



10º Simpósio Brasileiro de Computação Musical
The 10th Brazilian Symposium on Computer Music

Proceedings of the 10th Brazilian Symposium on Computer Music

Current Frameworks for Music Information Representation

Pontifícia Universidade Católica de Minas Gerais
Campus Belo Horizonte – Unidade São Gabriel

October 3-6, 2005.

Organized by:



Promoted by:



Sponsored by:



SBCM 2005 (2005 : Belo Horizonte, MG)

Web site: <http://www.cefala.org/sbcm2005/>

Anais do X Simpósio Brasileiro de Computação Musical
Proceedings of the 10th Brazilian Symposium on Computer Music, FuMARC,
Belo Horizonte, 2005.

Current Frameworks for Music Information Representation

Edited by Hugo Bastos de Paula and Hani Camille Yehia

393p.: 14,8 x 21 cm.

Index included

1. COMPUTAÇÃO – Congressos
2. MÚSICA – Congressos
3. ARTE COMPUTACIONAL – Congressos, Concertos



Foreword from the Chairmen

First of all, welcome to the 10th Brazilian Symposium on Computer Music and to the city of Belo Horizonte. The 10th SBCM will take place from October the 3rd to 6th, and will be held at the Catholic University - PUC Minas in São Gabriel. The computer music symposia are promoted by the Brazilian Computer Society (SBC) and your tenth edition will be coordinated together by the Institute of Informatics of PUC Minas and the School of Music of UFMG.

The 10th SBCM is aimed at the scientific and academic communities from the fields of Computer Science, Music, Psychology, Engineering, Linguistics, Education, and many others. Through discussion panels, paper presentations and tutorials, the SBCM will promote the reflection and information interchange among researchers of Brazil and above, that represent the state-of-the-art of the scientific and artistic research that are currently being carried out.

The event's theme is Current Frameworks for Music Information Representation. The decision to focus on the representation of musical information follows the fast increase of research on Music Information Retrieval (MIR) techniques. This development reflects the explosion of the amount of data related to music that is available nowadays on the internet and on local networks, and the need to better represent this information in order to make processing, indexing and content searching fast and easy.

The event will have a talk from Professor Petri Toiviainen, from Jyväskylä University and will have the OpenMusic 5.0 Tutorial lectured by Carlos Agon, Jean Bresson and Olivier Warusfel, from IRCAM. IRCAM will also give a prize for the best student paper presented during SBCM. Two awards will be given: one for the best technical paper and another for the best music paper.

The 10th SBCM will also give three concerts. The opening concert will present works from young Brazilian Musicians in a unique show, alternating instrumental music and songs. The performers of this concert are Antonio Loureiro, Kristoff Silva, Rafael Macedo e Rafael Martini (composers and players), and Pedro Trigo (acoustic bass) and Vinícius Augustus (sax and flute). Using sophisticated harmony and Brazilian rhythms, these young artists manage to keep the roots of the local music of the state of Minas Gerais and at the same time open new pathways in between the classical and popular, the acoustical and electronical, the pop and the contemporary styles. The two other concerts will promote an exhibition of current developments in electroacoustic and electronic music in Brazil, presenting recent compositions from Brazilian composers such as Sergio Freire, Felipe Amorim, Rodolfo Caesar, Fernando Iazzetta, Denise Garcia, Didier Guigue, Jonatas Manzzolli, Silvio Ferraz.

I am confident that the Symposium will provide a most exciting Forum for the Computer Music community of Brazil and beyond and I sincerely hope that you enjoy your time in Belo Horizonte.

Hugo de Paula and Maurício Loureiro (Chairmen, October/2005).

Foreword from the Program Chairs

In its 10th edition, the Brazilian Symposium on Computer Music (SBCM) is moving towards maturity, showing an increase both in the quality and quantity of papers submitted. The preparation for the 2005 edition of SBCM started in October 2004 with the constitution of the Program Committee (PC). As in previous editions, the PC is composed not only of 19 top Brazilian researchers in the field but also of 24 world-class researchers from renowned institutions such as IRCAM, Stanford, UIUC, UC Berkeley, and Carnegie-Mellon. The PC has members from Argentina, Belgium, Canada, Finland, France, Greece, Hong Kong, Ireland, Italy, Netherlands, New Zealand, South Africa, Sweden, UK, Uruguay, and USA.

We received a total of 54 submissions in four categories: full technical paper, full music paper, short paper, and poster. The full technical papers describe original research with scientific contributions; we received 35 full technical paper submissions, which were carefully reviewed by the PC. Each technical paper was reviewed by at least 3 reviewers and, in some cases, we had up to 5 reviewers for a single paper. The PC selected 18 of the 35 submissions to be presented as full papers, yielding an acceptance rate of 51%.

Music papers aim to discuss aesthetic and poetic issues concerning the musical discourse and the experience of composers using computational tools to develop compositional strategies, to control sonic process in real-time and to digital processing of instrumental sounds. The music paper committee, constituted of members from UFMG, USP, UNICAMP, UFPA, UFRJ, U. Louisiana, UIUC, U. Nacional de Tres de Febrero, and IRCAM, selected 7 papers to be presented during the symposium. These selected works cover improvisation and real-time audio processing, mixed electroacoustic composition, computational environments for image and sound, and Brazilian electroacoustic music.

Short papers and posters were selected in two phases. First, the best full paper submissions that were not accepted were invited to be presented as either short papers or posters. Second, we had a new call for short papers and posters in which we received additional submissions. In the end, 17 short papers and 6 posters were accepted.

We are grateful for the valuable help provided by Lisandro Granville, Diego Contessa, Leo Ueda, and Nelson Lago with the management of the JEMS online submission system. We also thank the PC members and additional reviewers, who completed over 150 reviews, helping both select the papers and improve their quality with valuable feedback for the authors.

We are very happy with the quality and diversity of the works presented in this Symposium. The range of topics covered is very wide including: Acoustics; Artificial Intelligence; Evolutionary Systems; Sound Diffusion Systems; Software Tools for Composition and Sound Synthesis; Music Analysis and Pattern Recognition; Music Representation, Retrieval, and Classification; Psychoacoustics, and Cognitive Modeling. During the symposium, a special committee will select the best student technical paper and the best student music paper based on the final camera-ready versions submitted by authors. Both of them will be awarded a one-year subscription of the IRCAM Forum.

We sincerely hope that all participants have a lot of fun during these four days in Belo Horizonte and that the papers in these proceedings, also available in the SBCM repository (<http://gsd.ime.usp.br/sbcm>), be very valuable for the Computer Music community.

Jônatas Manzoli, Regis Rossi Faria, and Fabio Kon, October/2005.

SBCM 2005 Conference Committee

General Chairs

Hugo de Paula, *PUC Minas*, Brazil.

Maurício Loureiro, *UFMG*, Brazil.

Program Chairs

Technical paper chair: Fabio Kon, *IME/USP*, Brazil.

Music papers chair: Jônatas Manzolli, *Unicamp*, Brazil.

Short papers and Posters chair: Regis Rossi Faria, *LSI/USP*, Brazil.

Concert chair

Sergio Freire, *UFMG*, Brazil.

Proceedings Editors

Hani C. Yehia, *CEFALA-DELT/UFMG*, Brazil

Hugo de Paula, *PUC Minas*, Brazil.

