then, a better coding gain can be achieved. The first subband rotation angle is processed independently and its mean removed value

$\theta_{MR}[n, 1] = \theta[n, 1] - \bar{\theta}[n]$ is also considered for the quantization process.

Considering the estimated $PCAR$, we assume that the ambience signal has globally low energy level which decreases along frequencies. As a result, we also propose the quantization of the differential subband $PCAR$, $PCAR_D[n, b] = PCAR[n, b] - PCAR[n, b - 1]$ if $b > 1$.

These parameters are then uniformly quantized. According to perceptual criteria, the quantization process aims at introducing quantization errors which are just inaudible.

For the rotation angles, the minimum audible angle (MAA), as described in [10], is considered. The performance in localization is a complex function which depends on the nature of the stimulus and also on the stimulus frequency. For instance, sinusoidal signals can be discriminated with $\pm 1^\circ$ precision for a source at 0° position (direct front of the listener) and this only for frequencies lower than 1 kHz. More precisely, the localization accuracy is decreasing when the sound source is moving away from the direct front of the listener and also when the frequency of the sound source is increasing. For example, a sinusoidal signal at 30° position (on the left side of a stereo setup) can be discriminated with at least $\pm 5^\circ$ precision for stimulus frequencies higher than 2 kHz, still extracted from [10]. So, considering real sound sources with spatial position belonging to the stereo setup interval $[-30; 30]^\circ$, the corresponding precision for the mean rotation angle $\bar{\theta}$ varying in the interval $[0; 90]^\circ$ is chosen equal to at worst $\pm 3^\circ$. The first subband mean removed rotation angle θ_{MR} and the differential subband rotation angles θ_D values are varying in different intervals with a dynamics decreasing along frequencies. Moreover, these modified subband rotation angles do not need the same precision as the vital mean rotation angle; they only aim at refining the spatial position of the sound sources. We propose to attribute different level of precision for these mean removed and differential subband angles. Therefore, considering the subband differential rotation angles θ_D , we follow the academic results presented in [10] with a precision decreasing when frequency increases:

- $\pm 4^\circ$ with frequency: $f < 1$ kHz,
- $\pm 7.5^\circ$ with frequency: $f < 5$ kHz,
- $\pm 10^\circ$ with frequency: 5 kHz $< f < fs$.

Considering the $PCAR$ expressed in decibels (dB) as the energy difference between the principal component and the ambience signal, audible difference between reconstructed and original stereo signals are avoided with an energetic precision about ± 3 dB. More precisely, the precision allocated to the original $PCAR$ of the first subband is slightly higher than the precision allocated to the subband differential $PCAR_D$. The $PCAR$ quantization step is less critical than just noticeable differences usually used by frequency quantizers because $PCAR$ is used to synthesize the ambience spectral envelope.

The precision allocated for each parameter determines the amount of bits needed by the quantization processes. This quantization step is followed by a Huffman coding which achieves a reduction of the mean code length and then, delivers a mean bit rate of this parametric stereo coding method. From a training basis, this parameter mean bit rate has been estimated and compared with parameter mean bit rates estimated from other well known stereo signals extracted from MPEG audio basis. The resulting mean bit rates estimated from the MPEG audio basis are on average (2.8 kbps) slightly lower than the mean bit rate estimated from

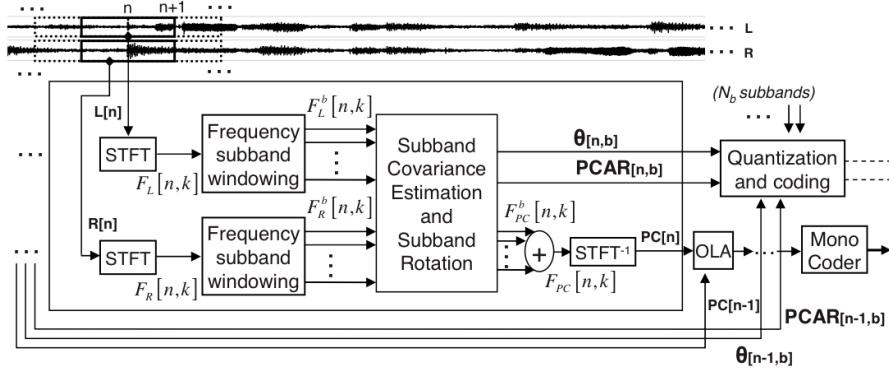


Figure 2: Parametric coding of stereo signals based on subband frequency PCA processing.

the training basis (3 kbps). So, we can conclude that the quantization process is weakly sensitive to the training basis.

In the final implementation of the parametric stereo coder, the quantized values of the subband rotation angles are used to achieve subband PCA. Afterwards, subband *PCAR* are extracted from the PCA outputs, quantized and transmitted with the quantized rotation angles. In order to reduce even more the bit rate, a joint polar quantization of the rotation angles and *PCAR* could be considered.

3.2. Ambience synthesis to achieve inverse PCA

The decoding scheme is based on the generation of an ambience signal A' , from the decoded signal PC' and the dequantized parameters (see Figure 3). Therefore, the inverse PCA can then synthesize a stereo signal perceptually as close as possible from the original stereo. Due to the PCA property of decorrelation, the decoder should generate an ambience signal weakly correlated to the decoded principal component. However, the frequency synthesis of subband signal A' from the decoded principal component PC' and the energy parameters $PCAR^Q[n, b]$, only provide A' spectral envelope. To achieve weak correlation between PC' and A' , we propose the use of random phase all-pass filters as described in [11]. We perform a subband filtering (filter H on Figure 3) of $F_{A'}^b[n, k]$ spectra.

So, the decoder can operate the inverse subband PCA from the signal subband spectra $F_{PC'}^b[n, k]$ and $F_{A'_H}^b[n, k]$ and the dequantized rotation angles $\theta^Q[n, b]$ (see Figure 3). Afterwards, the sum of all subband signals, obtained from inverse subband PCA, is transformed to time domain by inverse STFT to generate the stereo signal (L', R') .

4. SUBJECTIVE LISTENING TEST

In order to evaluate the PCA based parametric stereo coding method, a subjective listening test has been conducted. This subjective test aims at comparing the PCA-based parametric stereo coding method to the parametric stereo coding method used in the state-of-the-art High-Efficiency Advanced Audio Coding Parametric Stereo (HE-AACv2). Then, to achieve accurate comparison, several stereo signals encoded by HE-AACv2 at 24 kbps are perceptually compared to stereo signals processed by the PCA-based parametric stereo coding method described in section 3. More precisely, the principal component signal is encoded by the mono-

phonic profile of the HE-AAC encoder at 22 kbps and the quantized parameters are transmitted at a bit rate around 3 kbps. A lower bit rate of the HE-AAC mono profile would have generated monophonic signal with lower bandwidth than the bandwidth (16 kHz) of the stereo signals generated by the HE-AACv2 operating at 24 kbps. Then, the overall bit rate for the PCA-based parametric stereo coding method (PCA-HE-AAC) is about 25 kbps using 20 frequency subbands and a parameter update rate linked to the 4096 sample block length.

A subjective listening test following the MUSHRA methodology has been conducted. Thirteen listeners participated in this headphone listening experiment. All listeners were instructed to evaluate both the spatial audio quality as well as other audio coding artefacts. Listeners had to rate the perceived quality of seven processed items against the original stereo on a 100-point scale with two anchors corresponding to 3.5 kHz and 7 kHz low pass filtered version of the original stereo. Five original audio items were extracted from the MPEG audio basis and the other two stereo signals were chosen for their impressive stereo image. A comparison of the MUSHRA scores is presented on Figure 4-(a)-(b).

The stereo items processed by the HE-AACv2 have slightly higher score than the stereo audio processed by the PCA-HE-AAC (see Figure 4-(a)). This small difference can be explained by the fact that the PCA-HE-AAC codec uses a fixed and higher parameter update rate (4096 samples) than the HE-AACv2, which results in degraded transients; this is the case for the stereo item "Guitar+Castanets".

Although, the time-frequency resolution needs to be considered and improved to avoid coding artefacts, the spatial impression given by stereo signals processed by the PCA-based parametric coder is very close to the original stereo image. The spatial synthesis of the PCA-based parametric coder is the main advantage of this coding method and has been considered, by the listeners, as more stable than the spatial impression delivered by the HE-AACv2 coder. This explains that on average, there is no significant difference between both parametric stereo coding methods (see Figure 4-(b)).

5. CONCLUSIONS

In this paper, an instantaneous mixture model for multichannel audio is defined as the sum of weighted directional sources and ambiances. Then, PCA of stereo signals following the model has been considered. *PCAR* estimation has shown how a frequency

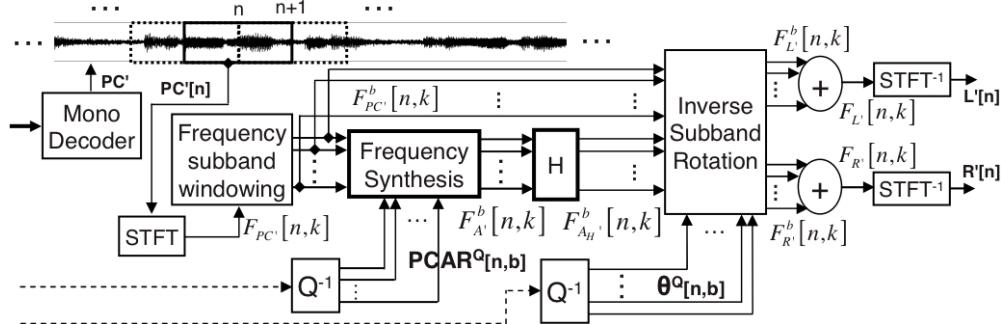


Figure 3: Parametric decoding of stereo signals based on inverse subband PCA processing.

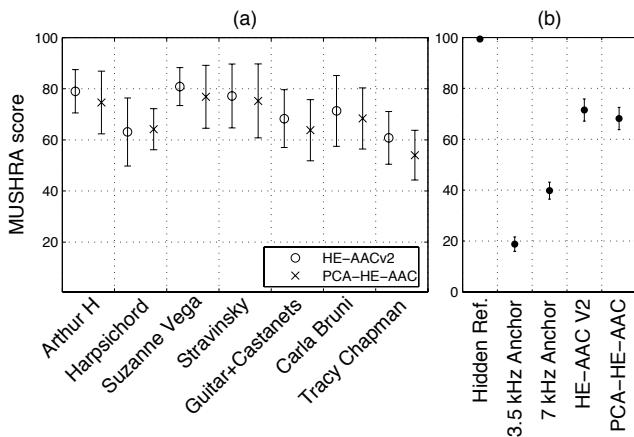


Figure 4: (a)-MUSHRA mean gradings and 95% confidence intervals over all listeners as a function of test item and the parametric stereo coding method. (b)-MUSHRA mean gradings and 95% confidence intervals over all listeners and all items.

subband PCA yields more efficient power concentration than classical time-domain PCA. Stereo audio is represented as a principal component, related to directional sources, plus a less energetic ambience signal, related to decorrelated ambiences, which provides spatial impression.

The main point addresses a parametric coding method of stereo audio based on subband PCA output signals. A stereo signal can be represented as a principal component and rotation angles which both permit a basic audio stereo reconstruction. Moreover, the ambience signal, related to the spatial impression components, has globally low energy level and no perceptually relevant phase information. Therefore, this ambience signal allows a parametric coding of stereo audio by means of PCA rotation angles and ambience energy parameters. To achieve better coding gain, subband differential parameters are uniformly quantized according to perceptual criteria. Then, Huffman coding of the quantized parameters is achieved in order to estimate a mean bit rate (about 3 kbps) of the parametric stereo coding method. The analysis has showed that the parameter quantization is not sensitive to a training basis.

Finally, the conducted listening test yield to the conclusion that there is no perceptually significant difference comparing the HE-AACv2 codec and the parametric stereo coding method based on PCA associated with the monophonic profile of the HE-AAC

codec. The parameter update rate should be adapted to the audio content to improve the audio quality.

6. REFERENCES

- [1] R. Dressler, "Dolby surround Prologic II decoder: Principles of operation," Dolby Laboratories, Tech. Rep., 2000, [Online] http://www.dolby.com/assets/pdf/tech_library/209_Dolby_Surround_Pro_Logic_II_Decoder_Principles_of_Operation.pdf.
- [2] C. Faller, "Parametric coding of spatial audio," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, July 2004, thesis No. 3062, [Online] <http://library.epfl.ch/theses/?nr=3062>.
- [3] B. R. Glasberg and B. C. J. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hearing Research*, vol. 47, pp. 103–138, 1990.
- [4] R. Irwan and R. M. Aarts, "Two-to-five channel sound processing," *J. Audio Eng. Soc.*, vol. 50, no. 11, pp. 914–926, Nov. 2002.
- [5] Y. Li and P. F. Driessens, "An unsupervised adaptative filtering approach of 2-to-5 channel upmix," in *119th Conv. Audio Eng. Soc.*, New York, USA, Oct. 2005, preprint 6611.
- [6] Producers & Engineers Wing Surround Sound Recommendations Committee, "Recommendations for surround sound production," The National Academy of Recording Arts & Sciences, Tech. Rep., 2004.
- [7] D. T. Yang, C. Kyriakakis, and C. C. Jay Kuo, *High-Fidelity Multichannel Audio Coding*. EURASIP Book Series on Sig. Proc. and Communications, 2004.
- [8] M. Briand, D. Virette, and N. Martin, "Parametric representation of multichannel audio based on Principal Component Analysis," in *120th Conv. Audio Eng. Soc.*, Paris, France, May 2006, preprint 6813.
- [9] R. G. van der Waal and R. N. J. Veldhuis, "Subband coding of stereophonic digital audio signals," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'91)*, Toronto, Canada, May 1991, pp. 3601–3604.
- [10] B. C. J. Moore, *Psychology of Hearing*. Academic Press, 1993.
- [11] M. Bouéri and C. Kyriakakis, "Audio signal decorrelation based on a critical band approach," in *117th Conv. Audio Eng. Soc.*, San Francisco, USA, Oct. 2004, preprint 6291.

EXACT DISCRETE-TIME REALIZATION OF A DOLBY B ENCODING/DECODING ARCHITECTURE

Federico Avanzini

Dipartimento di Ing. dell'Informazione
Università di Padova
Via Gradenigo 6/A, Padova 35131, Italy
avanzini@dei.unipd.it

Federico Fontana

Dipartimento di Informatica
Università di Verona
Strada le Grazie 15, Verona 37134, Italy
federico.fontana@univr.it

ABSTRACT

An algebraic technique which computes nonlinear, delay-free digital filter networks is applied to model the Dolby B in the discrete-time. The model preserves the topology of the analog system, and imports the characteristics of the nonlinear processing blocks which are responsible of the peculiar functioning of Dolby B. The resulting numerical system exhibits qualitatively similar dynamic behavior and performance – full compliance with the Dolby B specifications would be achieved by deriving, from comprehensive data sheets of the system, accurate discrete-time models of the analog processing blocks. Results demonstrate that the computation converges if proper iterative methods are employed.

1. INTRODUCTION

The history of audio effects design traces back to the world of analog circuits. It was not long after the advent of digital architectures that scientists considered the possibility to reproduce in the digital domain the analog and electro-acoustic mechanisms the early audio effects were based upon.

Converting a continuous-time process into a sequence of discrete computations inevitably introduces approximations, which in some cases can generate intolerable problems, like heavy artifacts in the system response, or even instability of the discrete-time model. The *delay-free loop problem* [1, sec. 6.1.3] refers to the presence in a network of feedback paths that are not computable, meaning with this that the computation cannot be executed sequentially due to the lack of pure delays along the loop. This problem can appear in particular during conversion to the digital domain of analog filter networks, or even in digital-to-digital domain transformations (such as frequency-warping mappings [2]).

If the network is linear, various techniques can be used to convert a continuous-time system into an equivalent numerical one, working either in the time or in the Laplace domains. As an example, wave methods [3] and transfer function models [4] have been widely applied to the numerical simulation of acoustic systems. Moreover, a linear network can be always rearranged into a new one in which delay-free paths are solved by composing the filters belonging to them into bigger linear structures that “embed” the loop [1]. Nevertheless there are cases where this rearrangement is deprecated (e.g., situations in which the access to the filter parameters becomes too complicated after the rearrangement). Furthermore, the elimination of a delay-free path implies that all the branches belonging to it cannot be used any longer as input/output points where to inject/extract the signal to/from the system: this

point is particularly relevant in the design of virtual musical instruments by physical modeling.

When nonlinearities exist in the continuous-time system, however, the discretization procedure must preserve stability and must ensure a precise simulation of the nonlinear characteristic. Moreover, if a nonlinearity is part of a delay-free path there is no general procedure to rearrange the loop to realize a new linear structure in which to embed the delay-free path.

A technique to compute linear delay-free paths without topology rearrangement was proposed in [5]. It was applied to warped IIR filter computation [2] and to magnitude-complementary parametric equalizers [6], and generalized to linear filter networks with arbitrary delay-free path configurations [7]. It was then extended to networks containing nonlinear blocks [8, 9]. The technique assumes that each linear and nonlinear block has been already individually modeled in the discrete time domain.

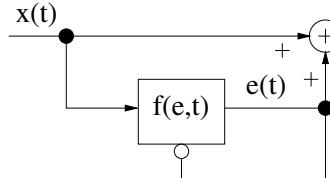
In this paper we analyze the Dolby B codec as a challenging example of analog system that includes nonlinear feedback loops. We show that the system can be realized *exactly* in the discrete-time domain by employing the numerical technique presented in [8, 9]. Sections 2 and 3 review the functioning of the analog system and of existing discrete-time realizations. Sections 4 and 5 discuss the exact realization proposed in this paper and analyze the convergence of the iterative schemes used to solve the nonlinear digital network. Finally, results from numerical simulations of the system are presented in Section 6.

2. THE DOLBY B CODEC

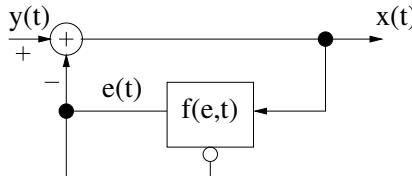
At the mid of the sixties Dolby Laboratories introduced a family of noise reduction systems that had a strong impact on the industry of consumer audio which, at that time, was experiencing the explosion of the compact cassette [10].

Dolby noise reduction systems gave the most successful solution to the problem of noise floor generated by the tape, clearly audible in the mid and high frequencies unless masked by a moderately loud audio message, and, hence, preventing music listening at Hi-Fi standards. While convincingly reducing noise floor, the Dolby coding paradigm did not heavily mask the audio content. In this way music from a Dolby encoded cassette could still be listened to even by a normal tape player [11]. Due to this versatility, Dolby systems marked a commercial advantage against other noise reduction architectures such as DBX, Tel Com by Telefunken and DNL by Philips.

Among the many noise reduction systems licensed by Dolby Laboratories [10], the B architecture was one of the most successful. As its predecessor, the A system, Dolby B is based on the



ENCODER



DECODER

Figure 1: Dolby B codec.

idea that tape noise floor has to be canceled only when the dynamics of the audio signal is too low to mask it, otherwise the music message can be left untouched. For this reason, before recording the signal on the tape, the encoder emphasizes the mid and high frequency range under low dynamics conditions. Coherently, the decoder de-emphasizes the same frequencies during audio tape reproduction under similar dynamic conditions. The overall result is a proportional decrease of the cassette noise.

The Dolby B codec is shown in Figure 1. According to this scheme, the encoded signal $y(t)$ is obtained by summing, to the unencoded signal $x(t)$, a filtered version $e(t)$ of the same unencoded signal:

$$y(t) = x(t) + e(t) = \{1 + f(e, t)\} * x(t). \quad (1)$$

In Eq. (1) we recognize the filter f to be nonlinear and time-varying: both characteristics are mandatory if we want to process the signal depending on its dynamics. More in detail, we see that the encoded message depends on the characteristics of the signal captured at the time-varying filter output (the more obvious choice of reading the dynamics before encoding was implemented by Dolby A, then abandoned). In particular, the higher the amplitude of e , the smoother the high-pass characteristic of f . In the limit case when e has a very pronounced dynamics, then f becomes nearly transparent, i.e., $f(e, t) \approx 1$.

Conversely, and assuming the inverse transfer characteristic of $1 + f(e, t)$ to be stable, the decoder realizes the following nonlinear transfer function against the noisy version $\tilde{y}(t)$ of the encoded signal:

$$\tilde{x}(t) = \tilde{y}(t) - \tilde{e}(t) = \tilde{y}(t) - f(\tilde{e}, t) * \tilde{x}(t). \quad (2)$$

In Eq. (2) we have neglected transmission delay. Noticeably, if no noise or whatever audio artifact superimposes to $y(t)$, i.e., if $\tilde{y}(t) = y(t)$, then decoding is error-free, i.e., $\tilde{x}(t) = x(t)$. Of course noise-free transmission is unrealistic: should it exist, then noise reduction systems would have turned out to be unnecessary.

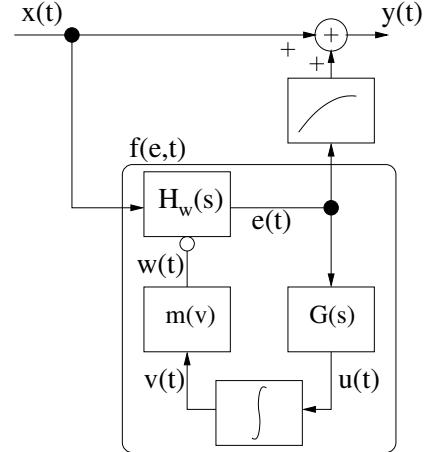


Figure 2: Dolby B encoder.

3. DIGITAL REALIZATIONS OF DOLBY B

Figure 1 immediately shows that implementing a Dolby B encoder in the digital domain is difficult due to the existence of a delay-free path connecting the time-varying filter output to its control input.

A more detailed inspection of the filter (see the schematic in Figure 2, disclosed by Dolby Labs [10]) reveals that control is realized by passing the signal $e(t)$ through a smooth high-pass linear filter $G(s)$ yielding a signal $u(t)$, and, then, through an envelope follower. This stage rectifies the audio message into a control signal $v(t)$ that instantaneously drives, by means of the output w from the map $m(v)$, the high-pass characteristic of a time-varying linear filter $H_w(s)$ whose output, depending on the value taken by w , is finally added to the original signal to form the encoded message.

In this explanation (but not in Figure 2) we have omitted the presence of a compressor, located between the time-varying filter and the adder, whose role is to remove from $e(t)$ overshoots arising when the input suddenly switches from very low to very high dynamics. In this case, a short but audible time window occurs in which the mid and high frequencies in $x(t)$ are mistakenly amplified until the system responds to the high dynamics by positioning the time-varying filter $H_w(s)$ to almost transparent behavior. This compressor, hence, prevents such high-frequency boosts to come out from the system by filtering them out from $e(t)$ until its amplitude falls within the correct dynamic range. Otherwise, e.g. for medium and low dynamics, the compressor is transparent.

In spite of this fairly complicated network, a digital encoder can be (at least in principle) realized by assuming that $v(t)$ is enough slowly varying to allow for the inclusion of a fictitious unit delay between the envelope follower and the map. Since the signal rectification performed by the envelope follower indeed results in a slowly varying output, then a computational scheme for a discrete-time Dolby B encoder can be figured out which, starting from an initial state plus an initial value for w (e.g. those resulting from a null input) computes e known x and, then, the new state and samples for all the block outputs, including y .

It is clear that this computing procedure requires that every analog processing block in Figure 2 has been somehow converted to the digital domain, possibly taking the unavoidable distortions introduced by the presence of the fictitious delay into account.

The digital conversion of the decoder is a much more com-

plicate matter. If one reports the internal structure of $f(e, t)$ (see Figure 2) into the block scheme of the Dolby B decoder (see Figure 1), then a double feedback can be immediately noticed: one at the subtraction point $e(t) - y(t)$ and one, already explained, at the input control of the time-varying filter. The former, in particular, cannot be translated into a corresponding discrete time feedback loop by including a unit delay, since the audio signal flowing along it is by no means slowly varying. Furthermore, both feedbacks include nonlinear blocks in between.

Alternative structures aimed at realizing a real-time Dolby B decoder in the digital domain have been successfully engineered by Philips [12] and STMicroelectronics [13, 14]. Such structures include a fictitious time delay to account for the instantaneous feedback at the time-varying filter control, and rearrange the other feedback loop into an alternative, feed-forward topology. As a consequence, most of the parameters characterizing the circuitry of the Dolby B analog blocks must be carefully adapted for the resulting digital decoder to match the specification requirements imposed by Dolby Labs.

In particular, the discrete-time Dolby B proposed by STMicroelectronics [13, 14] contains careful transpositions of all analog stages into corresponding digital blocks. The transfer characteristic $X(s)/Y(s)$ is modeled by a time-varying feed-forward digital filter whose transfer function is given by

$$\frac{X(z)}{Y(z)} = \frac{1}{1 + H_w(z)}. \quad (3)$$

The filter $G(z)$, digital counterpart of $G(s)$, is driven by a discrete-time model of the transfer characteristic $E(s)/Y(s)$:

$$\frac{E(z)}{Y(z)} = \frac{H_w(z)}{1 + H_w(z)} = 1 - \frac{X(z)}{Y(z)}. \quad (4)$$

In this way a single time-varying digital filter, whose transfer function is given by (3), is used both to compute the system output and, via Eq. (4), to feed the filter $G(z)$. Obviously, such a realization requires to re-design the nonlinear map $m(v)$ properly.

4. EXACT TRANSPOSITION OF THE ENCODING/DECODING NETWORK

In [8] we have presented an algebraic technique which allows to compute every nonlinear filter network, regardless of the existence of delay-free paths located in between processing blocks. The same technique also promises to add insight on the inherent stability properties of a nonlinear system structured, like Dolby B, as an interconnection of input/output blocks [9].

In order to exploit this technique to realize a Dolby B architecture exactly, we will develop a ‘mock-up’ of the system which, in particular, preserves all the nonlinearities and topological details of the analog structure. However, we will avoid to carefully transpose into discrete time the transfer characteristics of the Dolby B analog blocks. Instead, we will rely on simplified digital versions of the same blocks without sacrificing in generality.

As a result, we have realized a digital “Dolby B-like” system working at 44100 Hz which, in spite of its resemblance to the real Dolby B both in structure and performance, does not comply with the requirements of Dolby Labs due to discrepancies between the transfer characteristics of the analog blocks forming the original system and their digital transpositions. As to the question whether

a Dolby B codec can be *exactly* realized in discrete time, our answer is ‘yes’ as far as every block of the analog architecture is exactly translated into the digital domain using proper transformation methods [15, 1].

In the remainder of this section we adopt the notation used in [8, 9]: linear blocks are defined through their transfer functions H_i and nonlinear blocks are defined through their nonlinear characteristics f_i , with $i = 1, 2, \dots$. Inputs and outputs are denoted as x_{Li}, y_{Li} (linear blocks) and x_{Ni}, y_{Ni} (nonlinear blocks). We employ the following digital blocks (refer to Figure 2 for nomenclature of the continuous-time blocks):

- The time-varying filter $H_w(s)$ is replaced by a digital high-frequency shelving filter [1]

$$\begin{aligned} H_w(z) &= H_1(z) + wH_2(z) \\ &= \frac{1}{2}\{1 - A(z)\} + \frac{w}{2}\{1 + A(z)\} \end{aligned} \quad (5)$$

with

$$A(z) = \frac{\alpha - z^{-1}}{1 - \alpha z^{-1}} \quad \text{and} \quad \alpha = -0.9. \quad (6)$$

In this filter the coefficient w controls the high-frequency gain. We will consider this gain to be a time-varying parameter $w(nT)$.

- The high-pass filter $G(s)$ in the feedback control loop is replaced with a digital equivalent [14]:

$$H_3(z) = \frac{U(z)}{E(z)} = \frac{b_0 - b_1 z^{-1}}{1 - a_1 z^{-1}}, \quad (7)$$

with $b_0 = 0.55$, $b_1 = -0.46$, $a_1 = 0.014$;

- The envelope follower f is replaced by a digital equivalent that rectifies the signal $u(nT)$ according to the nonlinear function [16]:

$$\begin{aligned} v(nT) &= f_1(u(nT), v(nT - T)) \\ &= \{1 - b(nT)\}|u(nT)| + b(nT)v(nT - T) \end{aligned} \quad (8)$$

with

$$b(nT) = \begin{cases} b_{up} & |u(nT)| > v(nT - T) \\ b_{down} & \text{otherwise} \end{cases} \quad (9)$$

in which $b_{up} = 0.995$ and $b_{down} = 0.9998$;

- The control on the time-varying shelving filter is realized as a nonlinear map that defines the gain $w(nT)$ as a function of the envelope $v := y_{N1}$ and multiplies w by the output y_{L2} of H_2 :

$$f_2(y_{N1}(nT), y_{L2}(nT)) = y_{L2}(nT) \cdot w(y_{N1}(nT)), \quad (10)$$

with

$$w(y_{N1}) = \begin{cases} a - b(y_{N1})^e & 0 \leq y_{N1} \leq 3 \\ 1 + c/(1 - dy_{N1}) & \text{otherwise} \end{cases} \quad (11)$$

This equation was determined empirically by observing that w must have high values for small values of the envelope, while $w \rightarrow 1$ (the high-pass filter becomes transparent) for high values of the envelope. Parameters in Eq. (11) were determined by interpolating over two values and by requiring w to be $\mathcal{C}^{(1)}$ at $y_{N1} = 1$. The chosen values are $a = 7.8$, $b = 1.4 \cdot 10^{-14}$, $e = 30$, $c = 0.57$, $d = 2.86$.

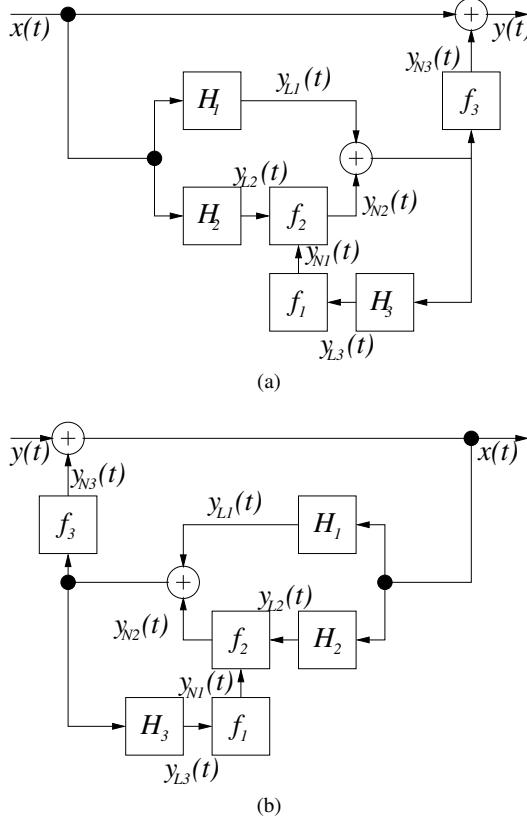


Figure 3: Digital Dolby B-like system; (a) encoder; (b) decoder.

- The overshoot characteristics realizes the following nonlinear compression function:

$$f_3(e(nT)) = \quad (12)$$

$$\begin{cases} e(nT), & |e(nT)| \leq T_1 \\ T_1 + \frac{T_2 - T_1}{T_3 - T_1} \{e(nT) - T_1\}, & T_1 < |e(nT)| \leq T_3 \\ T_2, & |e(nT)| > T_3 \end{cases}$$

with $T_1 = 80$, $T_2 = 100$, $T_3 = 200$. We give reason of such parameters by considering that our system is designed to work in the dynamic range $(-10, 30)$ dB. Hence, the amplitude of $e(t)$ is considered acceptable until staying within 40 dB, i.e., absolute value equal to 100, otherwise an overshoot caused by the time-varying filter is detected.

Using the characteristic (6) for the time-varying filter guarantees that the transfer function $1 + H_w(z)$ is minimum-phase for every choice of w . This fact in practice ensures the stability of the decoder, where the output depends on the input according to the relation explained by (3).

In summary, both the encoder and the decoder comprise three linear blocks H_i and three nonlinear blocks f_i . The update of linear blocks is written in matrix form as

$$\mathbf{y}_L[n] = \mathbf{B}\mathbf{x}_L[n] + \mathbf{q}[n], \quad (13)$$

where the vectors \mathbf{x}_L and \mathbf{y}_L collect inputs and outputs of blocks, $\mathbf{q}_i[n] = b_{1,i}\mathbf{x}_{Li}[n-1] + a_{1,i}\mathbf{y}_{Li}[n-1]$ collect contributions of

past components, and \mathbf{B} is a diagonal matrix containing the linear coefficients $b_{0,i}$.

The update of nonlinear blocks is written in matrix form as

$$\mathbf{y}_N[n] = \mathbf{f}(\mathbf{x}_N[n], \mathbf{p}[n]), \quad (14)$$

where the vectors \mathbf{x}_N , \mathbf{y}_N , collect inputs and outputs of the nonlinear blocks, while p_i contains the contribution of historical components in the functions. According to equations (9,11,12), f_1 has a non-null historical component $p_1[n] = v(nT - T)$, while f_2 and f_3 are algebraic nonlinearities.

The topology of the network and the external inputs to each block are specified by the equation:

$$\begin{vmatrix} \mathbf{x}_N \\ \mathbf{x}_L \end{vmatrix} = \mathbf{C} \begin{vmatrix} \mathbf{y}_N \\ \mathbf{y}_L \end{vmatrix} + \begin{vmatrix} \mathbf{u}_N \\ \mathbf{u}_L \end{vmatrix}, \quad \mathbf{C} = \begin{vmatrix} \mathbf{C}_{NN} & \mathbf{C}_{NL} \\ \mathbf{C}_{LN} & \mathbf{C}_{LL} \end{vmatrix}, \quad (15)$$

where $\mathbf{u}_{N,L}$ represent external inputs to the nonlinear and linear blocks, respectively. The topology matrix \mathbf{C} specifies connections between block inputs and outputs, and the four sub-matrices $\mathbf{C}_{NN, NL, LN, LL}$ account for nonlinear-to-nonlinear, linear-to-nonlinear, nonlinear-to-linear, and linear-to-linear connections, respectively. The encoder and the decoder systems differ only on the matrix \mathbf{C} specifying the topology:

$$\mathbf{C} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & c_{5,3} & 0 & 0 & 0 \\ 0 & 0 & c_{6,3} & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{vmatrix}, \quad (16)$$

where $c_{5,3} = c_{6,3} = 0$ for the encoder, and $c_{5,3} = c_{6,3} = -1$ for the decoder. Note that in both cases there is no direct connection between the input and the output of the same block.

In summary, the encoder and the decoder satisfy all the hypotheses for the applicability of the procedure described in [9]. The digital systems representing the encoder and the decoder are represented in Figure 4(a) and 4(b), respectively.

5. ANALYSIS OF CONVERGENCE

We have shown in previous works [8, 9] that straightforward algebra leads to the following equations for the system inputs:

$$\mathbf{x}_L = \mathbf{F}_{LL}^{-1} \mathbf{C}_{LN} \mathbf{f}(\mathbf{x}_N, \mathbf{p}) + \mathbf{F}_{LL}^{-1} (\mathbf{C}_{LL} \mathbf{q} + \mathbf{u}_L), \quad (17a)$$

$$\mathbf{x}_N = \mathbf{W}_1 \mathbf{f}(\mathbf{x}_N, \mathbf{p}) + \mathbf{W}_2 \mathbf{q} + \mathbf{W}_3 \mathbf{u}_L + \mathbf{u}_N, \quad (17b)$$

where the matrices \mathbf{F}_{LL} , \mathbf{W}_i are defined from \mathbf{B} and \mathbf{C} (see [9] for details). From equations (14) and (17b), one can write

$$\mathbf{y}_N[n] = \mathbf{f}(\mathbf{W}_1 \mathbf{y}_N[n] + \tilde{\mathbf{x}}_N[n], \mathbf{p}), \quad (18)$$

where $\tilde{\mathbf{x}}_N[n] = \mathbf{W}_2 \mathbf{q}[n] + \mathbf{W}_3 \mathbf{u}_L[n] + \mathbf{u}_N[n]$ collects the contribution of known quantities to the input \mathbf{x}_N .

Note that the only unknown in (18) is $\mathbf{y}_N[n]$. The presence of delay-free paths in the network causes \mathbf{W}_1 to be non-null, and consequently Eq. (18) defines $\mathbf{y}_N[n]$ implicitly. The network is computable if $\mathbf{y}_N[n]$ can be computed from (18). More precisely, the computation can be decomposed into the following steps:

- $\mathbf{x}_N[n]$ and $\mathbf{y}_N[n]$ are computed from (17b) and (18) using external inputs $\mathbf{u}[n]$ and historical components $\mathbf{p}[n], \mathbf{q}[n]$;

2. $x_L[n]$ and $y_L[n]$ are computed from (17a) and (13), respectively;
3. $p[n+1]$ and $q[n+1]$ are computed from known variables; in particular $q[n+1]$ can be computed by feeding each filter with a null signal [5]. No computation is needed if the filters are realized in transposed direct form [1, 7].

In [8] we discussed the use of Newton-Raphson (NR) iteration for the solution of step 1. in the computational scheme outlined above. The NR algorithm [17] searches a local zero of the function

$$f(\mathbf{W}_1 \mathbf{y}_N + \tilde{\mathbf{x}}_N, p) - \mathbf{y}_N. \quad (19)$$

In [9] we have proposed an approach based on fixed-point (FP) iteration [17]. In this case we try to solve the fixed-point problem $\mathbf{y}_N = g_p(\mathbf{y}_N)$, where the function g_p has been defined as

$$g_p(\mathbf{y}_N) = f(\mathbf{W}_1 \mathbf{y}_N + \tilde{\mathbf{x}}_N, p). \quad (20)$$

FP iteration is preferable over NR iteration in terms of efficiency and ease of implementation. However, convergence of FP iteration is ensured only if the nonlinear function g_p given in equation (20) satisfies more restrictive hypothesis. Namely, g_p must be a *contraction*, i.e. it possesses a Lipschitz constant $0 \leq L_{g_p} < 1$ such that $\|g_p(\mathbf{y}) - g_p(\mathbf{y}^*)\| \leq L_{g_p} \|\mathbf{y} - \mathbf{y}^*\|$. We have shown [9] that L_{g_p} can be estimated from above as $L_{g_p} \leq L_{f_p} \|\mathbf{W}_1\|$, where L_{f_p} is a Lipschitz constant for $f(\cdot, p)$. In the remainder of this section we refine this analysis and apply it to the Dolby B. Specifically we show that FP iteration can be applied to the encoder topology but not to the decoder topology.

We restrict our analysis to the second component g_2 of g_p , as it is easy to show that this is the critical one. The function g_2 is $C^{(1)}$, since $g_2(\mathbf{y}_N) = f_2(\mathbf{W}_1 \mathbf{y}_N + \tilde{\mathbf{x}}_N)$. In order to estimate whether g_2 is a contraction, it suffices to estimate its derivatives $\partial g_2 / \partial y_{Ni}$ ($i = 1, 2, 3$) [9]. If the condition

$$\sup_{\mathbf{y}_N \in Y} \left| \frac{\partial g_2}{\partial y_{Ni}} \right| > 1 \quad (21)$$

holds for some i in a given range Y , then g_2 is locally not a contraction in Y and FP iteration is not convergent. The derivatives are computed as

$$\frac{\partial g_2}{\partial y_{Ni}} = \sum_j \frac{\partial f_2}{\partial x_{Nj}} \frac{\partial x_{Nj}}{\partial y_{Ni}} = \sum_j \frac{\partial f_2}{\partial x_{Nj}} [\mathbf{W}_1]_{j,i}. \quad (22)$$

Substituting values of the coefficients $[\mathbf{W}_1]_{j,i}$ for the encoder and the decoder yields

$$\frac{\partial g_2}{\partial y_N} = \begin{cases} \left[y_{L1} \cdot \frac{\partial w}{\partial v}, 0, 0 \right] & \text{(encoder)} \\ \left[y_{L1} \cdot \frac{\partial w}{\partial v}, 0, 0.95 \cdot w(v) \right] & \text{(decoder)} \end{cases} \quad (23)$$

Note in particular that the third component of this derivative is not null for the decoder: this is a consequence of the decoder topology, in which the output from the compressor f_3 influences the input to the time-varying high-frequency gain f_2 through the main feedback loop. Unfortunately $w(v)$ is well above 1 in a neighborhood of the origin, therefore $\partial g_2 / \partial y_{N3}$ is locally greater than 1 for the decoder and FP iteration cannot be applied safely. On the contrary, g_p is globally a contraction for the encoder, and FP iteration can be applied in this case.