Technical Papers Program Committee

Adolfo Maia Jr., *Unicamp*, Brazil
 Aluizio Arcela, *UNB*, Brazil
 Andrew Horner, *HKUST*, Hong Kong
 Chris Chafe, *Stanford U.*, USA
 Edilson Ferneda, *UCB*, Brazil
 Eduardo Reck Miranda, *U. Plymouth*, UK
 Emiliós Cambouropoulos, *AUTH*, Greece
 Fabio Kon, *USP*, Brazil
 Flavio S. C. Silva, *USP*, Brazil
 François Déchelle, *IRCAM*, France
 Furio Damiani, *Unicamp*, Brazil
 Geber Ramalho, *UFPE*, Brazil
 Gérard Assayag, *IRCAM*, France
 Hani C. Yehia, *UFMG*, Brazil
 Henkjan Honing, *U. Amsterdam*, Netherlands
 Ian Whalley, *U. Waikato*, New Zealand
 Jürgen Bräuninger, *U. KwaZulu-Natal*, S. Africa

Lelio Camilleri, *U. Firenze*, Italy
 Luis Jure, *U. Republica*, Uruguay
 Marcelo Gomes Queiroz, *USP*, Brazil
 Marcelo Soares Pimenta, *UFRGS*, Brazil
 Marcelo Wanderley, *McGill U.*, Canada
 Márcio Brandão, *UNB*, Brazil
 Matthew Wright, *UC Berkeley*, USA
 Maurício Loureiro, *UFMG*, Brazil
 Oscar Pablo di Liscia, *UNQ*, Argentina
 Palle Dahlstedt, *Chalmers U. Tech.*, Sweden
 Peter Beyls, *Hogeschool Gent*, Belgium
 Petri Toivainen, *U. Jyväskylä*, Finland
 Roger Dannenberg, *CMU*, USA
 Rosa Maria Vicari, *UFRGS*, Brazil
 Sever Tipei, *U. Illinois UC*, USA
 Victor Lazzarini, *NUI*, Ireland

Music Papers Program Committee

Carlos Palombini, *UFMG*, Brazil
 Didier Guigue, *UFPB*, Brazil
 Fernando Iazzetta, *USP*, Brazil
 Garnett Guy, *U. Illinois UC*, USA
 Heinrich K. Taube, *U. Illinois UC*, USA
 Mikhail Malt, *IRCAM*, France

Ricardo dal Farra, *Un. Nacional de Tres de Febrero*, Argentina
 Robert Willey, *University of Louisiana*, USA
 Rodolfo Cesar, *UFRJ*, Brazil
 Sílvio Ferraz, *Unicamp*, Brazil

Table of Contents

Foreword from the Chairmen	III
Foreword from the Program Chairs	IV
SBCM 2005 Conference Committee	VI
Software tutorial	
OpenMusic 5: A Cross-Platform Release of the Computer-Assisted Composition Environment.	1
<i>Jean Bresson, Carlos Agon, Gerard Assayag</i>	
Artificial Intelligence, Music Representation and Evolutionary Music Systems	
A Memetic Approach to the Evolution of Rhythms in a Society of Software Agents	13
<i>Marcelo Gimenes, Eduardo Reck Miranda, Chris Johnson</i>	
Cin:Balada um Laboratório Multiagente de Geração de Ritmos de Percussão	24
<i>Pablo Sampaio, Patrícia Tedesco, Geber Ramalho</i>	
EV Multilevel Music Knowledge Representation and Programming	36
<i>Jesus L. Alvaro, Eduardo R. Miranda, and Beatriz Barros</i>	
Classificação Automática de Gêneros Musicais Utilizando Métodos de Bagging e Boosting	48
<i>Carlos N. Silla Jr. , Celso A. A. Kaestner , Alessandro L. Koerich</i>	
Software Tools for Composition and Sound Synthesis	
Fuzzy Granular Synthesis Controlled by Walsh Functions	58
<i>Adolfo Maia Jr., Eduardo R. Miranda</i>	
Bibliotecas Java Aplicadas a Computação Musical	70
<i>Leandro L. Costalonga, Evandro M. Miletto, Luciano V. Flores, Rosa M. Vicari</i>	
Construindo Protótipos Musicais Cooperativamente na Web	82
<i>Evandro M. Miletto, Marcelo S. Pimenta, Leandro Costalonga, Rosa Vicari</i>	
Self-Organizing Topological Timbral Design Methodology Using a Kohonen Neural Network	94
<i>Marcelo Caetano, César Costa, Jônatas Manzolini, Fernando Von Zuben</i>	

Sound Diffusion Systems, Acoustics and Computational Hearing

AUDIENCE - Audio Immersion Experiences in the CAVERNA Digital	106
<i>Regis Rossi A. Faria, Leandro F. Thomaz, Luciano Soares, Breno T. Santos, Marcelo K. Zuffo, João Antônio Zuffo</i>	
A User-Friendly Graphical System for Room Acoustics Measurement and Analysis	118
<i>Leo Kazuhiro Ueda, Fábio Leão Figueiredo, Fernando Iazzetta, Fabio Kon</i>	
Localização de Fontes e Ouvintes em Salas de Escuta	130
<i>Luciana Dias, Marcelo Queiroz</i>	
Tararira Sistema de búsqueda de música por melodía cantada	142
<i>Ernesto López, Martín Rocamora</i>	

Music Analysis, Psychoacoustics and Cognitive Modeling –

Extracting Patterns from Guitar Accompaniment Data Some Experimental Results	154
<i>Ernesto Trajano de Lima, Søren Tjagvad Madsen, Márcio Dahia, Gerhard Widmer, Geber Ramalho</i>	
Similarity Measures for Rhythmic Sequences	164
<i>João M. Martins, Marcelo Gimenes, Jônatas Manzolli, Adolfo Maia Jr</i>	
Um Sistema Automático de Transcrição Melódica	174
<i>Adriano Mitre e Marcelo Queiroz</i>	
A computational model of harmonic chord recognition	186
<i>Riana Walsh, Dr Donncha O' Maidin</i>	
Modelling Issues for Fine Musical Timbre Characterization	195
<i>Mauricio A. Loureiro, Hugo B. de Paula, Hani C. Yehia</i>	

Music Papers I

“Enquanto eles riem” para Clarinete e computador rodando MAXMSP	207
<i>Cristiano Figueiró, Anselmo Guerra de Almeida</i>	
Implementação da síntese FM em uma linha de atraso variável e suas possíveis aplicações no processamento de áudio em tempo real	219
<i>Sérgio Freire</i>	
Bodyweave ambiente computacional para um encontro poético entre corpo e sonoridades	226
<i>Andrea C. B. Krotoszynski I, Jarbas de Moraes Neto, Jônatas Manzolli</i>	

Music Papers II

gemini flux:Música electrónica, improvisación y código abierto	237
<i>Andrés Cabrera</i>	
Silicon Child	249
<i>Eloi Fernando Fritsch, Rafael de Oliveira, Rodrigo Avellar Muniagurria</i>	
Interação Timbrica na Música Eletroacústica Mista	256
<i>Ignacio de Campos</i>	
Cidade de Gilberto Mendes o toca-discos e gravador como instrumentos musicais	264
<i>Denise Garcia, Clayton Rosa Mamedes</i>	

Short Papers I

New Feature For Automatic Speech Music Discrimination	271
<i>Jayme Garcia Arnal Barbedo, Amauri Lopes</i>	
Sistema Especialista para Detecção do Timbre da Clarineta	275
<i>Leonardo Carneiro de Araújo</i>	
SOAL Ferramenta para análise musical no ambiente Open Music	279
<i>André Lira Rolim, Bruno Jefferson, Didier Guigue</i>	
Waveforms Synthesis by Evolutionary Processes	283
<i>José Fornari, Jônatas Manzolli, Adolfo Maia Jr.</i>	
Funchal a System for Automatic Functional Harmonic Analysis	287
<i>Ricardo Scholz, Vítor Dantas, Geber Ramalho</i>	
Transcrição Automática de Sinais de Áudio Monofônicos	291
<i>Narciso Trevilatto Junior, Jayme Garcia Arnal Barbedo, Amauri Lopes</i>	

Short Papers II

Validating the Lattice Boltzmann Method for the Characterization of Impedances in Pipes	295
<i>Andrey R. da Silva, Gary Scavone</i>	
Um Aplicativo em Max/MSP para Geração de Material Musical Utilizando Técnicas de Síntese Granular	299
<i>Rafael de Oliveira, Luciano Vargas Flores, Eloi Fernando Fritsch</i>	

Rythmus Environment um Ambiente para Construção de Ferramentas Educacionais de Instrumentos de Percussão 303

Suzana Mesquita de Borba Maranhão, Érika Pessoa Araújo, Matheus Cabral de Araújo Gois, Virgínia Barbosa, Geber Lisboa Ramalho

Desenvolvimento e implementação de uma codificação para definir estruturas musicais 307

Pedro Kröger

Poucas Linhas de Ana Cristina de Silvio Ferraz uma breve análise a partir de conceitos do próprio compositor 311

Flávio Ferreira da Silva, Maurício Alves Loureiro

Short Papers III

Impact of Distance in Pitch Class Profile Computation 319

Giordano Cabral, Jean-Pierre Briot, François Pachet

Interações Musicais em Rede 325

Fábio Furlanete, Jonatas Manzolli

Rumo à formalização da Teoria das Árvores Harmônicas 329

Edilson Fernalda, Marcio Brandão, Fernando W. Cruz, Fernando S. Goulart Jr, Luciênio de M. Teixeira, Karen P. de Sousa, Marcio G. V. de Souza, Paullus M. de S. N. Castro, Maria Carolina F. da Silva

Parsing Incremental para Acompanhamento em Tempo Real 333

Giordano Cabral, Jean-Pierre Briot, François Pachet

Sistemas Dinâmicos Não-Lineares e Organicidade no Material Musical 338

Daniel Fils Puig

Poster Session

A brief history of auditory models 346

Leonardo C. Araújo, Tairone N. Magalhaes, Damares P. M. Souza, Hani C. Yehia, Maurício A. Loureiro

Revisitando Waveshaping implementando um plugin VST para distorcer sons de guitarra 352

André Luiz Luvizotto, Ricardo Ichizo, Jônatas Manzolli

Classification of Triadic Chord Inversions Using Kohonen Self-organizing Maps 355

Luis Felipe de Oliveira, Luis Guilherme Pereira Lima, André Luiz Gonçalves de Oliveira, Rael Bertarelli Gimenes Toffolo

Estudo e Implementação de Métodos de Medição de Resposta Impulsiva em Salas de Pequeno Porte	364
<i>Bruno Sanches Masiero, Fernando Iazzetta</i>	
Estudo Experimental da Sonoridade Chalumeau da Clarineta através de Projeto Fatorial	370
<i>Luís Carlos Oliveira, Ricardo Goldemberg, Jônatas Manzolli</i>	
Simulação de Performances de Violão por Agentes Artificiais	374
<i>Leandro L. Costalonga, Luciano V. Flores, Evandro M. Miletto, Rosa M. Vicari</i>	
Index of authors	380

OpenMusic 5: A Cross-Platform Release of the Computer-Assisted Composition Environment.

Jean Bresson, Carlos Agon, Gerard Assayag

Musical Representations Team

IRCAM – Centre G. Pompidou – 1, place I. Stravinsky – 75004 Paris – France

{bresson,agon,assayag}@ircam.fr

Abstract. *This paper presents the computer-assisted composition environment OpenMusic and introduces OM 5.0, a new cross-platform release. The characteristics of this system will be exposed, with examples of applications in music composition and analysis.*

Resumo. *Este artigo apresenta o ambiente de composição assistida por computador OpenMusic, e introduz OM 5.0, uma nova versão do programa, compatível Mac e PC. Serão expostas também algumas características do sistema tendo em vista aplicações para a composição e análise musical.*

1. Introduction

The Ircam's Computer-Aided Composition (CAC) project aims at connecting formal computing tools and musical thought, in order to provide the composers with programming languages adapted to their specific needs, and to allow them to formalize, develop, and experiment their musical ideas [Assayag 1998]. Our experience demonstrates that the definition of computing models corresponding to musical situations can lead to a rich creative and analytic approach of composing processes.

In this framework, several environments have been developed, around the core concept of programming languages, facilitating the formulation of musical concepts: Formes (an object oriented environment in Lisp for high level control of sound synthesis, [Rodet and Cointe 1984]), PreForm/Esquisse (an extension of Formes with graphical interfaces, providing multidimensional objects and functions to manipulate them), Crime (a CAC environment providing musical abstractions and symbolic score descriptions, [Assayag et al. 1985]), PatchWork and OpenMusic [Agon et al. 1999]. These languages have been enriched and extended with graphical interfaces, along with the evolution of computer science, allowing the introduction of visual programming.

OpenMusic is an example of a complete programming language dedicated to music composition. In this article, we present an overview of this programming/composition environment (section 2) and some examples of applications (section 3). In section 4 we introduce OM 5, the new cross-platform release of OpenMusic.

2. The OpenMusic visual programming language

2.1. A visual language based on Lisp

PatchWork [Laurson and Duthen 1989] is a graphical interface of the Common Lisp programming language, in which programs are developed graphically by means of functional boxes and connections. In this environment, any Lisp function can be represented as a graphical box, and connected to other boxes to constitute a program. This graphical program is a connected acyclic graph which corresponds to a Lisp form. The data structures resulting from the functional boxes can be retained and associated with graphical editors, which allows their visualization and edition in a musical notation.

OpenMusic [Agon 1998] was designed and developed in CLOS (Common Lisp Object System [Steele 1998]) on top of the PatchWork model. It is now a powerful visual language, allowing various programming models and applications.

2.2. Functional programming

The basis of OpenMusic, inherited from PatchWork, is purely functional. Functional programming is an intuitive approach that has been widely used by the composers. It allows to define operations that are applied on multidimensional data structures, in order to create new ones.

In OpenMusic, a *patch* is a visual algorithm, in which boxes represent functional calls, and connections are functional compositions. The evaluation of a box causes a sequence of successive evaluations, corresponding to the execution of the program.

The visual language provides a set of graphical control structures, such as loops and conditional controls, as well as the possibility to manipulate programming concepts like abstraction or recursion. Through functional abstraction, some elements of the program can become variables, which leads to the definition of functional objects. These objects can then be embedded into other programs or abstractions (see Figure 1).

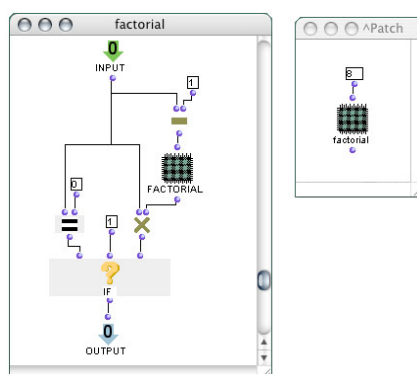


Figure 1. Functional programming in OpenMusic: the example of the "factorial" algorithm illustrates some of the language features : conditional controls, abstraction, recursion. To create an abstraction, the user adds inputs and outputs to the patch, which will then appear on the patch abstraction box. This abstraction can then be applied in another patch or in the patch itself.

2.3. Object-Oriented programming

Built on the Common Lisp Object System, the visual language is an object-oriented environment enriched with classes and generic functions [Agon 2003].

Some basic classes are provided by the environment to represent musical structures (notes, chords, sounds, break-point functions, etc.) These classes can be used in the development of the graphical programs through the use of *factory* boxes. A factory is a special box which inlets correspond to the public slots of the corresponding class. Its evaluation creates and stores an instance of this class (see Figure 2).

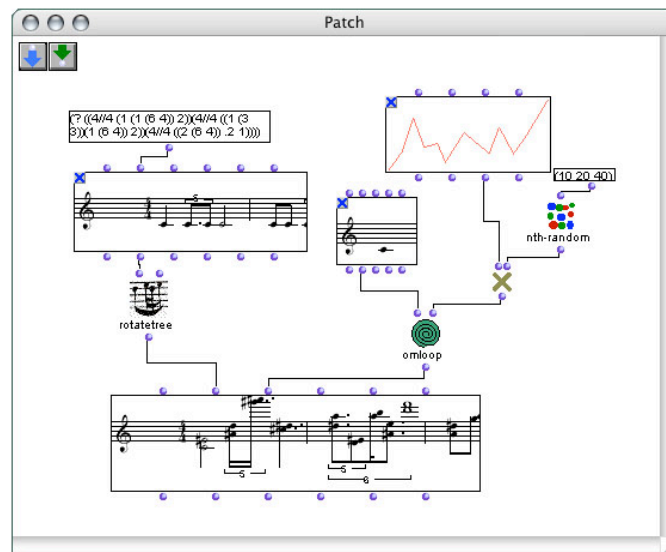


Figure 2. A patch using some of the OM classes. In this example, a *voice* instance is created by computing a rhythmic pattern and a melodic profile.

The user can also create his/her own classes, using slot definition tools, and by setting inheritance relationships. The OpenMusic classes can thus be extended with user defined subclasses (Figure 3-a). The polymorphism feature of CLOS is integrated in the language through the possibility to create generic functions and methods (Figure 3-b).