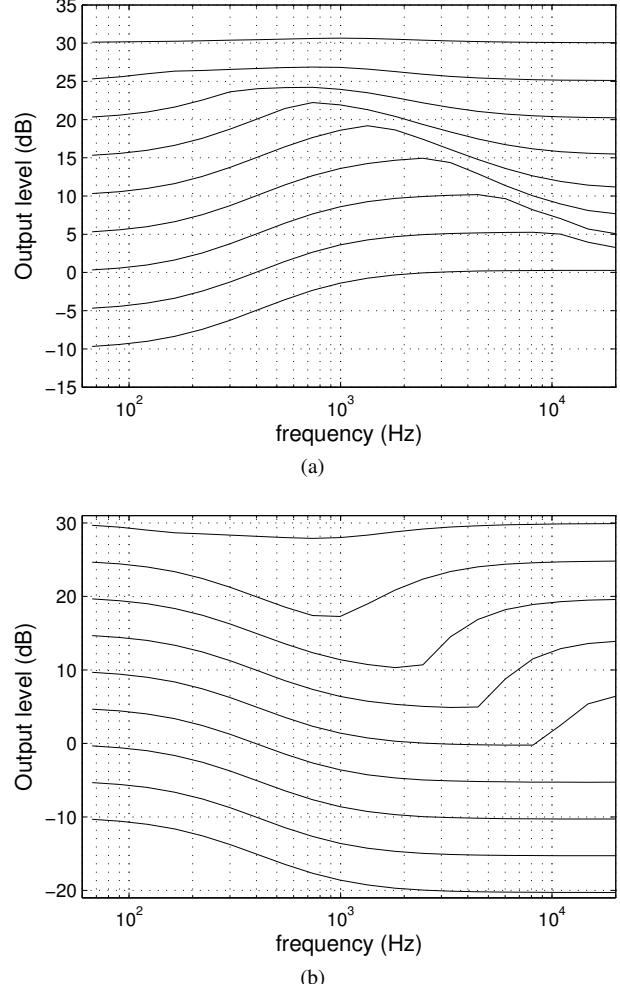


Figure 4: Digital system response; (a) encoder, (b) decoder.

6. NUMERICAL SIMULATIONS

The Dolby B-like codec was implemented as a set of Octave/Matlab functions. We first experimentally verified the correctness of the analysis reported in the previous section: numerical simulations show that FP iteration converges when the system topology is the one given in figure 4(a), while this is not the case for the topology of figure 4(b) in accordance to our analytical study on g_p .

We then tested the response of the encoder and the decoder by feeding both systems with a 9×20 matrix of sinusoidal inputs $x(t)$, containing 9 equally spaced input levels between -10 and 30 dB, and 20 exponentially spaced frequencies between 20 and 20000 Hz. In light of the results presented above, NR iteration was used for the solution of both systems.

For each input sinusoid, the average number of NR iterations per sample was computed. Results show that both the encoder and the decoder require in average ~ 2 NR iterations per sample. The average iteration number, not surprisingly, increases with the frequency of the input sinusoid, with a maximum of 2.25 for the encoder and 2.38 for the decoder. The same number decreases for higher input levels, reaching a minimum of 1.21 for the encoder

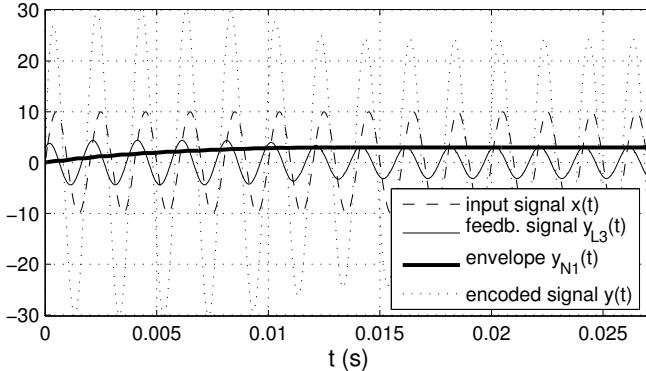


Figure 5: Example of an encoded sinusoid.

and 1.28 for the decoder: this is also a predictable effect, since the lower the input level is, the higher is the time-varying gain w .

The responses of the two systems are plotted in figure 5(a) and 5(b), respectively. For each element of the input matrix, the frequency and amplitude of the resulting output sinusoid determined the interpolation point which the responses of figure 4 intersect. Note in particular that for very low input levels the encoder provides a maximum of about 10 dB boost above 4000 Hz, which is qualitatively in accordance with the specification requirements imposed by Dolby Labs [10]. Note also that the systems are transparent to high input signal levels.

Figure 5 provides an example of the encoding process on a test sinusoid at 20 dB and 500 Hz. The encoded sinusoid is boosted, but as the envelope signal y_{N1} rises up the gain of the shelving filter is lowered and the output level is consequently reduced.

Numerical simulations made by cascading the decoder after the encoder show that the decoded sinusoids are exact reconstructions of the input signals. The presence of a very short and small transient encoding/decoding error, lasting less than five samples and invisible in Figure 5, disappears as soon as the initial input discontinuity arising when the null signal turns into a sinusoid is forgotten by the digital codec.

7. CONCLUSIONS

An exact realization of a Dolby B encoding/decoding architecture was made possible by implementing a previously known algebraic procedure that allows to compute nonlinear filter networks containing delay-free paths. The use of this procedure sheds light on some computational aspects of the Dolby B: specifically, the analytical results reported in section 5 provide a quantitative link between the inherent structure of the system and the robustness of the computational procedure.

Ongoing work is focusing on investigating the properties of the computational procedure further, and specifically on the relations between the topology of a generic nonlinear numerical system (17a,17b) and its computational behavior.

8. REFERENCES

- [1] S. K. Mitra, *Digital Signal Processing. A computer-Based Approach*. New York: McGraw-Hill, 1998.
- [2] 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, Nov. 2000.
- [3] S. Bilbao, *Wave and Scattering Methods for the Numerical Integration of Partial Differential Equations*. New York: John Wiley & Sons., 2004.
- [4] L. Trautmann and R. Rabenstein, *Digital Sound Synthesis by Physical Modeling Using the Functional Transformation Method*. New York: Kluwer Academic/Plenum Publishers, 2003.
- [5] A. Härmä, "Implementation of frequency-warped recursive filters," *EURASIP Signal Processing*, vol. 80, no. 3, pp. 543–548, Mar. 2000.
- [6] F. Fontana and M. Karjalainen, "A digital bandpass/bandstop complementary equalization filter with independent tuning characteristics," *IEEE Sig. Proc. Letters*, vol. 10, no. 4, pp. 88–91, Apr. 2003.
- [7] F. Fontana, "Computation of linear filter networks containing delay-free loops, with an application to the waveguide mesh," *IEEE Trans. Speech and Audio Proc.*, vol. 11, no. 6, pp. 774–782, Nov. 2003.
- [8] F. Fontana, F. Avanzini, and D. Rocchesso, "Computation of nonlinear filter networks containing delay-free paths," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, Oct. 2004, pp. 113–118.
- [9] F. Avanzini, F. Fontana, and D. Rocchesso, "Efficient computation of nonlinear filter networks with delay-free loops and applications to physically-based sound models," in *Proc. 4th Int. Workshop on Multidimensional Systems (NDS 2005)*, Wuppertal, July 2005, pp. 110–115. [Online]. Available: <http://www.dei.unipd.it/~avanzini>
- [10] Dolby Laboratories Inc., "San Francisco, CA," Retrieved June 29th, 2006, [Online] <http://www.dolby.com>.
- [11] Audiotools, "Noise Reduction Systems," Retrieved June 29th, 2006, [Online] <http://audiotools.com/noise.html>.
- [12] U. Sauvagerd, "Circuit arrangement for influencing the frequency response of a digital audio signal," Oct. 1999, US Patent US5974156.
- [13] F. Fontana and M. Bricchi, "Process for noise reduction, particularly for audio systems, device and computer program product therefore," US Patent US2003004591, Jan. 2003.
- [14] M. Bricchi and F. Fontana, "A process for noise reduction, particularly for audio systems, device and computer program product therefore," EU Patent EP1271772, Jan. 2003.
- [15] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.
- [16] P. Dutilleux and U. Zölzer, "Nonlinear processing," in *DAFx – Digital Audio Effects*, U. Zölzer, Ed. New York: J. Wiley & Sons, 2002, pp. 93–136.
- [17] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*. New York: John Wiley & Sons, 1991.

REAL-TIME BAYESIAN GSM BUZZ REMOVAL

Han Lin, Simon Godsill

Signal Processing Group, Department of Engineering
 University of Cambridge
 Trumpington Street, Cambridge CB2 1PZ, UK
 {hl309|sjg}@eng.cam.ac.uk

ABSTRACT

In this paper we propose an iterative audio restoration algorithm based on an autoregressive (AR) model with modeling of the noise pulse template to detect and restore Cell-phone electromagnetic interference (EMI) patterns known as “GSM buzz”. The algorithm is purely software based and does not require the aid of any hardware providing side information. The only assumption is that individual pulses are similar to scaled versions of the known template. With this assumption, the algorithm can fully detect and restore noisy interference signals in real time with almost no audible artifacts and improve the signal to noise ratio by as much as 50dB.

1. INTRODUCTION

TDMA, CDMA and GSM are the three most common digital signals used by cellular service providers. These wireless transmission protocols send out strong electromagnetic pulses as control or data signals during the registration process as well as during receiving, transmitting, and hand-over procedures. Once transmitted, interference pulses are received by audio amplifiers and line-in circuits, which generates audible sound distortion in the form of “GSM buzz” (217 Hz for full rate and 108 Hz for half rate) [1] [2].

Although audio products such as car stereos, telephones, recorders, portable audio players, medical devices [3], and hearing aids [4] all suffer from this kind of interference, such interference is often ignored. However, as the popularity of cellular phones grows, the problem can no longer be overlooked, especially for high-end consumer audio products and hearing aids that place emphasis on the clarity of the sound.

Reducing Cell-phone transmission power, changing the transmission protocol, equipping a telecoil [5], as well as shielding the audio circuit can alleviate the problem. However, these solutions cannot be implemented without changing the existing hardware designs, which can often be very difficult and expensive. In addition, building an interference-free audio circuit that is immune from the noises caused by electromagnetic pulses transmitted using various protocols (GSM, TDMA, and CDMA) is nearly impossible. Further, interference can happen in any phase of the audio transmission path, and once the audio waveform is interfered with, there is no existing solution to filter out the noise without distorting the original audio.

For many industry applications, internal subtractive noise cancellation where a regenerated noise pulse is fed back to the circuit and subtracted from the signal provides an effective solution [6] [7]. However, the design is only practical when the regenerated noise pulse is exact. Generally this is achieved with the aid of a hardware detector and a synchronized hardware clock generator.

Other solutions such as notch filter based restoration [2] typically require frequency domain operation, produce a metallic sound, and most importantly, rely on the fact that individual pulses arrive at exact and regular intervals (217 Hz for full rate) without any delays, which is rare when a mobile phone is trying to connect with the base stations during call initialization and hand-over [8].

Many efficient real-time audio restoration algorithms are based on autoregressive (AR) model, where a stationary random audio signal is modeled as the output of an order P all-pole filter excited by white noise. In the AR model, the output of a linear time invariant filter is restricted to a weighted sum of past output values and a white noise input $e_{AR}(n)$ [9].

$$s(n) = \sum_{i=1}^P a(i)s(n-i) + e_{AR}(n)$$

A statistical signal processing approach based on the AR model with prior knowledge of noise pulse template is applied to restore Cell-phone interfered audio files. The approach intends to detect and remove Cell-phone interfered noise pulses before the audio data is played back on the speaker/headphone, or after the audio files are recorded. The restoration process is extremely effective and it is performed at the last stage of the audio circuit.

2. ANALYSIS OF NOISE PULSE AND THE RESTORATION MODEL

When Cell-phone induced electromagnetic interference pulses, in the form of a square wave, interfere with an audio signal, the pulses are superimposed onto the original waveform. A typical corrupted, Cell-phone interfered, audio signal $x(n)$ appears as in Fig. 1. We can see in Fig. 2 a typical noise pulse $g(n)$ consists of two main parts: a *central pulse* and a *decaying tail*. The *central pulse* is caused by the electromagnetic (EM) excitation and the *decaying tail* is due to the capacitance associated with the audio circuit. Therefore, we consider an idealized model of the corrupted signal. This model is used in section 3 to devise an detector. In section 4, an iterative restoration procedure loosely based on the idealized model is provided.

$$x(n) = bg(n-m) + s(n) + e(n) \quad (1)$$

Where:

- $x(n)$ is the observed corrupted, Cell-phone interfered signal,
- $g(n)$ is the known interference pattern template of Cell-phone interference noise pulse,

- b is a constant scaling factor to compensate for the amplitude difference between noise and template,
- $e(n)$ is a white output noise,
- $s(n)$ is the original signal without interference.

We can see that if the exact start location m of the Cell-phone interference template $g(n)$ is known, and if we can also determine the scaling factor b , we can then restore the original signal $s(n)$.

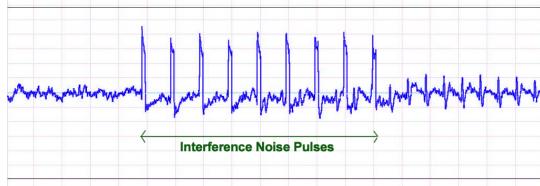


Figure 1: $x(n)$ - A typical section of cell-phone interfered audio.

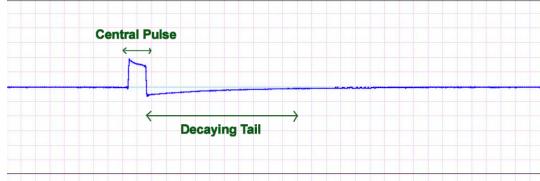


Figure 2: $g(n)$ - A typical cell-phone interference noise pulse.

3. DETECTION OF NOISE PULSES

Numerous detection methods can be used to determine the exact location of the noise pulses, for example:

1. Hardware electromagnetic wave detector,
2. Threshold detection from signal,
3. Threshold detection from signal slope,
4. Cross correlation/matched filter detector,
5. Bayesian step detector [10] (chapter 5),
6. Autoregressive (AR) detector [9] (chapter 5),
7. The *Bayesian template detector* (see section 3.1)

For real-time applications, a hardware electromagnetic wave detector can be an effective solution. For software implementations, threshold detection from signal amplitude can be a simple and efficient method. Threshold detection from signal slope can also be very effective.

3.1. The Bayesian Template Detector

To increase accuracy, the exact location of the *central pulse* can be calculated with a *Bayesian template detector* model. The model helps to predict the probability associated with each possible pulse location. We can then detect the location of the noise pulse by determining the value of m which produces the maximum probability, where m is the start of the noise pulse. Here we create

the *Bayesian template detector* model by simplifying Eq. (1) with $e(n) = 0$:

$$x(n) = bg(n-m) + s(n) \quad (2)$$

$s(n)$ is assumed to be autoregressive:

$$s(n) = \sum_{i=1}^P a(i)s(n-i) + e_{AR}(n)$$

And we can write:

$$\mathbf{e}_{AR} = \mathbf{A}\mathbf{s}$$

\mathbf{A} is a $((N-P) \times N)$ matrix containing autoregressive coefficients (see [9] for detailed description of \mathbf{A}). We can then rewrite Eq. (2) as:

$$\mathbf{e}_{AR} = \mathbf{A}(\mathbf{x} - bg)$$

Where $\mathbf{g} = [g(-m) \ g(-m+1) \ \dots \ g(N-1-m)]^T$. We assume \mathbf{e}_{AR} is a zero mean independent Gaussian vector with variances σ_1^2 , i.e. $P(\mathbf{e}_{AR}(n)) = N(0, \sigma_1^2)$. b is a Gaussian variable with mean b' and variance $k\sigma_1^2$, i.e. $P(b) = N(b', k\sigma_1^2)$. k is generally a large constant. We now obtain the equation for $P(\mathbf{x}|m, b)$:

$$P(\mathbf{x}|m, b, \sigma_1) \propto P_{e_{AR}}(\mathbf{A}(\mathbf{x} - bg)) \propto (2\pi\sigma_1^2)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma_1^2}(\mathbf{e}_{AR}^T \mathbf{e}_{AR})}$$

We wish to integrate out parameters b and σ_1 in the detector to obtain an equation in just variable m . We define $\mathbf{u} = \mathbf{Ax}$ and $\mathbf{v} = \mathbf{Ag}$. We can then define the probability model for the *Bayesian template detector*:

$$P(m|\mathbf{x}, \mathbf{g}) = \int p(\mathbf{x}|m, b, \sigma_1)p(m)p(b)p(\sigma_1)dbd\sigma_1 \quad (3)$$

Assigning uniform prior to $p(m)$ and Jeffrey's prior $\frac{1}{\sigma_1}$ for $p(\sigma_1)$ in Eq. (3), σ_1 can be integrated out using the gamma integral [11], b can be obtained using properties of Gaussian integrals, and after rearranging, we have the general solution for the *Bayesian template detector*:

$$P(m|\mathbf{x}, \mathbf{g}) \propto \frac{1}{k\sqrt{\mathbf{v}^T \mathbf{v} + \frac{1}{k^2}}} \cdot \left[\frac{(\mathbf{u}^T \mathbf{v} + \frac{1}{k^2}b')^2}{(\mathbf{v}^T \mathbf{v} + \frac{1}{k^2})} - (\mathbf{u}^T \mathbf{u} + \frac{1}{k^2}b') \right]^{-(\frac{N-P-1}{2})}$$

Empirically, k can be set to a large value and b' is generally zero due to the fact that b can be both positive or negative and no particular value is favored *a priori*.

3.2. Other Detection Methods

Other suitable detection methods include cross-correlation detector, Bayesian step detector, and autoregressive (AR) detector, which are summarized below: A cross-correlation detector correlates the observed signal $x(n)$ with the Cell-phone interference template $g(n)$. A Bayesian step detector was described in detail in [10]. For example, if we assume the observed signal $x(i)$ has mean μ_1 before change point m and mean μ_2 after point m ,

$$x(i) = \begin{cases} \mu_1 + \varepsilon(i) & \text{if } i < m \\ \mu_2 + \varepsilon(i) & \text{otherwise} \end{cases}$$

The solution of the simplified Bayesian step detector is then:

$$p(m|\mathbf{x}) \propto \frac{[\mathbf{x}^T \mathbf{x} - \mathbf{x}^T \mathbf{L} (\mathbf{L}^T \mathbf{L})^{-1} \mathbf{L}^T \mathbf{x}]^{-\frac{(N-M)}{2}}}{\sqrt{\det(\mathbf{L}^T \mathbf{L})}}$$

where \mathbf{L} has two columns. The first column contains m rows of 1s followed by $N-m$ rows of 0s and the second column consists of m rows of 0s followed by $N-m$ rows of 1s. The autoregressive (AR) detector was described in detail in [9] with the following steps:

1. Calculate prediction error $\epsilon_n = x_n - \sum_{i=1}^P a_i x_{n-i}$, where x_n is the observed data, ϵ_n is a white noise input, a_i is the AR coefficient, and P is the AR model order,
2. If $|\epsilon_m| > r\sigma_e$, where r is around 3 and σ_e is the standard deviation of the white noise input, m is a possible location for the start of the *central pulse*.

3.3. Comparison of the Different Detectors

For the *Bayesian template detector* and the Bayesian step detector, we plot the probability of noise pulse location according to different possible locations m . For the AR detector, the errors ϵ_n are plotted. On the other hand, for the cross-correlation detector, the aligned cross-correlation results are plotted. For the plots described above, if the values are above a certain threshold, they show indications of possible noise pulse locations. The maximum value indicates the most probable noise pulse location. We can see in Fig. 3 that the Bayesian template detector and the cross correlation detector perform best in finding the noise pulse locations. On the other hand, Fig. 10 shows the AR detector and the simple Bayesian step detector will try to detect both the rising and the falling edges of the noise pulse. This sometimes causes confusion and sacrifices the performance of the detector. Empirical results show that the Bayesian template detector is most precise and robust for most scenarios owing to the fact that location m tends to show up as a very sharp peak. In addition, the Bayesian template detector can be very flexible in the sense that by setting $g(n-m)$ to zero, we can also determine the probability of the "no interference" condition.

4. REMOVAL OF CELL-PHONE INTERFERED NOISE PULSES WITH THE AR TEMPLATE INTERPOLATOR

Once the exact locations of the Cell-phone interference template $g(n)$ are known, say m' , we can attempt to remove the noise pulses. We propose a sequential iterative *AR template interpolator* that restores the noise pulses one by one. We first divide the observed corrupted signal $x(n)$ into overlapping frames. The frames should be small enough so that only one pulse can be present at a time. Then for each frame, find the *central pulse* location with any of the detection methods mentioned above. We then restore the *central pulse* location with the AR model and obtain the initial estimate $s^0(n)$ of the clean original signal $s(n)$. Subsequently, we subtract $s^0(n)$ from the observed corrupted signal $x(n)$ to determine the estimated interference signal $r^0(n)$. Further, we fit the interference template $g(n-m')$ to this estimated interference signal $r^0(n)$ to determine the first scaling factor b^1 . Then we subtract the scaled template $b^1 g(n-m')$ from the observed corrupted signal $x(n)$ and interpolate over the *central pulse* location again to re-estimate the original signal. The algorithm can then be iterated

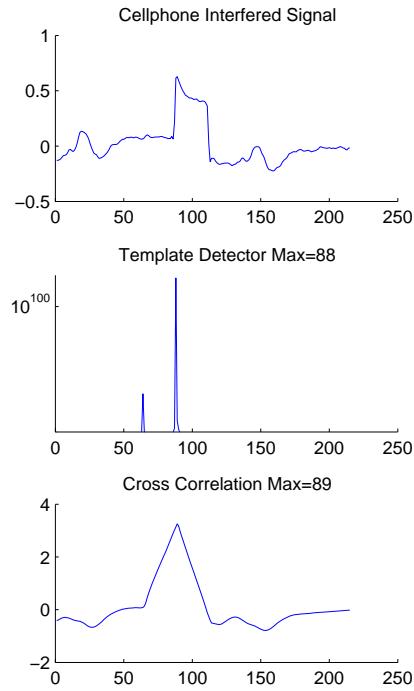


Figure 3: *Detection results of Log of Bayesian template detector and cross correlation detector.*

by using this "re-estimated" signal to determine the scaling factor, which later determines another estimate of the original signal and so on. The exact steps of the *AR template interpolator* are described below:

1. **LSAR** interpolation over the central region of the noise pulse, according to the formula:

$$\mathbf{s}^i = \mathbf{LSAR}(x(n) - b^i g(n-m')) \quad (4)$$

where i starts from 0 and $b^0 = 0$. Therefore **LSAR** initially restores the *central pulse* (see Fig. 2) location using the AR model [9] ($x(n) = \sum_{i=1}^P a(i)x(n-i) + e_{AR}(n)$) with a low order (typical $P = 20$). We know from [9], for example, the solution for the LSAR interpolator is

$$\mathbf{s}^0 = -(\mathbf{A}_{(i)}^T \mathbf{A}_{(i)})^{-1} \mathbf{A}_{(i)}^T \mathbf{A}_{-(i)} \mathbf{x}_{-(i)}.$$

See [9] for detailed descriptions of \mathbf{A} , $\mathbf{A}_{(i)}$, and $\mathbf{A}_{-(i)}$. $\mathbf{x}_{-(i)}$ denotes known/ uncorrupted samples of the observed signal \mathbf{x} . This is the initial estimate of the original signal $s^0(n)$ (see Fig. 4). Note that $s^0(n)$ is the result of an LSAR interpolator with no prior knowledge of the noise template $g(n)$.

2. We introduce a new variable $r^i(n)$ which represents the estimated interference signal:

$$r^i(n) \equiv x(n) - s^i(n)$$

3. Calculate the next estimate of the scaling factor b^{i+1} ; one simple way to do this is to solve the following equation by

minimizing $e(n)$ within the *central pulse* location (see Fig. 6):

$$r^i(n) = b^{i+1}g(n - m') + e(n) \quad (5)$$

4. Finally we set $i = i + 1$ and iterate from 1.

We can see that $s^1(n) = \text{LSAR}(x(n) - b^1g(n - m'))$ where the fitted interference pulse $b^1g(n - m)$ has $n \in$ *central pulse* and *decaying tail*. The first estimate of the original signal $s^1(n)$ is shown in Fig. 5. We can also see from Fig. 4, 5 that the first estimate of the original signal $s^1(n)$ is closer to the actual values of the original signal $s(n)$ than the initial estimate of the original signal $s^0(n)$.

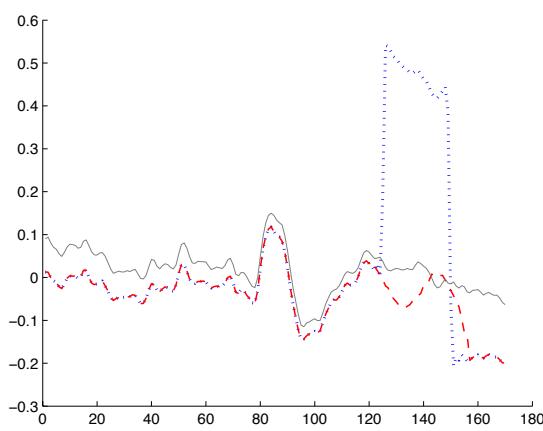


Figure 4: Example implementation of Eq. (4) : Observed corrupted signal $x(n)$ (Dotted), Original signal $s(n)$ (Gray), Initial estimate of the original signal $s^0(n)$ (Dashed).

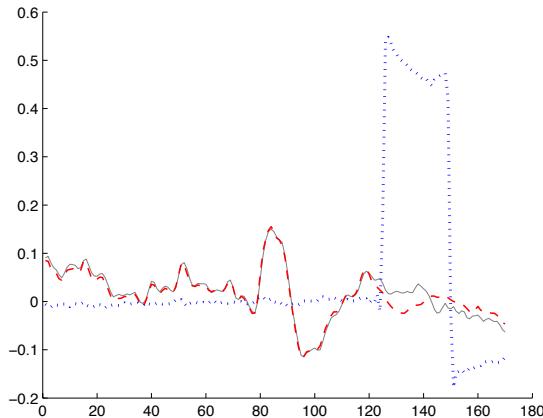


Figure 5: Example implementation of Eq. (4) : Predicted interference pulse $b^1g(n - m')$ (Dotted), Original signal $s(n)$ (Gray), First estimate of the original signal $s^1(n)$ (Dashed).

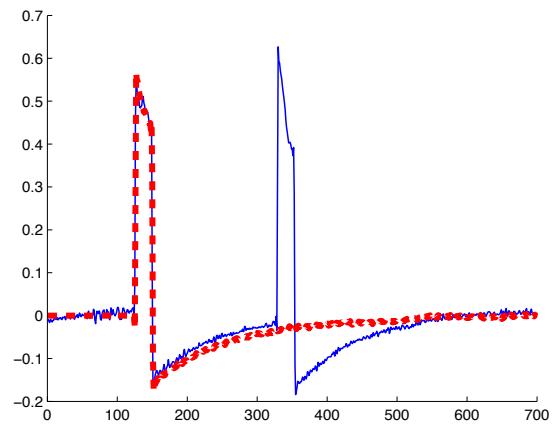


Figure 6: Eq. (6) : Estimated interference signal $r^0(n)$ (Blue), Fitted interference pulse $b^1g(n)$ (Dashed).

5. CONCLUSIONS AND RESULTS

Empirical results show the proposed Bayesian template detector is very effective in detecting the interference pulses. Once the pulses are properly detected, the proposed AR template interpolator can typically improve the signal to noise ratio (SNR) of the observed corrupted signal by more than 50dB (Table 1). The AR template interpolator outperforms the general LSAR interpolator by as much as a 25dB improvement in SNR. Subjective listening tests show that the AR template interpolator restored signal displays no audible artifact when compared to the original signal. The general LSAR interpolator does not take into account the *decaying tail*. We can see from Fig. 7, 11 that the general LSAR interpolator restored signal has a “dip” at the end of the noise pulses. This contributes to the audible artifacts associated with the general LSAR interpolator. Audio samples of the results of the AR template interpolator can be found at:

<http://www-sigproc.eng.cam.ac.uk/~hl309>

- Recorded on a full rate GSM interfered stereo			
- All units in (dB)		Type of audio signal	
Restoration techniques	Jazz	Pop	Speech
Noisy signal $x(n)$	-17.0	-25.5	-22.8
LSAR interpolator	19.0	-3.4	1.6
AR template (1st iter)	37.8	18.3	22.8
AR template (5th iter)	38.5	19.5	23.5

Table 1: SNR performance of different interpolators.

For time-critical applications, we can use a threshold detector or hardware electromagnetic detectors and restore the corrupted signal with only one iteration of the AR template interpolator and no overlap between frames. Then the dominating factor for the complexity is the matrix inversion step in the AR template interpolator. The performance of the AR template interpolator is of the same order as the general LSAR interpolator. If we use the Levinson-Durbin recursion [12] for the matrix inversion steps we arrive at a computational complexity of $O(L^2)$. For example, if we need to restore a CD quality (44 kHz sampling rate), full rate

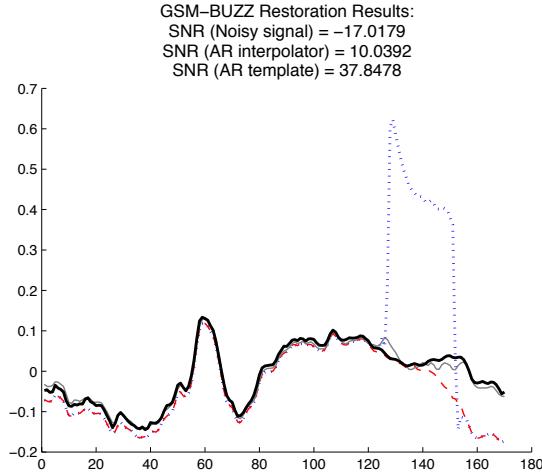


Figure 7: Example of GSM buzz restoration of a corrupted trumpet sequence (Jazz) : $x(n)$ (Dotted), $s(n)$ (Gray), $s^0(n)$ (Dashed), $s^1(n)$ (Black).

217 Hz GSM buzz interfered audio file, L is around 25 to 75 samples (25 samples are the length of the *central pulse*). We can see that with such computational complexity, the algorithm can be efficiently implemented on most microprocessors to run on real-time.

6. FUTURE WORK

The model for removal of Cell-phone interfered noise pulses can be elaborated if we also want to recover the information from the corrupted signal $x(n)$ in the *central pulse* Fig. 2. By observing the patterns of the *central pulse* and the *decaying tail* of the Cell-phone interference noise pulse, we introduce a new variable $y(n)$ to represent the interference noise. We can see from Fig. 8, $y(n)$ can be modeled as two exponential decays. We propose the *Exponential Decay Model* for the observed corrupted signal $x(n)$. We can then estimate the parameters jointly with Bayesian motived techniques such as *Expectation Maximization* or *Gibbs Sampler* [9][10].

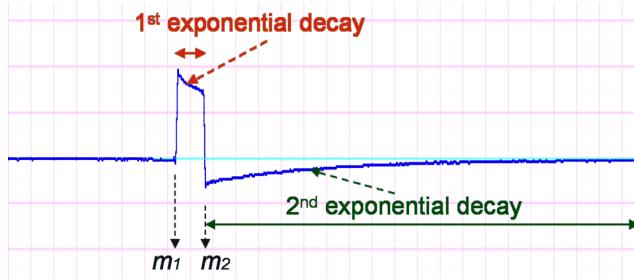


Figure 8: Example of modeling the noise pulse $y(n)$ as 2 exponential decays.

The Cell-phone interfered noise pulse restoration can be further expanded to the Multi-Channel case, where N channels of audio samples are observed. We can model the noise pulse of one channel as a scaled version of the noise pulse of the other channel (see Fig. 9). We can now extend the AR template interpola-

tor to Multi-Channel [13]. In addition, we can iterate over all the available channels with the *Exponential Decay Model* mentioned earlier.

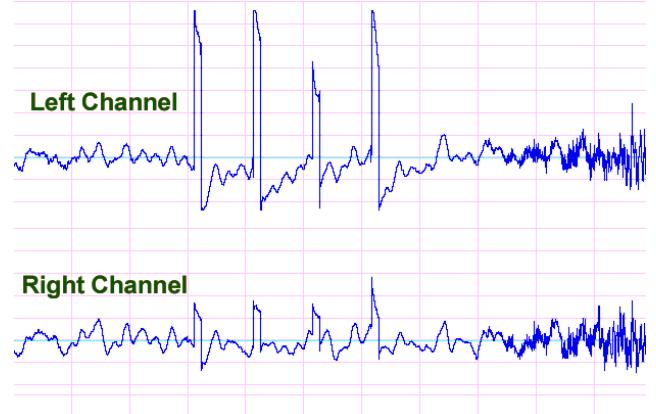


Figure 9: Example of a cell-phone interfered audio signal where noise pulse of the left channel can be modeled as a scaled version of the noise pulse of the right channel.

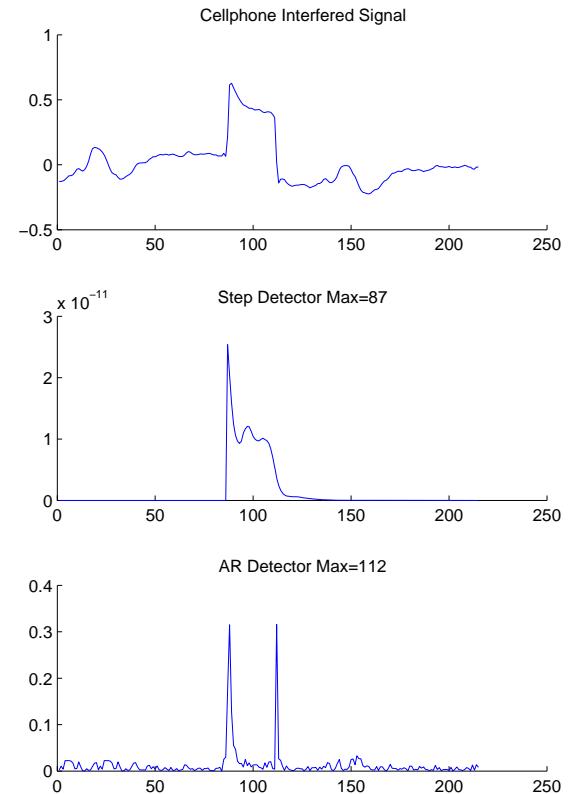


Figure 10: Detection results of Bayesian step detector and AR detector.

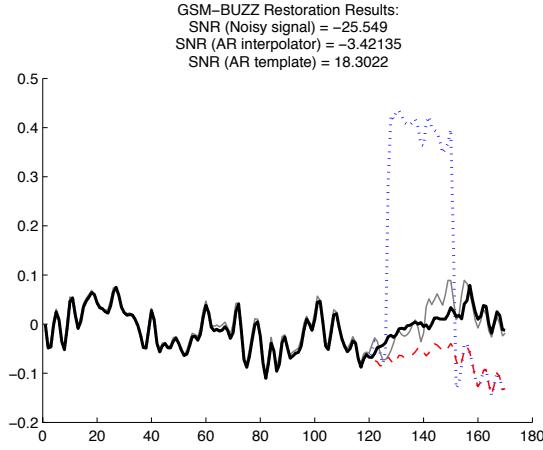


Figure 11: Example of GSM buzz restoration of a corrupted vocal sequence (Pop): $x(n)$ (Dotted), $s(n)$ (Gray), $s^0(n)$ (Dashed), $s^1(n)$ (Black).

7. REFERENCES

- [1] B. Poole, "Reducing audio "buzz" in GSM cell phones," *EDN*, Feb. 2005, [Online] <http://www.edn.com/article/CA498768.html>.
- [2] I. Claesson and A. Nilsson, "GSM TDMA frame rate internal active noise cancellation," *Inter. J. Acoust. and Vibration*, vol. 8, no. 3, pp. 159–166, 2003.
- [3] J. J. Morrissey, M. Swicord, and Q. Balzano, "Characterization of electromagnetic interference of medical devices in the hospital due to cell phones," *Health Phys*, vol. 82, pp. 45–51, 2002.
- [4] M. Skopec, "Hearing aid electromagnetic interference from digital wireless telephones," *IEEE Trans. Rehabil. Eng.*, vol. 6, no. 2, pp. 235–239, june 1998.
- [5] M. Ross, "Telecoil and telephones the most commonly misunderstood "assistive listening device"," *Hearing Loss; The Journal of Self Help for Hard of Hearing People*, vol. 23, no. 1, pp. 16–19, Jan./Feb. 2002.
- [6] G. B. B. Chaplin and R. A. Smith, "Method and apparatus for cancelling vibrations," U.S. Pat. 4490841, Dec. 1984.
- [7] ———, "Hearing aid and method for operating a hearing aid to suppress electromagnetic disturbance signals," U. S. Pat. 20010762875, Aug. 1999.
- [8] GSM Standard (GSM 05.01 version 7.0.0 Release 1998), "Digital cellular telecommunications system (phase 2+), Physical layer on the radio path," 1998.
- [9] S. J. Godsill and P. J. W. Rayner, *Digital audio restoration – A Statistical model-based approach*. Springer-Verlag, 1998.
- [10] J. J. K. O'Ruanaid and W. J. Fitzgerald, *Numerical Bayesian methods applied to signal processing*. Springer-Verlag, 1996.
- [11] J. N. Bernardo and A. F. M. Smith, *Bayesian Theory*. Wiley, 1994.
- [12] J. R. Bunch, "Stability of methods for solving Toeplitz systems of equations," *SIAM J. Sci. Stat. Comput.*, vol. 6, pp. 349–364, 1985.
- [13] H. Lin and S. J. Godsill, "Audio in the new millennium," in *IEEE Workshop on Audio and Acoustics*, Mohonk, NY State, Oct. 2005, pp. 1–4.

APPLICATION OF RASTER SCANNING METHOD TO IMAGE SONIFICATION, SOUND VISUALIZATION, SOUND ANALYSIS AND SYNTHESIS

Woon Seung Yeo, Jonathan Berger

Center for Computer Research in Music and Acoustics
Stanford University, California, U.S.A.
woony@ccrma.stanford.edu

ABSTRACT

Raster scanning is a technique for generating or recording a video image by means of a line-by-line sweep, tantamount to a data mapping scheme between one and two dimensional spaces. While this geometric structure has been widely used on many data transmission and storage systems as well as most video displaying and capturing devices, its application to audio related research or art is rare.

In this paper, a data mapping mechanism of raster scanning is proposed as a framework for both image sonification and sound visualization. This mechanism is simple, and produces compelling results when used for sonifying image texture and visualizing sound timbre. In addition to its potential as a cross modal representation, its complementary and analogous property can be applied sequentially to create a chain of sonifications and visualizations using digital filters, thus suggesting a useful creative method of audio processing.

Special attention is paid to the rastrogram - raster visualization of sound - as an intuitive visual interface to audio data. In addition to being an efficient means of sound representation that provides meaningful display of significant auditory features, the rastrogram is applied to the area of sound analysis by visualizing characteristics of loop filters used for a Karplus-Strong model. A new sound synthesis method based on texture analysis/synthesis of the rastrogram is also suggested.

1. INTRODUCTION

Data conversion between the visual and audio domain has been an active area of scientific research and various multimedia arts. Examples include waveforms, spectrograms, and numerous audio visualization plug-ins, as well as visual composition and image sonification software such as Metasynth [1] and Audiosculpt [2].

Since these conversions essentially represent data mappings, it is crucial to understand and utilize the nature of datasets in both audio and visual domains to design an effective mapping scheme. The temporal nature of a sound and the time-independent, two-dimensional nature of an image requires that data mappings between the two media address these fundamental differences.

1.1. Raster Scanning as a Data Mapping

Raster scanning is a technique for generating or recording the elements of a display image by sweeping the screen in a line-by-line manner. More specifically, it scans the whole area, generally from left to right, while progressing from top to bottom of the imaging sensor or the display monitor, as shown in figure 1.

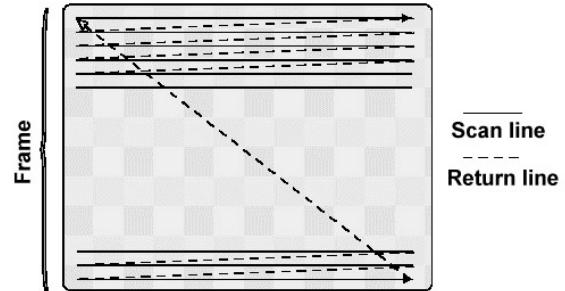


Figure 1: Path of raster scanning.

Geometric framework of raster scanning - a mapping between one- and two-dimensional data spaces - can be found in other places such as communication and storage systems of two-dimensional datasets. This is, in fact, the property that receives primary attention in our choice of raster scanning as a new mapping framework between image and sound.

Raster scanning provides an intuitive, easy to understand mapping scheme between one- and two-dimensional data spaces. This simple, one-to-one mapping also makes itself a totally reversible process: data converted into one representation can be reconstructed without any loss of information.

2. REVIEW OF COMPARABLE WORKS

In terms of geometric framework, applications of raster mapping to sound-image conversions can rarely be found. However, there are some comparable works based on different types of scanning methods.

2.1. Sound Scanner

In [3], Kock used a "sound scanner" to make sound visible: he put a small microphone to a long motorized arm which swept out a raster-like arc pattern. Attached to the microphone was a small neon light driven by an amplifier connected to the microphone. Sound received by the microphone would then light up the lamp, which was photographed in darkness with a long exposure. The resulting picture, therefore, depicted the sound level as bright patterns.

While the result of our raster visualization is time dependent, Kock's work was spatial rather than temporal; although the light source swept the space over a period of time, its results were standing waves. In terms of loudness to brightness conversion, however,

they are based on a similar mapping rule.

2.2. Spiral Visualization of Phase Portrait

To visualize period-to-period difference of a sound, Chafe [4] projected phase portrait of a sound onto a time spiral, thereby separating each period. In this mapping, time begins at the perimeter and spirals inward, one orbit corresponding to one full period of the waveform. Also, trace can be colored according to specific spectral qualities of the sound.

Spiral drawing path, together with the use of spectral transform, sets this technique apart from raster visualization. However, it should be noted that this was proposed as an analysis tool for designing physical synthesis models, especially with pitch-synchronicity in mind. Application of the raster mapping method to a similar problem will be discussed in §5.

2.3. Wave Terrain Synthesis

In wave terrain synthesis [5], a "wave surface" is scanned in an "orbit" (a closed path), and movement of the orbit causes variations in the generated sound. Obviously this is a mapping from two- to one-dimensional data space, which is applicable to image sonification.

Techniques that scan a wave terrain have been explored by several researchers, including [6]. None of them, however, are similar to the raster scanning path we propose.

2.4. Scanned Synthesis

Scanned synthesis [7] is a sound synthesis technique which "scans" a closed path in a data space periodically to create a sound. Due to its emphasis on the performer's control of timbre, data to be scanned is usually generated by a slow dynamic system whose frequencies of vibration are below about 15 [Hz], whereas the pitch is determined by the speed of the scanning function. The system is directly manipulated by motions of the performer, therefore can be looked upon as a dynamic wavetable control.

While scanned synthesis can be characterized by various scanning patterns and data controllability, raster sonification features a fixed geometric framework dedicated to converting rectangular images to sound.

2.5. Research on Mapping Geometry

In [8], Yeo proposed several new image sonification mappings whose scanning paths and color mappings are different from those of the inverse spectrogram method that is most widely used. Examples include vertical scanning, and scanning along a virtual "perpendicular" axis, with horizontal panning.

This research for mapping geometry has been further refined in [9] to provide the concept of *pointer - path* pair, which serves as the basis of a general framework for mapping classification.

2.6. Significance and Contributions

In summary, the following can be proposed as possible contributions of this research in relation to existing works.

- By adopting raster scanning method, we can construct a simple and reversible mapping framework between image and sound with complementarity. This enables us not only

to utilize images as sound libraries (or sounds as image libraries), but also to edit sounds by modifying corresponding images (or vice versa), in a highly predictable way.

- Raster sonification creates a sound which evokes the visual texture of the original image in detail. Although it lacks the freedom of control provided by scanned synthesis, its geometric framework proves to be highly effective with two-dimensional images.
- Raster visualization is also proposed as an intuitive tool for timbre visualization, sound analysis, and filter design for digital waveguide synthesis.
- Moreover, combination of raster sonification and visualization not only suggests a new concept of sound analysis and synthesis based on image processing techniques, but also has strong implications for artistic applications, including cross-modal mapping and collaborative paradigm.

3. IMAGE SONIFICATION

Currently, raster mapping for sonification is defined as follows:

- Brightness values of grayscale image pixels, ranging from 0 to 255 (8-bit) or 65535 (16-bit), are linearly scaled to fit into the range of audio sample values from -1.0 to 1.0.
- One image pixel corresponds to one audio sample.

Figure 2 illustrates these rules.

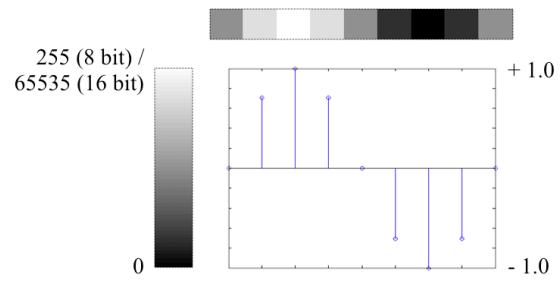


Figure 2: Rules of raster mapping.

3.1. Basic Properties

Because of the natural periodicity found in the majority of images, the sonified result of an image sounds "pitched;" the width of an image determines the period (thereby the pitch) of its sonified sound. Also, by the one-to-one sample mapping, area of an image corresponds to the duration of its sonified sound.

In addition to width, pattern changes in the vertical direction are represented as similar changes in timbre over time. This is depicted in figure 3: images in 4(a) and 4(b) sonifies into sounds with constant sound quality, whereas 4(c) and 4(d) are examples of time-varying timbre.

3.2. Image Textures

Raster mapping proves to be highly effective when used for sonifying the fine "texture" of an image. Sonified sounds preserve the feeling of the original images quite similarly in the auditory domain, and are quite useful for discriminating relative differences between various image textures. Figure 4 illustrates four images

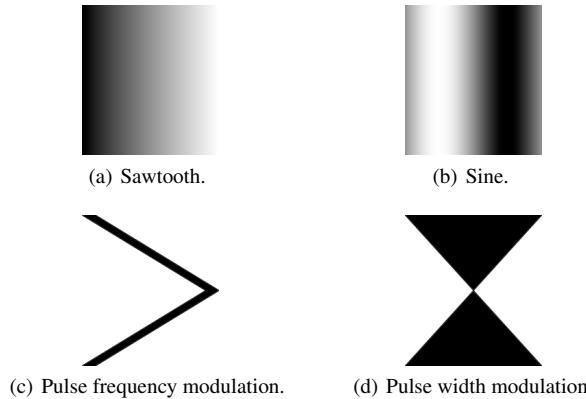


Figure 3: *Images with constant and changing vertical patterns. Sonified results are specified individually.*

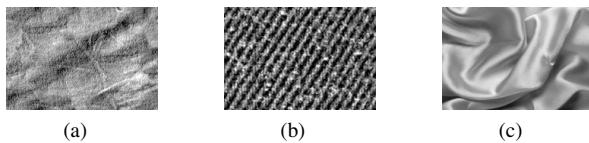


Figure 4: *Images used to compare textures by sound.*

with contrasting textures: a number of tests have shown that most people could correctly match the original images with their sonified sounds when they were given all of them simultaneously.

Providing an absolute auditory reference to a particular visual texture, however, is a challenging task. To this end, construction of a large set of image-sound pairs as a mapping library, together with their classification and training, is desirable.

3.3. Effects of Visual Filters

Raster mapping also produces interesting and impressive results when used for sonifying the effects of various visual filters. Figure 5 contains an image together with its filtered results. Sonified results of these images are very convincing. Compared with the sound generated from the original image of figure 6(a), sound of figure 6(b) produces the auditory image of small bumpy texture, while that of figure 6(c) feels more noisy and grainy.

Sonification of visual filter effects could be further developed as the concept of "sound manipulation using image processing techniques" when combined with corresponding raster visualization.

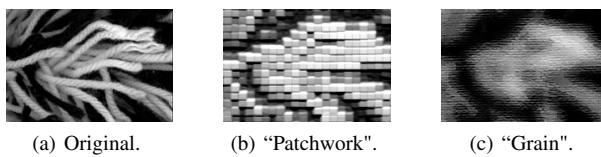


Figure 5: *Comparison of visually filtered images.*

4. SOUND VISUALIZATION: RASTROGRAM

Visualization of sound with raster mapping is an inverse process of raster sonification.

- Values of an audio samples (from -1.0 to 1.0) are linearly scaled to fit into the range of brightness values of image pixels (within the range from 0 to 255).
- One audio sample corresponds to one image pixel.

A new term *rastrogram* is proposed as its name.

Compared with ordinary waveform displays, rastrogram can be considered to be highly space-efficient, even without any loss of samples.

4.1. Width And Pitch Estimation

Rastrogram is basically a representation of short segments of audio samples stacked from top to bottom over time. Since it shows the phase shift between each of those segments, rastrogram can be a useful tool for visualizing changes of pitch over time.

To make this effective, its width should be properly chosen to match the length of one period of sound as closely as possible, thereby being *pitch-synchronous*. In case of an exact match, every "stripe" of rastrogram should align on a perfectly vertical direction. Due to the limited precision of frequency values obtained by integer-only image widths, however, unwanted "drifts" can be introduced by round-off errors: stripes will usually slope in either way, depending on the instantaneous pitch value.

Figure 6 shows three rastrograms that are generated from the same violin sound, but of different width. Obviously 7(b) is most well-synchronized to its pitch, which is supposed to be around $44, 100/170 = 259.4$ [Hz]. From a closer inspection of this, we see the followings:

- Inclination at a certain point provides the amount of relative delay to the width of rastrogram, which makes it possible to derive more precise pitch. Figure 7(b) shows that the original sound could be largely segmented into five different pitch sections, with roughly estimated pitch values of 259.3 [Hz] (I), 259.5 [Hz] (II), 259.4 [Hz] (III), 259.2 [Hz] (IV), and 259.6 [Hz] (V), respectively.
- In addition, it is clearly shown that there are a number of short, subtle changes of pitch throughout the duration of sound. These include the relatively big one in the beginning of the sound (O), which introduces temporary pitch shift down to about 257.9 [Hz].

It should be noted that rastrogram visualizes fine details of pitch variations with extremely high precision *in both time and frequency*, which can hardly be achieved by spectrogram alone.

4.2. Timbre Visualization

Figure 7 illustrates rastrograms of recordings of a flute, a French horn, and a piano. In addition to their pitch changes over time, differences in timbre (especially the complexity of piano sound in figure 8(c)) are clearly visible.

4.3. Frequency Modulation

Rastrograms of frequency-modulated sounds with different values of modulation frequency are depicted in figure 8. With the same modulation index, threshold of image "clarity" lies around 10 [Hz]

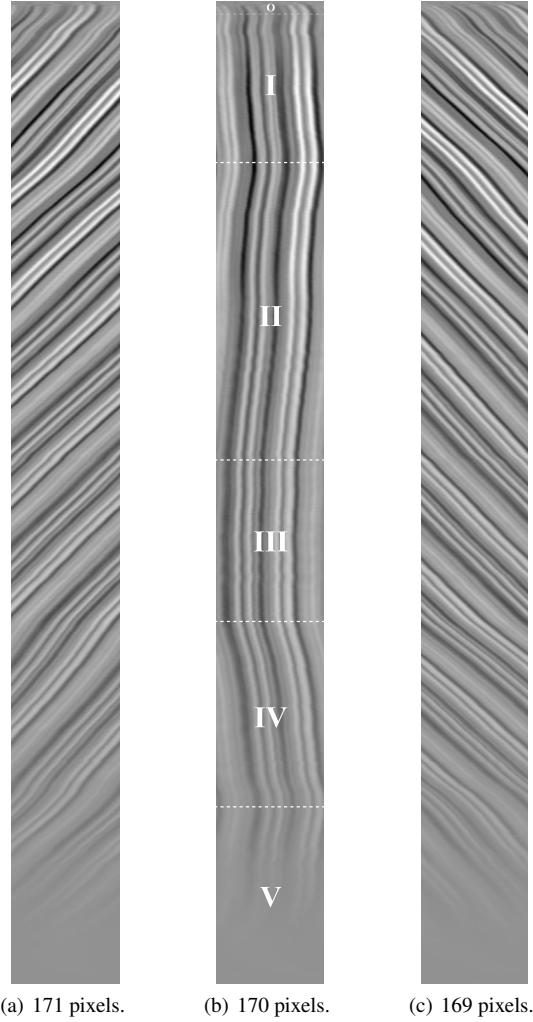


Figure 6: Comparison of rastrograms from the same violin sound, but of different widths (as specified). Sound sample was obtained from the Musical Instrument Samples of Electronic Music Studios, University of Iowa.

of modulation frequency, which roughly matches the perceptual characteristics in the auditory domain. On the other hand, figure 9 shows the results of various modulation indices.

Rastrograms of FM sounds with higher modulation frequency and index values are generally complex, and require further research to be analyzed and fully understood.

On the contrary, sounds with relatively low modulation frequency and moderate modulation index produce quite simple rastrograms, from which both parameters could be derived. Also, they show a wood-like texture. This, in turn, means that selected woodgrain images as in figure 11(a) could be raster sonified to synthesize FM-like sounds. Figure 11(b) shows a simplified "synthetic" rastrogram, which is generated from an FM synthesizer whose parameters were designed to emulate this natural wood-grain. Except for the fine details of the natural woodgrain side, sounds of both figures are quite similar to each other in terms of pitch.

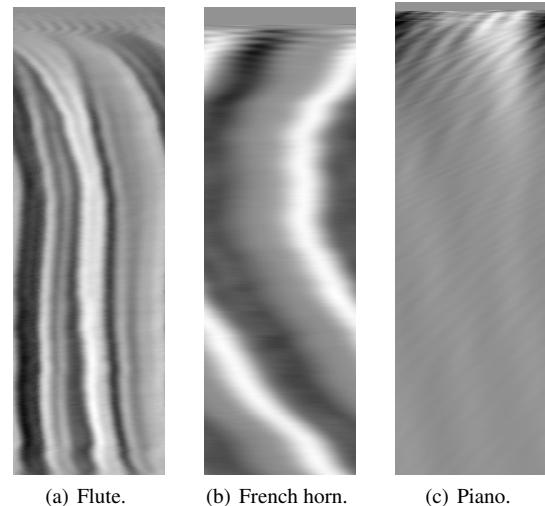


Figure 7: 2 seconds of rastrograms of different instruments, as individually specified.

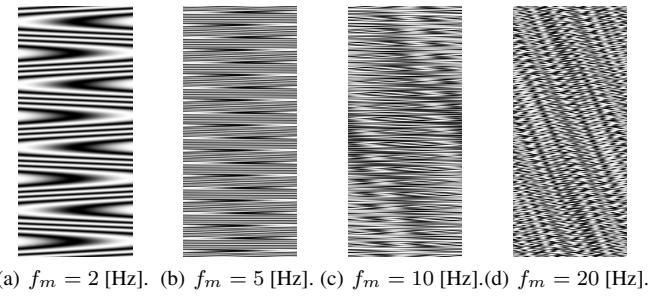


Figure 8: Rastrograms of frequency modulated sounds, $y(t) = \cos[2\pi\{f_c t + I \cdot f_m \cdot \sin(2\pi f_m t)\}]$, with $f_c = 220.5$ [Hz], $I = 1$, but different f_m values (as specified).

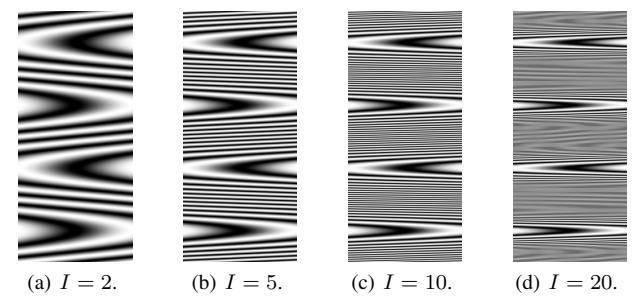
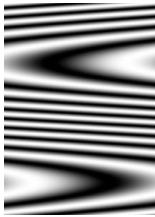


Figure 9: Rastrograms of FM sounds from the same equation as in figure 8. Each sound is generated with $f_c = 220.5$ [Hz] and $f_m = 1$ [Hz], but with different I (as specified). Note that $I = 1$ would be the same as figure 9(a).



(a) Natural woodgrain image.



(b) "Synthetic" woodgrain created from an FM sound with $f_c = 220.5$ [Hz], $f_m = 0.7$ [Hz], and $I = 0.6$.

Figure 10: *Natural and synthetic woodgrain images.*

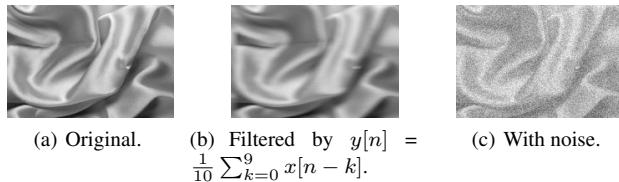


Figure 11: *Rastograms of sounds modified by different audio processes.*

4.4. Effects of Audio Filters

We suggest the idea of “visualizing the effect of audio filters” as a dual to its counterpart in the other domain. In figure 11, sound generated from 12(a) is lowpass-filtered and visualized back to create the rastrogram in 12(b), while 12(c) is converted from the same sound of 12(a) with added white noise. Naturally, this could develop further into the idea of “image manipulation with audio filters.”

5. VISUALIZATION OF KARPLUS-STRONG MODEL

As mentioned in 4.1, rastrogram shows short segments of audio samples in order of time. For digital waveguide sound synthesis models, it becomes a visual record of delay line status at every cycle and depicts particular characteristics of the system. This, therefore, becomes an ideal method to visualize sounds generated from Karplus-Strong algorithm [10] [11], whose diagram with a simple (or, possibly the simplest) filter is shown in figure 12.

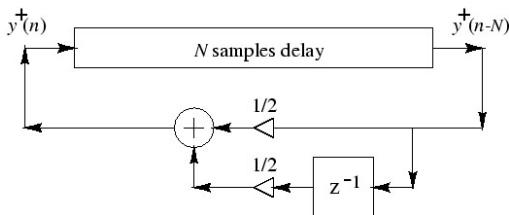
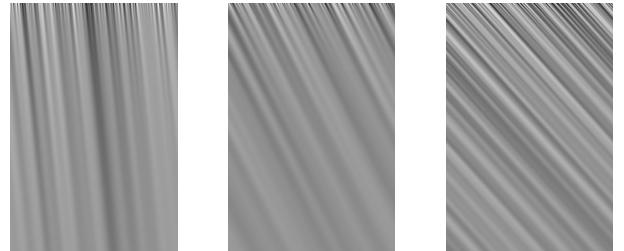


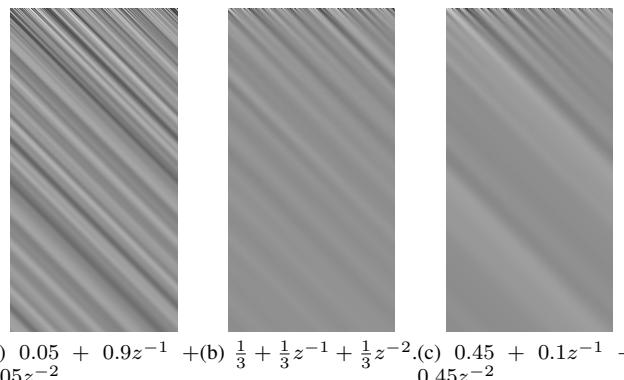
Figure 12: *System diagram of Karplus-Strong algorithm, with a simple loop filter.*

Figure 13 illustrates rastograms generated from a Karplus-Strong model using different filters in the feedback path. In this comparison, inclined lines in each rastrogram receive primary at-



(a) $0.9 + 0.1z^{-1}$. (b) $0.5 + 0.5z^{-1}$. (c) $0.1 + 0.9z^{-1}$.

Figure 13: *Rastograms of plucked-string sounds generated from Karplus-Strong algorithm, with different loop filter $H(z)$ s as individually specified. Delay line length N is 337, which corresponds to about 130.9 [Hz].*



(a) $0.05 + 0.9z^{-1} + 0.05z^{-2}$. (b) $\frac{1}{3} + \frac{1}{3}z^{-1} + \frac{1}{3}z^{-2}$. (c) $0.45 + 0.1z^{-1} + 0.45z^{-2}$.

Figure 14: *Rastograms of the same Karplus-Strong model as figure 4.1, with different second-order loop filter $H(z)$ s that introduce one sample delay.*

tention. Inclination in the rastrogram of a waveguide represents an additional delay whose length can be determined by degree, thereby showing instantaneous pitch change, as discussed in §4.1. The loop filters used here can be considered as *fractional delays* using linear interpolation: delay sizes of the filters used in 14(a), 14(b), and 14(c) are 0.1, 0.5, and 0.9, respectively. From inspection of figure 13, we see that each of these coincides the inclination of corresponding rastrogram expressed as $\Delta x / \Delta y$.

In Figure 14, Karplus-Strong rastograms with second-order loop filters are depicted. Each of these can be considered as a second-order interpolation that is equivalent to one sample delay, thereby having the same degree of inclination. Differences between their amplitude response characteristics, however, are clearly visualized: obviously, the result of 15(a) is not much affected, while 15(c) is the most lowpass-filtered.

From the results presented in this section, we can see that rastrogram produces an effective visualization of particular features of the loop filter used in Karplus-Strong algorithm, including the length of interpolated delay. Although rastrogram does not provide any precise measurement of filter parameters by itself, it has a strong potential as a visual interface of a filter analysis/design

tool.

6. IMAGE PROCESSING TECHNIQUES FOR SOUND SYNTHESIS

When used together, raster sonification and visualization make it possible to modify/synthesize a sound using various image processing techniques.

6.1. Basic Editing and Filtering

From the aforementioned geometric properties of raster mapping, it is obvious that *time scale modification* and *frequency shifting* of a sound can be performed by resizing its rastrogram. More precise control of frequency would be possible by parallelogram-shaped *skew* transforms to consider the roundoff drift. Other geometric modifications, such as rotation, can change the timbre and pitch: as a special case, rotation by 180° produces a time-reversed sound.

In addition to these processes, visual filters can be applied not only to replace corresponding audio filters but also to create unique variations of the original sound, as suggested by examples in §3.3.

6.2. Texture Analysis and Synthesis

Numerous algorithms for analysis and synthesis of visual texture have been developed in the field of computer graphics, as summarized in [12]. We believe that these techniques can be used for analyzing the rastrogram of a sound to create a new synthesized one, which then can be raster-sonified to produce a *visually synthesized* version of the original sound. Advantages of using texture synthesis techniques for sound include the followings:

- While preserving the quality of the original image, synthesized textures can be made of any size, providing full control over the pitch and duration of raster sonified sound.
- Texture synthesis can also produce *tileable* images by properly handling the boundary conditions: this enables us to eliminate any unwanted noise components introduced by abrupt jumps at the edge.
- Potential applications of image texture synthesis include de-noising, occlusion fill-in, and compression. Therefore, techniques for these could be applied to similar problems in audio domain.

Future research on this topic will be focused on constructing a set of *visual eigenfunctions* which can span various rastograms with “auditory significance”. We believe this will provide a simpler and more effective algorithm for analysis and resynthesis of complex tones.

7. ONLINE EXAMPLES

Examples presented in this paper, together with sound files, are also available at [13].

8. CONCLUSION

We have here proposed raster scanning method as a new mapping framework for image sonification and sound visualization. Raster sonification proves to be a powerful method for creating a sound which contains the “feeling” of the original image, and becomes

the elementary background for sonifying the effects of visual filters. Rastrogram, on the other hand, has a strong potential as a visual interface to audio: in addition to being a space-efficient audio data display, it can intuitively visualize the timbre and some fundamental auditory properties of a sound, and filter characteristics as well. Together, both sides of raster mappings form a complete circle of conversion between audio and visual data, thereby making it possible to utilize image processing methods for sound analysis and synthesis.

Future works will also include artistic applications of raster mappings. In addition to the simple idea of using images as sound libraries, sonification of a painting will be proposed as an auditory clue for its visual patterns. Also, the chain of audio-visual conversion will be developed into a new framework of collaborative art.

9. REFERENCES

- [1] U&I Software, “Metasynth 4,” Retrieved June 29th, 2006, [Online] <http://uisoftware.com/MetaSynth/>.
- [2] Ircam, “Audiosculpt,” Retrieved June 29th, 2006, [Online] <http://forumnet.ircam.fr/>.
- [3] W. Kock, *Seeing Sound*. New York: Wiley-Interscience, 1971.
- [4] C. Chafe, “Adding vortex noise to wind instrument physical models,” in *Proc. Int. Comp. Music Conf. (ICMC’95)*, Banff, Canada. ICMA, Sept. 1995, pp. 57–60.
- [5] J. Bischoff, R. Gold, and J. Horton, “Music for an interactive network of microcomputers,” *Computer Music J.*, vol. 2, no. 3, pp. 24–29, 1978.
- [6] A. Borgonovo and G. Haus, “Musical sound synthesis by means of two-variable functions: Experimental criteria and results,” in *Proc. Int. Comp. Music Conf. (ICMC’84)*, Paris, France. ICMA, 1984, pp. 35–42.
- [7] B. Verplank, M. Mathews, and R. Shaw, “Scanned synthesis,” in *Proc. Int. Comp. Music Conf. (ICMC’00)*, Berlin, Germany. ICMA, Sept. 2000, pp. 368–371.
- [8] W. S. Yeo, “Image sonification: Image to sound,” 2001, [Online] <http://www.mat.ucsb.edu/~woony/research/winter01/mat310/>.
- [9] W. S. Yeo and J. Berger, “Application of image sonification methods to music,” in *Proc. Int. Comp. Music Conf. (ICMC’05)*, Barcelona, Spain. ICMA, Sept. 2005, pp. 219–222.
- [10] K. Karplus and A. Strong, “Digital synthesis of plucked string and drum timbres,” *Computer Music J.*, vol. 7, no. 2, pp. 43–45, 1983.
- [11] D. A. Jaffe and J. O. Smith, “Extensions of the Karplus-Strong plucked string algorithm,” *Computer Music J.*, vol. 7, no. 2, pp. 56–69, 1983.
- [12] L. Wei, “Deterministic texture analysis and synthesis using tree structure vector quantization,” in *Proc. ACM SIGGRAPH Los Angeles, USA*. ACM, 1999, pp. 207–213.
- [13] W. S. Yeo, “Raster scanning,” Retrieved June 29th, 2006, [Online] <http://ccrma.stanford.edu/~woony/research/raster/>.

CECILIA AND TCLCSOUND

Jean Piché

Faculté de Musique
Université de Montréal
Montréal, Canada
jean@piche.com

Victor Lazzarini

Music Technology Laboratory
National University of Ireland
Maynooth
Victor.Lazzarini@nuim.ie

ABSTRACT

This article discusses some developments relating to environments for Csound programming, composition and performance. It introduces the Csound 5 API and discusses its use in the development of a Tcl/Tk scripting interface, TclCsound. The three components of TclCsound are presented and discussed. A number of applications, from simple transport control of Csound to client-server networking are explained in some detail. The new multi-platform version of CECILIA is presented. Cecilia is the first Csound frontend to use the functionalities of TclCsound.

1. INTRODUCTION

The Csound music programming system [1] is currently the most complete of the text-based audio processing systems in terms of its unit generator collection. Csound hails from a long tradition in Computer Music. Together with cmusic [2], it was one of the first modern C-language-based portable sound compilers [3], when it was released in 1986. Due to its source-code availability, first from an MIT ftp server and then from the DREAM site at Bath, it was adopted by composers and developers world-wide. These brave people developed Csound into a formidable tool for sound synthesis, processing and computer music composition. Its latest version, Csound 5 [4] has close to one thousand opcodes, ranging from the basic table lookup oscillator to spectral signal demixing unit generators.

The usability of command line languages for music creation has always made it difficult to promote their wide-spread use by composers. In the hope of bridging this approachability gap, a graphical and highly interactive front end was developed for the language. Cecilia [5] is built as a tcl/tk program that issues calls to an independent Csound process through Unix pipes. It proposes a graphical interface, sliders, buttons and an editor to make orchestras and scores. Cecilia also includes a simple algorithmic score generator known as Cybil. We have developed a new version of the interface where the functionalities of Cecilia are extended to all platforms.

Many important changes have been introduced in Csound5, which involved a complete redesign of the software. This resulted not only in a better software, from an engineering perspective, but in the support for many new possible ways of using and interacting with Csound. An important development has been the availability of a complete C API (the so-called ‘Host API’, which was, in fact, already partially present in earlier versions). The API can be used to instantiate and control Csound from a host application, such as TclCsound/Cecilia.

2. THE CSOUND 5 API

The Csound 5 Host API allows the embedding of the audio processing system under other ‘host’ software. Effectively, Csound is now a library, `libcsound`, that can provide audio services, such as synthesis and processing, for any application. This allows for complete control over the functioning of the audio engine, including transport control, loading of plugins, inter-application software bus, multi-threading, etc.. A ‘classic’ Csound command-line program can now be written based only on a few API calls:

```
#include <csound.h>

int main(int argc, char **argv) {
    int result;

    /* the csound instance */
    CSOUND *cs;

    /* initialise the library */
    csoundInitialize(&argc, &argv, 0);

    /* create the csound instance */
    cs = csoundCreate(NULL);

    /* compile csound code */
    result = csoundCompile(cs, argc, argv);

    /* this is the processing loop */
    if(result) while(csoundPerformKsmpls(cs)==0);

    /* destroy the instance */
    csoundDestroy(cs);

    return 0;
}
```

The Csound API can be used in many applications; the development of frontends is the most obvious of these. A good example of its application is found on the `csoundapi~` Class, which provides a multi-instantiable interface to Csound 5 for Pure Data. The Csound API is the basis for TclCsound [5], a Tcl/Tk extension, discussed in the next section.

3. TCLCSOUND

The classic interface to Csound gives you access to the program via a command-line such as

```
csound -odac hommage.csd
```

This is a simple yet effective way of making sound. However, it does not give you neither flexibility nor interaction. With the

advent of the API, a lot more is possible. At this stage, TclCsound was introduced to provide a simple scripting interface to Csound. Tcl is a simple language that is easy to extend and provide nice facilities such as easy file access and TCP networking. With its Tk component, it can also handle a graphic and event interface. TclCsound provides three ‘points of contact’ with Tcl:

1. a csound-aware tcl interpreter (`cstclsh`)
2. a csound-aware windowing shell (`cswish`)
3. a csound-commands module for Tcl/Tk (`tclcsound` dynamic lib)

3.1. The Tcl interpreter: `cstclsh`

With `cstclsh`, it is possible to have interactive control over a Csound performance. The command starts an interactive shell, which holds an instance of Csound. A number of commands can then be used to control it. For instance, the following command can compile Csound code and load it in memory ready for performance:

```
csCompile -odac hommage.csd -m0
```

Once this is done, performance can be started in two ways: using `csPlay` or `csPerform`. The command

```
csPlay
```

will start the Csound performance in a separate thread and return to the `cstclsh` prompt. A number of commands can then be used to control Csound. For instance,

```
csPause
```

will pause performance; and

```
csRewind
```

will rewind to the beginning of the note-list. The `csNote`, `csTable` and `csEvent` commands can be used to add Csound score events to the performance, on-the-fly. The `csPerform` command, as opposed to `csPlay`, will not launch a separate thread, but will run Csound in the same thread, returning only when the performance is finished. A variety of other commands exist, providing full control of Csound.

3.2. Cswish: the windowing shell

With Cswish, Tk widgets and commands can be used to provide graphical interface and event handling. As with `cstclsh`, running the `cswish` command also opens an interactive shell. For instance, the following commands can be used to create a transport control panel for Csound:

```
frame .fr
button .fr.play -text play -command csPlay
button .fr.pause -text pause -command csPause
button .fr.rew -text rew -command csRewind
pack .fr .fr.play .fr.pause .fr.rew
```

Similarly, it is possible to bind keys to commands so that the computer keyboard can be used to play Csound.

Particularly useful are the control channel commands that TclCsound provides. For instance, named IO channels can be registered with TclCsound and these can be used with the `invalue`, `out-value` opcodes. In addition, the Csound API also provides a complete software bus for audio, control and string channels. It

is possible in TclCsound to access control and string bus channels (the audio bus is not implemented, as Tcl is not able to handle such data). With these TclCsound commands, Tk widgets can be easily connected to synthesis parameters.

3.3. A Csound server example

In Tcl, setting up TCP network connections is very simple. With a few lines of code a Csound server can be built. This can accept connections from the local machine or from remote clients. Not only Tcl/Tk clients can send commands to it, but TCP connections can be made from other software, such as, for instance, Pure Data (PD). A Tcl script that can be run under the standard `tclsh` interpreter is shown below. It uses the `TclCsound` module, a dynamic library that adds the Csound API commands to Tcl.

```
# load tclcsound.so
#(OSX: tclcsound.dylib, Windows: tclcsound.dll)
load tclcsound.so Telcsound
set forever 0

# This arranges for commands to be evaluated
proc ChanEval { chan client } {
    if {[catch { set rtn [eval [gets $chan]]} err]}{
        puts "Error:$err"
    } else {
        puts $client $rtn
        flush $client
    }
}

# this arranges for connections to be made
proc NewChan { chan host port } {
    puts "Csound_server:_connected_to_$host_on_port"
    $port_($chan)"
    fileevent $chan readable [list ChanEval $chan $host]
}

# this sets up a server to listen for
# connections
set server [socket -server NewChan 40001]
set sinfo [fconfigure $server -sockname]
puts "Csound_server:_ready_for_connections_on"
port_[lindex $sinfo 2]"
vwait forever
```

With the server running, it is then possible to set up clients to control the Csound server. Such clients can be run from standard Tcl/Tk interpreters, as they do not evaluate the Csound commands themselves. Here is an example of client connections to a Csound server, using Tcl:

```
# connect to server
set sock [socket localhost 40001]

# compile Csound code
puts $sock "csCompile_-odac_hommage.csd"
flush $sock

# start performance
puts $sock "csPlay"
flush $sock

# stop performance
puts $sock "csStop"
flush $sock
```

As mentioned before, it is possible to set up clients using other software systems, such as PD. Such clients need only to connect

to the server (using a netsend object) and send messages to it. The first item of each message is taken to be a command. Further items can optionally be added to it as arguments to that command.

3.4. TclCsound as a language wrapper

It is possible to use TclCsound at a slightly lower level, as many of the C API functions have been wrapped as Tcl commands. For instance it is possible to create a ‘classic’ Csound command-line frontend completely written in Tcl. The following script demonstrates this:

```
#!/usr/local/bin/cstclsh
set result 1
csCompileList $argv
while { $result != 0 } {
    set result csPerformKsmps
}
```

This script is effectively equivalent to the C program shown in section 2. If saved to, say, a file called `csound.tcl`, and made executable, it is possible to run it as in

```
csound.tcl & $odac hommage.csd
```

With TclCsound, it is possible to transform the popular text editor *emacs* into a Csound scripting/performing environment. When in Tcl mode, the editor allows for Tcl expressions to be evaluated by selection and use of a simple escape sequence (ctrl-C ctrl-X). This facility allows the integrated editing and performance of Csound and Tcl/Tk code.

3.5. TclCsound as a Csound performance environment

In Tcl it is possible to write score and orchestra files that can be saved, compiled and run by the same script, under the *emacs* environment. The following example shows a Tcl script that builds a Csound instrument and then proceeds to run a Csound performance. It creates 10 slightly detuned parallel oscillators, generating sounds similar to those found in Risset’s *Inharmonique*.

```
load tclcsound.so Tclcsound

# set up some intermediary files
set orcfile "tcl.orc"
set scofile "tcl.sco"
set orc [open $orcfile w]
set sco [open $scofile w]

# This Tcl procedure builds an instrument
proc MakeIns { no code } {
    global orc sco
    puts $orc "instr_$no"
    puts $orc $code
    puts $orc "endin"
}

# Here is the instrument code
append ins "asum_init_0_\n"
append ins "ifreq_=p5_\n"
append ins "iamp_=p4_\n"

for { set i 0 } { $i < 10 } { incr i } {
    append ins "a$i_oscili_iamp,"
    append ins "ifreq+ifreq*[expr $i*0.002],_1\n"
}

for { set i 0 } { $i < 10 } { incr i } {
```

```
    if { $i } {
        append ins "_+a$i"
    } else {
        append ins "asum_=a$i_"
    }
}

append ins "\nk1_linen_1,_0.01,_p3,_0.1_\n"
append ins "out_asum*k1"

# build the instrument and a dummy score
MakeIns 1 $ins
puts $sco "f0_10"

close $orc
close $sco

# compile
csCompile $orcfile $scofile -odac -d -m0

# set a wavetable
csTable 1 0 16384 10 1 .5 .25 .2 .17 .15 .12 .1

# send in a sequence of events and perform it
for {set i 0} { $i < 60 } { incr i } {
    csNote 1 [expr $i * 0.1] .5 \
        [expr ($i * 10) + 500] [expr 100 + $i * 10]
}

csPerform

# it is possible to run it interactively as
# well
csNote 1 0 10 1000 200
csPlay
```

The use of such facilities as provided by *emacs* can emulate an environment not unlike the one found under the so-called ‘modern synthesis systems’, such as SuperCollider (SC). In fact, it is possible to run Csound in a client-server set-up, which is one of the features of SC3. A major advantage is that Csound provides about three or four times the number of unit generators found in that language (as well as providing a lower-level approach to signal processing, in fact these are but a few advantages of Csound).

4. THE NEW CECILIA

Over the years, Cecilia [6] has made the utilization of Csound much more intuitive and productive. Cecilia provides a library of common sound processing modules that are ready to use for the By providing an environment where one can quickly build interfaces from tk widgets linked to Csound parameters, explorations of sound processing algorithms becomes much more convivial and interactive.