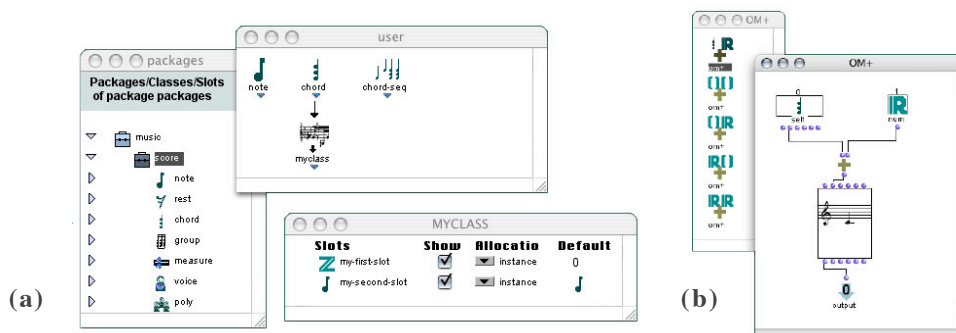


Figure 3. Object-Oriented programming - (a) Inheritance: definition of a class extending an OM class - (b) Polymorphism: definition of a new method for the *om+* generic function.

The object-oriented language is extended with a graphical MOP (Meta-Object Programming) [Agon and Assayag 2003], allowing one to interact with the language elements and providing access to the visual components' properties and behavior.

2.4. Constraint programming

Constraint programming aroused a great interest among composers and computer music researchers. In OpenMusic, composers can graphically define a constraint satisfaction problem (CSP), and try to solve it using different constraint resolution systems: Situation [Rueda and Bonnet 1998], Screamer [Siskind and McAllester 1993], or OMClouds [Truchet et al. 2003]).

2.5. Music representation and notation

The representation of musical structures is a concrete expression of the information transmitted across the system. In OpenMusic this representation can be audibly or visually rendered with audio and MIDI players (the MIDI manipulations and renderings are implemented using the MidiShare system [Orlarey and Lequay 1989]) or with graphical editors associated to the main OM classes (musical structures, break-point functions, sounds, etc.) Musical notation editors provide the user with interactive editing and navigation into the hierarchical structure of the score (see Figure 4).

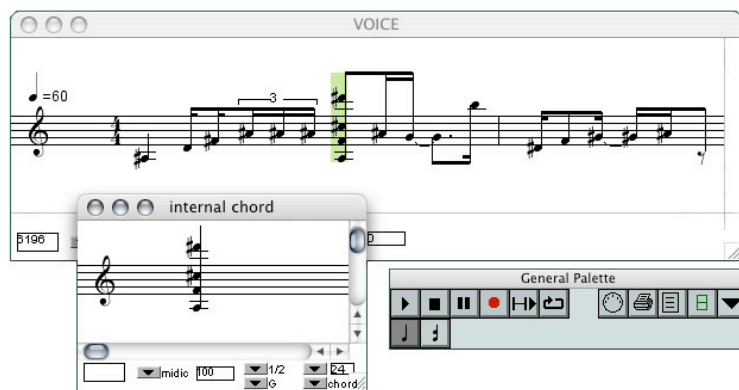


Figure 4. OpenMusic score editor.

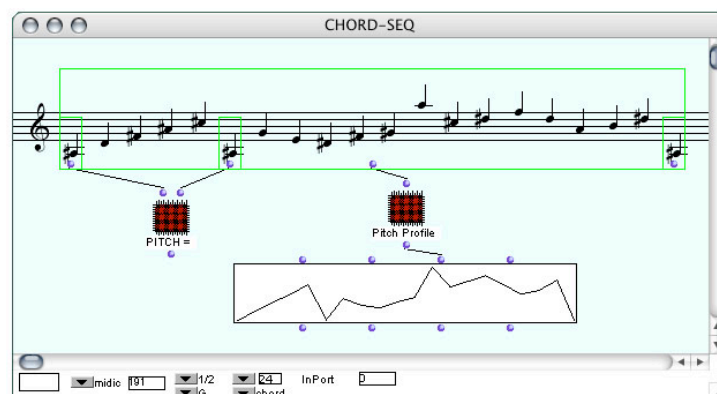


Figure 5. Programming inside the score editor.

The integration of musical editors with visual programs can be done in different ways:

- By including the editors in patches; as a way to control the value of the components of a computation tree (as in Figure 2).
- By including visual programs within the score editor itself; by defining relationships between different sub-structures of a score (Figure 5).

2.6. Temporal aspects: the *Maquette*

The *maquette* [Agon and Assayag 2002] is an original concept designed to unify both the program and the score in a common representation. A *maquette* is a bi-dimensional space containing blocks called "temporal boxes" (see Figure 6). Such boxes can contain either :

- simple musical objects (instances of the OM musical classes),
- temporal patches (patches having an output in the temporal context),
- embedded *maquettes* (for constructing hierarchical temporal structures).

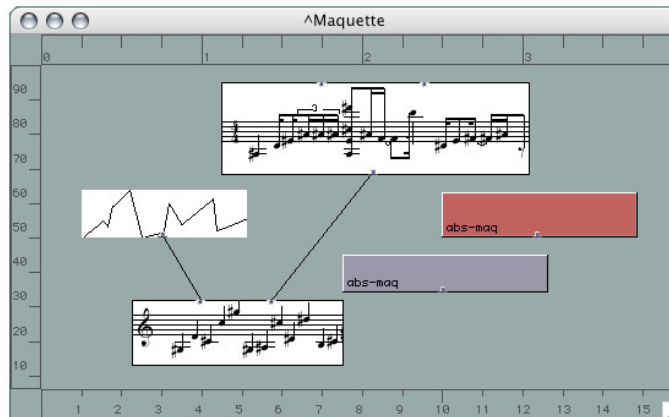


Figure 6. A *maquette*.

In the *maquette*, the horizontal axis represents time, so that the position and horizontal extension of the temporal boxes can be associated with offsets and duration in the *maquette*'s time referential. The MOP allows one to access the graphical characteristics of a temporal box and to put it in relation to its contents. For example, the vertical position and extent, can be assigned to any musical (frequency, intensity) or functional (an input for the programs execution) parameter.

Boxes can have inputs and outputs and be connected in the *maquette*. It allows functional relationships between the temporal objects, revealing further types of musical semantics. A *maquette* can then be evaluated as a graphical program which takes into account the temporal properties of its elements.

Eventually, the computed *maquette* can be executed (i.e. played) by MIDI and audio players.

3. OpenMusic Applications

3.1. Models for musical formalisms: examples of compositional applications

OpenMusic has been used in a large panel of applications and musical pieces. Many projects and toolboxes, dedicated to special musical purposes and specific formal systems, have been developed. Some of these projects are available in the software release as "user libraries".

The *Esquisse* library, for example, was created by Tristan Murail for his works on harmonic structures. *OMChaos* and *OMAlea* were developed by Mikhail Malt to build harmonic materials with stochastic processes and non-linear models [Malt 1994]. They were used for the composition of *Lambda 3.99* and *Actrinou*. The *Profile* and *Morphologie* libraries were used by Jacopo Baboni-Schilingi for the creation and manipulation of melodic profiles.

Rhythmic issues (quantification, rhythmic manipulation, rhythmic canons construction) also inspired the work of many composers. For instance, Karim Haddad manipulates the OpenMusic rhythmic trees representation [Agon et al. 2002] to create complex rhythmic structures with the *OMTrees* library (e.g. in *...und wozu Dichter in durftiger Zeit?...*).

Various musical applications of constraint programming have also been developed in OpenMusic, for example by Örjan Sandred with the *OMRC* library which is specialized for finding structures corresponding to rhythmic constraints (used for example in the piece *Kalejdoskop*.) The OM constraint systems were also used with harmonic constraints (e.g. by Antoine Bonnet in *Epitaphe*, with the *Situation* constraint solver).

3.2. Control of sound synthesis and writing electronic music

The development of Digital Signal Processing technologies brought to music creation a new field of exploration, and extended compositional activities to the composition of the sound itself [Risset 2002]. The connection between musical signals, and symbolic objects and concepts became an interesting challenge for both musicians and researchers. This problem is one of our current research axes in OpenMusic. We consider it from a variety of angles.

The importation of sound description data in the compositional environment allows the composer to use musical material coming from real sounds, by transforming it into symbolic entities. Sound-related material can be imported either as *sound* boxes (from which data can be extracted as simple sound samples or analysed to produce sound analysis data), or as *SDIF* boxes. The SDIF interface [Bresson and Agon 2004] allows one to import SDIF sound description data [Wright et al. 1998] coming from an analysis processed by external tools and softwares, and to convert them into symbolic musical structures (see Figure 7-a).