4.1. Key Cecilia concepts

Technically, Cecilia consists of time variant functions that are sent to Csound from real-time sliders on screen, from physical sliders such as MIDI fader boxes or from a screen graphing window. These objects use global Csound variables to send their data to the relevant instrument parameters. All data from interface objects is communicated to Csound with i-time stdin console messages. A system of a single tagged instrument per controller allows for coding a large number of simultaneous channels of control data.

All controllers, whether graphical or physical, are assigned a unique instrument number and, at performance time, will receive data exclusively from the Csound process stdin. Every time a gesture is detected on the interface or a physical slider, the following tcl process is called and the current value of the object is passed through the stdin with an i-time "i" message.

```
proc passData {instr val}
global csoundID
puts $csoundID " i$instr 0.00 0.0001 $val"
}
```

csoundID being the Unix pipe identifier for the previously launched Csound instance, "instr" being the instrument number attached to the controller and "val" being the current value of the slider.

Cecilia also offers a complete text editor to facilitate the elaboration of orchestras and scores. The editor offers services such as syntax and keyword highlighting and direct on-line documentation.

4.2. Cecilia and TclCsound

The new version of Cecilia communicates with the Csound engine via the TclCsound commands only thus eliminating the need for the Unix pipeline. TclCsound's csChannelIn Value command eliminates the need for console-driven events to update values.

The improvement of performance from the use of internal pipelines is notable but the main benefit is to make Cecilia completely platform independent. This means Cecilia can finally run on the Windows operating systems the same way it runs on UNIX systems. The interface and the synthesis engine run under a single process.

4.3. New Functionalities of Cecilia

It is now possible to make multi-layered arrangements of Cecilia processing modules by simply chaining the output of one module as the input signal to another, effectively paving the way to extremely complex signal processing networks. The interface controls for "stacked" modules are presented in a tabbed window for easy access.

Most Cecilia sound processing modules can now be applied to live input audio streams instead of just sound files.

It is now possible to record, display and edit any gestural controller data entered via MIDI input. The Cecilia grapher window now contains a bezier function editor and a data-reduction method

to properly implement a keyframe based automation system. This feature is borrowed from video editing software like Adobe After Effects.

4.4. Cecilia for audio-visual composition

The advent of audiovisual composition as one of the most promising areas of computer assisted creative disciplines is influencing our design decisions. It is now possible to export Cecilia grapher functions as keyframe maps directly to Adobe After Effects, Processing or other image processing software. This feature is expected to be an important advance for the establishment of audio-visual mapping strategies where time varying parameters of the audio stream can be used to control time variant parameters of image processing software.

5. CONCLUSIONS

Csound5, TclCsound and Cecilia are an example of the future development of audio processing software where intuitive and very responsive interfaces are built around powerful, general and highly programmable DSP engines. It is thereby shown that the power of low level signal processing languages is only accessible to non-specialist composers and creators inside responsive interaction contexts, a combination readily provided by TclCsound and Cecilia.

6. REFERENCES

- [1] B. Vercoe, *Csound: A Manual of the Audio Processing System*. MIT Media Lab, 1986.
- [2] F. R. Moore, *Elements of Computer Music*. Englewood Cliffs, N.J.: Prentice Hall, 1990.
- [3] S. T. Pope, "Machine tongues XV: Three packages for software sound synthesis," *Computer Music J.*, vol. 17, no. 2, pp. 23–55, 1993.
- [4] J. ffitch, "On the design of Csound 5," in *Proc. 3rd Linux Audio Conf.*, 2005, pp. 37–42.
- [5] V. Lazzarini, "Scripting Csound 5," in *Proc. 4th Linux Audio Conf.*, 2005, pp. 50–55.
- [6] J. Piché and A. Burton, "Cecilia: A production interface to Csound," *Computer Music J.*, vol. 22, no. 2, pp. 52–55, 1998.

A NEW PARADIGM FOR SOUND DESIGN

Ananya Misra, Perry R. Cook[†], Ge Wang

Department of Computer Science ([†]also Music)
Princeton University, Princeton, USA
{amisra|prc|gewang}@cs.princeton.edu

ABSTRACT

A sound scene can be defined as any “environmental” sound that has a consistent background texture, with one or more potentially recurring foreground events. We describe a data-driven framework for analyzing, transforming, and synthesizing high-quality sound scenes, with flexible control over the components of the synthesized sound. Given one or more sound scenes, we provide well-defined means to: (1) identify points of interest in the sound and extract them into reusable templates, (2) transform sound components independently of the background or other events, (3) continually re-synthesize the background texture in a perceptually convincing manner, and (4) controllably place event templates over the background, varying key parameters such as density, periodicity, relative loudness, and spatial positioning. Contributions include: techniques and paradigms for template selection and extraction, independent sound transformation and flexible re-synthesis; extensions to a wavelet-based background analysis/synthesis; and user interfaces to facilitate the various phases. Given this framework, it is possible to completely transform an existing sound scene, dynamically generate sound scenes of unlimited length, and construct new sound scenes by combining elements from different sound scenes.

URL: <http://taps.cs.princeton.edu/>

1. INTRODUCTION

Many sound synthesis techniques focus on generating foreground sounds, which by themselves do not generally give a listener a strong sense of being in a real-world environment. This paper introduces techniques and paradigms for working with the totality of foreground and background sounds that compose a sound scene. Existing methods that deal with pre-recorded sound do not provide suitable analysis and synthesis techniques for a sound scene to be composed from selected components of different existing sounds. Naive approaches such as repeatedly playing or combining raw segments of original recordings do not sound convincing, while more complex synthesis methods lack flexibility both in creating scenes and in the amount of user control needed.

Given one or more existing sound scenes, our task is to generate from these any amount of perceptually convincing sound, arbitrarily similar to or different from the original sounds, that can be parametrically controlled to fit the user’s specifications. One of our aims is to provide a flexible tool for easily modeling and generating sound scenes for entertainment (movies, TV, and games), Virtual and Augmented Reality, and art projects such as live performances and installations. Towards this aim, we introduce TAPESTREA: Techniques and Paradigms for Expressive Synthesis, Transformation and Rendering of Environmental Audio. Our approach is based on the notion that sound scenes are composed of events

and background texture, which are best modeled separately. We separate a sound scene into the following components: (1) *Deterministic events*: composed of highly sinusoidal components, often perceived as pitched events, such as a bird’s chirp or a baby’s cry; (2) *Transient events*: brief non-sinusoidal events, such as footsteps; (3) *Stochastic background*: the “din” or residue remaining after the removal of deterministic and transient components, such as wind, ocean waves, or street noise.

TAPESTREA analyzes and synthesizes each component separately, using algorithms suitable to the component type. It applies spectral modeling [1] to extract deterministic events, and time-domain analysis to detect transient events. Each event can then be transformed and synthesized individually. The stochastic background is obtained by removing deterministic and transient events from the given sound and filling in the holes left by transient removal; background is then dynamically generated using an improved wavelet tree learning algorithm [2].

TAPESTREA is distinct from other sound analysis and synthesis methods in that it allows users to: (1) point at a sound or a part of a sound, extract it, and request more or less of it in the final scene, (2) transform that sound independently of the background, (3) flexibly control important parameters of the synthesis, such as density, periodicity, relative gain, and spatial positioning of the components (4) construct novel sounds in a well-defined manner.

The rest of the paper is structured as follows: In Section 2 we describe related work. Section 3 provides an overview of our approach along with an example highlighting how it can be used. Section 4 describes the analysis stage of our framework, section 5 describes the possible transformations on events, and section 6 describes the synthesis phase. Section 7 provides details about our user interface and section 8 summarizes results and contributions. Section 9 describes our conclusions and directions for future work.

2. RELATED WORK

2.1. Simulated and Model-based Foreground Sounds

Simulation and model-based sound synthesis techniques are based on physical models of the objects, the world, and/or the interactions between these [3]. Physically based models have been used to generate foreground sounds caused by object interactions, including walking sounds [4], sounds caused by the motion of solid objects [5], complex sounds due to individual objects and gestures [6], and contact sounds [7] such as colliding, rolling, and sliding.

2.2. Background Sound Textures

A sound texture can be described as a sound with structural elements that repeat over time, but with some randomness. The sound

of rain falling or applause are examples of sound textures. Textures often form a large part of the background of sound scenes.

Athineos and Ellis [8] modeled sound textures composed of very brief granular events known as *micro-transients*, such as fire crackling or soda being poured out of a bottle. Zhu and Wyse [9] extended their technique to separate the foreground transient sequence from the background din in the source texture and resynthesized these separately. Both these methods are effective on textures that primarily contain micro-transients, but do not generalize well to other sounds. For instance, the foreground-background separation misses spectral foreground events, as it does not take frequency into account while identifying events.

Miner and Caudell [10] used wavelets to decompose, modify, and re-synthesize sound textures, concentrating on the perceptual effects of various transformations. Dubnov et. al. [2] also used a wavelet decomposition to analyze and generate more of a sound texture. Their method works well for sounds that are mostly stochastic or have very brief pitched portions. However, sounds with continuous components, such as a formula-one racecar engine, sometimes get chopped up, while rhythmic sounds may lose their rhythm during synthesis. The stochastic model is also not suitable for sounds with many sinusoidal components.

In general, these existing approaches work only for mostly stochastic sounds and do not allow flexible control over the output—either the entire texture is transformed or segments are shuffled and concatenated. Hence these methods are insufficient for sounds that have various foreground events and background playing simultaneously. Our approach overcomes these limitations by isolating and removing pitched sounds, performing modified wavelet tree learning [2] on the remaining stochastic part, and re-inserting the extracted components afterwards. We separate the pitched components from the sound texture using spectral modeling.

2.3. Spectral Modeling

Spectral modeling [1] extends the original sinusoidal modeling algorithm [11] by posing the concept of “sines plus noise,” based on the notion that some components of sound fit a sinusoidal model while others are better modeled by spectrally shaped noise. While Serra and Smith [1] initially applied this to musical instrument sounds, we use it to extract *deterministic events* from any recorded sound scene. Sinusoidal modeling also enables modification of the original sound before re-synthesis, for instance by pitch-shifting and time-stretching. Other related work on spectral analysis includes alternatives to the Fourier transform for estimating the spectra of specific kinds of signals [12, 13].

Existing tools for spectral analysis and re-synthesis, such as SPEAR [14] and the CLAM library [15], allow high-level sinusoidal analysis, transformations and re-synthesis, but do not offer the level of parametric control over these stages suitable for analyzing and creating sound scenes. Further, they lack a framework for processing transients and stochastic background components.

2.4. Sound Editors

Current tools for commercial or home audio production include a range of sound editors. Free or inexpensive commercially available software such as Audacity and GoldWave perform simple audio production tasks. Midline audio editing systems, including Peak, Logic, and Cubase, are geared towards music production

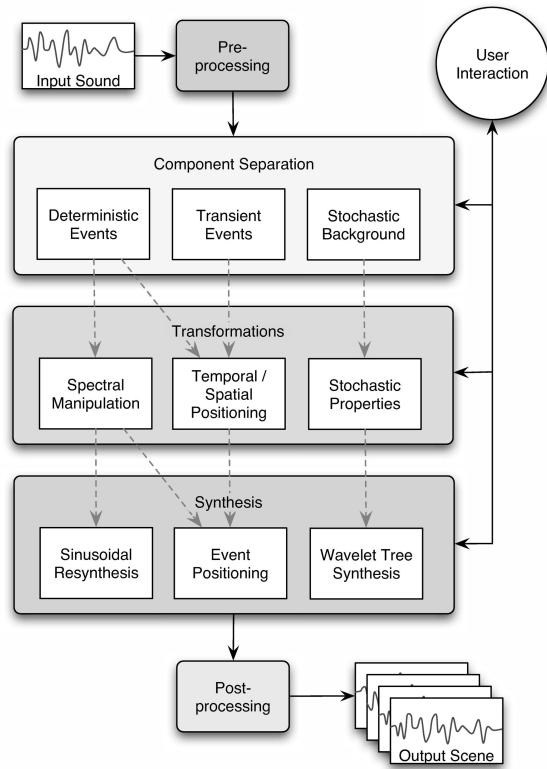


Figure 1: *Stages in our pipeline: (1) preprocessing, (2) analysis, (3) transformation, (4) synthesis.*

and often offer real-time MIDI sequencing capability. At the high end are digital audio production hardware/software systems such as Pro Tools, geared towards commercial sound production. Most of these products support Virtual Studio Technology (VST) plugins that perform synthesis algorithms and apply effects such as reverb. However, none of them provides one real-time, extensible, integrated analysis-transformation-synthesis workspace.

3. EXAMPLE AND OVERVIEW OF OUR APPROACH

The TAPESTREA system starts by loading a 5–15 seconds or longer existing sound scene, such as the sound of a city street, seagulls by the ocean or children playing in a park. Sound events in the park scene may include children yelling, a ball bouncing, and geese honking in a nearby pond. The background texture might consist of the general din of the surroundings.

Figure 1 depicts the phases in the TAPESTREA pipeline. The existing sound scene first undergoes a basic preprocessing phase involving sample-rate/data-depth conversion as needed, channel information, DC blocking and data normalization. Next, it passes through the analysis phase, where the user extracts deterministic (children yelling, geese honking), transient (ball bouncing) and stochastic background (general din) components by specifying analysis parameters. Each component can be played back separately and stored as a template for future use. For example, one bounce of the ball can be stored as a transient template while individual yells can be saved as deterministic event templates. In the transformation and synthesis phase, the system or user

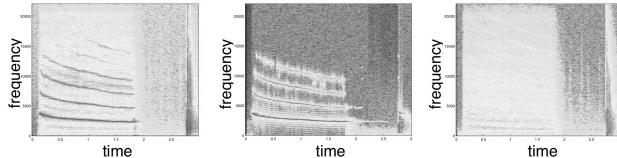


Figure 2: Separating sinusoidal tracks from stochastic residue: (a) original sound; (b) sinusoidal tracks; (c) residue.

parametrically specifies how to construct the output sound scene. Transformations are applied to individual templates and these templates are combined in specified ways to generate a complete sound scene. For instance, the output sound scene can consist of a repeatedly bouncing ball and many children yelling at different pitches and times over a continuous general din, to simulate a children's game with enthusiastic spectators in a park without geese. The output sound can also include templates from other existing sound scenes, such as a referee's whistle. The synthesized sound scene can be written to a file or played continuously in real-time for as long as needed. TAPESTREA also includes a graphical user interface for interactive control of the analysis, transformation and synthesis parameters. The following sections provide more in-depth information on the processing phases and the user interface.

4. EVENT IDENTIFICATION AND ISOLATION

The first step in our framework is to identify and separate foreground events from background noise. Foreground events are parts of the scene perceived as distinct occurrences, and include both *deterministic events* (the sinusoidal or pitched components of a sound) and *transient events* (brief bursts of stochastic energy). Removing these leaves us with the *stochastic background*.

4.1. Sinusoidal Modeling

Deterministic events are identified through sinusoidal analysis based on the spectral modeling framework. The input sound scene is read in as possibly overlapping frames, each of which is transformed into the frequency domain using the FFT and processed separately. The maximum and average magnitudes of the spectral frame are computed and stored. The following steps are then repeated until either a specified maximum number (N) of peaks have been located or no more peaks are present: (1) The maximum-magnitude bin in the frame, within the specified frequency range, is located. (2) If the ratio of its magnitude to the average magnitude of the frame is below a specified threshold, it is assumed to be noise and we deduce that no more peaks are present. (3) If its magnitude is above a specified absolute threshold, it is added as a sinusoidal peak and the bins it covered are zeroed out in the analysis frame.

All the sinusoidal peaks and FFT frames can also be precomputed and stored. All peaks in a frame are found by locating bins where the derivative of the magnitude changes from positive to negative. The peaks for each frame are stored in decreasing magnitude order. At run-time, the top N peaks that satisfy any frequency and threshold bounds are selected per frame for peak matching.

Once the top N peaks in all the frames have been collected, peaks are matched from frame to frame if they occur at sufficiently similar frequencies. Over time this yields *tracks* of peaks lasting

across frames. The matching and updating of tracks takes place as follows: (1) Each existing track from previous frames selects a current frame peak closest to it in frequency. If the difference in frequency is above a specified threshold, that track is dormant and the selected peak remains unmatched. (2) All unmatched peaks are added as new tracks, and all existing tracks that have not found a continuation are removed if they have remained dormant for a specified number of frames. (3) Tracks that continue across a specified minimum number of frames are retained.

Finally, TAPESTREA can parametrically group related tracks [16, 17] to identify events. A track is judged to belong in an existing group if it has a minimum specified time-overlap with the group and either: (1) its frequency is harmonically related to that of a track in the group, (2) its frequency and amplitude change proportionally to the group's average frequency and amplitude, or (3) it shares common onset and offset times with the group average. If a track fits in multiple groups, these groups are merged. While the grouping could benefit from a more sophisticated algorithm or machine learning, it may be fine-tuned for specific sounds by manipulating error thresholds for each grouping category. Groups that last over a specified minimum time span are considered deterministic events. If grouping is not selected, all the tracks found are together considered a single event. Each deterministic event is defined a list of sinusoidal tracks, with a history of each track's frequency, phase and magnitude, and onset and completion times.

The residue, or the sound with deterministic components removed, is extracted after the sinusoidal tracks have been identified. TAPESTREA eliminates peaks in a sinusoidal track from the corresponding spectral frame by smoothing down the magnitudes of the bins beneath the peak. It also randomizes the phase in these bins. Figure 2 shows sinusoidal separation results.

4.2. Transient Detection and Separation

Transients are brief stochastic sounds with high energy. While a sinusoidal track looks like a near-horizontal line on a spectrogram, a transient appears as a vertical line, representing the simultaneous presence of information at many frequencies. Transients are usually detected in the time domain by observing changes in signal energy over time [18, 19]. TAPESTREA processes the entire sound using a non-linear one-pole envelope follower filter with a sharp attack and gradual decay to detect sudden increases in energy. Points where the ratio of the envelope's derivative to the average frame energy is above a user-specified threshold mark transient onsets. A transient's length is also user-specified and can thus include any amount of the decay. Other real-time analysis parameters include the filter attack and decay coefficients, and aging amount in computing average frame energy. Transient events, being brief and noisy, are represented as raw sound clips, although they can also be modeled by peak picking in the time domain [18].

Detected transients are removed, and the resulting "holes" are filled by applying wavelet tree resynthesis [2]. The nearest transient-free segments before and after a transient event are combined to estimate the background that should replace it. Wavelet tree learning generates more of this background, which is overlap-added into the original sound to replace the transient. The residue from the sinusoidal analysis, with transients removed in this way, is saved to file and used for stochastic background generation in the synthesis phase. Figure 3 demonstrates the hole-filling.

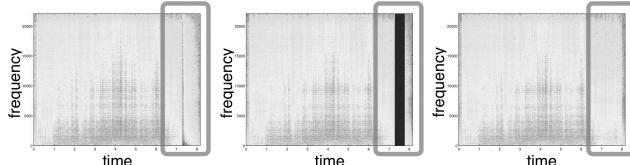


Figure 3: Transient removal and hole filling: (a) firework with pop (at 7.2 sec); (b) pop removed; (c) hole filled.

5. TRANSFORMATIONS

We now have *deterministic events* isolated in time and frequency from the background, *transient events*, and *stochastic background texture*. Output sound scenes are parametrically constructed from these templates. The parametric model lets each transformation be applied to each component independently of others.

5.1. Event Transformations

By stretching or compressing spectral data, we can raise or lower the **frequency** content of a sound without affecting its duration. For deterministic events with sinusoidal tracks, TAPESTREA linearly scales the frequency at each point in each track, giving high fidelity frequency warping for almost any factor (limited by our range of hearing). For transients, it uses a standard phase vocoder [20] to similarly scale the frequency for each frame.

The track-based representation of deterministic events allows us to robustly change the **duration** of each track by almost any factor without producing artifacts, by scaling the time values in the time-to-frequency trajectories of their tracks. Both time-stretching and frequency-warping can take place in real-time for deterministic events. Time-stretching for transients once again uses a phase vocoder to stretch or shorten the temporal overlap between frames.

TAPESTREA offers control over the **temporal placement** of an individual event, explicitly or using a probability distribution. Explicitly, an event instance can be placed on a timeline at a specified onset time. The timeline may also include other event instances and background sound. Repeating events can be defined by a mean event density and desired repetition periodicity, and generated according to these parameters by a Gaussian or other distribution. Events can also be panned across two speakers.

5.2. Stochastic Background Transformations

It is possible to interactively control the similarity between an extracted background and the synthesized background generated from its template. The similarity or randomness is governed by the wavelet tree learning (Section 6.2) parameters. Also, the generated background can play for any arbitrary amount of time.

6. SYNTHESIS

TAPESTREA synthesizes a sound scene following the specified transformations. The background component and the events are synthesized separately and combined to produce the final scene. Each component can also be heard in isolation so that a user can determine its role in the final scene. Although we discuss transformation and synthesis in separate sections for clarity, these two aspects are closely related. For example, components can be transformed in certain ways even while they are being synthesized.

6.1. Event Synthesis

Deterministic events are synthesized from their defining tracks with sinusoidal re-synthesis. The system linearly interpolates frequency and magnitude between consecutive frames before computing the time-domain sound from these. Transient events are directly played back or, if a frequency-warping or time-stretching factor is specified, analyzed and synthesized through a phase vocoder.

6.2. Stochastic Background Generation

The background is generated using an extension of the wavelet tree learning algorithm by Dubnov et. al. [2]. The extracted stochastic background is decomposed into a wavelet tree (Daubechies, 5 vanishing moments), where each node represents a wavelet coefficient. A new wavelet tree is learned, with nodes selected from the original tree by their context, within a specified randomness range.

We added the option of incorporating randomness into the first step of the learning, and modified the amount of context used ('k') to depend on the node's depth. We also found that we can avoid learning the coefficients at the highest resolutions, without perceptually altering the results. Since the wavelet tree is binary, learning at the highest levels takes longer, but randomizes mainly high-frequency information. The optimization let us build a real-time version of wavelet tree learning, with interactive control over the learning parameters. The wavelet tree learning also works better with the separated stochastic background as input since the harmonic events it would otherwise garble have been removed.

6.3. Putting It All Together

To construct a sound scene, extracted background and events are combined to the user's preference. A scene of a specified length can be generated by placing templates on a timeline of the desired length. Infinitely long sound scenes can also be generated and modified on-the-fly. The improved wavelet tree algorithm synthesizes unlimited background texture, while event templates can be temporally placed against the background either with fine control or in an automated manner (see Section 5.1).

This framework adapts to many techniques for synthesizing the final sound. A user may craft a sound scene by listening to and adjusting the components separately, based on how they sound as a group or individually. The combined sound can then be similarly sculpted. On the other hand, the synthesis can also be driven from a game or animation algorithm that specifies transformations according to parameters drawn from the game or animation itself.

7. USER INTERFACE

The user interface (Figure 4) is separated into two phases: analysis and synthesis. In the analysis stage, the user can load a sound file and view its waveform, frame-by-frame spectrum and spectrogram. These views allow the user to visually identify events and perform analysis on appropriate time and frequency regions to extract specific events. Time and frequency bounds for the analysis can be specified by adjusting range sliders in the waveform and spectrum views or by selecting a rectangle in the spectrogram view. The frame-by-frame spectrum also shows the sinusoidal analysis threshold. Direct control over many other analysis parameters (Section 4) is also available. Having adjusted the analysis

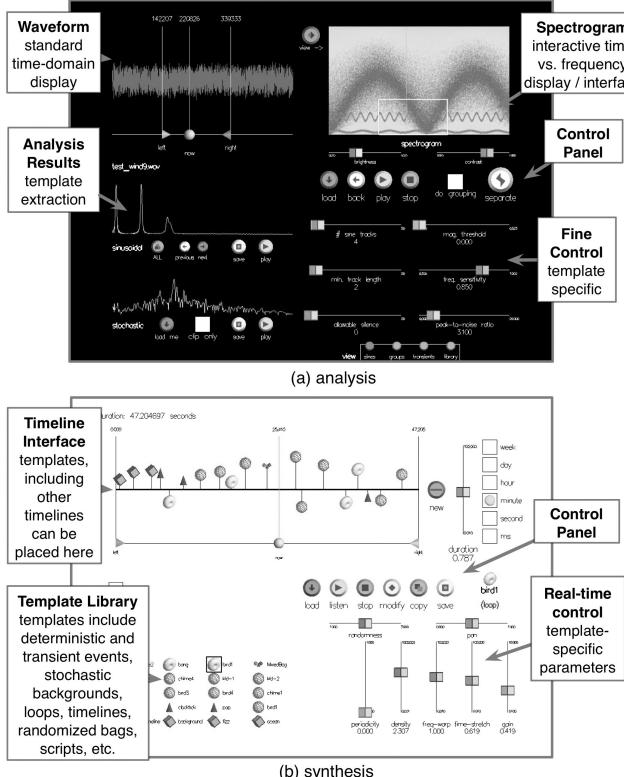


Figure 4: Screen shots of user interface.

parameters, the user starts analysis by clicking a button. The extracted events are played separately, along with a frame-by-frame view of their spectrum (for deterministic events) or a zoomed in view of their waveform (for transient events). The stochastic background is similarly played and viewed, or loaded for further analysis. An extracted event or background can be saved as a template for use in the synthesis phase. The user may then perform further analysis on the same source sound or a different one, or move on to the synthesis phase.

The synthesis stage of the interface offers a framework for applying transformations and synthesizing the resulting sounds. Templates saved from the analysis stage are available in the synthesis stage for listening, transforming, and placing in a sound scene. Templates include the following types: (1) *deterministic events*, (2) *transient events*, (3) *stochastic background*, (4) *loops*, and (5) *timelines*. The first three are imported directly from the analysis results, while loops and timelines are as described in Section 5.1. Any event can be saved as a loop, with parameters specifying how often it repeats and how periodic the repetition is. Event instances within a loop can also be randomly transformed within a controllable range, so that every iteration of the loop sounds slightly different. This is useful in generating ‘crowd’ sounds, such as a flock of birds constructed from a single extracted chirp, or many people from a single voice. While loops parametrically repeat a single event, timelines control the explicit temporal placement of any number of components for a specified duration. Any existing template can be dragged on to a timeline; its location on the timeline determines when it is synthesized. When a timeline is played, each template on it is synthesized at the appropriate time step and played for its duration or until the timeline ends. It is also possible to place timelines within timelines, to capture details of

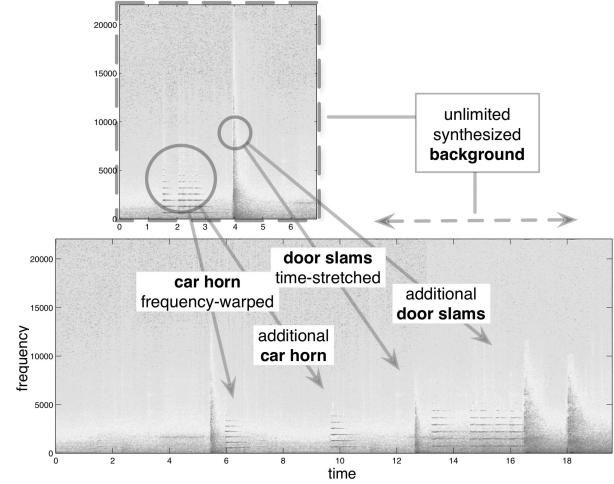


Figure 5: Existing sound scene (top) lengthened and transformed (bottom) with time/frequency warps and continuous background.

a sound scene at different temporal resolutions. Any synthesized sound scene can be written to file while it plays, or play forever.

8. RESULTS AND CONTRIBUTIONS

Figure 5 shows an example where a single existing sound scene is transformed into a different one. An original 6 second recording from a warehouse environment, with some background noise, a multi-toot horn, and a door slamming was analyzed. The multi-toot horn was extracted and saved as multi-toot and single-toot deterministic event templates. The door-slam transient and final stochastic background sound were separated. A new scene of length 19 seconds was constructed using randomized non-looping wavelet tree re-synthesis for the background. The new scene combines multiple and overlapping versions of frequency and time shifted single horns, a multi-toot horn, and door slams (some transformed so greatly that they sound like explosions).

Our main contributions comprise of the approach and framework for analysis, transformation, and synthesis of sound scenes. In particular, they are (1) techniques and paradigms for interactive selection and extraction of templates from a sound scene, (2) techniques for parametrically transforming components independently of each other, (3) a framework for flexible resynthesis of events and synthesis of novel sound scenes, (4) an interface to facilitate each analysis and synthesis task. Furthermore, we have refined several of the core algorithms employed in our system, as follows.

Firstly, we extend the wavelet tree algorithm to continually resynthesize the background component, by speeding up the learning. Tests show a 4x speedup in total running time between the original algorithm (15 levels) and our modified version (stopping at 9 levels), for a 1 minute 58 second sound clip to be generated from an 18 second clip sampled at 11 kHz.

Secondly, we refine the sinusoidal extraction process by letting users parametrically extract specific events. The data structures for grouping and storing sinusoidal tracks as objects are also a first step towards object classification and computational auditory scene analysis [21].

Thirdly, we use wavelet tree learning to fill in the gap left by transient removal (Section 4.2). A clear sonic difference can be

discerned between attenuating the transient segment to the noise floor versus automatically replacing it with a stochastic sound clip produced by wavelet tree learning.

9. CONCLUSION AND FUTURE WORK

We have described a framework for extracting specific parts of existing sound scenes and flexibly reusing these to create new sound scenes of arbitrary length. It allows users to interactively highlight points of interest in the input sound, separating it into well-defined components. This allows greater control over the synthesized scene, letting elements from different sound scenes be transformed independently and combined. We have also demonstrated an interactive paradigm for building new sound scenes, which includes iterative refinement of components, interactive previews of transformations, grouping, and placement in time and space. Due to the separation, our system is effective in analyzing and synthesizing many classes of sounds. It is, to our knowledge, the first to provide a comprehensive approach for extracting / transforming / resynthesizing the different component templates, first individually, then into cohesive sound scenes.

While our system has no fundamental restriction on the type of input sound to analyze, there are some limitations. When two events overlap in both time and frequency, it can be hard for the analysis to distinguish between them. Further, it is difficult to extract an event's reverberation. Also, when events have strong deterministic as well as stochastic components, these components get separated and may be difficult to regroup. Future work includes overcoming these limitations by (1) using more sophisticated event tracking and grouping methods, and (2) extending the idea of objects to include composite events with both deterministic and transient components. In addition, we plan to combine machine learning techniques to classify events and to improve performance without human assistance over time.

To sum up, our main contributions comprise the approach, system, and interface for selective extraction, transformation, and resynthesis of sound scenes. While there is plenty of scope for future work, TAPESTREA makes it possible to create novel sound scenes from existing sounds in a flexible and parametrically controlled manner, providing a new paradigm for both real-time and offline sound production.

10. ACKNOWLEDGEMENTS

Thanks to members of the Princeton soundlab and graphics group.

11. REFERENCES

- [1] X. Serra, "A system for sound analysis / transformation / synthesis based on a deterministic plus stochastic decomposition," Ph.D. dissertation, Stanford University, USA, 1989.
- [2] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Synthesizing sound textures through wavelet tree learning," *IEEE Computer Graphics and Applications*, vol. 22, no. 4, pp. 38–48, 2002.
- [3] P. R. Cook, *Real Sound Synthesis for Interactive Applications*. A. K. Peters, 2002.
- [4] ——, "Modeling BILL'S GAIT: analysis and parametric synthesis of walking sounds," in *Proc. Audio Eng. Soc. 22nd Conf. Virtual, Synthetic and Entertainment Audio*, 2002, pp. 73–78.
- [5] J. O'Brien, P. Cook, and G. Essl, "Synthesizing sounds from physically based motion," in *Proc. ACM SIGGRAPH*, Los Angeles, USA, 2001, pp. 529–536.
- [6] D. Rocchesso, R. Bresin, and M. Fernström, "Sounding objects," *IEEE Multimedia*, vol. 10, no. 2, pp. 42–52, 2003.
- [7] K. van den Doel, P. G. Kry, and D. K. Pai, "FOLEYAUTOMATIC: Physically-based sound effects for interactive simulation and animation," in *Proc. ACM SIGGRAPH*, Los Angeles, USA, 2001, pp. 537–544.
- [8] M. Athineos and D. P. Ellis, "Sound texture modelling with linear prediction in both time and frequency domains," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'03)*, Hong Kong, China, 2003, pp. 648–651.
- [9] X. Zhu and L. Wyse, "Sound texture modeling and time-frequency LPC," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 345–349.
- [10] N. E. Miner and T. P. Caudell, "Using wavelets to synthesize stochastic-based sounds for immersive virtual environments," in *Proc. 4th Int. Conf. on Auditory Display (ICAD'97)*, Palo Alto, USA, 1997, pp. 69–76.
- [11] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [12] Y. Qi, T. P. Minka, and R. W. Picard, "Bayesian spectrum estimation of unevenly sampled nonstationary data," MIT Media Lab Technical Report Vismod-TR-556, Oct. 2002.
- [13] H. D. Thornburg and R. J. Leistikow, "Analysis and resynthesis of quasi-harmonic sounds: An iterative filterbank approach," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003, pp. 129–134.
- [14] M. Klingbeil, "Software for spectral analysis, editing, and synthesis," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 107–110.
- [15] X. Amatriain and P. Arumi, "Developing cross-platform audio and music applications with the CLAM framework," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 403–410.
- [16] D. P. W. Ellis, "A computer implementation of psychoacoustic grouping rules," in *Proc. 12th Int. Conf. Pattern Recognition*, 1994, pp. 108–112.
- [17] K. Melih and R. Gonzalez, "Source segmentation for structured audio," in *Proc. IEEE Int. Conf. Multimedia and Expo (II)*, 2000, pp. 811–814.
- [18] T. S. Verma and T. H. Meng, "An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'98)*, Seattle, USA, 1998, pp. 12–15.
- [19] 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 and Audio Proc.*, vol. 13, no. 5, pp. 1035–1047, 2005.
- [20] M. B. Dolson, "The phase vocoder: A tutorial," *Computer Music J.*, vol. 10, no. 4, pp. 14–27, 1986.
- [21] A. S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, 1990.

SOUND PROCESSING IN OPENMUSIC

Jean Bresson

IRCAM - CNRS, Music Representations Team
Paris, France
bresson@ircam.fr

ABSTRACT

This article introduces some new possibilities of audio manipulations and sound processing in the Computer-Aided Composition environment OpenMusic. Interfaces with underlying sound processing systems are described, with an emphasis on the use of the symbolic and visual programming environment for the design of sound computation processes.

1. INTRODUCTION

OpenMusic (OM) [1] is a visual programming language specialized in the symbolic processing of musical objects that provides composers with high-level musical representations. Following the tradition of the IRCAM's Computer-Aided Composition (CAC) systems [2], the main proposal of OM is that of the formalization of the musical structures, which makes it possible the creation of the corresponding computing models. However, the idea of a symbolic manipulation of musical data, close to the compositional concepts, historically tended to make these systems diverge from the sound-related concerns.

The development of contemporary musical practices, which incorporate sounds in the compositional processes, led to a renew interest for bringing together these two different fields. In this context, the OM environment provides possibilities for bridging the compositional and signal processing domains.

In a first step, this connection has been implemented with the automatic generation of scripting or parameter files bound to external processing tools. Various OM libraries were developed in this scope, in relation with different software synthesizers (generation of Csound scores [3], of Diphone [4] scripts, of SuperVP parameter files [5], etc.) Some direct interfaces with these sound processing tools were then created, which allowed an easier experimentation, by launching sound processing tasks directly from the OM visual programs (*patches*) [6]. More than a simple convenience of use, this allows to unite sound processing programs and symbolic musical processes in a same environment and in a common computational flow, which induces improved possibilities for the control and integration of these programs in compositional models (e.g. [7]).

Our current works in OpenMusic are heading towards new tools and interfaces for adapting the computational and symbolic tools to sound processing applications. The articulation between sound signal and musical symbolism in composition is one of the main problematics of this project, which points at exploring the symbolic resources bestowed by computers and CAC in order to describe audio signals and processes.

This article presents some new possibilities of sound manipulations in OpenMusic. The IRCAM's analysis/synthesis tools SuperVP and pm2 are particularly targeted, following the idea of an

integrated programming environment where various cycles of the musical materials, from sound to symbolic data, and from symbolic data to sounds, could be developed.

2. NEW AUDIO INTERFACE IN OPENMUSIC

Since version 5.0, the OpenMusic audio architecture is based on the GRAME's LibAudioStream library [8]. This library provided a better control for sounds scheduling and rendering through the concept of sound stream structures. The sound objects now can be individually assigned volume or panoramic values. They are possibly split up into different audio tracks, which allows a global interactive control through a graphical mixing console (Figure 1).

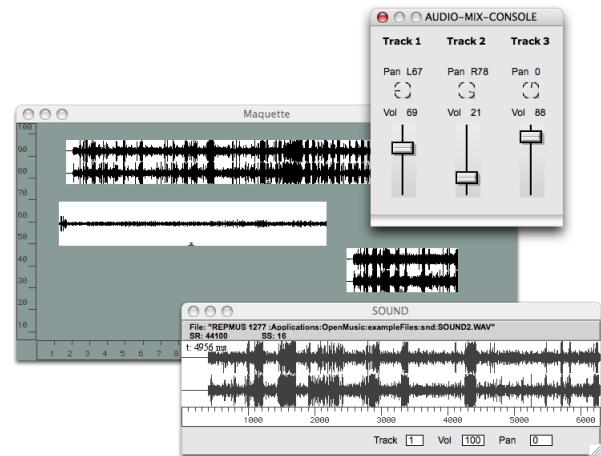


Figure 1: Audio interface in OM 5.

3. A COMPOSITIONAL APPROACH OF SOUNDS

3.1. The Digital Sound

Once it is recorded, the sound can become an abstract compositional object, thanks to the “material” support [9]. From this perspective, the digitalization constitutes a major revolution in the realm of the possibilities for a compositional approach of sounds.

An audio file, digital support of a sound, is made of a sequence of samples; each one of them carries a small quantity of information, but the sequence constitutes a whole bigger than the sum of all the individual samples: a stable acoustic waveform that can be observed and manipulated. From a theoretical point of view, this idea can be compared with the symbolic/sub-symbolic distinction formulated in [10]. In this division, sub-symbolic data are void of

sense out of their context, while symbolic data are structured and bring semantic information: they have an intrinsic meaning and are straightforward to be used in musical constructions.

3.2. Sound Objects and Symbolic Processing

Considered as a continuous stream, the sound samples constitute a coherent data set (a digital waveform), close to the acoustic reality and to the musical material.

The basic musical manipulations on this sound representation are the cut, mixing, and sequencing, as could be done with a "physical" tape. In the context of visual programming, these operations correspond to functional call boxes on which can be connected sound objects. They are permitted by the LibAudioStream functionalities. Sound streams can thus be created, cut, mixed, and sequenced in visual programs (see Figure 2). The programming tools allow to make algorithmically the "montage" and mixing of sounds, generally "hand-made" using sequencing environments after recording, processing, or synthesizing them.

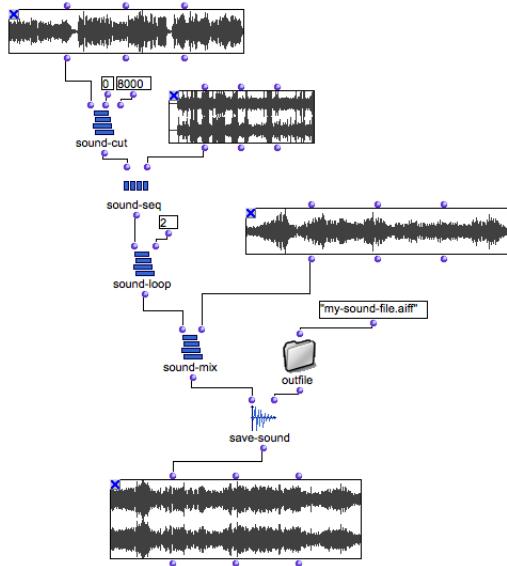


Figure 2: Manipulations of sounds with the OM audio tools. A section of the first sound is sequenced with the second one and this sequence is repeated two times. The result is then mixed with a third sound.

3.3. Abstraction of Sound

An abstraction of a sound can be any symbolic data that represent this sound in a compositional purpose. Between the symbolic and sub-symbolic domains, the data extracted from a sound following a scientific or musical demarche constitute the successive steps of abstraction that allow for its compositional manipulation.

The segmentation can be a first method for considering a sound in a symbolic manner, since it identifies primitives for a symbolic processing (the sound segments). The segmentation can be integrated in the sound objects by the use of temporal markers. In OM, markers can be assigned to a sound either by a program (in a patch), or manually inside the sound editor (see Figure 3).

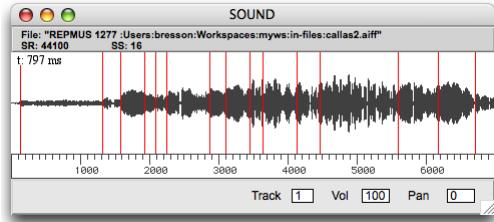


Figure 3: Editing of temporal markers in an audio file.

However, the size of the segments that will allow for a symbolic manipulation remains hard to decide. Segments of two samples remain sub-symbolic, while an intelligent segmentation, that discriminates syllables for example, constitutes symbolic data. Between these two limits, the size of a sound analysis window (e.g. 512 samples), or the units of a concatenative sound synthesis algorithm, can be considered either in one or another category.

In another general approach, a set of isolated samples, extracted from a waveform by any formalized way (e.g. downsampling, see Figure 4), can also be regarded as an abstraction of the digital sonic material.

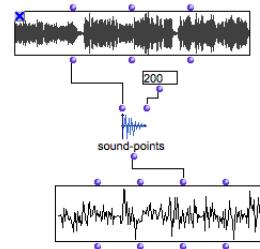


Figure 4: Extracting sound samples from an audio file with a given sampling rate.

This method was used for example by K. Haddad in the piece *Adagio for String Quartet*, described in [11].

Sound analysis is another way to get sound representations in compositional processes. This will be detailed in the next section.

4. SOUND ANALYSIS

Sound analysis data are frequently used in computer-assisted composition programs, either for being transformed and dispatched in sound synthesis processes, or for being used as basic compositional material. The SDIF file format [12] allows the different computer music tools and software to share sound descriptions data: a special class (*SDIFFile*) is used to represent it in OpenMusic. This class constitutes an abstract support for sound descriptions [13]; it will generally correspond to an intermediate object in the visual program, used to store the analysis data coming from the sound processing kernels. SDIF file objects can be visualized using SDIF-EDIT [14], and inspected in order to extract the description data .

SuperVP [15] and pm2 are two sound processing tools developed by the Analysis/Synthesis team of IRCAM. They are the processing kernels of the AudioSculpt software [16]. Used autonomously in OM, they provide various types of sound analysis,

and return SDIF description files.

4.1. Spectral Analysis

Most of the sound processing algorithms in the SuperVP phase vocoder [17] are based on the short time Fourier transform (STFT), which provides a time/frequency representation of sounds. This analysis function can be called in OpenMusic by the *FFT* box (see Figure 5). The analysis data is then transferred into the OM program via an SDIF file.

For this analysis, the main input data to provide to the *FFT* box (in addition to the sound file to be analyzed) are the FFT size, and the size, shape and step of the STFT analysis window. These parameters should be adjusted depending on the initial sound and the expected characteristics of the analysis data.

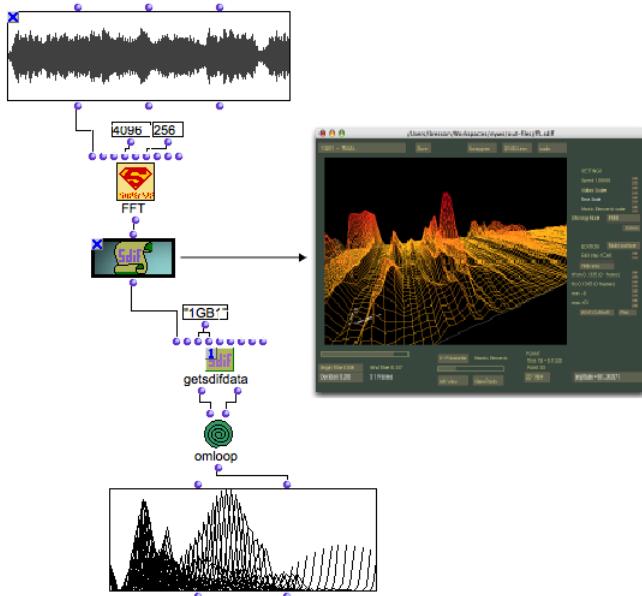


Figure 5: SuperVP FFT analysis in OpenMusic. The SDIF analysis data can be viewed using SDIF-EDIT and extracted in the patch.

The fundamental frequency estimate (F0) analysis [18] is another description provided by SuperVP and which is likely to be used in compositional and sound processing programs: its direct musical meaning helps musical interpretations and manipulations. In addition to the FFT settings, some supplementary parameters must be set to the *f0-estimate* box (frequency ranges for analysis, noise threshold, smoothing order, etc.) An example will be given in section 7 (Figure 11).

OpenMusic can also invoke the transient detection analysis [19] from SuperVP. This analysis provides an automatic temporal segmentation of the sound files. This segmentation can be useful for correlating sound descriptions to symbolic rhythms or temporal structures, as will be illustrated in the example of Figure 10.

4.2. Additive Analysis

Pm2 is an additive analysis/synthesis software that can process partial tracking analysis of sound files. The partial tracking consists in the detection of the most important partials (sinusoidal

components of a sound signal) and the following of their frequency and dynamic evolutions [20].

This analysis can be either harmonic, in which case a fundamental frequency evolution must be provided and the partials will be considered as harmonics of this fundamental frequency, or inharmonic, in which case each partial constitutes an independent component with individual beginning and ending times, frequency and amplitude evolutions.

The results of the pm2 partial tracking analysis are also stored in SDIF files, and can then be converted into a *chord-sequence* object [21], as illustrated in Figure 6.

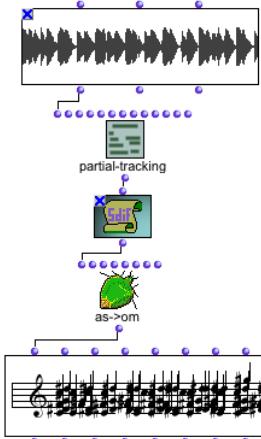


Figure 6: Partial tracking analysis of a sound and conversion into a chord sequence.

The *chord-sequence-analysis* is another kind of additive analysis based on a set of temporal markers. These markers can be issued from the sound object (see Figure 3), or being processed by a preliminary transient detection analysis. The additive components are thus averaged frequencial components that can be directly converted into chords or notes.

5. TREATMENTS

We call treatment the fact of synthesizing a sound by applying a transformation to an initial sound. Such treatments can be applied directly on the waveform representation, or by the analysis/processing/synthesis method.

5.1. Sound Streams Processing

Transforming a sound by processing its samples stream is what is generally called an effect. We currently experiment the integration of such process in OM using Faust [22], a language for the design and compilation of digital sound processing (DSP) objects. The symbolic primitives and syntax of this language make it a suitable tool for the design of sound treatments in a compositional context. The LibAudioStream interface then allows to apply Faust compiled effects to the sound streams (see Figure 7).

5.2. Analysis-driven Processing: SuperVP

The analysis/treatments/synthesis method proposed by SuperVP and AudioSculpt allows for original and powerful manipulations

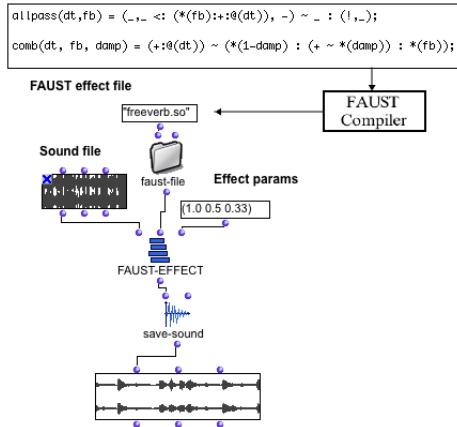


Figure 7: Applying a transformation to a sound using Faust. The framed code is a fragment of the Faust effect definition file.

of sounds. Thanks to a treatments sequencer, various kinds of sound transformations can be created and combined, and eventually applied in a processing phase for creating a new sound.

In the OM visual programming framework, the corresponding system is centred on the *supervp-processing* function, a general box for transforming a sound using various combinations of the SuperVP treatments. Each available treatment is represented with another particular box. These boxes include the time stretching, pitch shifting, transposition, band filter, formant filter, breakpoint filter, clipping, and freeze treatments. Each treatment may require some special parameters settings, and can be applied to the entire sound or to a time interval in it. They can be used alone or combined, connected to the general *supervp-processing* box as a list of treatments (see Figure 8).

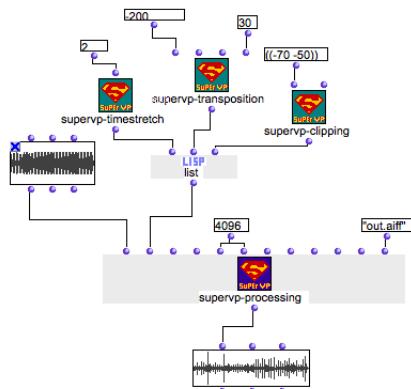


Figure 8: Applying treatments (time stretch, transposition, and clipping) with the supervp-processing box.

The parameters (e.g. stretching factors, transposition intervals, etc.) can be set with constant values or with time-varying data using imported parameter files or break point functions (BPF). These data can thus come from complex compositional processes implemented in OM. Section 7 will give examples of sound processing patches.

6. SYNTHESIS

Several works have been carried out regarding sound synthesis models and interfaces in OM, most often using Csound [23] which provides a wide range of sound synthesis possibilities. Some interfaces have also been developed with the IRCAM's sound processing tools.

In addition to the treatments processing discussed in the previous section, the *supervp-cross-synthesis* and *supervp-sourcefilter-synthesis* are other boxes that call the corresponding SuperVP functions (see Figure 9).

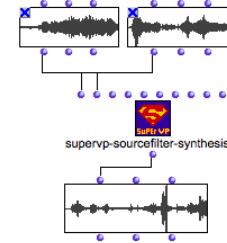


Figure 9: : Source-Filter synthesis with SuperVP.

The pm2 additive synthesis, performed from an SDIF additive description file, also exists as a function call box.

The CHANT synthesizer [24] can be used in OM as well, by formatting SDIF parameters files and calling the synthesis kernel. Future works should improve this interface in order to allow for easier and better use of this synthesizer's potentialities.

Finally, the OM-Modalys library allows the design of Modalys physical models synthesis in OM programs [25].

Most of these tools are "direct" interfaces that do not exactly represent the abstract methods generally targeted in CAC applications. They might nonetheless provide a base for the development of higher-level control interfaces.

7. USING OPENMUSIC FOR SOUND PROCESSING

The interest in creating sound processing programs in OM relies on two principal points that we will detail in this section.

7.1. Computation of Processing Parameters

Using sound processing tools in OM allows to set the parameters of the treatments by complex programs and with the symbolic data structures provided by the environment. We will show two examples, where the sound analysis tools presented above are used together with symbolic data structures, for the computation of some processing parameters.

In the example of Figure 10, a time-stretching factor evolution is computed starting from the attack transients detected in a sound file and from a targeted rhythm: this variable factor will stretch the sound segments in order to make them match with this rhythm.

The example of Figure 11 is another program which transposes a sound using time-varying data coming from the F0 analysis. The transposition data is computed by an operation of symmetry between the F0 and a given pitch, so that the result is a sound with a constant fundamental frequency (the value of the reflection axis), which keeps all the other characteristics (timbre,

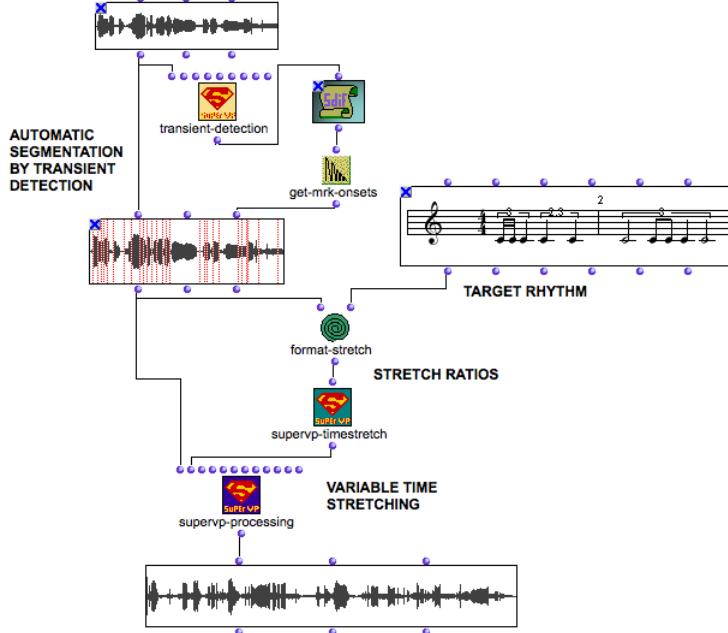


Figure 10: Variable time stretching: matching sound segments with a rhythm.

transients, noise, etc.) from the original sound.

In the same perspective, the OM-AS library functions [5], dedicated to the automatic generation of SuperVP treatment parameters files, can also be connected to the corresponding treatment boxes, providing a symbolic mean to generate these parameters (e.g. computing a sound transposition parameter file starting from a *chord-seq* melody, etc.)

7.2. Iterative Processes

The visual programming tools may also be used for managing multiple processing operations. A sound, or a sound bank, can be processed iteratively using the *omloop* control structure, in order to create another set of sound files depending on fixed or dynamically generated parameters (see Figure 12).

The output sounds can then possibly be arranged in a *maquette* [1] (Figure 12), or directly mixed/sequenced as a single sound file using the audio processing tools discussed in section 3.2.

8. CONCLUSION

We presented the possibilities of an algorithmic and symbolic manipulation of sounds in the OpenMusic visual programming environment, using the audio framework (manipulation of the sound objects) and the IRCAM sound processing tools interfaces (fft analysis, F0 analysis, transient detection, partial tracking analysis, sound treatments, cross-synthesis, etc.) All these tools are available and documented with tutorial patches in the latest OM releases, either included in the OM core distribution (audio features) or in the new OM-SuperVP library (SuperVP and pm2 tools).

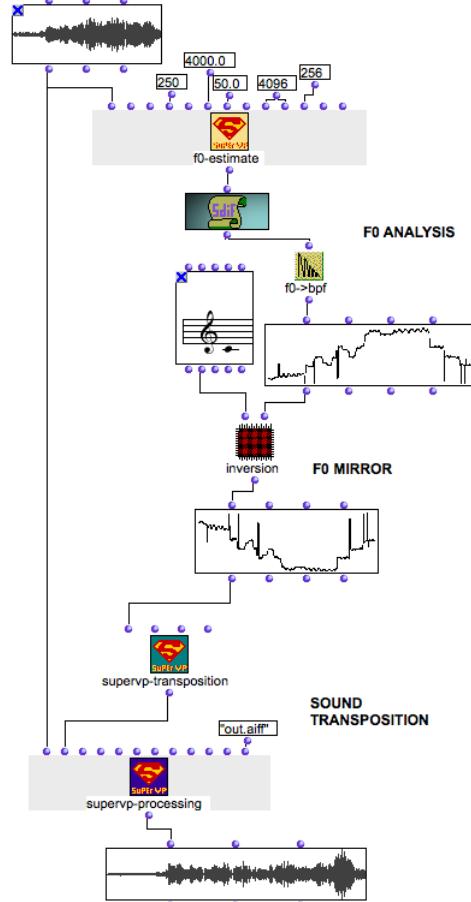


Figure 11: Fundamental frequency analysis (F0) of a sound and transposition using the mirror of the F0.

Graphical programming interfacing with sound manipulations in a compositional context. The iterative processes, coupled with visual representations and audio processing tools, allow for efficient and intuitive experiments. We have in mind that more interfaces with external tools could be added to this framework, which will be completed with higher-level abstractions and assort with further musical experiences and applications.

9. ACKNOWLEDGEMENTS

We would like to thank Stephane Letz from GRAME for his help and support with the LibAudioStream library. The SuperVP boxes and the examples presented in this article have been created with Jean Lochard from the IRCAM pedagogic department. SuperVP and pm2 are currently developed and maintained by Axel Röbel in the IRCAM Analysis/Synthesis team.

10. REFERENCES

- [1] C. Agon, "OpenMusic: Un langage visuel pour la composition assistée par ordinateur," Ph.D. dissertation, University of Paris 6, 1998.

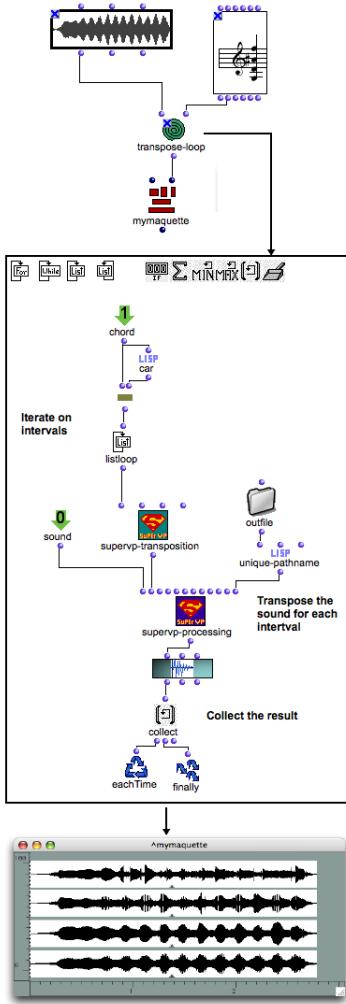


Figure 12: Iterative creation and superposition of sounds by transposing a sound file following the intervals of a chord.

- [2] G. Assayag, "Computer-assisted composition today," in *First Symposium on Music and Computers*, Corfu, 1998.
- [3] K. Haddad, "OpenMusic OM2CSound," *Ircam Software Documentation*, 1999.
- [4] X. Rodet and A. Lefèvre, "The Diphone program: New features, new synthesis engines and experience of musical use," in *Proc. Int. Comp. Music Conf. (ICMC'97)*, Thessaloniki, Greece, 1997, pp. 418–421.
- [5] H. Tutschku, "OpenMusic OM-AS Library," *Ircam Software Documentation*, 1998.
- [6] J. Bresson, M. Stroppa, and C. Agon, "Symbolic control of sound synthesis in computer assisted composition," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 303–306.
- [7] C. Agon, M. Stroppa, and G. Assayag, "High level musical control of sound synthesis in openmusic," in *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, 2000, pp. 332–335.
- [8] LibAudioStream, Retrieved June 29th, 2006, [Online] <http://libAudioStream.sourceforge.net/>.
- [9] P. Schaeffer, *Traité des Objets Musicaux*. Paris: Seuil, 1966.
- [10] M. Leman, "Symbolic and subsymbolic description of music," in *Music Processing*, G. Haus, Ed. Oxford University Press, 1993, pp. 119–164.
- [11] K. Haddad, "Timesculpt in OpenMusic," in *The OM Composer's Book*, C. Agon, G. Assayag, and J. Bresson, Eds. Ircam – Delatour, 2006, pp. 45–62.
- [12] Ircam, "Sound Description Interchange Format," Retrieved June 29th, 2006, [Online] <http://www.ircam.fr/sdif/>.
- [13] J. Bresson and C. Agon, "SDIF sound description data representation and manipulation in computer assisted composition," in *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, 2004, pp. 520–527.
- [14] Ircam, "SDIF-Edit: Visualization of SDIF sound description files," 2001, [Online] <http://recherche.ircam.fr/equipes/repmus/bresson/sdifiedit/sdifiedit.html>.
- [15] P. Depalle and G. Poirot, "A modular system for analysis, processing and synthesis of sound signals," in *Proc. Int. Comp. Music Conf. (ICMC'91)*, Montréal, Canada, 1991, pp. 161–164.
- [16] N. Bogaards and A. Röbel, "An interface for analysis-driven sound processing," in *119th Conv. Audio Eng. Soc.*, New York, USA, 2004.
- [17] M. Dolson, "The phase vocoder: A tutorial," *Computer Music J.*, vol. 10, no. 4, pp. 14–27, 1986.
- [18] B. Doval and X. Rodet, "Estimation of fundamental frequency of musical sound signals," in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'91)*, Toronto, Canada, 1991, pp. 3657–3660.
- [19] A. Röbel, "Transient detection and preservation in the phase vocoder," in *Proc. Int. Comp. Music Conf. (ICMC'03)*, Singapore, 2003, pp. 247–250.
- [20] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.
- [21] P. Hanappe and G. Assayag, "Intégration des représentations temps/fréquence et des représentations musicales symboliques," in *Recherches et applications en informatique musicale*, M. Chemillier and F. Pachet, Eds. Hermès, 1998, pp. 199–207.
- [22] Y. Orlarey, D. Fober, and S. Letz, "Syntactical and semantic aspects of Faust," *Soft Computing*, vol. 8, no. 9, pp. 623–632, 2004.
- [23] R. Boulanger, *The Csound Book*. MIT Press, 2000.
- [24] X. Rodet, Y. Potard, and J.-B. Barrière, "The CHANT project: From the synthesis of the singing voice to synthesis in general," *Computer Music J.*, vol. 8, no. 3, pp. 15–31, 1984.
- [25] N. Ellis, J. Bensoam, and R. Caussé, "Modalys demonstration," in *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005, pp. 101–102.

Full Bibliography

This bibliography is a compilation of all bibliographic references from each DAFx-06 paper. Page numbers that appear at the end of each entry link to the bibliography sections that include it. Please click on the URL or on the page number to access the linked item.

- Abe, M. and Smith, J. AM/FM rate estimation for time-varying sinusoidal modeling. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'05)*, Philadelphia, USA, 2005. 258
- Abel, J. and Berners, D. On peak-detecting and RMS feedback and feed-forward compressors. In *115th Conv. Audio Eng. Soc.*, New York, USA. Preprint 5914, 2003. 168
- Abel, J. S., Smith, J. O., and Bensa, J. An allpass filter design method with application to piano string synthesis. In *Proc. 149th Meeting Acoust. Soc. Am.*, Vancouver, Canada, 2005. 18
- Abramowitz, M. and Stegun, I. A. *Handbook of mathematical functions*. New York: Dover, 1970. 12
- Agon, C. *OpenMusic: Un Langage Visuel pour la Composition Assistée par Ordinateur*. PhD thesis, University of Paris 6, 1998. 330
- Agon, C., Stroppa, M., and Assayag, G. High level musical control of sound synthesis in openmusic. In *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, pp. 332–335, 2000. 330
- Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *Data Structures and Algorithms*, pp. 392–407. Series in Computer Science and Information Processing. Addison-Wesley, 1983. 186
- Algazi, V. R., Duda, R. O., Thompson, D. P., and Avendano, C. The CIPIC HRTF database. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, pp. 99–102, 2001. 232, 238
- Allen, J. B. and Berkley, D. A. Image method for efficiently simulating small-room acoustics. *J. Acoust. Soc. Am.*, 65(4), pp. 943–950, 1979. 220
- Allen, J. B. and Rabiner, L. R. A unified approach to short-time Fourier analysis and synthesis. *Proc. IEEE*, 65, pp. 1558–1564, 1977. 180
- Alonso, M., Badeau, R., David, B., and Richard, G. Musical tempo estimation using noise subspace projection. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, pp. 95–98, 2003. 148
- Altman, E. and Wyse, L. *Emergent Semantics from Media Blending*. Srinivasan and Nepal (Eds.), The Idea Group Inc., 2004. 48
- Amatriain, X. and Arumi, P. Developing cross-platform audio and music applications with the CLAM framework. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 403–410, 2005. 324
- Amatriain, X., Bonada, J., Loscos, A., Arcos, J. L., and Verfaille, V. Content-based transformations. *J. New Music Research*, 32(1), pp. 95–114, 2003. 113
- Amatriain, X., Bonada, J., Loscos, A., and Serra, X. Spectral processing. In U. Zölzer (Ed.), *DAFX – Digital Audio Effects* pp. 429–431. J. Wiley & Sons, 2002. 99
- Ampwares. 5F6-A schematic. [Online] http://www.ampwares.com/ffg/bassman_narrow.html, Retrieved June 29th, 2006. 5
- Anderton, C. *Electronic Projects for Musicians* (second edition ed.). New York: Amsco Publications, 1980. 168
- Ando, Y. *Concert hall acoustics*. Berlin: Springer Series in Electrophysics, 1985. 226
- Andrews, D., Peck, W., Agron, J., Preston, K., Komp, E., Finley, M., and Sass, R. hthreads: A hardware/software co-designed multithreaded RTOS kernel. In *10th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Catania, Italy, 2005. 290
- ANSI. *Information technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced Audio Coding (AAC)*. INCITS/ISO/IEC 13818-7, 2003. 252
- Arfib, D. Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves. *J. Audio Eng. Soc.*, 27(10), pp. 757–779, 1979. 198
- Arfib, D., Couturier, J., Kessous, L., and Verfaille, V. Strategies of mapping between gesture data and synthesis model. *Organised Sound*, 7(2), pp. 122–144, 2002. 48
- Arfib, D., Keiler, F., and Zölzer, U. Source-filter processing. In U. Zölzer (Ed.), *DAFX – Digital Audio Effects* pp. 361–362. J. Wiley & Sons, 2002. 99
- Arfib, D., Keiler, F., and Zölzer, U. Time-frequency processing. In U. Zölzer (Ed.), *DAFX – Digital Audio Effects* pp. 237–97. J. Wiley & Sons, 2002. 113
- Askenfelt, A. and Galembro, A. S. Study of the spectral inharmonicity of musical sound by the algorithms of pitch extraction. *Acoust. Physics*, 46(2), pp. 121–132, 2000. 76
- Assayag, G. Computer-assisted composition today. In *First Symposium on Music and Computers*, Corfu, 1998. 330
- Athineos, M. and Ellis, D. P. Sound texture modelling with linear prediction in both time and frequency domains. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'03)*, Hong Kong, China, pp. 648–651, 2003. 324
- Atig, M. *Non-linéarité acoustique localisée à l'extrémité ouverte d'un tube. Mesure, modélisation et application aux instruments à vent*. PhD thesis, Université du Maine, 2004. 88
- Aucouturier, J.-J. and Pachet, F. Ringomatic: A real-time interactive drummer using constraint-satisfaction and drum sound descriptors. In *Proc. Int. Conf. Music Information Retrieval (ISMIR'05)*, London, UK, pp. 412–419, 2005. 282
- Audiotools. Noise Reduction Systems. [Online] <http://audiotools.com/noise.html>, Retrieved June 29th, 2006. 302
- Auger, F. and Flandrin, P. Improving the readability of time-frequency and time-scale representations by the reassignment method. *IEEE Trans. Sig. Proc.*, 43(5), pp. 1068–1089, 1995. 148, 258
- Aurora. Plug-ins Home Page v4.0. [Online] <http://pcfarina.eng.unipr.it/aurora/home.htm>, Retrieved June 29th, 2006. 226
- Avanzini, F., Fontana, F., and Rocchesso, D. Efficient computation of non-linear filter networks with delay-free loops and applications to physically-based sound models. In *Proc. 4th Int. Workshop on Multidimensional Systems (NDS 2005)*, Wuppertal, pp. 110–115, 2005. 302
- Avendano, C. Frequency-domain source identification and manipulation in stereo mixes for enhancement, suppression, and re-panning applications. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, pp. 55–58, 2003. 238
- Badeau, R., David, B., and Richard, G. High resolution spectral analysis of mixtures of complex exponentials modulated by polynomials. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 54(4), pp. 1341–1350, 2006. 245
- Bank, B., Avanzini, F., Borin, G., Poli, G. D., Fontana, F., and Rocchesso, D. Physically informed signal processing methods for piano sound synthesis: A research overview. *EURASIP J. on Applied Sig. Proc.*, 2003(10), pp. 941–952, 2003. 76
- Bank, B. and Sujbert, L. Generation of longitudinal vibrations in piano strings: From physics to sound synthesis. *J. Acoust. Soc. Am.*, 117(4), pp. 2268–2278, 2005. 82, 120
- Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., and Werman, M. Texture mixing and texture movie synthesis using statistical learning. *IEEE Trans. Visualization and Computer Graphics*, 7(2), pp. 120–135, 2001. 36
- Barnsley, M. and Rising, H. *Fractals Everywhere*. Boston Academic Press Professional, 1993. 36
- Barthet, M., Guillemain, P., Kronland-Martinet, R., and Ystad, S. On the relative influence of even and odd harmonics in clarinet timbre. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 351–354, 2005. 24

- Bassuet, A. Acoustics of early music spaces from the 11th to 18th century (abstract). *J. Acoust. Soc. Am.*, 115(5, pt 2), pp. 2582, 2004. 226
- Bateman, L. A. Soprano, style and voice quality: Acoustic and laryngographic correlates. Master's thesis, University of Victoria, 2004. 160
- Bağcı, U. and Erzin, E. Boosting classifiers for music genre classification. In *20th Int. Symp. on Computer and Information Sciences (ISCIS 2005)*, also appeared in *Lecture Notes in Computer Science (LNCS)* by Springer-Verlag, Istanbul, pp. 575–584, 2005. 156
- Beauchamp, J. W. Synthesis by spectral amplitude and "brightness" matching of analyzed musical instrument tones. *J. Audio Eng. Soc.*, 30(6), pp. 396–406, 1982. 24
- Beeson, M. J. and Murphy, D. T. Virtual room modelling using hybrid digital waveguide mesh techniques. In *Proc. 147th Meeting Acoust. Soc. Am.*, New York, USA, volume 115(5), pp. 2515. Abstract, 2004. 164
- Beller, G., Hueber, T., Schwarz, D., and Rodet, X. Speech rates in french expressive speech. In *Speech Prosody*, Dresden, Germany, pp. 672–675, 2006. 282
- Beller, G., Schwarz, D., Hueber, T., and Rodet, X. A hybrid concatenative synthesis system on the intersection of music and speech. In *Journées d'Informatique Musicale (JIM)*, MSH Paris Nord, St. Denis, France, pp. 41–45, 2005. 282
- Bello, J., Duxbury, C., Davies, M., and Sandler, M. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Sig. Proc. Letters*, 11(6), pp. 553–556, 2004. 137
- Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., and Sandler, M. B. A tutorial on onset detection in music signals. *IEEE Trans. Speech and Audio Proc.*, 13(5), pp. 1035–1047, 2005. 137, 324
- Beltrán, J. R. and Beltrán, F. Additive synthesis based on the continuous wavelet transform: a sinusoidal plus transient mode. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003. 220
- Beltrán, J. R., de León, J. P., and Estopiñán, E. Intermodulation effects analysis using complex bandpass filterbanks. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, pp. 149–154, 2005. 220
- Benade, A. and Kouzoupis, S. The clarinet spectrum: Theory and experiment. *J. Acoust. Soc. Am.*, 83(1), pp. 292–304, 1988. 24
- Benade, A. H. Equivalent circuits for conical waveguides. *J. Acoust. Soc. Am.*, 83(5), pp. 1764–1769, 1988. 94
- Benade, A. H. *Fundamentals of musical acoustics* (2nd ed.). New York, NY, USA: Dover Publications, Inc., 1990. 120
- Bencina, R. The Metasurface – applying natural neighbour interpolation to two-to-many mapping. In *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, pp. 101–104, 2005. 48
- Benesty, J., Gänslер, T., and Eneroeth, P. Multi-channel sound, acoustic echo cancellation, and multi-channel time-domain adaptive filtering. In S. L. Gay & J. Benesty (Eds.), *Acoustic Signal Processing for Telecommunication* chapter 6, pp. 101–120. Kluwer Academic Publishers, 2000. 266
- Bensa, J. *Analysis and Synthesis of Piano Sounds using Physical and Signal Models*. PhD thesis, Université de la Méditerranée, France, 2003. 76
- Bensa, J. Stiff piano string modeling: Computational comparison between finite differences and digital waveguide. In *Proc. 148th Meeting Acoust. Soc. Am.*, San Diego, USA. Nov. 15–19, 2004. 18
- Bensa, J., Bilbao, S., Kronland-Martinet, R., and Smith III, J. O. The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides. *J. Acoust. Soc. Am.*, 114(2), pp. 1095–1107, 2003. 82
- Berdahl, E., Backer, S., and Smith III, J. O. If I had a hammer: Design and theory of an electromagnetically-prepared piano. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 81–84, 2005. 168
- Berdahl, E. and Smith III, J. O. Active damping of a vibrating string. In *Proc. Int. Symp. Active Noise and Vibration Control, Adelaide, Australia*, to appear in Aug. 2006. 168
- Berkhout, A. J., de Vries, D., and Vogel, P. Acoustic control by Wave Field Synthesis. *J. Acoust. Soc. Am.*, 93, pp. 2764–2778, 1993. 214
- Bernardo, J. N. and Smith, A. F. M. *Bayesian Theory*. Wiley, 1994. 308
- Bertrand, J., Bertrand, P., and Ovarlez, J. The Mellin transform. In A. Poularikas (Ed.), *The Transforms and Applications Handbook* pp. (11–53)–(11–54). CRC Press LLC, Boca Raton, Florida, U.S. in cooperation with IEEE Press, 2000. 99
- Besnainou, C. Transforming the voice of musical instruments by active control of the sound radiation. In *Proc. Int. Symp. Active Noise and Vibration Control*, Fort Lauderdale, Florida, 1999. 168
- Betlehem, T. and Williamson, R. C. Acoustic beamforming exploiting directionality of human speech sources. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'03)*, Hong Kong, China, pp. 365–368, 2003. 208
- Beurmann, A. and Schneider, A. Sonological analysis of harpsichord sounds. In *Proc. Stockholm Music Acoustics Conf. (SMAC'03)* Stockholm, Sweden, pp. 167–170, 2003. 120
- Bevilacqua, F., Müller, R., and Schnell, N. MnM: a Max/MSP mapping toolbox. In *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, pp. 85–88, 2005. 48
- Bidlack, R. Chaotic systems as simple (but complex) compositional algorithms. *Computer Music J.*, 16(3), pp. 33–47, 1992. 198
- Bilbao, S. *Wave and Scattering Methods for the Numerical Integration of Partial Differential Equations*. New York: John Wiley & Sons., 2004. 302
- Bilbao, S. Conservative numerical methods for nonlinear strings. *J. Acoust. Soc. Am.*, 118(5), pp. 3316–3327, 2005. 82
- Bilbao, S. Robust physical modeling sound synthesis for nonlinear systems: Direct numerical simulation and the energy method. Under review, IEEE Signal Processing Magazine. Invited article., 2006. 82
- Bilbao, S. [Online] Plate sound synthesis examples at http://www.music.ed.ac.uk/Contacts/DrStefanBilbao_soundExamples.htm, Retrieved June 29th, 2006. 82
- Bilbao, S. and Smith III, J. O. Energy conserving finite difference schemes for nonlinear strings. *Acta Acustica united with Acustica*, 91, pp. 299–311, 2005. 82
- Biles, J. A. GenJam, [Online] <http://www.it.rit.edu/~jab/> including numerous publications on GenJam and evolutionary music. 29
- Birnbaum, D., Fiebrink, R., Malloch, J., and Wanderley, M. Towards a dimension space for musical devices. In *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, pp. 192–195, 2005. 52
- Bischoff, J., Gold, R., and Horton, J. Music for an interactive network of microcomputers. *Computer Music J.*, 2(3), pp. 24–29, 1978. 314
- Blauert, J. *Spatial hearing: The psychophysics of human sound localization* (Revised ed.). Cambridge, Massachusetts, USA: The MIT Press, 1997. 232
- Blesser, B. An interdisciplinary synthesis of reverberation viewpoints. *J. Audio Eng. Soc.*, 49(10), pp. 867–903, 2001. 113
- Bloit, J. Analyse temps réel de la voix pour le contrôle de synthèse audio. Master-2/SAR ATIAM, UPMC (Paris 6), Paris, 2005. 282
- Blu, T. A new design algorithm for two-band orthonormal rational filter banks and orthonormal rational wavelets. *IEEE Trans. Sig. Proc.*, 46(6), pp. 1494–1503, 1998. 191
- Boden, M. Modeling human sound source localization and the cocktail-party-effect. *Acta Acustica*, 1, 1, pp. 43–55, 1993. 232
- Boersma, P. and Weenink, D. Praat: A system for doing phonetics by computer. *Glot International*, 5(9/10), pp. 341–345, 2001. 160
- Bogaards, N. and Röbel, A. An interface for analysis-driven sound processing. In *119th Conv. Audio Eng. Soc.*, New York, USA, 2004. 330
- Bohn, D. and Pennington, T. Constant-*Q* graphic equalizers. RaneNote 101, Rane Corporation. [Online] <http://www.rane.com/pdf/note101.pdf>, 1987. 40