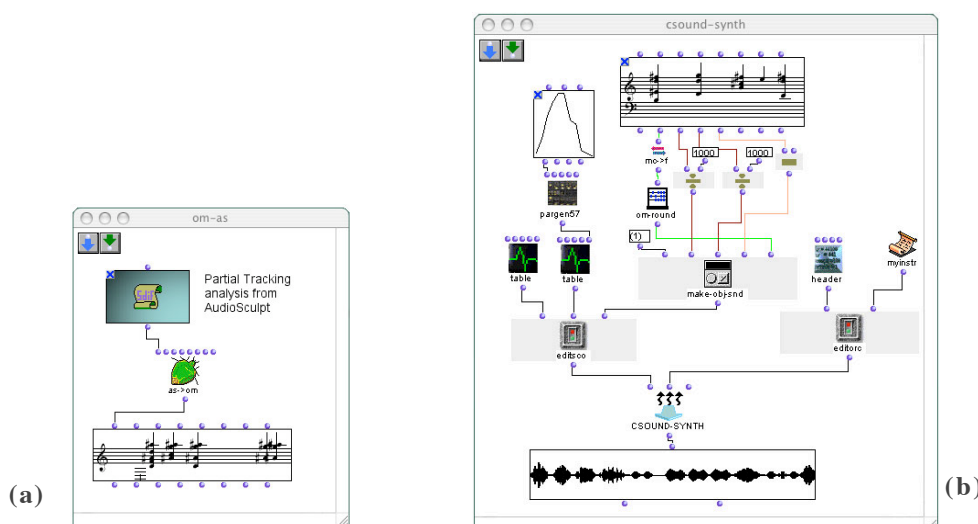


Figure 7. Sound analysis/synthesis in OM - (a) An importation of partial tracking analysis data and conversion into a polyphonic musical sequence [Hannape 1995] - (b) Sound synthesis with CSound using symbolic data and operations.

The control of sound synthesis processes in OM allows one to take advantage of the computational and expressive power of its visual language for creating and manipulating sound synthesis parameter data. OM then generally delegates the sound synthesis processing to external synthesizers, either by formatting parameters files, or

through direct interfaces. Some of these interfaces are associated with dedicated libraries: OM2CSound for the graphical design of CSound instruments and scores [Boulanger 2000] (see Figure 7-b), OM-AS for creating parameter files for the SuperVP phase vocoder [Depalle and Poirot 1991], OMModalys for creating Modalys physical synthesis patches [Eckel et al. 1995]. These tools have been used in the creation of several electro-acoustic pieces of Karim Haddad, Hans Tutschku, Mauro Lanza, and more.

OMChroma is an other original approach to high level control of sound synthesis based on Marco Stroppa's *Chroma* system [Agon et al. 2000]. Using object-oriented programming techniques, this system allows a powerful instantiation of classes representing complex data sets which can be sent to several external synthesizers. OMChroma is used in several compositions by Marco Stroppa (e.g. *Come Natura di Foglia*, etc.)

Sound spatialization can also be involved in computer-assisted composition. OpenMusic provides special classes for designing three dimensional curves and trajectories. This spatialization data can be sent to the Ircam's *Spatialisateur* [Jot and Warusfel 1995] through the *OMSpat* library (e.g. Brian Ferneyhough, in *Stelae for the failed times*).

3.3. Musical analysis

OpenMusic is used in musicology as a support for experiments on formal and mathematical models of music. The reconstitution of musical formalisms and musical pieces in this environment provides an intuitive and interactive approach of music analysis. Interesting models of works by Iannis Xenakis, for example, (*Achorripsis*, *Herma*, *Nomos Alpha*), could be recreated [Agon et al. 2004] (see Figure 8).

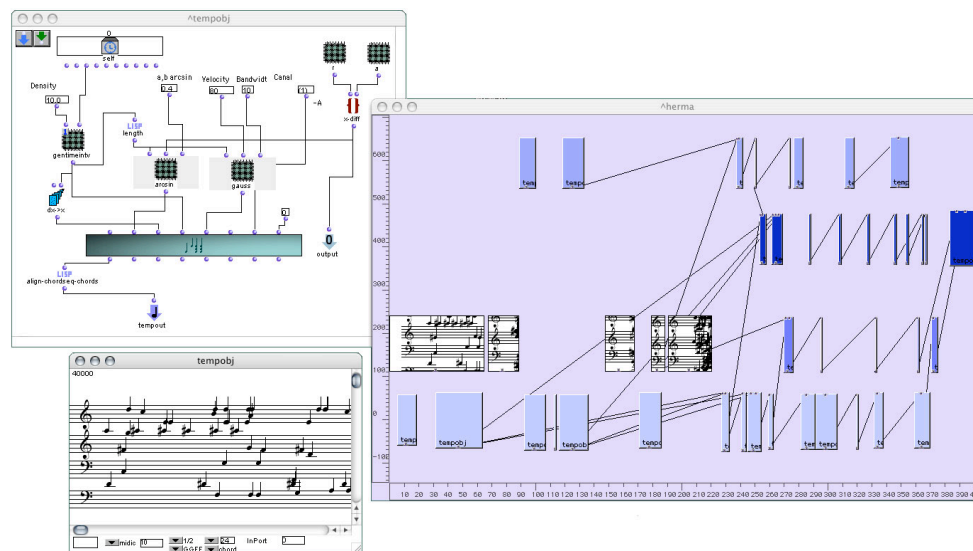


Figure 8. *Herma* (I. Xenakis). The composition's structure is represented in a *maquette*: the pitch set operations and probabilistic rules are implemented by means of graphical programs and connections between blocks. Each box can be considered as a graphical program (top-left window) in the computational flow, or as a musical object in time (bottom-left window).

OpenMusic is also an experimental environment for various types of research on harmonic analysis, pattern recognition, neural networks, and musical style analysis and simulations, in several institutions and universities.

4. OpenMusic 5

4.1. A cross-platform environment

The OpenMusic 5.0 release contains major internal changes. In this new version, the code has been reorganized and divided in two distinct parts: the *API* code, and the *OM* code.

The *API* (Application Programming Interface) code contains low level primitives and structures which have been identified as being dependent on the System and/or on the Lisp implementation. These related primitives are generally non Common Lisp code (previously specific to the Macintosh platform). They can be grouped into several categories: windows, frames and graphic objects; dialog items and menus; graphical primitive structures; drawing tools; images and icons; user interactions (keyboard, mouse), drag&drop, and events handling; meta-objects programming tools; system tools and communication with external libraries. OpenMusic was historically developed on Digitool's Macintosh Common Lisp (MCL), so this new API has been specified following the MCL model in order to maintain the original OpenMusic code structure and functional system.

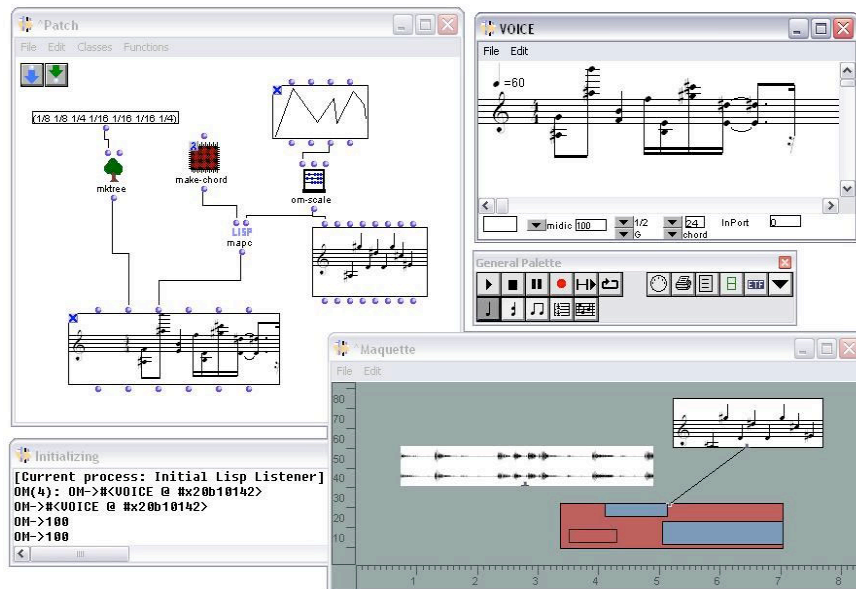


Figure 9. OpenMusic on Windows XP.

The *OM* code contains the functional kernel of OpenMusic (the visual programming language), and some "projects" (code modules dedicated to specific applications : music, sound synthesis, musical analysis, constraint programming, etc.) This part represents the core of OpenMusic, and has been rewritten following the API specification.

The API now allows an abstraction of the system-related constraints while writing OpenMusic code. This code is thus interpreted for a target platform or Lisp implementation, providing the API is implemented in it. This is the case for Macintosh (with MCL) and Windows (with Allegro Common Lisp) (see Figure 9). A Linux version would need the adaptation of the API on this platform; preliminary trials have been done using SBCL and GTK.

Patches from older versions of OM can still be loaded and are automatically converted, when saved, following the new specifications. Patches can also be transferred between the Mac and PC versions.

4.2. Examples of new features in OM 5.0

The OM 5 audio library, based on GRAME's *LibAudioStream* project, extends the audio support in OpenMusic. Advanced audio stream processing operations (mixing, sequencing, effects, etc.) are available. The sound objects can now be assigned to an audio channel, so that audio tracks can be simulated and controlled using a graphical mixing console (see Figure 10).

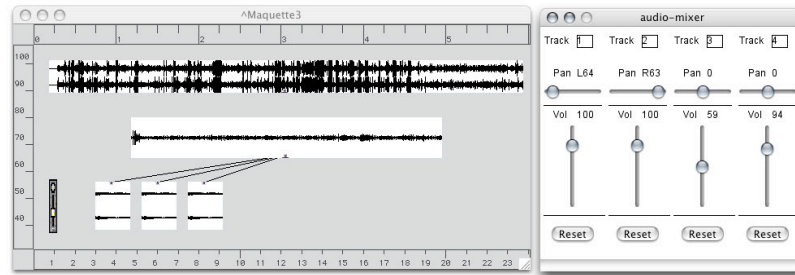


Figure 10. Interactive control of audio objects.

The different available tools for sound analysis and synthesis have been gathered around direct external interfaces (CSound, SuperVP, OSC, etc.) and a set of shared globals variables and preferences. The SDIF toolbox is also enriched with new classes and functions allowing one to store and write structured SDIF data within patches (see Figure 11.)

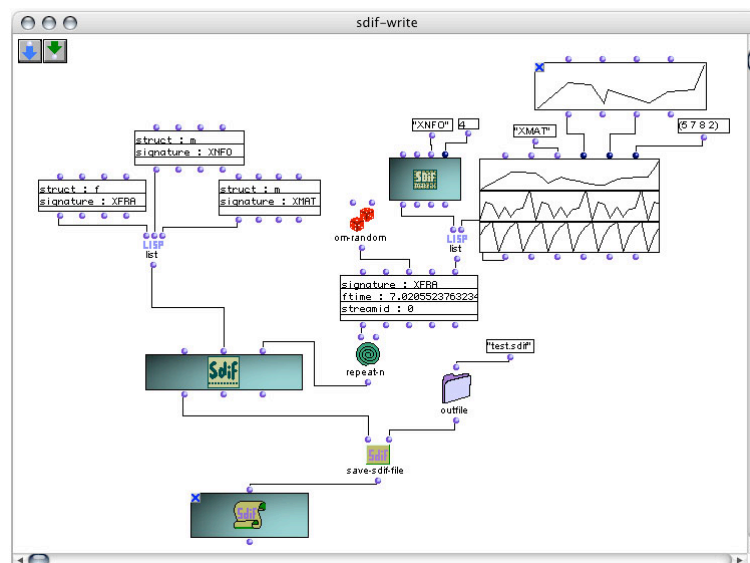


Figure 11. Programming with SDIF data structures.

Research in mathematical music theory carried out with OpenMusic (set theory, classifications, canons, etc.) has been implemented in a new set of mathematical tools.

The OM musical classes have also been extended with a tonality model. By following the hierarchical architecture of the musical objects, this tonal model contains knowledge about relative tonality, which can be used for musical notation in score editors (see Figure 12), or to compute properties and operations in the tonal field (tonal transpositions, etc.)

Microtonal notation is now available in the score editors. Micro-intervals up to 16th of a tone can be manipulated and displayed (Figure 12).

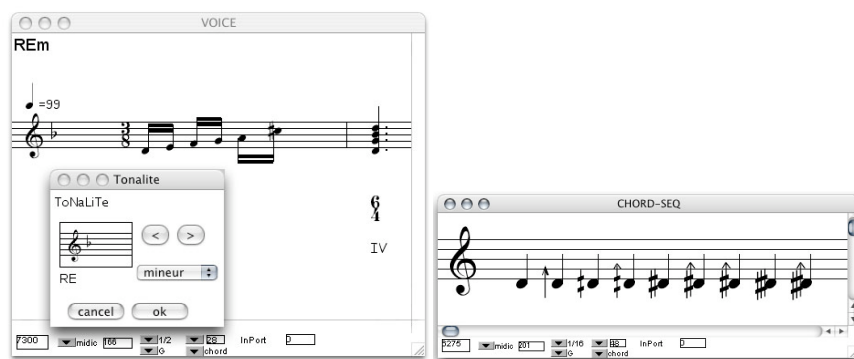


Figure 12. Representation of tonality and microtonality in score editors.

The *maquette*'s flexibility was also improved for example with the possibility to set functional relationships through the hierarchical levels. Current developments are oriented towards a better integration of the programs in the *maquette*.

5. Conclusion

OpenMusic has proven to be a practical and innovative tool for music composition, allowing musical material and musical processes to coexist in the same representation.

Many composers have been using OpenMusic, each with his/her specific tools and methodologies. The *OM Composer's Book* [Agon et al. 2005], to be published in 2005-2006, is a collection of user experiences in which each chapter, written by a composer, describes his/her usage of OpenMusic for composition.

OpenMusic is also used in musical research, and is taught in composition classes in several universities and conservatories (Columbia University, Harvard, Stanford, Musikhochschule Stuttgart, Berkeley, CNSM Paris, etc.)

OM 5.0 constitutes a new step in the development of OpenMusic, and may promote its diffusion with the extension to new platforms.

References

- Agon, C. (1998) "OpenMusic: Un Langage Visuel pour la Composition Assistée par Ordinateur", PhD. Thesis, Université Paris VI.
- Agon, C., Assayag, G., Laurson, M. and Rueda, C. (1999) "Computer Assisted Composition at Ircam: PatchWork & OpenMusic", *Computer Music Journal* 23(5).
- Agon, C., Stroppa, M. and Assayag, G. (2000) "High Level Musical Control of Sound Synthesis in OpenMusic", *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2000.
- Agon, C. and Assayag, G. (2002) "Programmation Visuelle et Editeurs Musicaux pour la Composition Assistée par Ordinateur", *IHM'02*, Poitiers, France, ACM Computer Press.

- Agon, C., Haddad, K. and Assayag, G. (2002) "Representation and Rendering of Rhythmic Structures", WedelMusic Darmstadt, IEEE Computer Press.
- Agon, C. (2003) "Object-Oriented Programming in OpenMusic", in *Topos of Music*, Mazzola G., Birkhäuser Verlag Ed.
- Agon, C. and Assayag, G. (2003) "OM: A Graphical Extension of CLOS using the MOP", *Proceedings ICL '03*, New York.
- Agon, C., Andreatta, M., Assayag, G. and Schaub, S. (2004) "Formal Aspects of Iannis Xenakis' "Symbolic Music": A Computer-Aided Exploration of Compositional Processes", *Journal of New Music Research*, 33(2).
- Agon, C., Assayag, G. and Bresson, J. (ed.) (2005) "The OM Book", Delatour Editions, to be published.
- Assayag, G., Castellengo, M. and Malherbe, C. (1985) "Functional Integration of Complex Instrumental Sounds in Music Writing", *Proceedings of the International Computer Music Conference*, Burnaby, Canada, 1985.
- Assayag, G. (1998) "Computer Assisted Composition Today", *1st Symposium on Music and Computers*, Corfu, 1998.
- Boulanger, R. (ed.) (2000) "The Csound Book", MIT Press.
- Bresson, J. and Agon, C. (2004) "SDIF Sound Description Data Representation and Manipulation in Computer Assisted Composition", *Proceedings of the International Computer Music Conference*, Miami, USA, 2004.
- Depalle, Ph. And Poirot, G. (1991) "A Modular System for Analysis, Processing and Synthesis of Sound Signals", *Proceedings of the International Computer Music Conference*, Montreal, Canada, 1991.
- Eckel, G., Iovino, F. and Caussé, R. (1995) "Sound Synthesis by Physical Modelling with Modalys", *Proceedings of the International Symposium on Music Acoustics*, Dourdan, France.
- Hannape, P. (1995) "Integration des représentations temps/fréquence et des représentations musicales symboliques", in "Recherches et applications en informatique musicale", M. Chemillier and F. Pachet (ed.), 1995.
- Jot, J.-M. and Warusfel, O. (1995) "A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications", *Proceedings of the International Computer Music Conference*, Banff, Canada, 1995.
- Laurson, M. and Duthen, J. (1989) "Patchwork, a Graphic Language in PreForm", *Proceedings of the International Computer Music Conference*, Ohio State University, USA, 1989.
- Malt, M. (1994) "Modelos Matemáticos e composição Assistida por Computador, Sistemas Estocásticos e Sistemas Caóticos", in *Primeiro Simposio de Computação e Música*, Caxambu, MG, Brazil
- Orlarey, Y. and Lequay, H. (1989) "MidiShare", *Proceedings of the International Computer Music Conference*, Columbus, USA, 1989.
- Risset, J.C. (2002) "Computing Musical Sounds", in Assayag et al. (ed.) "Mathematics and Music", Springer, 2002.
- Rodet, X. and Cointe, P. (1984) "Formes: Composition and Scheduling of Processes", *Computer Music Journal* Vol. 8(3).
- Siskind, J. M. and McAllester, D. (1993) "Nondeterministic Lisp as a Substrait for Constraint Logic Programming", *AAAI-93*, pp133-138.