- Bökesson, S. The Cosmos model, an event generation system for synthesizing emergent sonic structures. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 259–262, 2005. 202
- Boll, S. F. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 27(2), pp. 113–120, 1979. 232
- Bonada, J. and Loscos, A. Sample-based singing voice synthesizer by spectral concatenation. In *Proc. Stockholm Music Acoustics Conf. (SMAC'03)*, Stockholm, Sweden, pp. 439–442, 2003. 44
- Bongers, B. *Trends in Gestural Control in Music*, chapter Physical interaction in the electronic arts: Interaction theory and interfacing techniques for real-time performance. France: Ircam, 2000. 52
- Borgonovo, A. and Haus, G. Musical sound synthesis by means of two-variable functions: Experimental criteria and results. In *Proc. Int. Comp. Music Conf. (ICMC'84)*, Paris, France, ICMA, pp. 35–42, 1984. 314
- Bosi, M., Brandenburg, K., Quackenbush, S., Fielder, L., Akagiri, K., Fuchs, H., Dietz, M., Herre, J., Davidson, G., and Oikawa, Y. Iso/iec mpeg-2 advanced audio coding. In *101st Conv. Audio Eng. Soc.*, Los Angeles, USA, pp. 789–814, 1996. 270
- Bou-Ghazale, S. E. and Hansen, J. H. L. A comparative study of traditional and newly proposed features for recognition of speech under stress. *IEEE Trans. Speech and Audio Proc.*, 8(4), pp. 429–442, 2000. 160
- Bouéri, M. and Kyriakakis, C. Audio signal decorrelation based on a critical band approach. In *117th Conv. Audio Eng. Soc.*, San Francisco, USA. Preprint 6291, 2004. 296
- Boulanger, R. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming*. Cambridge, Massachusetts, USA: MIT Press, 2000. 66, 82, 330
- Bowcott, P. High level control of granular synthesis using concepts of inheritance and social interaction. In *Proc. Int. Comp. Music Conf. (ICMC'90)*, Glasgow, Scotland, pp. 50–52, 1990. 202
- Braasch, J. A loudspeaker-based 3d sound projection using virtual microphone control (ViMiC). In *118th Conv. Audio Eng. Soc.*, Barcelona, Spain, pp. 968–979, 2005. 274
- Brandt, E. Hard sync without aliasing. In *Proc. Int. Comp. Music Conf. (ICMC'01)*, Havana, Cuba, pp. 365–368, 2001. 172
- Bregman, A. *Auditory Scene Analysis: The Perceptual Organization of Sound* (Second ed.). Cambridge: MIT Press, 1999. 202, 324
- Bresson, J. and Agon, C. SDIF sound description data representation and manipulation in computer assisted composition. In *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, pp. 520–527, 2004. 330
- Bresson, J., Stroppa, M., and Agon, C. Symbolic control of sound synthesis in computer assisted composition. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 303–306, 2005. 330
- Briand, M., Virette, D., and Martin, N. Parametric representation of multichannel audio based on Principal Component Analysis. In *120th Conv. Audio Eng. Soc.*, Paris, France. Preprint 6813, 2006. 296
- Bricchi, M. and Fontana, F. A process for noise reduction, particularly for audio systems, device and computer program product therefore. EU Patent EP1271772, 2003. 302
- Brown, J. Calculation of a constant Q spectral transform. *J. Acoust. Soc. Am.*, 89(1), pp. 425–434, 1991. 131
- Bunch, J. R. Stability of methods for solving Toeplitz systems of equations. *SIAM J. Sci. Stat. Comput.*, 6, pp. 349–364, 1985. 308
- Buzan, T. and Buzan, B. *Mind Map Book*. Plume, 1996. 113
- Cadoz, C., Luciani, A., and Florens, J.-L. CORDIS-ANIMA: a modeling and simulation system for sound and image synthesis – The General formalism. *Computer Music J.*, 17(1), pp. 19–29, 1993. 82
- Cafagna, V. and Caterina, G. On a theorem of B. Friedman, in preparation. 191
- Cafagna, V., Caterina, G., and Vicinanza, D. Strategies of spectral approximation by boolean functions, in preparation. 191
- Cafagna, V. and Vicinanza, D. Synthesis by mathematical models: elliptic functions and lacunary series. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 339–344, 2004. 191
- Cage, J. *Silence: Lectures and Writings*. Middletown: Wesleyan University Press, 1961. 52
- Cage, J. and Charles, D. *For The Birds: John Cage in Conversation with Daniel Charles*. Marion Boyers, 1981. 82
- Cakewalk. Sonitus fx:Compressor. [Online] <http://www.cakewalk.com/Products/Sonitus/sonitus.asp>, Retrieved June 29th, 2006. 66
- Campbell, M. and Greated, C. *The Musician's Guide to Acoustics*. New York, NY, USA: Schirmer Books, 1988. 120
- Campos, G. R. and Howard, D. M. On the computational efficiency of different waveguide mesh topologies for room acoustic simulation. *IEEE Trans. Speech and Audio Proc.*, 13(5), pp. 1063 – 1072, 2005. 164
- Camurri, A., De Poli, G., Friberg, A., Leman, M., and Volpe, G. The MEGA project: analysis and synthesis of multisensory expressive gesture in performing art applications. *J. New Music Research*, 34(1), pp. 5–21, 2005. 36
- Camurri, A., Poli, G. D., and Rocchesso, D. A taxonomy for sound and music computing. *Computer Music J.*, 19(2), pp. 4–5, 1995. 113
- Canazza, S., De Poli, G., Drioli, C., Roda, A., and Vidolin, A. Modeling and control of expressiveness in music performance. *Proc. IEEE*, 92(4), pp. 286–701, 2004. 44
- Cano, P. Fundamental frequency estimation in the SMS analysis. In *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, pp. 99–102, 1998. 44
- Cano, P., Loscos, A., Bonada, J., de Boer, M., and Serra, X. Voice morphing system for impersonating in karaoke applications. In *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, pp. 109–12, 2000. 44
- Cariou, B. Design of an alternative controller from an industrial design perspective. In *Proc. Int. Comp. Music Conf. (ICMC'92)*, San Francisco, USA, pp. 366–367, 1992. 52
- Cariou, B. The aXiO midi controller. In *Proc. Int. Comp. Music Conf. (ICMC'94)*, Århus, Denmark, pp. 163–166, 1994. 52
- Carvalho, A., Desarnaualds, V., and Loerincik, Y. Acoustic behavior of ceramic posts used in middle age worship spaces, a laboratory analysis. In *9th Int. Congress Sound and Vibration*. [Online] <http://paginas.fe.up.pt/~carvalho/icsv9.pdf>, 2002. 226
- Casey, M. Acoustic lexemes for organizing internet audio. *Contemporary Music Review*, 24(6), pp. 489–508, 2005. 282
- Casey, M. A. Separation of mixed audio sources by independent subspace analysis. In *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, pp. 154–161, 2000. 262
- Caterina, G. *Fuzzy Sets: some analytic, algebraic and geometric Aspects*. PhD thesis, University of Salerno, 2006. 191
- Cemgil, A. T. and Godsill, S. J. Probabilistic phase vocoder and its application to interpolation of missing values in audio signals. In *13th European Sig. Proc. Conf.*, Antalya, Turkey. [Online] <http://www.sigproc.eng.cam.ac.uk/~atc27/papers/cemgil-godsill-em-restore-eusipco.pdf>, 2005. 180
- Chafe, C. Adding vortex noise to wind instrument physical models. In *Proc. Int. Comp. Music Conf. (ICMC'95)*, Banff, Canada, ICMA, pp. 57–60, 1995. 314
- Chaigne, A. and Askenfelt, A. Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods. *J. Acoust. Soc. Am.*, 95(2), pp. 1112–8, 1994. 82
- Chamberlin, H. *Musical Applications of Microprocessors*, Second Edition. Hayden Books, 1985. 56
- Champagne, B., Bedard, S., and Stephenne, A. Performance of time-delay estimation in the presence of room reverberation. *IEEE Trans. Speech and Audio Proc.*, 4(2), pp. 148–152, 1996. 208

- Chaplin, G. B. B. and Smith, R. A. Method and apparatus for cancelling vibrations. U.S. Pat. 4490841, 1984. 308
- Chaplin, G. B. B. and Smith, R. A. Hearing aid and method for operating a hearing aid to suppress electromagnetic disturbance signals. U. S. Pat. 20010762875, 1999. 308
- Charpentier, F. J. Pitch detection using the short-term phase spectrum. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'86)*, Tokyo, Japan, pp. 113–116, 1986. 252
- Chasaide, A. N. and Gobl, C. *Handbook of Phonetic Sciences*, chapter Voice source variation, pp. 427–461. W. J. Hardcastle and J. Laver Eds. Blackwell, 1997. 160
- Chen, J., Benesty, J., and Huang, Y. A. Time delay estimation in room acoustic environments: an overview. *EURASIP J. on Applied Sig. Proc.*, 2006, Special issue on Advances in Multi-Microphone Speech Proc., article ID 26503, 19 pages, 2006. 208
- Chen, S., Donoho, D., and Saunders, M. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1), pp. 33–61, 1999. 270
- Childers, D. G. and Lee, C. K. Vocal quality factors: Analysis, synthesis, and perception. *J. Acoust. Soc. Am.*, 90(5), pp. 2394–2410, 1991. 160
- Choi, I. A chaotic oscillator as a musical signal generator in an interactive performance system. *J. New Music Research*, 26(1), pp. 17–47, 1997. 202
- Chuan, C.-H. and Chew, E. Fuzzy analysis in pitch class determination for polyphonic audio key finding. In *Proc. Int. Conf. Music Information Retrieval (ISMIR'05)*, London, UK, pp. 296–303, 2005. 131
- Claesson, I. and Nilsson, A. GSM TDMA frame rate internal active noise cancellation. *Inter. J. Acoust. and Vibration*, 8(3), pp. 159–166, 2003. 308
- Clark, R. J. *Investigation into digital audio equalizer systems and the effects of arithmetic and transform errors on performance*. PhD thesis, Univ. of Plymouth. [Online] <http://www.tech.plym.ac.uk/spmc/pdf/audio/RobClarkPhD.pdf>, 2001. 60
- Clarke, E. F. *Generative Processes in Music*, chapter Generative principles in music performance, pp. 1–26. Oxford: J. A. Sloboda (Ed), Clarendon Press, 1988. 52
- Cohen, A. and Kovacevic, J. Wavelets: The mathematical background. *Proc. IEEE*, 84(4), pp. 514–522, 1996. 220
- Cohen, L. The scale representation. *IEEE Trans. Sig. Proc.*, 41(12), pp. 3275–3291, 1993. 99
- Collins, N. A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions. In *118th Conv. Audio Eng. Soc.*, Barcelona, Spain, 2005. 137
- Conklin, H. A. Generation of partials due to nonlinear mixing in a stringed instrument. *J. Acoust. Soc. Am.*, 105(1), pp. 536–545, 1999. 120
- Cook, P. R. Toward the perfect audio morph? Singing voice synthesis and processing. In *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, pp. 223–230, 1998. 160
- Cook, P. R. Modeling BILL'S GAIT: analysis and parametric synthesis of walking sounds. In *Proc. Audio Eng. Soc. 22nd Conf. Virtual, Synthetic and Entertainment Audio*, pp. 73–78, 2002. 324
- Cook, P. R. *Real Sound Synthesis for Interactive Applications*. A. K. Peters, 2002. 324
- Corteel, E. *Caractérisation et extensions de la Wave Field Synthesis en conditions réelles d'écoute*. PhD thesis, Paris 6 University, Paris, France. [Online] <http://mediatheque.ircam.fr/articles/textes/Corteel04a/>, 2004. 214
- Corteel, E., Horbach, U., and Pellegrini, R. S. Multichannel inverse filtering of multiexciter distributed mode loudspeaker for Wave Field Synthesis. In *112th Conv. Audio Eng. Soc.*, Munich, Germany. Preprint Number 5611, 2002. 214
- Couturier, J.-M. *Utilisation avancée d'interfaces graphiques dans le contrôle gestuel de processus sonores*. PhD thesis, University of Aix-Marseille II, 2004. 36
- Couturier, J.-M., Kessous, L., and Arfib, D. Expressiveness and digital musical instrument design. *J. New Music Research, special issue on Expressive Gesture in Performing Arts and New Media*, 34(1), pp. 125–136, 2005. 36
- CRC. CRC-Seq. [Online] <http://www.crc.ca/fr/html/aas/home/products/products.html>, 2000. 245
- Cremer, M. and Derboven, C. A system for harmonic analysis of polyphonic music. In *AES 25th Int. Conf.*, London, UK, pp. 115–120, 2004. 131
- Curtis, D. V., Chapman, K. L., Adams, C. C., and Fender Musical Instruments. Simulated tone stack for electric guitar. United States Patent 6222110, 2001. 5
- Dalmont, J., Nederveen, C., Dubos, V., Ollivier, S., Méserette, V., and te Sligte, E. Experimental determination of the equivalent circuit of an open side hole: linear and non linear behaviour. *Acustica - Acta Acustica*, 88, pp. 567–575, 2002. 88
- Daniel, P. and Weber, R. Psychoacoustical roughness: Implementation of an optimized model. *Acustica*, 83, pp. 113–123, 1997. 180
- Dattorro, J. Effect design, part 1: Reverberator and other filters. *J. Audio Eng. Soc.*, 45(9), pp. 660–684, 1997. 152
- Dattorro, J. Effect design, part 2: Delay-line modulation and chorus. *J. Audio Eng. Soc.*, 45(10), pp. 764–788, 1997. 152
- Daubechies, I. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Reg. Conf. Series in Applied Math*. Philadelphia: SIAM, 1992. 106
- Daudet, L., Molla, S., and Torrésani, B. Towards a hybrid audio coder. In *Proc. Third Int. Conf. on Wavelet Analysis and Applications*, pp. 13–24, 2004. 270
- David, B., Richard, G., and Badeau, R. An EDS modelling tool for tracking and modifying musical signals. In *Stockholm Music Acoustics Conf.*, Stockholm, Sweden, pp. 715–718, 2003. 148
- Davies, M. and Daudet, L. Sparse audio representations using the MCLT. *Signal Processing*, 86(3), 2006. 270
- De Bonet, J. S. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proc. ACM SIGGRAPH*, New Orleans, USA, pp. 361–368, 1997. 36
- de Chevigné, A. and Henrich, N. Fundamental frequency estimation of musical sounds. *J. Acoust. Soc. Am.*, 111(4), pp. 1917–1930, 2002. 282
- De Sena, A. and Rocchesso, D. A fast Mellin transform with applications in DAFx. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 65–69, 2004. 99
- Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via EM algorithm. *J. Royal Stat. Soc.*, 39(1), pp. 1–38, 1977. 238
- Depalle, P., Garcia, G., and Rodet, X. Tracking of partials for additive sound synthesis using Hidden Markov Models. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'93)*, Minneapolis, USA, volume 1, pp. 225–228, 1993. 176, 245
- Depalle, P. and Poirier, G. A modular system for analysis, processing and synthesis of sound signals. In *Proc. Int. Comp. Music Conf. (ICMC'91)*, Montréal, Canada, pp. 161–164, 1991. 330
- Desainte-Catherine, M. and Marchand, S. High precision Fourier analysis of sounds using signal derivatives. *J. Audio Eng. Soc.*, 48, pp. 654–667, 2000. 258
- Desarnaulds, V., Loerincik, Y., and Carvalho, A. Efficiency of 13th Century acoustic ceramic pots in two Swiss churches. In *Noise-Con*. [Online] <http://paginas.fe.up.pt/~carvalho/nc01.pdf>, 2001. 226
- Desoer, C. A. and Kuh, E. S. *Basic Circuit Theory*. New York: McGraw-Hill, 1969. 94
- D'haes, W., van Dyck, D., and Rodet, X. PCA-based branch and bound search algorithms for computing k nearest neighbors. *Pattern Recognition Letters*, 24(9–10), pp. 1437–1451, 2003. 282
- Di Scipio, A. Composition by exploration of non-linear dynamic systems. In *Proc. Int. Comp. Music Conf. (ICMC'90)*, Glasgow, Scotland, pp. 324–327, 1990. 198

- Di Scipio, A. Synthesis of environmental sound textures by iterated nonlinear functions. In *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-99)*, Trondheim, Norway, 1999. 198
- Di Scipio, A. Iterated nonlinear engines as a sound-generating engine. *Leonardo Journal*, 34(3), pp. 249–254, 2001. 202
- DiBiase, J., Silverman, H., and Bransttein, M. *Robust Localization in Reverberant Rooms in Microphone Arrays: Signal Processing Techniques and Applications*, chapter 8, pp. 157–180. New York: Springer, 2001. 208
- Ding, Y. and Qian, X. Processing of musical tones using a combined quadratic polynomial-phase sinusoid and residual (QUASAR) signal model. *J. Audio Eng. Soc.*, 45(7/8), pp. 571–584, 1997. 186
- Dixon, S. Automatic extraction of tempo and beat from expressive performances. *J. New Music Research*, 30(1), pp. 39–58, 2001. 24
- Dixon, S. Learning to detect onsets of acoustic piano tones. In *Proc. MOSART Workshop on Current Dir. in Computer Music Res.*, IUA-UPF, Barcelona, pp. 147–151, 2001. 137
- Dixon, S. On the analysis of musical expression in audio signals. *Storage and Retrieval for Media Databases, SPIE-IS&T Electronic Imaging*, 5021, pp. 122–132, 2003. 24
- Dobson, R. and Fitch, J. Experiments with chaotic oscillators. In *Proc. Int. Comp. Music Conf. (ICMC'95)*, Banff, Canada, pp. 45–48, 1995. 198
- Dolby Laboratories Inc. San Francisco, CA. [Online] <http://www.dolby.com>. Retrieved June 29th, 2006. 302
- Dolson, M. The phase vocoder: A tutorial. *Computer Music J.*, 10(4), pp. 14–27, 1986. 36, 180, 278, 324, 330
- Doval, B. and Rodet, X. Estimation of fundamental frequency of musical sound signals. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'91)*, Toronto, Canada, pp. 3657–3660, 1991. 330
- Downie, J. 2005 MIREX contest results – audio onset detection. [Online] <http://www.music-ir.org/evaluation/mirex-results/audio-onset>, 2005. 137
- Downie, J., West, K., Ehmann, A., and Vincent, E. The 2005 music information retrieval evaluation exchange (MIREX 2005): Preliminary overview. In *Proc. Int. Conf. Music Information Retrieval (ISMIR'05)*, London, UK, pp. 320–323, 2005. 137
- Dressler, R. Dolby surround Prologic II decoder: Principles of operation. Technical report, Dolby Laboratories. [Online] http://www.dolby.com/assets/pdf/tech_library/209_Dolby_Surround_Pro_Logic_II_Decoder_Principles_of_Operation.pdf, 2000. 296
- Dubnov, S., Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., and Werman, M. Synthesizing sound textures through wavelet tree learning. *IEEE Computer Graphics and Applications*, 22(4), pp. 38–48, 2002. 324
- Dubnov, S., Tishby, N., and Cohen, D. Influence of frequency modulating jitter on higher order moments of sound residual with applications to synthesis and classification. In *Proc. Int. Comp. Music Conf. (ICMC'96)*, Hong Kong, pp. 378–385, 1996. 180
- Dubos, V., Kergomard, J., Khettabi, A., Dalmont, J., Keefe, D., and Nederveen, C. Theory of sound propagation in a duct with a branched tube using modal decomposition. *Acustica - Acta Acustica*, 85, pp. 153–169, 1999. 88
- Duncan Amps. Tone stack calculator. [Online] <http://www.duncanamps.com/tsc/>. Retrieved June 29th, 2006. 5
- Dunn, H. K. and Farnsworth, D. W. Exploration of pressure field around the human head during speech. *J. Acoust. Soc. Am.*, 10, pp. 184–199, 1939. 208
- Dutilleux, P., Poli, G. D., and Zölzer, U. Time-segment processing. In *DAFx – Digital Audio Effects*, pp. 201–236, 2002. 262
- Dutilleux, P. and Zölzer, U. Nonlinear processing. In U. Zölzer (Ed.), *DAFx – Digital Audio Effects* pp. 93–136. New York: J. Wiley & Sons, 2002. 302
- Duxbury, C., Bello, J., Davies, M., and Sandler, M. A combined phase and amplitude based approach to onset detection for audio segmentation. In *Proc. 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS-03)*, pp. 275–280, 2003. 137
- Duxbury, C., Davies, M., and Sandler, M. Separation of transient information in musical audio using multiresolution analysis techniques. In *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, 2001. 266
- Duxbury, C., Sandler, M., and Davies, M. A hybrid approach to musical note onset detection. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, pp. 33–38, 2002. 137
- Eckel, G., Rocchesso, D., and Strobl, G. Sound texture modeling: A survey. In *Proc. Sound and Music Computing (SMC) Int. Conf.*, Marseille, France. [Online] <http://assoc.orange.fr/gmem/smc06/papers/8-soundtexturemodelingSMC06.pdf>, 2006. 36
- Efros, A. and Leung, T. Texture synthesis by non-parametric sampling. In *Proc. IEEE Int. Conf. on Computer Vision, Kerkyra, Greece*, pp. 1033–1038, 1999. 36
- Ellis, D. P. W. A computer implementation of psychoacoustic grouping rules. In *Proc. 12th Int. Conf. Pattern Recognition*, pp. 108–112, 1994. 324
- Ellis, N., Bensoam, J., and Caussé, R. Modalys demonstration. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 101–102, 2005. 330
- Essl, G. Circle maps as a simple oscillators for complex behavior: I. Basics, submitted to *Int. Comp. Music Conf. (ICMC'06)*, New Orleans, USA, 2006. 198
- Etter, W. and Moschytz, G. S. Noise reduction by noise-adaptive spectral magnitude expansion. *J. Audio Eng. Soc.*, 42, pp. 341–349, 1994. 232
- Evangelista, G. Dyadic warped wavelets. *Adv. in Imaging and Electron Physics*, 117, pp. 73–171, 2001. 106
- Evangelista, G. and Cavaliere, S. Discrete frequency warped wavelets: Theory and applications. *IEEE Trans. Sig. Proc.*, 46(4), pp. 874–885. special issue on Theory and Applications of Filter Banks and Wavelets, 1998. 106
- Evangelista, G. and Cavaliere, S. Frequency warped filter banks and wavelet transform: A discrete-time approach via Laguerre expansions. *IEEE Trans. Sig. Proc.*, 46(10), pp. 2638–2650, 1998. 106
- Evangelista, G., Testa, I., and Cavaliere, S. The role of frequency warping in physical models of sound sources. In *Proc. Forum Acusticum*, Budapest, Hungary, pp. 715–720, 2005. 106
- Fabig, L. and Janer, J. Transforming singing voice expression – the sweetness effect. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 70–75, 2004. 160
- Faller, C. *Parametric Coding of Spatial Audio*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. Thesis No. 3062, [Online] <http://library.epfl.ch/theses/?nr=3062>, 2004. 232, 296
- Faller, C. and Merimaa, J. Source localization in complex listening situations: Selection of binaural cues based on interaural coherence. *J. Acoust. Soc. Am.*, 116(5), pp. 3075–3089, 2004. 238
- Fant, G., Liljencrants, J., and Lin, Q. A four-parameter model of glottal flow. Technical report, STL-QPSR, Vol. 4, pp. 1–13, 1985. 160
- Farina, A. Simultaneous measurement of impulse response and distortion with a swept sine technique. In *108th Conv. Audio Eng. Soc.*, Paris, France. Preprint No. 5093, 2000. 226
- Farina, A. and Ayalon, R. Recording concert hall acoustics for posterity. In *Proc. AES 24th Int. Conf. on Multichannel Audio*, Banff, Canada. [Online] <http://pcfarina.eng.unipr.it/Public/Papers/185-AES24.PDF>, 2003. 226
- Farina, A., Glasgal, R., Armelloni, E., and Torger, A. Ambiphonic principles for the recording and reproduction of surround sound for music. In *Proc. AES 19th Int. Conf. on Surround Sound, Techniques, Technology and Perception*, Schloss Elmau, Germany, pp. 26–46, 2001. 226
- Farina, A. and Tronchin, L. Advanced techniques for measuring and reproducing spatial sound properties of auditoria. In *Int. Symp. Room Acoustics: Design and Science*, Kyoto, Japan. [Online] <http://pcfarina.eng.unipr.it/Public/Papers/190-RADS2004.pdf>, 2004. 226
- Farner, S., Kronland-Martinet, R., Voinier, T., and Ystad, S. Timbre variations as an attribute of naturalness in clarinet play. In *Proc. 3rd Computer Music Modelling and Retrieval Conf. (CMMR05)*, Pisa, Italy, pp. 45–53, 2005. 24

- Fettweis, A. A simple design of maximally flat delay digital filters. *IEEE Trans. Audio and Electroacoustics*, 20(2), pp. 112–114, 1972. 76
- Fettweis, A. Wave digital filters: Theory and practice. *Proc. IEEE*, 74, pp. 270–327, 1986. 5
- ffitch, J. On the design of Csound 5. In *Proc. 3rd Linux Audio Conf.*, pp. 37–42, 2005. 278, 318
- ffitch, J., Bradford, R., and Dobson, R. Sliding is smoother than jumping. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 287–290, 2005. 278
- Filatriau, J.-J. and Arfib, D. Instrumental gestures and sonic texture. In *Proc. Sound and Music Computing (SMC) Int. Conf.*, Salerno, Italy. [Online] http://www.tele.ucl.ac.be/~jjfil/smcc05_Filatriau_Arfib.pdf, 2005. 36
- Fink, R. The story of ORCH5, or, the classical ghost in the hip-hop machine. *Popular Music*, 24(3), pp. 339–356, 2005. 52
- Fitz, K. and Haken, L. Sinusoidal modeling and manipulation using Lemur. *Computer Music J.*, 20(4), pp. 44–59, 1996. 245
- Flanagan, J. L. and Golden, R. M. Phase vocoder. *Bell System Tech. J.*, 45, pp. 1493–1509, 1966. 180, 252
- Fleiss, J. L., Levin, B., and Paik, M. C. *Statistical Methods for Rates and Proportions* (Third ed.). John Wiley & Sons, 2003. 152
- Fletcher, H., Blackham, E. D., and Stratton, R. Quality of piano tones. *J. Acoust. Soc. Am.*, 34(6), pp. 749–761, 1962. 76
- Fletcher, N. H. Analysis of the design and performance of harpsichords. *Acustica*, 37, pp. 139–147, 1977. 120
- Fletcher, N. H. and Rossing, T. D. *The Physics of Musical Instruments*. New York: Springer-Verlag, 1991. 18, 120
- Fontana, F. Computation of linear filter networks containing delay-free loops, with an application to the waveguide mesh. *IEEE Trans. Speech and Audio Proc.*, 11(6), pp. 774–782, 2003. 302
- Fontana, F., Avanzini, F., and Rocchesso, D. Computation of nonlinear filter networks containing delay-free paths. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 113–118, 2004. 302
- Fontana, F. and Bricchi, M. Process for noise reduction, particularly for audio systems, device and computer program product therefore. US Patent US2003004591, 2003. 302
- Fontana, F. and Karjalainen, M. A digital bandpass/bandstop complementary equalization filter with independent tuning characteristics. *IEEE Sig. Proc. Letters*, 10(4), pp. 88–91, 2003. 302
- Fontana, F. and Rocchesso, D. Physical modeling of membranes for percussive instruments. *Acustica United with Acta Acustica*, 84, pp. 529–542, 1998. 164
- Fontana, F., Rocchesso, D., and Apollonio, E. Using the waveguide mesh in modelling 3D resonators. In *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-00)*, Verona, Italy, pp. 229–232, 2000. 164
- Freed, A., Rodet, X., and Depalle, P. Synthesis and control of hundreds of sinusoidal partials on a desktop computer without custom hardware. In *Proc. ICSPAT*, pp. 98–101, 1992. 186
- Freed, A., Rodet, X., and Depalle, P. Performance, synthesis and control of additive synthesis on a desktop computer using FFT⁻¹. In *Proc. Int. Comp. Music Conf. (ICMC'93)*, Tokyo, Japan, pp. 98–101, 1993. 245
- Friberg, A. *A Quantitative Rule System for Musical Performance*. PhD thesis, Department of Speech, Music and Hearing, Royal Institute of Technology, Stockholm, 1995. 24
- Friedman, B. Fourier coefficients of bounded functions. *Bull. Amer. Math. Soc.*, 47, pp. 84–92, 1941. 191
- Frossard, P., Vanderghenst, P., Ventura i Figueras, R., and Kunt, M. A posteriori quantization of progressive matching pursuit streams. *IEEE Trans. Sig. Proc.*, 52(2), pp. 525–535, 2004. 270
- Fujishima, T. Realtime chord recognition of musical sound: a system using common lisp music. In *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, pp. 464–467, 1999. 131
- Gabor, D. Theory of communications. *J. Inst. Elect. Eng.*, 93(III), pp. 429–457, 1946. 262
- Gabor, D. Acoustical quanta and the theory of hearing. *Nature*, 159(4044), pp. 591–594, 1947. 262
- Gabor, D. *Lectures on communication theory*, Technical Report 238, Research Laboratory of Electronics, Cambridge, Massachusetts, Massachusetts Institute of Technology. 238, 1952. 262
- Galemba, A., Askenfelt, A., Cuddy, L. L., and Russo, F. A. Perceptual relevance of inharmonicity and spectral envelope in the piano bass range. *Acta Acustica united with Acustica*, 90(3), pp. 528–536, 2004. 76
- Garcia, G. and Pampin, J. Data compression of sinusoidal modeling parameters based on psychoacoustic masking. In *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, pp. 40–43, 1999. 245
- Garnier, M., Dubois, D., Poitevineau, J., Henrich, N., and Castellengo, M. Perception et description verbale de la qualité vocale dans le chant lyrique : une approche cognitive. In *Journées d'Étude sur la Parole (JEP04)*, Fes, Maroc, 2004. 126
- Gäumann, T. The pretransient of the harpsichord sound. I. vertical movement of the plectrum. In *Proc. Stockholm Music Acoustics Conf. (SMAC'03)* Stockholm, Sweden, pp. 163–166, 2003. 120
- Gaver, W. W. What in the world do we hear? an ecological approach to auditory event perception. *Ecological Psychology*, 5(1), pp. 1–29, 1993. 113
- Geronimus, J. On the trigonometric moment problem. *Ann. of Math.*, 47(4), pp. 742–761, 1946. 191
- Ghizzetti, A. Sui coefficienti di Fourier di una funzione limitata, compresa fra limiti assegnati. *Ann. Scuola Norm. Sup. Pisa*, 4, pp. 131–156, 1950. 191
- Girin, L., Marchand, S., di Martino, J., Röbel, A., and Peeters, G. Comparing the order of a polynomial phase model for the synthesis of quasi-harmonic audio signals. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, pp. 193–196, 2003. 186, 245
- Glasberg, B. R. and Moore, B. C. J. Derivation of auditory filter shapes from notched-noise data. *Hearing Research*, 47, pp. 103–138, 1990. 296
- Glazier, J. A. and Libchaber, A. Quasi-periodicity and dynamical systems: An experimentalist's view. *IEEE Trans. Circuits and Systems*, 35(7), pp. 790–809, 1988. 198
- Godsill, S. J. and Rayner, P. J. W. *Digital audio restoration – A Statistical model-based approach*. Springer-Verlag, 1998. 308
- Goetze, T. caps, the C Audio Plugin Suite. [Online] <http://quitte.de/dsp/caps.html>, Retrieved June 29th, 2006. 5
- Gogins, M. Iterated functions systems music. *Computer Music J.*, 15(1), pp. 40–48, 1991. 198
- Golden, R. M. Digital filter synthesis by sampled-data transformation. *IEEE Trans. Audio and Electroacoustics*, 16(3), pp. 321–329, 1968. 40
- Gomez, E. Melodic description of audio signals for music content processing. Technical report, Pompeu Fabra University, Barcelona. [Online] <http://www.iua.upf.es/mtg/publications/Phd-2002-Emilia-Gomez.pdf>, 2002. 24
- Gomez, E. Tonal description of polyphonic audio for music content processing. *INFORMS J. on Computing. Special cluster on music computing*, 18(3), to appear in 2006. 131
- Goodwin, M. Matching pursuit with damped sinusoids. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'97)*, Munich, Germany, volume 3, pp. 2037–2040, 1997. 270
- Gordon, J. W. and Smith, J. O. A sine generation algorithm for VLSI applications. In *Proc. Int. Comp. Music Conf. (ICMC'85)*, Burnaby, Canada, pp. 165–168, 1985. 186

- Gordon, N., Salmond, D., and Smith, A. F. M. Novel approach to nonlinear and non-Gaussian Bayesian state estimation. *Proc. IEE-F*, 140, pp. 107–113, 1993. 208
- Goto, M. An audio-based real-time beat tracking system for music with or without drum sounds. *J. New Music Research*, 30(2), pp. 159–171, 2001. 266
- Goto, M. A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43(4), pp. 311–329, 2004. 252
- Goto, M., Hashiguchi, H., Nishimura, T., and Oka, R. RWC music database: Popular, classical, and jazz music databases. In *Proc. Int. Conf. Music Information Retrieval (ISMIR'02)*, Paris, France, pp. 287–288, 2002. 176
- Goudeseune, C. Interpolated mappings for musical instruments. *Organised Sound*, 7(2), pp. 85–96, 2002. 48
- Gouyon, F., Klapuri, A., Dixon, S., Alonso, M., Tzanetakis, G., and Uhle, C. An experimental comparison of audio tempo induction algorithms. *IEEE Trans. Speech and Audio Proc.*, 14(5). to appear, 2006. 137
- Grey, J. M. Multidimensional perceptual scaling of musical timbres. *J. Acoust. Soc. Am.*, 61(5), pp. 1270–1277, 1977. 126
- Grey, J. M. and Gordon, J. Perceptual effects of spectral modification on musical timbres. *J. Acoust. Soc. Am.*, 63(5), pp. 1493–1500, 1978. 126
- Griffin, D. and Lim, J. A new model-based speech analysis/synthesis system. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'85)*, Tampa, USA, pp. 513–516, 1985. 36
- Grinsted, A., Moore, J. C., and Jevrejeva, S. Application of the cross wavelet transform and wavelet coherence to geophysical time series. *Nonlinear Processes in Geophysics*, 11, pp. 561–566, 2004. 220
- Grinsted, A., Moore, J. C., and Jevrejeva, S. Cross wavelet and wavelet coherence MATLAB toolbox. [Online] <http://www.pol.ac.uk/home/research/waveletcoherence/>, 2006. 220
- GSM Standard (GSM 05.01 version 7.0.0 Release 1998). Digital cellular telecommunications system (phase 2+), Physical layer on the radio path, 1998. 308
- Guastavino, C., Katz, B. F., Polack, J.-D., Levitin, D. J., and Dubois, D. Ecological validity of soundscape reproduction. *Acta Acustica united with Acustica*, 91(2), pp. 333–341, 2005. 113
- Guillemain, P. A digital synthesis model of double-reed wind instruments. *EURASIP J. on Applied Sig. Proc.*, 4(7), pp. 990–1000, 2004. 94
- Guillemain, P., Helland, R. T., Kronland-Martinet, R., and Ystad, S. The clarinet timbre as an attribute of expressiveness. In *Proc. 2nd Computer Music Modelling and Retrieval Conf. (CMMR04)*, Esbjerg, Denmark, pp. 246–259, 2004. 24
- Guillemain, P., Kergomard, J., and Voinier, T. Real-time synthesis of clarinet-like instruments using digital impedance models. *J. Acoust. Soc. Am.*, 118(1), pp. 483–494, 2005. 88, 94
- Guillemain, P. and Kronland-Martinet, R. Characterization of acoustic signals through continuous linear time-frequency representations. *Proc. IEEE*, 84(4), pp. 561–585, 1996. 220
- Guinness World Records. Longest lasting echo. [Online] http://www.guinnessworldrecords.com/content_pages/record.asp?recordid=47025&Reg=1, 2004, 2005. 226
- Haddad, K. OpenMusic OM2CSound. *Ircam Software Documentation*, 1999. 330
- Haddad, K. Timesculpt in OpenMusic. In C. Agon, G. Assayag, & J. Bresson (Eds.), *The OM Composer's Book* pp. 45–62. Ircam – Delatour, 2006. 330
- Hainsworth, S. W. and Macleod, M. D. On sinusoidal parameter estimation. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, pp. 151–156, 2003. 245
- Hajda, J. M., Kendall, R. A., Carterette, E. C., and Harshberger, M. L. Methodological issues in timbre research. In Deliège, I. and Sloboda, J. E. (Eds.), *Perception and Cognition of Music*, Psychology Press, pp. 253–286, 1997. 126
- Hall, D. E. String excitation: piano, harpsichord and clavichord. In *Proc. Stockholm Music Acoustics Conf. (SMAC'93)* Stockholm, Sweden, pp. 309–314, 1993. 120
- Hamanaka, M., Goto, M., Asoh, H., and Otsu, N. A learning-based quantization: Unsupervised estimation of the model parameters. In *Proc. Int. Comp. Music Conf. (ICMC'03)*, Singapore, pp. 369–372, 2003. 29
- Hamanaka, M., Hirata, K., and Tojo, S. Automatic generation of metrical structure based on GTTM. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 53–56, 2005. 29
- Hamman, M. Mapping complex systems using granular synthesis. In *Proc. Int. Comp. Music Conf. (ICMC'91)*, Montréal, Canada, pp. 475–478, 1991. 202
- Hamman, M. From symbol to semiotic: Representation, signification, and the composition of music interaction. *J. New Music Research*, 28, pp. 90–104, 1999. 52
- Hanappe, P. and Assayag, G. Intégration des représentations temps/fréquence et des représentations musicales symboliques. In M. Chemillier & F. Pachet (Eds.), *Recherches et applications en informatique musicale* pp. 199–207. Hermès, 1998. 330
- Hand, L. N. and Finch, J. D. *Analytical Mechanics*. Cambridge, UK: Cambridge University Press, 1998. 198
- Hanson, R. J., Anderson, J. M., and Macomber, H. K. Measurements of nonlinear effects in a driven vibrating wire. *J. Acoust. Soc. Am.*, 96(3), pp. 1549–1556, 1994. 168
- Hargreaves, D., Miell, D., and MacDonald, R. What do we mean by musical communication, and why it is important?, intro. of ‘musical communication (part 1)’, session, ICMPC CD-ROM. In *Proc. Int. Conf. on Music Perception and Cognition*, 2004. 113
- Härmä, A. Implementation of frequency-warped recursive filters. *EURASIP Signal Processing*, 80(3), pp. 543–548, 2000. 302
- Härmä, A., Karjalainen, M., Savioja, L., Välimäki, V., Laine, U. K., and Huopaniemi, J. Frequency-warped signal processing for audio applications. *J. Audio Eng. Soc.*, 48(11), pp. 1011–1031, 2000. 302
- Harris, S. SC1 Compressor. [Online] <http://www.plugin.org.uk/ladspa-swh/docs/ladspa-swh.html>, Retrieved June 29th, 2006. 66
- Harris, S. SC4 Compressor. [Online] <http://www.plugin.org.uk/ladspa-swh/docs/ladspa-swh.html>, Retrieved June 29th, 2006. 66
- Harte, C. and Sandler, M. Automatic chord identification using a quantised chromagram. In *AES 118th Convention*, Barcelona, Spain. Preprint 6412, 2005. 131
- Hartmann, W. M. Flanging and phasers. *J. Audio Eng. Soc.*, 26(6), pp. 439–443, 1978. 152
- Hartmann, W. M. *Signals, Sound, and Sensation*. Woodbury, NY: Modern Acoustics and Signal Processing, AIP Press, 1997. 144, 152
- Hasler, M. *Phénomènes non linéaires*. École Polytechnique Fédérale de Lausanne, 1999. 12
- Haykin, S. *Adaptive Filter Theory* (Fourth ed.). Englewood Cliffs, N. J.: Prentice Hall, 2001. 180, 266
- Heet, G. String instrument vibration initiator and sustainer. U.S. Pat. 4,075,921, 1978. 168
- Hélie, T. and Hasler, M. Volterra series for solving weakly non-linear partial differential equations: application to a dissipative Burgers’ equation. *Int. Journal of Control*, 77(12), pp. 1071–1082, 2004. 12
- Hiller, L. and Ruiz, P. Synthesizing musical sounds by solving the wave equation for vibrating objects. *J. Audio Eng. Soc.*, 19, pp. 462–72, 542–51, 1971. 82

- Hirokazu, K., Takuya, N., and Shigeki, S. Separation of harmonic structures based on tied gaussian mixture model and information criterion for concurrent sounds. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)*, Montreal, Canada, pp. 297–300, 2004. 238
- Holters, M. and Zölzer, U. Parametric recursive higher-order shelving filters. In *120th Conv. Audio Eng. Soc.*, Paris, France, Paris. Paper 6722, 2006. 40
- Huang, P., Serafin, S., and Smith, J. O. A waveguide mesh model for high-frequency violin body resonances. In *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, pp. 86–89, 2000. 164
- Hulsebos, E., de Vries, D., and Bourdillat, E. Improved microphone array configurations for auralization of sound fields by Wave-Field Synthesis. *J. Audio Eng. Soc.*, 50(10), pp. 779–790, 2002. 226
- Hunt, A. and Wanderley, M. M. Mapping performer parameters to synthesis engines. *Organised Sound*, 7(2), pp. 97–108, 2002. 48
- Huopaniemi, J., Savioja, L., and Karjalainen, M. Modeling of reflections and air absorption in acoustical spaces — a digital filter design approach. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, 1997. 164
- Huovilainen, A. Non-linear digital implementation of the Moog ladder filter. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 61–64, 2004. 12
- Hyvärinen, A., Karhunen, J., and Oja, E. *Independent Component Analysis*. Wiley, 2001. 262
- INRIA. Scilab. [Online] <http://www.scilab.org/>, Retrieved June 29th, 2006. 66
- Ircam. *AudioSculpt User's Manual* (Second ed.). Paris: Ircam, 1996. 245
- Ircam. SDIF-Edit: Visualization of SDIF sound description files. [Online] <http://recherche.ircam.fr/equipes/repmus/bresson/sdifedit/sdifedit.html>, 2001. 330
- Ircam. Audiosculpt. [Online] <http://forumnet.ircam.fr/>, Retrieved June 29th, 2006. 314
- Ircam. Sound Description Interchange Format. [Online] <http://www.ircam.fr/sdif/>, Retrieved June 29th, 2006. 330
- Irwan, R. and Aarts, R. M. Two-to-five channel sound processing. *J. Audio Eng. Soc.*, 50(11), pp. 914–926, 2002. 296
- Isidori, A. *Nonlinear control systems* (3rd edition ed.). Springer Verlag, 1995. 12
- ISO3382. Acoustics – measurement of reverberation time of rooms with reference to other acoustical parameters. Technical report, ISO, 1997. 226
- Izmirli, O. Template based key finding from audio. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 211–214, 2005. 131
- Jacob, R. J. K., Sibert, L. E., McFarlane, D. C., and Mullen Jr., M. P. Integrality and separability of input devices. *ACM Trans. Human Computer Interaction*, 1(1), pp. 3–26, 1994. 52
- Jaffe, D. A. and Smith III, J. O. Extensions of the Karplus-Strong plucked-string algorithm. *Computer Music J.*, 7(2), pp. 56–69, 1983. 76, 314
- Jahn, R. G., Devereux, P., and Ibison, M. Acoustical resonances of assorted ancient structures. *J. Acoust. Soc. Am.*, 99(2), pp. 649–658, 1996. 226
- Janer, J. Voice-controlled plucked bass guitar through two synthesis techniques. In *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, pp. 132–135, 2005. 44
- Järveläinen, H. and Karjalainen, M. Importance of inharmonicity in the acoustic guitar. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 363–366, 2005. 76
- Järveläinen, H., Välimäki, V., and Karjalainen, M. Audibility of the timbral effects of inharmonicity in stringed instrument tones. *Acoust. Research Letters Online*, 2(3), pp. 79–84, 2001. 76
- Jayant, N. S. and Noll, P. *Digital Coding of Waveforms – Principles and Applications to Speech and Video*. New Jersey: Prentice-Hall, 1984. 262
- Jehan, T. and Schoner, B. An audio-driven, spectral analysis-based, perceptually meaningful timbre synthesizer. In *110th Conv. Audio Eng. Soc.*, Amsterdam, the Netherlands, Amsterdam, Netherland, 2001. 44
- Jensen, K. *Timbre Models of Musical Sounds*. PhD thesis, Department of Computer Science, University of Copenhagen, 1999. 24
- Jing, Z. A new method for digital all-pass filter design. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 34(11), pp. 1557–1564, 1987. 18
- Johnson, N. L., Kotz, S., and Balakrishnan, N. *Continuous Univariate Distributions* (2nd. ed.). New York: John Wiley & Sons, Inc, 1994. 148
- Jordà, S. *Digital lutherie: Crafting musical computers for new musics' performance and improvisation*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2005. 36, 52
- Jullien, J.-P., Kahle, E., Marin, M., Warusfel, O., Bloch, G., and Jot, J.-M. Spatializer: a perceptual approach. In *96th Conv. Audio Eng. Soc.*, Amsterdam, the Netherlands, # 3465, pp. 1–13, 1993. 113
- Kabal, P. Ill-Conditioning and bandwidth expansion in linear prediction of speech. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'03)*, Hong Kong, China, pp. 824–827, 2003. 160
- Kammeyer, K.-D. and Kroschel, K. *Digitale Signalverarbeitung. Filterung und Spektralanalyse mit MATLAB-Übungen* (5 ed.). B. G. Teubner, 2002. 60
- Karjalainen, M. and Erkut, C. Digital waveguides versus finite difference structures: Equivalence and mixed modeling. *EURASIP Journal on Applied Signal Processing*, 7, pp. 978–989, 2004. 164
- Karplus, K. and Strong, A. Digital synthesis of plucked string and drum timbres. *Computer Music J.*, 7(2), pp. 43–45, 1983. 314
- Kay, S. M. *Fundamentals of Statistical Signal Processing – Estimation Theory*. Signal Processing Series. Prentice Hall, 1993. 144
- Keefe, D. H. Acoustical wave propagation in cylindrical ducts: Transmission line parameter approximations for isothermal and nonisothermal boundary conditions. *J. Acoust. Soc. Am.*, 75(1), pp. 58–62, 1984. 94
- Keiler, F. and Marchand, S. Survey on extraction of sinusoids in stationary sounds. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, pp. 51–58, 2002. 144, 176, 245, 252, 258
- Kemp, M. J. Analysis and Simulation of Analogue Dynamic Compressors and Limiters in the Digital Domain. In *Proc. 109th AES Convention*, Los Angeles. Preprint 5185, 2000. 66
- Klapuri, A. Sound onset detection by applying psychoacoustic knowledge. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, Phoenix, Arizona, 1999. 137
- Klingbeil, M. Software for spectral analysis, editing, and synthesis. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 107–110, 2005. 324
- Knapp, C. H. and Carter, G. C. The generalized correlation method for estimation of time delay. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 24, pp. 320–327, 1976. 208
- Kock, W. *Seeing Sound*. New York: Wiley-Interscience, 1971. 314
- Kottick, E. L. The acoustics of the harpsichord: Response curves and modes of vibration. *Galpin Soc. J.*, 38, pp. 55–77, 1985. 120
- Kottick, E. L., Marshall, K. D., and Hendrickson, T. J. The acoustics of the harpsichord. *Scientific American*, 264(2), pp. 94–99, 1991. 120
- Krimphoff, J., McAdams, S., and Winsberg, S. Caractérisation du timbre des sons complexes, II Analyses acoustiques et quantification psychophysique. *Journal de Physique IV, Colloque C5*, 4, 1994. 24
- Krstulovic, S. and Gribonval, R. GNU/GPL C++ Software, *the matching pursuit toolkit*. [Online] <http://gforge.inria.fr/projects/mptk/>, Retrieved June 29th, 2006. 270
- Krumhansl, C.-L. *Cognitive foundations of musical pitch*. New-York: Oxford University Press, 1999. 131

- Kuehnel, R. *Circuit Analysis of a Legendary Tube Amplifier: The Fender Bassman 5F6-A* (2nd ed.). Seattle: Pentode Press, 2005. 5
- Kvifte, T. and Jensenius, A. R. Towards a coherent terminology and model of instrument description and design. In *Int. Conf. New Interf. for Musical Expr. (NIME'06)*, Paris, France, pp. 220–225, 2006. 52
- Kwatra, V., Essa, I., Bobick, A., and Kwatra, N. Texture optimization for example-based synthesis. *ACM Trans. Graphics*, 24(3), pp. 795–802, 2005. 36
- La Burthe, A. and Peeters, G. Résumé sonore. In *Internship in the framework of the master ATIAM – final report IRCAM/INPG Grenoble*, Paris, France, 2002. 70
- Laakso, T. I., Välimäki, V., Karjalainen, M., and Laine, U. Crushing the delay – Tools for fractional delay filter design. Technical report, Helsinki University of Technology, Faculty of Electrical Engineering, Laboratory of Acoustics and Audio Signal Processing, Espoo, Finland, Report no. 35, 1994. 94
- Laakso, T. I., Välimäki, V., Karjalainen, M., and Laine, U. K. Splitting the unit delay: Tools for fractional delay filter design. *IEEE Sig. Proc. Magazine*, 13(1), pp. 30–60, 1996. 76, 82
- Laborie, A., Bruno, R., and Montoya, S. High spatial resolution multichannel recording. In *116th Conv. Audio Eng. Soc.*, Berlin, Germany. Poster Z7-3, 2004. 214
- Lagrange, M. *Sinusoidal Modeling of Polyphonic Sounds*. PhD thesis, University of Bordeaux 1, France, in French, 2004. 245
- Lagrange, M. and Marchand, S. Real-time additive synthesis of sound by taking advantage of psychoacoustics. In *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, pp. 5–9, 2001. 186
- Lagrange, M. and Marchand, S. Improving sinusoidal frequency estimation using a trigonometrical approach. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, pp. 59–64, 2005. 245
- Lagrange, M., Marchand, S., Raspaud, M., and Rault, J.-B. Enhanced partial tracking using linear prediction. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, pp. 141–146, 2003. 176
- Lagrange, M., Marchand, S., and Rault, J.-B. Sinusoidal parameter extraction and component selection in a non-stationary model. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, pp. 59–64, 2002. 258
- Lagrange, M., Marchand, S., and Rault, J.-B. Partial tracking based on future trajectories exploration. In *116th Conv. Audio Eng. Soc.*, Berlin, Germany. Preprint 6046 (10 pages), 2004. 245
- Lagrange, M., Marchand, S., and Rault, J.-B. Using linear prediction to enhance the tracking of partials. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)*, Montreal, Canada, volume 4, pp. iv/241 – iv/244, 2004. 245
- Lagrange, M., Marchand, S., and Rault, J.-B. Tracking partials for the sinusoidal modeling of polyphonic sounds. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'05)*, Philadelphia, USA, volume 3, pp. 229–232, 2005. 148, 245
- Laird, J. A. *The physical modelling of drums using digital waveguides*. PhD Thesis, University of Bristol, 2001. 164
- Lambert, J. D. *Numerical Methods for Ordinary Differential Systems*. New York: John Wiley & Sons, 1991. 302
- Lamnabhi-Lagarrigue, F. *Analyse des Systèmes Non Linéaires*. Editions Hermès. ISBN 2-86601-403-0, 1994. 12
- Landy, L. Reviewing the musicology of electroacoustic music: a plea for greater triangulation. *Org. Sound*, 4(1), pp. 61–70, 1999. 113
- Lane, J., Hoory, D., Martinez, E., and Wang, P. Modeling analog synthesis with DSPs. *Computer Music J.*, 21(4), pp. 23–41, 1997. 172
- Lang, M. Allpass filter design and applications. *IEEE Trans. Sig. Proc.*, 46(9), pp. 2505–2514, 1998. 18
- Lang, M. and Laakso, T. I. Simple and robust method for the design of allpass filters using least-squares phase error criterion. *IEEE Trans. Circuits and Systems—I: Fundamental Theory and Applications*, 41(1), pp. 40–48, 1994. 18
- Laroche, J. *Applications of Digital Signal Processing to Audio & Acoustics*, chapter Time and pitch scale modification of audio signals, pp. 279–309. M. Kahrs and K. Brandenburg (Eds.), Kluwer Academic Publishers, 1998. 113
- Laroche, J. and Dolson, M. New phase vocoder technique for pitch-shifting, harmonizing and other exotic effects. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, pp. 91–94, 1999. 36
- Laurson, M. and Kuuskankare, M. PWGL: A novel visual language based on common Lisp, CLOS and OpenGL. In *Proc. Int. Comp. Music Conf. (ICMC'02)*, Gothenburg, Sweden, pp. 142–145, 2002. 286
- Laurson, M. and Norilo, V. Recent developments in PWSynth. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, pp. 69–72, 2003. 286
- Laurson, M., Norilo, V., and Kuuskankare, M. PWGLSynth: A visual synthesis language for virtual instrument design and control. *Computer Music J.*, 29(3), pp. 29–41, 2005. 286
- Lauterborn, W. and Parlitz, U. Methods of chaos physics and their application to acoustics. *J. Acoust. Soc. Am.*, 84(6), pp. 1975–1993, 1988. 198
- Lazier, A. and Cook, P. MOSIEVIUS: Feature driven interactive audio mosaicing. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, pp. 312–317, 2003. 282
- Lazzarini, V. Extensions to the Csound language: from user-defined to plugin opcodes and beyond. In *Proc. of the 3rd Linux Audio Conf.*, pp. 13–20, 2005. 278
- Lazzarini, V. Scripting Csound 5. In *Proc. 4th Linux Audio Conf.*, pp. 50–55, 2005. 318
- Lazzarini, V., Timoney, J., and Lysagh, T. Time-stretching using the instantaneous frequency distribution and partial tracking. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 14–27, 2005. 278
- Le Brun, M. Digital waveshaping synthesis. *J. Audio Eng. Soc.*, 27(4), pp. 250–266, 1979. 198
- Lee, A., Kawahara, T., and Shikano, K. Julius — an open source real-time large vocabulary recognition engine. In *Proc. European Conf. on Speech Communication and Technology*, pp. 1691–1694, 2001. 44
- Lee, K. and Smith, J. O. Implementation of a highly diffusing 2-D digital waveguide mesh with a quadratic residue diffuser. In *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, pp. 309–315, 2004. 164
- Lehmann, E. A. and Williamson, R. C. Particle filtering design using importance sampling for acoustic source localisation and tracking in reverberant environments. *EURASIP J. on Applied Sig. Proc.*, 2006, Special issue on Advances in Multi-Microphone Speech Proc., article ID 17021, 9 pages, 2006. 208
- Leman, M. Symbolic and subsymbolic description of music. In G. Hause (Ed.), *Music Processing* pp. 119–164. Oxford University Press, 1993. 330
- Lemstrom, K. and Perttu, S. SEMEX – an efficient music retrieval prototype. In *Proc. Int. Symp. Music Information Retrieval (ISMIR'00)*, Plymouth, Massachusetts, USA. [Online] http://ismir2000.ismir.net/papers/lemstrom_paper.pdf, 2000. 29
- Leveau, P., Daudet, L., and Richard, G. Methodology and tools for the evaluation of automatic onset detection algorithms in music. In *Proc. Int. Conf. Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, pp. 72–75, 2004. 137

- Levine, H. and Schwinger, J. On the radiation of sound from an unflanged circular pipe. *Phys. Rev.*, 73(4), pp. 383–406, 1948. 94
- Levine, S. N., Verma, T. S., and Smith, J. O. Multiresolution sinusoidal modeling for wideband audio with modifications. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'98)*, Seattle, USA, volume 6, pp. 3585–3588, 1998. 245
- Levitin, D., McAdams, S., and Adams, R. L. Control parameters for musical instruments: a foundation for new mappings of gesture to sound. *Org. Sound*, 7(2), pp. 171–189, 2002. 52
- Li, T., Ogihara, M., and Li, Q. A comparative study on content-based music genre classification. In *Proc. 26th Int. ACM SIGIR conf. Research and Development in Information Retrieval*, pp. 282–289, 2003. 156
- Li, Y. and Driessen, P. F. An unsupervised adaptative filtering approach of 2-to-5 channel upmix. In *119th Conv. Audio Eng. Soc.*, New York, USA. Preprint 6611, 2005. 296
- LibAudioStream. [Online] <http://libAudioStream.sourceforge.net/>, Retrieved June 29th, 2006. 330
- Liénard, J.-S. and Benedetto, M.-G. D. Effect of vocal effort on spectral properties of vowels. *J. Acoust. Soc. Am.*, 106(1), pp. 411–422, 1999. 160
- Lin, H. and Godsill, S. J. Audio in the new millennium. In *IEEE Workshop on Audio and Acoustics*, Mohonk, NY State, pp. 1–4, 2005. 308
- Lindemann, E. Musical synthesizer capable of expressive phrasing. US Patent 6,316,710, 2001. 282
- Lippens, S., Martens, J. P., Mulder, T. D., and Tzanetakis, G. A comparison of human and automatic musical genre classification. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)*, Montreal, Canada, volume 4, pp. 233–236, 2004. 156
- Logan, B. Mel frequency cepstral coefficients for music modeling. In *Proc. Int. Symp. Music Information Retrieval (ISMIR'00)*, Plymouth, Massachusetts, USA, pp. 138–147, 2000. 156
- Lokki, T. *Physically-Based Auralization – Design, Implementation and Evaluation*. PhD thesis, Helsinki University of Technology, Espoo, Finland. ISBN 951-22-6157-X, 2002. 220
- Lokki, T. Auralization of simulated impulse responses in slow motion. In *118th Conv. Audio Eng. Soc.*, Barcelona, Spain. Paper no. 6500, 2005. 220
- Loscos, A. and Bonada, J. Emulating rough and growl voice in spectral domain. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 188–91, 1998. 160
- Loscos, A., Cano, P., and Bonada, J. Low-delay singing voice alignment to text. In *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, pp. 437–440, 1999. 44
- Lourdis, S. J. Decomposition of impulse responses using complex wavelets. *J. Audio Eng. Soc.*, 53(9), pp. 796–811, 2005. 220
- Makhoul, J. Linear prediction: A tutorial review. *Proc. IEEE*, 63, pp. 561–580, 1975. 160
- Maki-Patola, T., Laitinen, J., Kanerva, A., and Takala, T. Experiments with virtual reality instruments. In *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, pp. 11–16, 2005. 274
- Makino, S. and Kaneda, Y. Acoustic echo canceller algorithm based on the variation characteristics of a room impulse response. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, pp. 1133–1136, 1990. 266
- Mallat, S. *A Wavelet Tour of Signal Processing*. Cambridge, MA: Academic Press, 1997. 106, 262
- Mallat, S. and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Sig. Proc.*, 41(12), pp. 3397–3415, 1993. 270
- Malvar, H. A modulated complex lapped transform and its applications to audio processing. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'99)*, Phoenix, USA, volume 3, pp. 1421–1424, 1999. 270
- Manzolli, J., Damiani, F., Tatsch, P. J., and Maia, A. A non-linear sound synthesis method. In *Proc. 7th Brazilian Symp. on Computer Music*, Curitiba, 2000. 198
- Marchand, S. *Sound Models for Computer Music (Analysis, Transformation, and Synthesis of Musical Sound)*. PhD thesis, University of Bordeaux 1, France, 2000. 186
- Marchand, S. and Strandh, R. InSpect and ReSpect: Spectral modeling, analysis and real-time synthesis software tools for researchers and composers. In *Proc. Int. Comp. Music Conf. (ICMC'99)*, Beijing, China, pp. 341–344, 1999. 186, 245
- Markel, J. D. and Gray Jr., A. H. *Linear Prediction of Speech*. New York: Springer-Verlag Berlin Heidelberg, 1976. 56, 160
- Marui, A. and Martens, W. L. Perceptual and semantic scaling for user-centered control over distortion-based guitar effects. In *110th Conv. Audio Eng. Soc.*, Amsterdam, the Netherlands. Preprint 5387, 2001. 113
- Masri, P. *Computer Modelling of Sound for Transformation and Synthesis of Musical Signals*. PhD thesis, University of Bristol, UK, 1996. 137, 258
- Masri, P. and Bateman, A. Identification of nonstationary audio signals using the FFT, with application to analysis-based synthesis of sound. In *Proc. IEE Colloquium Audio Eng., Digest No. 1995/96*, pp. 11/1–11/6, 1995. 252
- Master, A. S. Nonstationary sinusoidal model frequency parameter estimation via fresnel integral analysis. Master's thesis, Stanford University, USA, 2002. 258
- Mathews, M. and Verplank, B. Scanned synthesis. In *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, pp. 368–371, 2000. 52
- Maxfield, C. *How Computers Do Math*. John Wiley & Sons, 2005. 290
- Maxim digital audio. mda dynamics. [Online] <http://www.mda-vst.com/>, Retrieved June 29th, 2006. 66
- McAulay, R. J. and Quatieri, T. F. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 34(4), pp. 744–754, 1986. 113, 144, 176, 186, 245, 258, 324, 330
- McAulay, R. J. and Quatieri, T. F. Shape invariant time-scale and pitch modification of speech. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 40(3), pp. 497–510, 1992. 245
- McCartney, J. Continued evolution of the Super-Collider real time environment. In *Proc. Int. Comp. Music Conf. (ICMC'98)*, Ann Arbor, USA, pp. 133–136, 1998. 286
- McGuinness, M. and Hong, Y. Arnold tongues in human cardiorespiratory systems. *Chaos*, 14(1), pp. 1–6, 2004. 198
- McIntyre, M. E., Schumacher, R. T., and Woodhouse, J. On the oscillations of musical instruments. *J. Acoust. Soc. Am.*, 74(5), pp. 1325–1345, 1983. 94
- Meine, N. and Purnhagen, H. Fast sinusoid synthesis for MPEG-4 HILN parametric audio decoding. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, pp. 239–244, 2002. 186, 245
- Melih, K. and Gonzalez, R. Source segmentation for structured audio. In *Proc. IEEE Int. Conf. Multimedia and Expo (II)*, pp. 811–814, 2000. 324
- Merimaa, J. and Pulkki, V. Spatial impulse response rendering 1: Analysis and synthesis. *J. Audio Eng. Soc.*, 53(12), 2005. 226
- Meron, Y. *High Quality Singing Synthesis using the Selection-based Synthesis Scheme*. PhD thesis, University of Tokyo, 1999. 44
- Métois, E. *Musical Sound Information: Musical Gesture and Embedding Synthesis*. PhD thesis, Massachusetts Institute of Technology, 1996. 44
- Miller, R. Equalization methods with true response. In *116th Conv. Audio Eng. Soc.*, Berlin, Germany. Paper 6088, 2004. 40
- Miner, N. E. and Caudell, T. P. Using wavelets to synthesize stochastic-based sounds for immersive virtual environments. In *Proc. 4th Int. Conf. on Auditory Display (ICAD'97)*, Palo Alto, USA, pp. 69–76, 1997. 324
- Mitra, S. K. *Digital Signal Processing. A computer-Based Approach*. New York: McGraw-Hill, 1998. 302
- Moffat, A., Neal, R. M., and Witten, I. H. Arithmetic coding revisited. *ACM Trans. Inf. Syst.*, 16(3), pp. 256–294, 1998. 270

Molino, J. Fait musical et sémiologie de la musique. <i>Musique en jeu</i> , 17, pp. 37–62, 1975.	113	Nuttall, A. H. Some windows with very good sidelobe behavior. <i>IEEE Trans. Acoust., Speech, and Signal Proc.</i> , ASSP-29(1), pp. 84–91, 1981.	252
Momeni, A. and Wessel, D. Characterizing and controlling musical material intuitively with geometric models. In <i>Int. Conf. New Interf. for Musical Expr. (NIME'03)</i> , Montréal, Canada, pp. 54–62, 2003.	48	O'Brien, J., Cook, P., and Essl, G. Synthesizing sounds from physically based motion. In <i>Proc. ACM SIGGRAPH</i> , Los Angeles, USA, pp. 529–536, 2001.	324
Moog, R. A. A voltage-controlled low-pass high-pass filter for audio signal processing. In <i>17th Conv. Audio Eng. Soc.</i> , New York, USA, pp. 1–12, 1965.	12	Ogata, K. <i>Modern Control Engineering</i> (4 ed.). Prentice Hall, 2001.	48
Moore, B. C. J. <i>Psychology of Hearing</i> . Academic Press, 1993.	296	Okazaki, M., Kunimoto, T., and Kobayashi, T. Multi-stage spectral subtraction for enhancement of audio signal. In <i>Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'04)</i> , Montreal, Canada, volume II, pp. 805–808, 2004.	144
Moore, B. C. J. and Glasberg, B. R. Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. <i>J. Acoust. Soc. Am.</i> , 74(3), pp. 750–753, 1983.	214	Oppenheim, A. V. and Schafer, R. W. <i>Digital Signal Processing</i> . Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.	18
Moore, B. C. J., Glasberg, B. R., and Baer, T. A model for the prediction of thresholds, loudness and partial loudness. <i>J. Audio Eng. Soc.</i> , 45, pp. 224–240, 1997.	120	Oppenheim, A. V. and Schafer, R. W. <i>Discrete-Time Signal Processing</i> . Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.	302
Moore, F. R. <i>Elements of computer music</i> . Englewood Cliffs, NJ: Prentice Hall, 1990.	113, 172, 318	Orfanidis, S. <i>Introduction to Signal Processing</i> . Prentice Hall Int. Editions, 1996.	113
Moorer, J. A. Signal processing aspects of computer music – a survey. <i>Computer Music J.</i> , 1(1), pp. 4–37, 1977.	186	Orfanidis, S. J. High-order digital parametric equalizer design. <i>J. Audio Eng. Soc.</i> , 53(11), pp. 1026–1046, 2005.	40
Moorer, J. A. The use of the phase vocoder in computer music applications. <i>J. Audio Eng. Soc.</i> , 26(1), pp. 42–45, 1978.	180	Orio, N., Schnell, N., and Wanderley, M. M. Evaluation of input devices for musical expression: Borrowing tools from HCI. <i>Computer Music J.</i> , 26(3), pp. 62–76, 2001.	52
Morrissey, J. J., Swicord, M., and Balzano, Q. Characterization of electromagnetic interference of medical devices in the hospital due to cell phones. <i>Health Phys.</i> , 82, pp. 45–51, 2002.	308	Orlarey, Y., Fober, D., and Letz, S. Syntactical and semantical aspects of Faust. <i>Soft Computing</i> , 8(9), pp. 623–632, 2004.	290, 330
Mouchtaris, A., Narayanan, S. S., and Kyriakakis, C. Virtual microphones for multichannel audio resynthesis. <i>EURASIP J. on Applied Sig. Proc.</i> , 2003, pp. 968–979, 2003.	274	O'Ruanaid, J. J. K. and Fitzgerald, W. J. <i>Numerical Bayesian methods applied to signal processing</i> . Springer-Verlag, 1996.	308
Moulines, E. and Charpentier, F. Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones. <i>Speech Communication</i> , 9(5/6), pp. 453–67, 1990.	113	Osborne, G. and Hoover, A. Sustainer for a musical instrument. U.S. Pat. 5,932,827, 1999.	168
Mourjopoulos, J. N., Kyriakis-Bitzaros, E. D., and Goutis, C. E. Theory and real-time implementation of time-varying digital audio filters. <i>J. Audio Eng. Soc.</i> , 38(7/8), pp. 523–536, 1990.	60	Paladin, A. and Rocchesso, D. A dispersive resonator in real-time on MARS workstation. In <i>Proc. Int. Comp. Music Conf. (ICMC'92)</i> , San Francisco, USA, pp. 146–149, 1992.	18
Murphy, D., Oxley, J., and Hildred, M. A Sense of Place. [Online] http://www.boothambar.org.uk/ , Retrieved June 29th, 2006.	226	Pallone, G. <i>Dilatation et Transposition sous contraintes perpectives des signaux audio: application au transfert cinema-video</i> . PhD thesis, Aix-Marseille II University, Marseille, 2003.	24
Murphy, D. T. Multi-channel impulse response measurement, analysis and rendering in archaeological acoustics. In <i>119th Conv. Audio Eng. Soc.</i> , New York, USA. Paper No. 6532, 2005.	226	Pampalk, E., Dixon, S., and Widmer, G. Exploring music collections by browsing different views. In <i>Proc. Int. Conf. Music Information Retrieval (ISMIR'03)</i> , Baltimore, Maryland, USA, pp. 201–208, 2003.	131
Nattiez, J.-J. <i>Fondements d'une sémiologie de la musique</i> . Paris: U. G. E., Coll. 10/18, 1975.	113	Papoulis, A. <i>Probability, Random Variables, and Stochastic Processes</i> (2nd ed.). McGraw-Hill, 1984.	144
Neter, J., Kutner, M. H., Nachtcheim, C. J., and Wasserman, W. <i>Applied Linear Statistical Models</i> (Fourth ed.). WCB/McGraw-Hill, 1996.	152	Pauws, S. Musical key extraction from audio. In <i>Proc. Int. Conf. Music Information Retrieval (ISMIR'04)</i> , Barcelona, Spain, pp. 96–99, 2004.	131
Nicol, R. <i>Restitution sonore spatialisée sur une zone étendue: Application à la téléprésence</i> . PhD thesis, Université du Maine, Le Mans, France. [Online] http://gyronymo.free.fr/audio3D/Guests/RozennNicol_PhD.html , 1999.	214	Peeters, G. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Technical report, Ircam – Centre Pompidou, 2004.	282
Nikolaidis, S. S., Mourjopoulos, J. N., and Goutis, C. E. A dedicated processor for time-varying digital audio filters. <i>IEEE Trans. Circuits and Systems—II: Analog and Digital Sig. Proc.</i> , 40(7), pp. 452–455, 1993.	60	Peeters, G. and Rodet, X. Sinusoidal characterization in terms of sinusoidal and non-sinusoidal components. In <i>Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)</i> , Barcelona, Spain, 1998.	148
Nordstrom, K. I., Driessen, P. F., and Rutledge, G. A. Influence of the LPC filter upon the perception of breathiness and vocal effort. In <i>IEEE Int. Symposium on Signal Processing and Information Technology (ISSPIT06)</i> , Vancouver, Canada, 2006.	160	Peeters, G. and Rodet, X. SINOLA: A new analysis/synthesis method using spectrum peak shape distortion, phase and reassigned spectrum. In <i>Proc. Int. Comp. Music Conf. (ICMC'99)</i> , Beijing, China, pp. 153–156, 1999.	258
Nordstrom, K. I., Rutledge, G. A., and Driessen, P. F. Using voice conversion as a paradigm for analyzing breathy singing voices. In <i>Pacific Rim Conf. on Communications, Computers and Sig. Proc. (PACRIM05)</i> , Victoria, Canada, pp. 428 – 431, 2005.	160	Peissig, J., Remmer ter Hasborg, J., Keiler, F., and Zelzer, U. Digital Emulation of Analog Companding Algorithms for FM Radio Transmission. In <i>Proc. Int. Conf. on Digital Audio Effects (DAFx-04)</i> , Naples, Italy, pp. 285–290, 2004.	66
Norman, D. A. <i>Some Observations on Mental Models</i> . D. Gentner and A. L. Stevens (eds.), Lawrence Erlbaum Associates, 1983.	48	Perlin, K. An image synthesizer. <i>Computer Graphics</i> , 19(5), pp. 287–296, 1985.	36
Perrot, D. and Gjerdingen, R. Scanning the dial: An exploration of factors in identification of musical style. In <i>Proc. Soc. Music Perception Cognition</i> , pp. 88, 1999.	156		

PhaseSpace Inc.	PhaseSpace motion capture.	[Online]	http://www.phasespace.com, Retrieved June 29th, 2006.	208	
Piché, J. and Burton, A.	Cecilia: A production interface to Csound.	<i>Computer Music J.</i> , 22(2), pp. 52–55, 1998.	318		245
Pienimaki, A. and Lemstrom, K.	Clustering symbolic music using paradigmatic and surface level analyses.	In <i>Proc. Int. Conf. Music Information Retrieval (ISMIR'04)</i> , Barcelona, Spain, pp. 175–178.	[Online] http://www.uia.upf.es/mtg/ismir2004/review/CRFILES/paper175-8b67ec9d48806099719fe50d8eb09f7e.pdf, 2004.	29	
Pierce, A.	<i>Acoustics</i> .	NY: McGraw-Hill, 1981.	88		
Plomp, R.	<i>The Intelligent Ear</i> (Second ed.).	London: Lawrence Erlbaum Associates, 2002.	126		
Poletti, M. A.	A unified theory of horizontal holographic sound systems.	<i>J. Audio Eng. Soc.</i> , 48(12), pp. 1155–1182, 2000.	226		
Poli, G. D., Picialli, A., Pop, S. T., and Roads, C.	<i>Musical Signal Processing</i> .	Eds. Swets & Zeitlinger, 1996.	113		
Polotti, P.	<i>Fractal additive synthesis</i> .	PhD thesis, EPFL, Lausanne, Switzerland, 2003.	106		
Polotti, P. and Evangelista, G.	Analysis and synthesis of pseudo-periodic 1/f-like noise by means of wavelets with applications to digital audio.	<i>EURASIP J. on Applied Sig. Proc.</i> , 2001(1), pp. 1–14, 2001.	106		
Polotti, P. and Evangelista, G.	Fractal additive synthesis by means of harmonic-band wavelets.	<i>Computer Music J.</i> , 25(3), pp. 22–37, 2001.	106		
Pompoli, R. and Prodi, N.	Guidelines for acoustical measurements inside historical opera houses: Procedures and validation.	[Online] http://acustica.ing.unife.it/ciarm/download.htm, Retrieved June 29th, 2006.	226		
Poole, B.	Reducing audio "buzz" in GSM cell phones.	<i>EDN</i> . [Online] http://www.edn.com/article/CA498768.html, 2005.	308		
Pope, S. T.	Machine tongues XV: Three packages for software sound synthesis.	<i>Computer Music J.</i> , 17(2), pp. 23–55, 1993.	318		
Portnoff, M.	Implementation of the digital phase vocoder using the fast Fourier transform.	<i>IEEE Trans. Acoust., Speech, and Signal Proc.</i> , 24(3), pp. 243–8, 1976.	113		
Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P.	<i>Numerical Recipes in C: The Art of Scientific Computing</i> , chapter 14.8.	Cambridge University Press, 1992.	220		
Pressing, J.	Nonlinear maps as generators of musical design.	<i>Computer Music J.</i> , 12(2), pp. 35–46, 1988.	198		
Pressing, J.	Some perspectives on performed sound and music in virtual environments.	<i>Presence</i> , 6(4), pp. 482–503, 1997.	274		
Pressnitzer, D. and Gnansia, D.	Real-time auditory models.	In <i>Proc. Int. Comp. Music Conf. (ICMC'05)</i> , Barcelona, Spain, pp. 295–298, 2005.	70		
Producers & Engineers Wing Surround Sound Recommendations Committee.	Recommendations for surround sound production.	Technical report, The National Academy of Recording Arts & Sciences, 2004.	296		
Puckette, M.	Combining event and signal processing in the MAX graphical programming environment.	<i>Computer Music J.</i> , 15(3), pp. 68–77, 1991.	70		
Puckette, M.	Low-dimensional parameter mapping using spectral envelopes.	In <i>Proc. Int. Comp. Music Conf. (ICMC'04)</i> , Miami, USA, pp. 406–408, 2004.	282		
Puckette, M. S.	Theory and techniques of electronic music.	[Online] http://www-crca.ucsd.edu/~msp/, 2006.	172		
Puckette, M. S. and Brown, J. C.	Accuracy of frequency estimates using the phase vocoder.	<i>IEEE Trans. Speech and Audio Proc.</i> , 6(2), pp. 166–176, 1989.	252		
Pulkki, V.	Virtual source positioning using vector base amplitude panning.	<i>J. Audio Eng. Soc.</i> , 45(6), pp. 456–466, 1997.	286		
Purnhagen, H.	Parameter estimation and tracking for time-varying sinusoids.	In <i>Proc. 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA)</i> , Leuven, Belgium, 2002.	245		
Purnhagen, H., Meine, N., and Edler, B.	Sinusoidal coding using loudness-based component selection.	In <i>Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'02)</i> , Orlando, USA, volume 2, pp. 1817–1820, 2002.	245		
Pye, D.	Content-based methods for managing electronic music.	In <i>Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'00)</i> , Istanbul, Turkey, pp. 2437 – 2440, 2000.	156		
Python.	Python programming language.	[Online] http://www.python.org/, Retrieved June 29th, 2006.	66		
Qi, Y., Minka, T. P., and Picard, R. W.	Bayesian spectrum estimation of unevenly sampled nonstationary data.	MIT Media Lab Technical Report Vismod-TR-556, 2002.	324		
Rabassó, C. A.	L'improvisation: du langage musical au langage littéraire.	<i>Intemporel: bulletin de la Société Nationale de Musique</i> , 15. [Online] http://catalogue.ircam.fr/hotes/snml/tpr15rabatxt.html, 1995.	113		
Rabenstein, R.	Minimization of transient signals in recursive time-varying digital filters.	<i>Circuits, Systems, and Sig. Proc.</i> , 7(3), pp. 345–359, 1988.	60		
Rabiner, L.	A tutorial on Hidden Markov Model and selected applications in speech.	<i>Proc. IEEE</i> , 77(2), pp. 257–285, 1989.	131		
Rasmussen, J.	<i>Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering</i> .	New York, NY, USA: Elsevier Science Inc., 1986.	52		
Raspaud, M., Marchand, S., and Girin, L.	A generalized polynomial and sinusoidal model for partial tracking and time stretching.	In <i>Proc. Int. Conf. on Digital Audio Effects (DAFx-05)</i> , Madrid, Spain, pp. 24–29, 2005.	186		
Rauhala, J. and Välimäki, V.	Parametric excitation model for waveguide piano synthesis.	In <i>Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'06)</i> , Toulouse, France, pp. 157–160, 2006.	76		
Rauhala, J. and Välimäki, V.	Tunable dispersion filter design for piano synthesis.	<i>IEEE Sig. Proc. Letters</i> , 13(5), pp. 253–256, 2006.	18, 76		
Ravindran, S. and Anderson, D.	Boosting as a dimensionality reduction tool for audio classification.	In <i>Proc. Int. Symp. Circuits and Systems (ISCAS '04)</i> , volume 3, pp. 465–468, 2004.	156		
Reddy, G. R. and Swamy, M. N. S.	Digital all-pass filter design through discrete hilbert transform.	In <i>Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'90)</i> , Albuquerque, USA, 1998.	18		
Renaissance.	The York Renaissance Project.	[Online] http://www.renaissanceyork.org.uk/, Retrieved June 29th, 2006.	226		
Rife, D. C. and Boorstin, R. R.	Single-tone parameter estimation from discrete-time observations.	<i>IEEE Trans. Information Theory</i> , IT-20, pp. 591–598, 1974.	144, 245		
Ris, C. and Dupont, S.	Assessing local noise level estimation methods: application to noise robust ASR.	<i>Speech Communication</i> , 34(2), pp. 141–158, 2001.	148		
Risset, J.-C.	Catalog of computer synthesized sound.	Murray Hill, Bell Telephone Laboratories, 1969.	198		
Roads, C.	<i>The Computer Music Tutorial</i> .	Cambridge, Massachusetts: MIT Press, 1996.	29, 113, 152, 168, 172, 198		
Roads, C.	Sound composition with pulsars.	<i>J. Audio Eng. Soc.</i> , 49(3), pp. 134–147, 2001.	202		
Roads, C.	<i>Microsound</i> .	Cambridge, Massachusetts: MIT Press, 2002.	262		
Röbel, A.	Estimating partial frequency and frequency slope using reassignment operators.	In <i>Proc. Int. Comp. Music Conf. (ICMC'02)</i> , Gothenburg, Sweden, pp. 122–125, 2002.	148		
Röbel, A.	Transient detection and preservation in the phase vocoder.	In <i>Proc. Int. Comp. Music Conf. (ICMC'03)</i> , Singapore, pp. 247–250, 2003.	330		
Röbel, A. and Rodet, X.	Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation.	In <i>Proc. Int. Conf. on Digital Audio Effects (DAFx-05)</i> , Madrid, Spain, pp. 30–35, 2005.	148		