- Steele, G. L. (1998) "Common LISP The language, second edition", Digital Press, USA.
- Truchet, C., Assayag, G. and Codognet, Ph (2003) "OMClouds, petits nuages de contraintes dans OpenMusic", Actes des Journées d'Informatique Musicale 2003, Montbéliard, France.
- Wright, M., Chaudhary, A., Freed, A., Wessel, D., Rodet, X., Virolle, D., Woehrmann, R. and Serra, X. (1998) "New applications of the Sound Description Interchange Format", Proceedings of the International Computer Music Conference, Ann Arbor, USA, 1998.

A Memetic Approach to the Evolution of Rhythms in a Society of Software Agents

Marcelo Gimenes, Eduardo Reck Miranda, Chris Johnson

Computer Music Research, School of Computing, Communications and Electronics
University of Plymouth, UK

{marcelo.gimenes, eduardo.miranda, c.johnson}@plymouth.ac.uk

Abstract. *We are developing RGeme (Rhythmic Meme Generator), an artificial intelligence system for the composition of rhythmic streams inspired by Richard Dawkin's theory of memes. The system is based on intelligent agents that learn from examples and interact by generating rhythms. The system has two broad stages. In the first one, the learning stage, which is the main focus of this paper, Agents are trained with examples of musical pieces in order to evolve a "musical worldview" which consists of a "Style Matrix" of basic rhythmic elements (or "rhythmic memes"). In the next (production) stage Agents are able to learn from each other's "compositions" and capable of evolving new rhythmic styles by adapting to each other's rhythms.*

1. Introduction

Computers have long been used for aiding musical composition in a number of possible ways. Some composers ([Cope 1991], [Dodge 1985], [Worral 2001] and [Xenakis 1971]) use mathematical models such as combinatorial systems, grammars, probabilities and fractals to create new pieces of music. Other systems apply standard Genetic Algorithm procedures for evolving musical materials such as melodies, rhythms, chords, and so on. One such example is Vox Populi [Manzollini 2000], which evolves populations of chords of four notes through the operations of crossover and mutation.

Evolutionary Computation models are also being used in many models. In one of them, CAMUS [Miranda 1993], the emergent behaviour of Cellular Automata (CA) is used to generate musical compositions in which case the co-ordinates of the cells are associated with the distances between the notes of a set of three musical notes.

Impett (2001) uses an Agent system to generate musical compositions. Through the interaction of embodied behaviours that co-exist and interact in the same world, Agents are adaptive to the changing environment to which they belong.

A growing number of researchers are developing computer models to study cultural evolution, including musical evolution [Blackmore 1999]. For instance, Miranda (1999) investigates how musical structures can originate and evolve in artificially created environments and inhabited by virtual communities of musicians and listeners.

Some rhythmic generating systems have already been proposed [Horowitz 1994]. Pachet (2000) describes an evolutionary model where a group of agents play rhythms together in real time without prior knowledge about the music to play. Agents

play in cycles to which transformation rules are applied in order to produce new variations.

The system that we present in this paper, RGeme, as described in the next Section, makes use of some of the previous experiences, in addition to concepts inspired on the theory of memes by Dawkins (1991). According to Dawkins, memes are basic units of cultural transmission in the same way that genes, in biology, are units of genetic information. The initial argument of this theory references musical aspects:

"Examples of memes are tunes, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperm and eggs, so memes propagate in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation."([Dawkins 1989], p. 206)

Cox (2001) asserts that the "memetic hypothesis" is based on the concept that the understanding that someone has on sounds comes from the comparison with the sounds already produced by this person. The process of comparison involves tacit imitation, or memetic participation that is based on the previous personal experience on the production of the sound.

Gabora (1997) explains that, in the same way that information patterns evolve through biological processes, mental representations, or memes, evolve through the adaptive exploration and transformation of an informational space through variation, selection and transmission. Our minds perform tasks on its replication through an aptitude landscape that reflects internal movements and a worldview that is continuously being updated through the renovation of memes.

Our aim with this research is to contribute to this trend by means of computational modelling of a memetic environment for the generation of rhythmic streams.

2. The Model

RGeme is an artificial intelligence system for the composition of rhythmic passages that uses the computational framework of intelligent Agents. Maes (1991) asserts that autonomous Agents are computer systems that inhabit a dynamic and complex environment, sense and act autonomously in this environment executing a series of goals and tasks for which they were devised.

These computational entities are designed to have the ability to perceive and to act in their environment in order to achieve certain targets [Russel 2002]. In our system, Agents are able to look for the existence of music compositions and to choose the ones with which they will interact; later on Agents parse and extract the rhythmic information. Conversely, Agents are also able to actuate in the system through the generation of new rhythmic streams.

At the beginning of a simulation a number of Agents are created. They are given an identity (name), a number of tasks ("Goal Matrix") and the criteria ("Evaluation Matrix") they will apply to choose the compositions with which they will interact ("Candidate Compositions").

Three types of agent tasks are envisioned (listen, practice and compose) to be accomplished during three different stages (Listener, Student, Composer) to which the Agents will belong during their lifetime. As Listeners, Agents can only execute listening tasks. In the Student stage, Agents can listen to and practice rhythms. In the last stage, as Composers, Agents can execute listening, practicing and composition tasks. Broadly speaking these stages and tasks split the model into two general concepts: the learning and the production phases. Obviously, listening and practicing tasks focus mainly on the learning phase whereas composition tasks focus mainly on the production phase.

Before the execution of listening and practicing tasks, the Agents choose the Candidate Music according to the Evaluation Matrix (composer's name and/or year of composition). An Evaluation Matrix can determine the same rules for the Agent's entire lifetime or can establish different ones according to the stage in which the Agent is at a specific moment. This last possibility will be employed in the simulation described in the next Section.

The basic difference between listening and practicing tasks is in the number of times an Agent parses the Candidate Music and adapts its internal states to the contents that it finds in the music. This feature is controlled by a 'number of rehearsals' variable, which is always 1 for the listening task and can be more than 1 for the practicing task.

Once the Candidate Music is chosen, Agents parse it in order to extract rhythmic memes (Candidate Memes). In the real world, the definition of the exact length/boundaries of a musical meme is a very complex subject for a number of reasons [Jan 2000 and 2004]. Roughly speaking, different individuals can identify different memes in the same or in different pieces of music in accordance with, among other factors, their previous personal musical background. Our model, however, was designed to produce musical material in artificially inhabited environments, although it has many features that were inspired in real life situations. Therefore, in order to keep it reasonably simple in the first steps of implementation, currently each rhythmic meme has a fixed length that corresponds to a music bar.

Agents store their musical knowledge in a Style Matrix in which every entry is related to a unique rhythmic structure (rhythmic meme) with the following information:

- the dates (represented in terms of a counter that calculates each interaction cycle) in which the memes were first and last listened to,
- the number of times the memes were listened to,
- the weight (importance) the memes hold due to the various interactions with the Candidate Memes and
- the Candidate Music the meme was listened from.