- Röbel, A., Zivanovic, M., and Rodet, X. Signal decomposition by means of classification of spectral peaks. In *Proc. Int. Comp. Music Conf. (ICMC'04)*, Miami, USA, pp. 446–449, 2004. 144, 148, 252
- Röber, N. and Masuch, M. Leaving the screen: New perspectives in audio-only gaming. In *Proc. 11th Int. Conf. on Auditory Display (ICAD 2005)*, Limerick, Ireland, pp. 92–98, 2005. 274
- Rocchesso, D., Bresin, R., and Fernström, M. Sounding objects. *IEEE Multimedia*, 10(2), pp. 42–52, 2003. 324
- Rocchesso, D. and Scalcon, F. Accurate dispersion simulation for piano strings. In *Proc. Nordic Acoust. Meeting (NAM'96)*, Helsinki, Finland, pp. 407–414, 1996. 18, 76
- Rocchesso, D. and Scalcon, F. Bandwidth of perceived inharmonicity for physical modeling of dispersive strings. *IEEE Trans. Speech and Audio Proc.*, 7(5), pp. 597–601, 1999. 76
- Rodet, X. Nonlinear oscillations in sustained musical instruments: Models and control. In *Proc. Euromech*, Hamburg, Germany, 1993. 198
- Rodet, X. Synthesis and processing of the singing voice. In *1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA 2002)*, Leuven, Belgium, 2002. 44
- Rodet, X. and Lefèvre, A. The Diphone program: New features, new synthesis engines and experience of musical use. In *Proc. Int. Comp. Music Conf. (ICMC'97)*, Thessaloniki, Greece, pp. 418–421, 1997. 330
- Rodet, X., Potard, Y., and Barrière, J.-B. The CHANT project: From the synthesis of the singing voice to synthesis in general. *Computer Music J.*, 8(3), pp. 15–31, 1984. 330
- Rodet, X. and Röbel, A. Real time signal transposition with envelope preservation in the phase vocoder. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 672–675, 2005. 278
- Ross, M. Telecoil and telephones the most commonly misunderstood “assistive listening device”. *Hearing Loss: The Journal of Self Help for Hard of Hearing People*, 23(1), pp. 16–19, 2002. 308
- Rovan, J., Wanderley, M. M., Dubnov, S., and Depalle, P. Instrumental gestural mapping strategies as expressivity determinants in computer music performance. Technical report, Synthesis Team/Real-Time Systems Group, Ircam, Paris, France, 1997. 48
- Rowland, I. and Howe, T. N., E. *Vitruvius: Ten Books on Architecture*. Cambridge: Cambridge University Press, 1999. 226
- Rugh, W. J. *Nonlinear System Theory, The Volterra/Wiener approach*. Baltimore: The Johns Hopkins University Press, 1981. 12
- Ruiz, P. A technique for simulating the vibrations of strings with a digital computer. Master's thesis, University of Illinois, 1969. 82
- Sankey, J. and Sethares, W. A. A consonance-based approach to the harpsichord tuning of Domenico Scarlatti. *J. Acoust. Soc. Am.*, 101(4), pp. 2332–2337, 1997. 120
- Sanz-Serna, J. Symplectic integrators for hamiltonian problems: An overview. *Acta Numerica*, 1, pp. 243–286, 1991. 82
- Satar-Boroujeni, H. and Shafai, B. A robust algorithm for partial tracking of music signals. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, pp. 202–207, 2005. 245
- Sauvagerd, U. Circuit arrangement for influencing the frequency response of a digital audio signal. US Patent US5974156, 1999. 302
- Savage, W. R., Kottick, E. L., Hendrickson, T. J., and Marshall, K. D. Air and structural modes of a harpsichord. *J. Acoust. Soc. Am.*, 91(4), pp. 2180–2189, 1992. 120
- Savioja, L., Huopaniemi, J., Lokki, T., and Väänänen, R. Creating interactive virtual acoustic environments. *J. Audio Eng. Soc.*, 47(9), pp. 675–705, 1999. 164
- Savioja, L., Rinne, T., and Takala, T. Simulation of room acoustics with a 3-D finite difference mesh. In *Proc. Int. Comp. Music Conf. (ICMC'94)*, Århus, Denmark, pp. 463–466, 1994. 164
- Savioja, L. and Välimäki, V. Interpolated rectangular 3-D digital waveguide mesh algorithms with frequency warping. *IEEE Trans. Speech and Audio Proc.*, 11(6), pp. 783–790, 2003. 164
- Scavone, G. and Cook, P. Real-time computer modeling of woodwind instruments. In *Proc. ISMA 1998*, Leavenworth, WA, USA, 1998. 88
- Scavone, G. P. *An Acoustic Analysis of Single-Reed Woodwind Instruments with an Emphasis on Design and Performance Issues and Digital Waveguide Modeling Techniques*. PhD thesis, Music Dept., Stanford University, 1997. 94
- Scavone, G. P. Time-domain synthesis of conical bore instrument sounds. In *Proc. Int. Comp. Music Conf. (ICMC'02)*, Gothenburg, Sweden, pp. 9–15, 2002. 94
- Schaeffer, P. *Traité des Objets Musicaux*. Paris: Seuil, 1966. 113, 330
- Schafer, R. M. *The Tuning of the World*. Knopf: New York, 1977. 113
- Schatter, G., Züger, E., and Nitschke, C. Synaesthetic approach for a synthesizer interface based on genetic algorithms and fuzzy sets. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 664–667, 2005. 48
- Scheirer, E. D. Tempo and beat analysis of acoustic musical signals. *J. Acoust. Soc. Am.*, 103(1), pp. 588–601, 1998. 266
- Scheirer, E. D. *Music-Listening Systems*. PhD thesis, Massachusetts Institute of Technology, 2000. 266
- Schimmel, J. Using nonlinear amplifier simulation in dynamic range controllers. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, pp. 49–52, 2003. 66
- Schnieder, A. and Wright, M. A query system for Open Sound Control. In *Proc. 2004 OSC Conf. [Online]* <http://www.opensoundcontrol.org/proceedings>, Accessed Mar. 21, 2006. 48
- Schnell, N. and Battier, M. Introducing composed instruments, technical and musicological implications. In *Int. Conf. New Interf. for Musical Expr. (NIME'02)*, Dublin, Ireland, pp. 138–142, 2002. 52
- Schnell, N., Borghesi, R., Schwarz, D., Bevilacqua, F., and Müller, R. FTM – complex data structures for Max. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 9–12, 2005. 70
- Schnell, N. and Schwarz, D. GABOR, multi-representation real-time analysis/synthesis. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, pp. 122–126, 2005. 70
- Schönle, M., Fliege, N., and Zölzer, U. Parametric approximation of room impulse responses based on wavelet decomposition. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, pp. 68–71, 1993. 220
- Schroeder, M. R. Natural-sounding artificial reverberation. *J. Audio Eng. Soc.*, 10(3), pp. 219–223, 1962. 94
- Schroeder, M. R. The "Schroeder frequency" revisited. *J. Acoust. Soc. Am.*, 99(5), pp. 3240–3241, 1996. 164
- Schwarz, D. A system for data-driven concatenative sound synthesis. In *Proc. COST-G6 Conf. on Digital Audio Effects (DAFx-00)*, Verona, Italy, pp. 97–102, 2000. 252
- Schwarz, D. *Data-Driven Concatenative Sound Synthesis*. Thèse de doctorat, Université Paris 6 – Pierre et Marie Curie, Paris, 2004. 282
- Schwarz, D. Concatenative sound synthesis: The early years. *J. New Music Research*, 35(1), pp. 3–22. Special Issue on Audio Mosaicing, 2006. 282
- Scipio, A. D. Synthesis of environmental sound textures by iterated non linear functions and its ecological relevance to perceptual modeling. *J. New Music Research*, 31(2), pp. 5–21, 2002. 36
- Scipio, A. D. ‘Sound is the interface’: Sketches of a constructivistic ecosystemic view of interactive signal processing. In *Proc. XIV Colloq. Musical Informatics (CIM-03)*, Firenze, Italy, pp. 259–262, 2003. 202
- Serra, X. *A System for Sound Analysis / Transformation / Synthesis Based on a Deterministic plus Stochastic Decomposition*. PhD thesis, Stanford University, USA, 1989. 252, 258, 324

- Serra, X. *Musical Signal Processing*, chapter Musical Sound Modeling with Sinusoids plus Noise, pp. 91–122. Studies on New Music Research. Lisse, the Netherlands: G. De Poli and A. Picialli and S. T. Pope and C. Roads Eds. Swets & Zeitlinger, 1997. 245, 278
- Serra, X. Spectral modeling synthesis: Past and present, keynote in *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK. [online] <http://www.iua.upf.es/~xserra/presentaciones/Spectral-Modeling-Synthesis-Past-and-Present.pdf>, 2003, 2003. 258
- Serra, X. and Bonada, J. Sound transformations based on the SMS high level attributes. In *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, pp. 138–142, 1998. 36
- Serra, X. and Smith, J. O. Spectral Modeling Synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music J.*, 14(4), pp. 12–24, 1990. 144, 176, 180, 186
- Sheh, A. and Ellis, D. Chord segmentation and recognition using EM-trained Hidden Markov Models. In *Proc. Int. Conf. Music Information Retrieval (ISMIR'03)*, Baltimore, Maryland, USA, pp. 183–189, 2003. 131
- Shepard, R. Circularity in judgements of relative pitch. *J. Acoust. Soc. Am.*, 36, pp. 2346–2353, 1964. 131
- Sijbers, J., den Dekker, A., Dyck, D. V., and Raman, E. Estimation of Signal and Noise from Rician Distributed Data. In *Proc. Int. Conf. Sig. Proc. and Communications*, pp. 140–142, 1998. 144
- Simon, B. *Orthogonal Polynomials on the Unit Circle, Part 1: Classical Theory*, volume 54 of *AMS Colloquium Publications*. Providence, RI: American Mathematical Society, 2004. 191
- Simon, B. *Orthogonal Polynomials on the Unit Circle, Part 2: Spectral Theory*, volume 54 of *AMS Colloquium Publications*. Providence, RI: American Mathematical Society, 2004. 191
- Sinyor, E. and Wanderley, M. M. Gyrotyre: A dynamic hand-held computer-music controller based on a spinning wheel. In *Int. Conf. New Interf. for Musical Expr. (NIME'05)*, Vancouver, Canada, pp. 42–45, 2005. 52
- Skopoc, M. Hearing aid electromagnetic interference from digital wireless telephones. *IEEE Trans. Rehabil. Eng.*, 6(2), pp. 235–239, 1998. 308
- Slaney, M., Covell, M., and Lassiter, B. Automatic audio morphing. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, pp. 1001–1004, 1996. 48
- Smaragdis, P. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *Int. Cong. Independent Component Analysis and Blind Signal Separation (ICA)*, pp. 494–499, 2004. 262
- Smith, J. Virtual acoustic musical instruments: Review of models and selected research. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY. Presentation Overheads, 2005. 12
- Smith, J. O. Efficient simulation of the reed-bore and bow-string mechanisms. In *Proc. Int. Comp. Music Conf. (ICMC'86)*, Den Haag, Netherlands, pp. 275–280, 1986. 94
- Smith, J. O. Physical modeling using digital waveguides. *Computer Music J.*, 16(4), pp. 74–91, 1992. 76, 120, 164
- Smith, J. O. Digital audio resampling home page. [Online] <http://www-ccrma.stanford.edu/~jos/resample/>, 2002. 172
- Smith, J. O. and Cook, P. R. The second-order digital waveguide oscillator. In *Proc. Int. Comp. Music Conf. (ICMC'92)*, San Francisco, USA, pp. 150–153, 1992. 186
- Smith, J. O. and Serra, X. PARSHL: an analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation. In *Proc. Int. Comp. Music Conf. (ICMC'87)*, Champaign-Urbana, USA, pp. 290–297, 1987. 245
- Smith, III, J. O. *Introduction to Digital Filters*. [Online] <http://ccrma.stanford.edu/~jos/filters/>. online book, 2005. 18
- Smith, III, J. O. *Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects*. [Online] <http://ccrma.stanford.edu/~jos/pasp/>, 2006. 18, 82, 94, 168
- Smith, III, J. O. and Abel, J. Bark and ERB bilinear transform. *IEEE Trans. Speech and Audio Proc.*, 7(6), pp. 697–708, 1999. 18, 70, 168
- Software, U. Metasynth. [Online] <http://www.uisoftware.com/MetaSynth/>, Retrieved June 29th, 2006. 36
- Sommen, P. C. W., Van Gerwen, P. J., Kotmans, H. J., and Janssen, A. J. E. M. Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function. *IEEE Trans. Circuits and Systems*, 34(7), pp. 788–798, 1987. 266
- SpACE-Net. The spatial audio creative engineering network – SpACE-Net. [Online] <http://www.space-net.org.uk/>, Retrieved June 29th, 2006. 226
- Stahl, V., Fischer, A., and Bippus, R. Quantile based noise estimation for spectral subtraction and wiener filtering. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'00)*, Istanbul, Turkey, pp. 1875–1978, 2000. 148
- Start, E. W. *Direct Sound Enhancement by Wave Field Synthesis*. PhD thesis, TU Delft, The Netherlands, 1997. 214
- Steeneken, H. J. M. and Houtgast, T. A review of the MTF concept in room acoustics and its use for estimating speech intelligibility in auditoria. *J. Acoust. Soc. Am.*, 77(3), pp. 1069–1077, 1985. 232
- Sterian, A. and Wakefield, G. H. A model-based approach to partial tracking for musical transcription. In *Proc. SPIE Int. Symp. Optical Science, Eng., and Instrumentation, San Diego, California*, San Diego, 1998. 245
- Stilson, T. and Smith, J. Alias-free digital synthesis of classic analog waveforms. In *Proc. Int. Comp. Music Conf. (ICMC'96)*, Hong Kong, pp. 332–335, 1996. 172
- Stilson, T. and Smith, J. Analyzing the Moog VCF with consideratons for digital implementation. In *Proc. Int. Comp. Music Conf. (ICMC'96)*, Hong Kong, pp. 398–401, 1996. 12
- Stinchcombe, T. E. Derivation of the transfer function of the Moog ladder filter. Technical report, [Online] http://mysite.wanadoo-members.co.uk/tstinchcombe/synth/Moog_ladder_tf.pdf, 2005. 12
- Strandh, R. and Marchand, S. Real-time generation of sound from parameters of additive synthesis. In *Proc. Journées d'Informatique Musicale*, pp. 83–88, 1999. 186
- Strikwerda, J. *Finite Difference Schemes and Partial Differential Equations*. Pacific Grove, Calif.: Wadsworth and Brooks/Cole Advanced Books and Software, 1989. 82
- Stuart, A. and Ord, J. K. *Kendall's Advanced Theory of Statistics, Vol. 1: Distribution Theory* (6th. ed.). New York: Oxford University Press, 1998. 148
- Sullivan, C. R. Extending the Karplus-Strong algorithm to synthesize electric guitar timbres with distortion and feedback. *Computer Music J.*, 14, pp. 26–37, 1990. 168
- Sundaram, H., Joshi, S., and Bhatt, R. Scale periodicity and its sampling theorem. *IEEE Trans. Sig. Proc.*, 45(7), pp. 1862–1864, 1997. 99
- Sundberg, J. *Integrated Human Brain Science: Theory, Method Application (Music)*, chapter Grouping and differentiation two main principles in the performance of music, pp. 299–314. Elsevier Science B.V., 2000. 24
- Swamy, M. N. S. and Thyagarajan, K. S. Digital bandpass and bandstop filters with variable center frequency and bandwidth. *Proc. IEEE*, 64(11), pp. 1632–1634, 1976. 40
- Szasz, O. On the partial sums of certain Fourier series. *Amer. J. Math.*, 59(3), pp. 696–708, 1937. 191
- Szegő, G. Beiträge zur theorie der Toeplitzschen formen, I. *Math. Z.*, 6, pp. 167–202, 1920. 191
- Szegő, G. Beiträge zur theorie der Toeplitzschen formen, II. *Math. Z.*, 9, pp. 167–190, 1921. 191
- Szilagyi, T. Tap Dynamics. [Online] <http://tap-plugins.sourceforge.net/ladspa/dynamics.html>, Retrieved June 29th, 2006. 66
- Talukdar, K. K. and Lawing, W. D. Estimation of the Parameters of the Rice Distribution. *J. Audio Eng. Soc.*, 89, pp. 1193–1197, 1991. 144

- Targett, D. S. and Fernström, M. Audio games: Fun for all? all for fun? In *Proc. 9th Int. Conf. on Auditory Display (ICAD 2003)*, Boston, USA, pp. 216–219, 2003. 274
- Temperley, D. What's key for key? the Krumhansl-Schmuckler key finding algorithm reconsidered. *Music Perception*, 17(1), pp. 65–100, 1999. 131
- Terroir, J. and Guillemain, P. A simple dynamic tone hole model for real-time synthesis of clarinet-like instruments. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, 2005. 88
- Thibault, F. and Depalle, P. Adaptive processing of singing voice timbre. In *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE2004)*, Niagara Falls, Ontario, Canada, pp. 871–874, 2004. 160
- Thiran, J.-P. Recursive digital filters with maximally flat group delay. *IEEE Trans. Circuit Theory*, 18(6), pp. 659–664, 1971. 76
- Thom, B. BoB: An improvisational music companion. In *Proc. Fourth Int. Conf. Autonomous Agents*, Barcelona, pp. 309–316, 2000. 29
- Thon, S. and Ghazanfarpour, D. Ocean waves synthesis and animation using real world information. *Computer Graphics*, 26(1), pp. 99–108, 2002. 36
- Thornburg, H. and Leistikow, R. Analysis and resynthesis of quasi-harmonic sounds: An iterative filterbank approach. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, pp. 129–134, 2003. 180, 324
- Tobudic, A. and Widmer, G. Playing Mozart phrase by phrase. Technical report, ÖFAI-TR-2003-02, 2003. 24
- Tolonen, T., Välimäki, V., and Karjalainen, M. Modeling of tension modulation nonlinearity in plucked strings. *IEEE Trans. Speech and Audio Proc.*, 8, pp. 300–310, 2000. 82
- Torrence, C. and Compo, G. P. A practical guide to wavelet analysis. *Bulletin Am. Meteorological Soc.*, 79(1), pp. 61–78, 1998. 220
- Toussaint, G. A comparison of rhythmic similarity measures. In *Proc. Int. Conf. Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, pp. 134–137. [Online] <http://www.iua.upf.es/mtg/ismir2004/review/CRFILES/paper134-3a3dbcdaf2f38eaa5d883c5bbbedf1092.pdf>, 2004. 29
- Traunmüller, H. and Eriksson, A. Acoustic effects of variation in vocal effort by men, women and children. *J. Acoust. Soc. Am.*, 107(6), pp. 3438–3451, 2000. 160
- Trausmuth, R. and Huovilainen, A. POWERWAVE – a high performance single chip interpolating wavetable synthesizer. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005. 290
- Trautmann, L. and Rabenstein, R. *Digital Sound Synthesis by Physical Modeling Using the Functional Transformation Method*. New York: Kluwer Academic/Plenum Publishers, 2003. 302
- Tropp, J. A. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Information Theory*, 50(10), pp. 2231–2242, Oct. 2004. 270
- Truax, B. Chaotic non-linear systems and digital synthesis: an exploratory study. In *Proc. Int. Comp. Music Conf. (ICMC'90)*, Glasgow, Scotland, pp. 100–103, 1990. 198
- Tutschku, H. OpenMusic OM-AS Library. *Ircam Software Documentation*, 1998. 330
- Tzanetakis, G. and Cook, P. Musical genre classification of audio signals. *IEEE Trans. Speech and Audio Proc.*, 10(5), pp. 293–302, 2002. 156
- Uhle, C. and Herre, J. Estimation of tempo, micro time and time signature from percussive music. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, 2003. 266
- U&I Software. Metasynth 4. [Online] <http://uisoftware.com/MetaSynth/>, Retrieved June 29th, 2006. 314
- Unique Recording Software. URS Plugins. [Online] <http://www.ursplugins.com/>, Retrieved June 29th, 2006. 66
- Unser, M. Splines: A perfect fit for signal and image processing. *Signal Processing Magazine*, 16(6), pp. 22–38, 1999. 99
- Usher, J., Cooperstock, J., and Woszczyk, W. A multi-filter approach to acoustic echo cancelation for teleconferencing. In *Proc. 147th Meeting Acoust. Soc. Am.*, New York, USA, 2004. 266
- Štěpánek, J. Evaluation of timbre of violin tones according to selected verbal attributes. In *32nd Int. Acoustical Conf., European Acoustics Association (EAA) Symp. "Acoustics Banská Štiavnica"*, pp. 129–132, 2002. 126
- Štěpánek, J. The study of violin timbre using spontaneous verbal description and verbal attribute rating. In *Forum Acusticum*, Sevilla, Spain, 2002. 126
- Štěpánek, J. Relations between perceptual space and verbal description in violin timbre. In *acústica 2004*. CD ROM, AFP 077-S, 2004. 126
- Štěpánek, J. and Moravec, O. Verbal description of musical sound timbre in Czech language and its relation to musicians profession and performance quality. In *Proc. Colloq. Interdisciplinary Musicology (CIM'05)*, Montréal, Québec, Canada, Electronic proceedings. [Online] http://www.oicm.umontreal.ca/cim05/cim05_articles/STEPANEK_J_CIM05_02.pdf, 2005. 126
- Štěpánek, J. and Otčenášek, Z. Listener common and group perceptual dimensions in violin timbre. In *Proc. Stockholm Music Acoustics Conf. (SMAC'03)* Stockholm, Sweden, pp. 663–666, 2003. 126
- Štěpánek, J. and Otčenášek, Z. Interpretation of violin spectrum using psychoacoustic experiments. In *CD of Proc. Int. Symposium on Musical Acoustics (ISMA2004)*, volume 4-S2-1, Nara, Japan, 2004. 126
- Štěpánek, J. and Otčenášek, Z. Acoustical correlates of the main features of violin timbre perception. In *Proc. Colloq. Interdisciplinary Musicology (CIM'05)*, Montréal, Québec, Canada, Electronic proceedings. [Online] http://www.oicm.umontreal.ca/cim05/cim05_articles/STEPANEK_J_CIM05_01.pdf, 2005. 126
- Štěpánek, J., Otčenášek, Z., and Melka, A. Comparison of five perceptual timbre spaces of violin tones of different pitches. In *CD-ROM of Joint Meeting 137th ASA, 2nd EAA Forum Acusticum 1999, 25th DAGA Berlin. 5aMUB5*, 1999. 126
- Štěpánek, J., Otčenášek, Z., Syrový, V., Täsch, C., and Angster, J. Spectral features influencing perception of pipe organ sounds. In *Forum Acusticum 2005*, Budapest, pp. 465–469, 2005. 126
- Štěpánek, J., Syrový, V., Otčenášek, Z., Täsch, C., and Angster, J. Verbal description of organ principal 8' sound. In *32. DAGA, Braunschweig, in print*, 2006. 126
- Vafin, R. *Towards Flexible Coding*. PhD thesis, KTH Stockholm, 2004. 270
- Vaidyanathan, P. *Multirate Systems and Filter Banks*. Upper Saddle River NJ: Prentice Hall Signal Processing Series, 1993. 106
- Välimäki, V. *Discrete-Time Modeling of Acoustic Tubes Using Fractional Delay Filters*. PhD thesis, Helsinki University of Technology, Faculty of Electrical Engineering, Laboratory of Acoustic and Audio Signal Processing, Espoo, Finland, Report no. 37. [Online] http://www.acoustics.hut.fi/~vpv/publications/vesan_vaitos, 1995. 60, 94
- Välimäki, V. Discrete-time synthesis of the sawtooth waveform with reduced aliasing. *IEEE Sig. Proc. Letters*, 12(3), pp. 214–217, 2005. 172
- Välimäki, V. and Laakso, T. I. Suppression of transients in time-varying recursive filters for audio signals. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'98)*, Seattle, USA, volume 6, pp. 3569–3572. [Online] <http://www.acoustics.hut.fi/~vpv/publications/icassp98-trel.pdf>, 1998. 60
- Välimäki, V., Pakarinen, J., Erkut, C., and Karjalainen, M. Discrete-time modelling of musical instruments. *Report on Progress in Physics*, 69(1), pp. 1–78, 2006. 99, 164
- Välimäki, V., Penttilä, H., Knif, J., Laurson, M., and Erkut, C. Sound synthesis of the harpsichord using a computationally efficient physical model. *EURASIP J. on Applied Sig. Proc.*, 2004(7), pp. 934–948, 2004. 120
- Välimäki, V. and Tolonen, T. Development and calibration of a guitar synthesizer. *J. Audio Eng. Soc.*, 46, pp. 766–778, 1998. 120

- Valette, C. The mechanics of vibrating strings. In A. Hirschberg, J. Kergomard, & G. Weinreich (Eds.), *Mechanics of Musical Instruments* pp. 116–183. New York: Springer, 1995. 82
- Valsamakis, N. and Miranda, E. R. Iterative sound synthesis using cross-coupled digital oscillators. *Digital Creativity*, 38(4), pp. 331–336, 2005. 198
- van den Doel, K., Kry, P. G., and Pai, D. K. FOLEYAUTOMATIC: Physically-based sound effects for interactive simulation and animation. In *Proc. ACM SIGGRAPH*, Los Angeles, USA, pp. 537–544, 2001. 324
- van der Waal, R. G. and Veldhuis, R. N. J. Subband coding of stereophonic digital audio signals. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'91)*, Toronto, Canada, pp. 3601–3604, 1991. 296
- van der Wal, M., Start, E. W., and de Vries, D. Design of logarithmically spaced constant-directivity transducer arrays. *J. Audio Eng. Soc.*, 44(6), pp. 497–507, 1996. 214
- Van Duyne, S. A. and Smith, J. O. Physical modeling with the 2-D digital waveguide mesh. In *Proc. Int. Comp. Music Conf. (ICMC'93)*, Tokyo, Japan, pp. 40–47, 1993. 164
- Van Duyne, S. A. and Smith, J. O. The tetrahedral digital waveguide mesh. In *Proc. IEEE Workshop Appl. of Dig. Sig. Proc. to Audio and Acoust.*, New Palz, NY, pp. 234–237, 1995. 164
- Van Duyne, S. A. and Smith, J. O. The 3D tetrahedral digital waveguide mesh with musical applications. In *Proc. Int. Comp. Music Conf. (ICMC'96)*, Hong Kong, pp. 9–16, 1996. 164
- Van Duyne, S. A. and Smith III, J. O. A simplified approach to modeling dispersion caused by stiffness in strings and plates. In *Proc. Int. Comp. Music Conf. (ICMC'94)*, Århus, Denmark, pp. 407–410, 1994. 76
- van Nort, D., Wanderley, M. M., and Depalle, P. On the choice of mappings based on geometric properties. In *Int. Conf. New Interf. for Musical Expr. (NIME'04)*, Hamamatsu, Japan, pp. 87–91, 2004. 48
- van Walstijn, M. and Scavone, G. The wave digital tonehole model. In *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, 2000. 88
- Ventura i Figueras, R., Vandergheynst, P., and Frossard, P. Low rate and flexible image coding with redundant representations. *IEEE Trans. Image Processing*, 15(3), pp. 726–739, 2006. 270
- Verblunsky, S. On the Fourier constants of a bounded function. *Proc. Cambridge Philos. Soc.*, 32, pp. 201–211, 1936. 191
- Vercoe, B. *Csound: A Manual of the Audio Processing System*. MIT Media Lab, 1986. 278, 318
- Verfaille, V. *Effets Audionumériques Adaptatifs : Théorie, Mise en Œuvre et Usage en Création Musicale Numérique*. PhD thesis, Université Aix-Marseille II, 2003. 113
- Verfaille, V. and Depalle, P. Adaptive effects based on STFT, using a source-filter model. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 296–301, 2004. 113, 278
- Verfaille, V., Guastavino, C., and Depalle, P. Perceptual evaluation of vibrato models. In *Proc. Colloq. Interdisciplinary Musicology (CIM'05)*, Montreal, Quebec, Canada, 2005. 152
- Verfaille, V., Wanderley, M. M., and Depalle, P. Mapping strategies for gestural control of adaptive digital audio effects. *J. New Music Research*, 35(1), pp. 71–93, 2006. 113
- Verfaille, V., Zölzer, U., and Arfib, D. Adaptive digital audio effects (ADAFx): A new class of sound transformations. *IEEE Trans. Audio, Speech and Language Proc.*, 14(5), pp. 1817–1831, 2006. 113
- Verhelst, W. and Nilens, P. A modified-superposition speech synthesizer and its applications. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'86)*, Tokyo, Japan, volume 3, pp. 2007–2010, 1986. 60
- Verma, T. *A Perceptually Based Audio Signal Model With Application to Scalable Audio Compression*. PhD thesis, Stanford University, 2000. 270
- Verma, T. S. and Meng, T. H. An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio. In *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'98)*, Seattle, USA, pp. 12–15, 1998. 324
- Vermaak, J. and Blake, A. Nonlinear filtering for speaker tracking in noisy and reverberant environments. *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, 5, pp. 3021–3024, 2001. 208
- Verplank, B., Mathews, M., and Shaw, R. Scanned synthesis. In *Proc. Int. Comp. Music Conf. (ICMC'00)*, Berlin, Germany, ICMA, pp. 368–371, 2000. 314
- Vincent, E., Gribonval, R., and Févotte, C. Performance measurement in blind audio source separation. *IEEE Trans. Audio, Speech and Language Proc.*, 14(4), pp. 1462–1469, 2006. 70
- Vinet, H. The semantic Hifi project. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, pp. 503–506, 2005. 131
- Virtanen, T. Sound source separation using sparse coding with temporal continuity objective. In *Proc. Int. Comp. Music Conf. (ICMC'03)*, Singapore, pp. 231–234, 2003. 262
- Virtanen, T. Separation of sound sources by convolutive sparse coding. In *ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing (SAPA)*, paper 55, 2004. 262
- Viste, H. *Binaural Localization and Separation Techniques*. PhD thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 2004. 238
- Vogel, P. *Application of Wave Field Synthesis in Room Acoustics*. PhD thesis, TU Delft, The Netherlands, 1993. 214
- von Bismarck, G. Timbre of steady sounds: A factorial investigation of its verbal attributes. *Acustica*, 30, pp. 146–159, 1974. 126
- Vos, J. and Rasch, R. The perceptual onset of musical tones. *Perception and Psychophysics*, 29(4), pp. 323–335, 1981. 137
- Wakefield, G. Mathematical representation of joint time-chroma distributions. In *SPIE Conf. Advanced Sig. Proc. Algorithms, Architecture and Implementations*, Denver, Colorado, USA, pp. 637–645, 1999. 131
- Wanderley, M. *Gesture and Sign Language in Human-Computer Interaction: International Gesture Workshop*, chapter Quantitative analysis of non-obvious performer gestures, pp. 241. Springer Berlin, Heidelberg, 2002. 24
- Wanderley, M. M. and Depalle, P. Gestural control of sound synthesis. *Proc. IEEE, Spec. Issue on Eng. and Music – Supervisory Control and Auditory Communication*, 92(4), pp. 632–644, 2004. 48, 52
- Wang, G. and Cook, P. On-the-fly programming: Using code as an expressive musical instrument. In *Int. Conf. New Interf. for Musical Expr. (NIME'04)*, Hamamatsu, Japan, pp. 138–143, 2004. 52
- Ward, D. B., Lehmann, E. A., and Williamson, R. C. Particle filtering algorithms for tracking an acoustic source in a reverberant environment. *IEEE Trans. Speech and Audio Proc.*, 11(6), pp. 826–836, 2003. 208
- Waves. Renaissance compressor. [Online] <http://www.waves.com>, Retrieved June 29th, 2006. 66
- Wei, L. Deterministic texture analysis and synthesis using tree structure vector quantization. In *Proc. ACM SIGGRAPH* Los Angeles, USA, ACM, pp. 207–213, 1999. 314
- Weinreich, G. and Caussé, R. Digital and analog bows: Hybrid mechanical-electrical systems. *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'86)*, Tokyo, Japan, 2, pp. 1297–1299, 1986. 168
- Wessel, D. and Wright, M. Problems and prospects for intimate musical control of computers. *Computer Music J.*, 26(3), pp. 11–22, 2002. 52, 274
- Weyer, R. D. Time-frequency-structures in the attack transients of piano and harpsichord sounds - I. *Acustica*, 35, pp. 232–252, 1976. 120
- Weyer, R. D. Time-varying time-frequency-structures in the attack transients of piano and harpsichord sounds - II. *Acustica*, 36(4), pp. 241–258, 1976. 120

- Wickelmaier, F. and Ellermeier, W. Scaling the perceived fluctuation strength of frequency-modulated tones. *Ac. Soc. of Am.*, 115(5), pp. 2600, 2004. 152
- Widmer, G. and Goebel, W. Computational models of expressive music performance: The state of the art. *J. New Music Research*, 33(3), pp. 203–216, 2004. 24, 44
- Wilson, S. G. Magnitude/phase quantization of independent Gaussian variates. *IEEE Trans. Communication*, COM-28(11), pp. 1924–1929, 1980. 270
- Wilson, T. A. and Beavers, G. S. Operating modes of the clarinet. *J. Acoust. Soc. Am.*, 56, pp. 653–658, 1974. 88
- Winsberg, S. and De Soete, G. A latent class approach to fitting the weighted euclidean model, CLASCAL. In *Psychometrika*, volume 58, pp. 315–330, 1993. 126
- Wise, D. A survey of biquad filter structures for application to digital parametric equalization. *Presented at the 105th AES convention*, preprint #4820, 1998. 56
- Wishart, T. *Audible Design*. Orpheus the Pantomime, York, 1996. 278
- Wolfe, P. J. and Godsill, S. J. Audio signal processing using complex wavelets. In *114th Conv. Audio Eng. Soc.*, Amsterdam, The Netherlands, 2003. 220
- Woodhouse, J. Plucked guitar transients: comparison of measurements and synthesis. *Acta Acustica united with Acustica*, 90(5), pp. 945–965, 2004. 76
- Woodworth, R. S. *Experimental Psychology*. New York: Holt, 1954. 238
- Wornell, G. W. and Oppenheim, A. V. Wavelet-based representations for a class of self-similar signals with applications to fractal modulation. *IEEE Trans. Information Theory*, 38(2), pp. 785–800, 1992. 106
- Wozniowski, M., Settel, Z., and Cooperstock, J. A framework for immersive spatial audio performance. In *Int. Conf. New Interf. for Musical Expr. (NIME'06)*, Paris, France, pp. 11–16, 2006. 274
- Wrigh, M. and Freed, A. Open SoundControl: A new protocol for communicating with sound synthesizers. In *Proc. Int. Comp. Music Conf. (ICMC'97)*, Thessaloniki, Greece, pp. 101–104, 1997. 48
- Wright, J. Synthesising band limited waveforms using wavetables. [Online] <http://www.musicdsp.org/files/bandlimited.pdf>, Retrieved June 29th, 2006. 172
- Wyse, L. A sound modeling and synthesis system designed for maximum usability. In *Proc. Int. Comp. Music Conf. (ICMC'03)*, Singapore, pp. 447–451, 2003. 48
- Xenakis, I. *Formalized Music* (Revised ed.). New York: Pendragon Press, 1992. 202
- Yang, D. T., Kyriakakis, C., and Jay Kuo, C. C. *High-Fidelity Multichannel Audio Coding*. EURASIP Book Series on Sig. Proc. and Communications, 2004. 296
- Yegnanarayana, B. Design of recursive group-delay filters by autoregressive modeling. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 30(4), pp. 632–637, 1982. 18
- Yeo, W. S. Image sonification: Image to sound. [Online] <http://www.math.ucsb.edu/~woony/research/winter01/mat310/>, 2001. 314
- Yeo, W. S. Raster scanning. [Online] <http://ccrma.stanford.edu/~woony/research/raster/>, Retrieved June 29th, 2006. 314
- Yeo, W. S. and Berger, J. Application of image sonification methods to music. In *Proc. Int. Comp. Music Conf. (ICMC'05)*, Barcelona, Spain, ICMA, pp. 219–222, 2005. 314
- Yilmaz, Ö. and Rickard, S. Blind separation of speech mixtures via time-frequency masking. *IEEE Trans. Sig. Proc.*, 52(7), pp. 1830–1847, 2004. 232, 238
- Zettenberg, L. H. and Zhang, Q. Elimination of transients in adaptive filters with application to speech coding. *Signal Processing*, 15(4), pp. 419–428, 1988. 60
- Zhu, X. and Wyse, L. Sound texture modeling and time-frequency LPC. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, pp. 345–349, 2004. 324
- Zölzer, U. *Digital Audio Signal Processing*. Chichester, UK: John Wiley & Sons, 1997. 40, 56, 120
- Zölzer, U. *DAFX - Digital Audio Effects*. New York, USA: John Wiley & Sons, 2002. 113, 152, 290
- Zölzer, U., Redmer, B., and Bucholtz, J. Strategies for switching digital audio filters. In *95th Conv. Audio Eng. Soc.*, New York, USA. preprint 3714, 1993. 60
- Zotkin, D. N. and Duraiswami, R. Accelerated speech source localization via a hierarchical search of steered response power. *IEEE Trans. Speech and Audio Proc.*, 12(5), pp. 499–508, 2004. 208
- Zwicker, E. and Fastl, H. *Psychoacoustics: Facts and Models*. Berlin: Springer-Verlag, 1990. 120, 152
- Zwicker, E., Flottorp, G., and Stevens, S. S. Critical bandwidth in loudness summation. *J. Acoust. Soc. Am.*, 29, pp. 548–557, 1957. 70

Index of Authors

A

- Abel, Jonathan S.....13
Arfib, Daniel.....31
Avanzini, Federico297

B

- Bağcı, Ulaş153
Barthet, Mathieu19
Beller, Grégory279
Berdahl, Edgar165
Berger, Jonathan309
Bilbao, Stefan77
Birnbaum, David49
Blaauw, Merlijn41
Blake, Andrew203
Bökesoy, Sinan199
Bonada, Jordi41
Bresson, Jean325
Briand, Manuel291
Britton, Sam279

C

- Cabrera, Andrés61
Cafagna, Vittorio187
Cook, Perry R.319
Cooperstock, Jeremy R.271
Corteel, Étienne209
Couturier, Jean-Michel31

D

- Daudet, Laurent267
De Sena, Antonio95
Depalle, Philippe177
Dixon, Simon133
Dressler, Karin247
Driessen, Peter F.157
Dusek, Christian287

E

- Eisenberg, Gunnar.....259
Erne, Markus227
Erzin, Engin153
Essl, Georg193
Evangelista, Gianpaolo101

F

- Faller, Christof227
Fallon, Maurice203
Favrot, Alexis227
ffitch, John77
Filatriau, Jehan-Julien31
Fontana, Federico297

G

- Geiger, Günter169
Godsill, Simon203, 303
Guastavino, Catherine107
Guillemain, Philippe83
Gurevich, Michael45

H

- Hanna, Pierre139
Hélie, Thomas7
Holters, Martin37

J

- Janer, Jordi41

K

- Kalinichenko, Victor57
Kelloniemi, Antti161
Kronland-Martinet, Richard19

L

- Lagrange, Mathieu239
Laurson, Mikael283
Lazzarini, Victor275, 315
Lin, Han303
Lokki, Tapiro215
Lysaght, Tom275

M

- Malloch, Joseph49
Marchand, Sylvain139, 181, 233, 239
Martens, William149
Martin, Nadine291
Marui, Atsushi149
Meurisse, Guillaume139
Misra, Ananya319
Mouba, Joan233
Müller, Remy67
Murphy, Damian T.221, 253

N

- Nordstrom, Karl I.157
Norilo, Vesa283

O

- Orlarey, Yann287

P

- Parson, Dale E.25
Peeters, Geoffroy127
Pendharkar, Chinmay45
Penttinen, Henri115
Piché, Jean315

R

- Rauhala, Jukka71
Ravelli, Emmanuel267
Röbel, Axel145
Robine, Matthias181
Rocchesso, Davide95

S

- Sandler, Mark173
Scavone, Gary P.89
Schnell, Norbert67
Schwarz, Diemo67, 279
Settel, Zack271
Sikora, Thomas259
Sinyor, Elliot49
Smith, Julius O.1, 13, 89, 165
Štěpánek, Jan121
Strandh, Robert181

T

- Terroir, Jonathan83
Timoney, Joe275
Traube, Caroline107
Trausmuth, Robert287

U

- Usher, John263

V

- Välimäki, Vesa71
Van Nort, Doug177
Verbrugghe, Bruno279
Verfaillie, Vincent107
Vesa, Sampo215
Vicinanza, Domenico187
Virette, David291

W

- Wanderley, Marcelo M.49
Wang, Ge319
Wells, Jeremy J.253
Wen, Xue173
Wise, Duane K.53
Wozniewski, Mike271
Wyse, Lonce45

Y

- Yeh, Chunghsin145
Yeh, David T.1
Yeo, Woon Seung309
Ystad, Sølvvi19

Z

- Zölzer, Udo37