Style Matrices also hold 'Composition Maps', which correspond to the ways the Candidate Memes are interconnected in the Candidate Compositions.

RGeme represents rhythmic memes coded as vectors whose entries are 0s and 1s (Figure 1), where 1 means the trigger of sounds and 0s are used to represent rests and as time placeholders.

Meme	dFL	dLL	nL	W
01011101	1	1	6	1.038
11011101	1	1	31	1.042
10001000	1	1	1	1.018
10010101	1	1	1	1.017
11011010	1	1	1	1.016
10011010	1	1	4	1.009
10011001	1	1	4	1.007
11111111	1	1	1	1.004
10000000	1	1	1	1.000

- dFL: date of first listening
- dLL: date of last listening
- nL: number of listening
- W: weight

Table 1: Extract from 1st Style Matrix

Table 2 shows the corresponding data in the Style Matrix after Agent ‘A’ listened to the second music (‘Matuto’, by same composer):

Meme	dFL	dLL	nL	W
01011101	1	2	7	1.070
11011101	1	2	37	1.079
10001000	1	1	1	1.024
10010101	1	2	21	1.059
11011010	1	2	2	1.040
10011010	1	1	4	1.016
10011001	1	2	10	1.044
11111111	1	1	1	1.011
10000000	1	2	2	1.027
00010101	2	2	1	1.035
10100101	2	2	1	1.025
11011111	2	2	2	1.022
10010111	2	2	4	1.024
10011111	2	2	2	1.021
11011000	2	2	1	1.009
10000101	2	2	2	1.009
11010101	2	2	1	1.007

Table 2: Extract from 2nd Style Matrix

Notice, for example, that:

- After the first cycle of interaction (Style Matrix 1 or SM1), meme 11011101 (second in the list) had been listened (nL) 31 times and its weight (W) was 1.042. After the second cycle of interactions (Style Matrix 2 or SM2) its number of listening was 37 and its weight had been raised to 1.079.
- In SM1 meme 11111111 (8th in the list) was listened only 1 time and its weight had been set to 1.004. In SM2, although the number of listening had been kept the same, the weight had been raised to 1.011. The additional weighting was due to the similarities that this meme had in comparison with the other ones that were listened to after it first appeared in any Style Matrix. Also, no ‘forgetting’ effect was applied because it was listened in two consecutive interaction cycles.

- Meme 11010101 (last in Table 2) only appears in SM2 and its weight was set to 1.007, which means that, after its first appearance (in which the weight had been set to 1), the weight raised due to the comparisons made with the other memes that were listened to afterwards.

Up to this point we described how the transformation algorithm alters the musical knowledge possessed by the Agents as a result of the execution of the listening and practicing activities.

Since the learning phase is the main focus of this paper, it suffices to say that in the production phase the Agents execute composition tasks mainly through the reassignment of the various Composition Maps according to the information previously stored in the learning phase. Composition tasks, beyond the production of new material, also have a transformation effect on the Style Matrix where all memes are updated according to the musical material used in the newly produced rhythms.

The model has the potential to execute intricate simulations with several Agents learning at the same time from rhythms by composers from inside and outside the system's environment.

In the next Section we present an experiment where some of the features of the system are evaluated and better explained.

3. A case study

A group of 20 pieces by Brazilian composers Ernesto Nazareth and Tom Jobim was selected. Nazareth embodies the 'chorinho' style, characteristic of the beginning of the twentieth century in Brazil. Jobim personifies the 'bossa nova', a rhythm that emerged as a softened variation of the 'samba' in the fifties.

RGeme was configured to create only one Agent (Agent 'A') to which a series of 100 tasks was given as shown in Table 3:

	Listen	Practice	Compose
Listener	50	n/a	n/a
Student	25	25	n/a
Composer	0	0	0

Table 3. Agent's 'A' Goal Matrix

The Evaluation Matrix for the first 50 tasks established the choice of only Nazareth's works. During the following 50 tasks only Jobim works should be chosen, as shown in Table 4:

	Composer	Year begin	Year end
Listener	Nazareth	-	-
Student	Jobim	-	-
Composer	n/a	n/a	n/a

Table 4. Agent's 'A' Evaluation Matrix

"Year begin" and "Year end" can usually be employed in an Evaluation Matrix to define a date interval. In this case they were not specified which means that the

algorithm returned all the compositions by Nazareth (in the Listener stage) and Jobim (in the Student stage). In each cycle Agent ‘A’ performed a task consisting of: one Candidate piece of Music was chosen, the Candidate Memes were parsed and the Style Matrix was transformed according with the transformation algorithm. The system generated a new Style Matrix after the accomplishment of each task and all the resulting Style Matrices were logged in the system in order to observe the behaviour of each meme during the interaction processes.

After the completion of the simulation, we observed that, during the first 50 tasks, Agent ‘A’ learned a total of 57 memes from the music by Nazareth. In the second half of the simulation, 76 memes were learned, which indicates that 19 new memes originated from the music by Jobim. Figure 2 shows the evolution in time of the number of memes that were learned by Agent ‘A’.

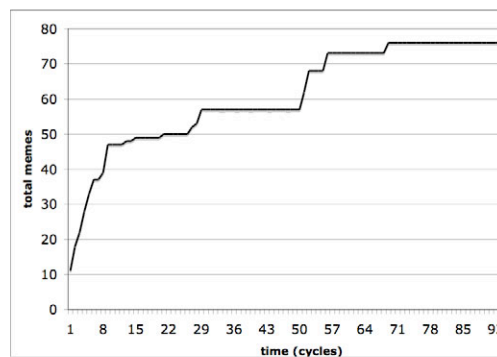


Figure 2. Number of memes learned in time

It was also possible to observe the number of times that each one of the memes were listened to by Agent ‘A’. The next Figure shows this amount for the first learned 20 memes.

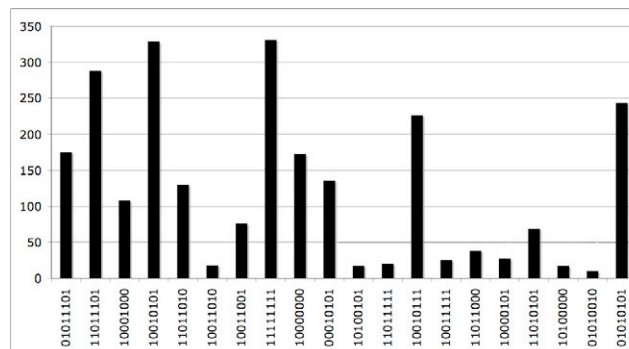


Figure 3. First 20 learned memes: number of listening.

One of the most interesting features that RGeme generates is the track of the evolution of the importance (weight) of each one of the memes during the learning phase of an Agent. The increase or decrease of the importance of the memes is the direct result of the number of times and the date they were listened to and/or practiced. The next Figure shows this analysis during Agent ‘A’s whole learning phase. Every time a new meme was learned a new line appeared. If a meme was not heard during a certain time, its curve started to fall (forgetting effect).

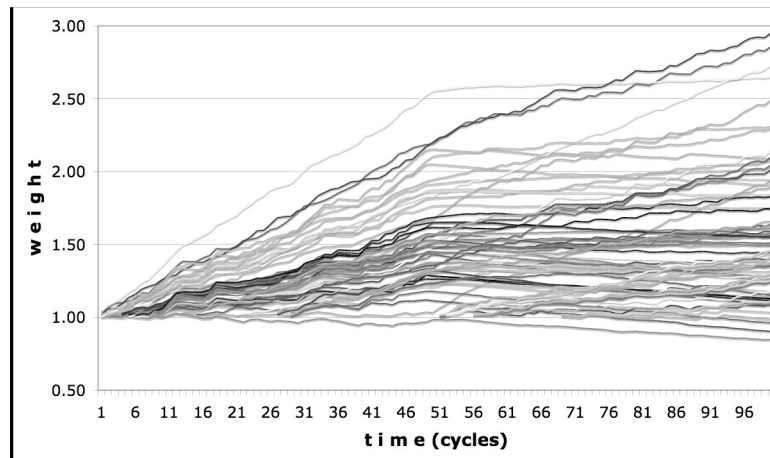


Figure 4. Memes curve of importance in time.

As it is obviously very difficult to visualize the evolution of all the 76 memes in the same graph, in Figure 5 we made a selection of a few of them. Some typical behaviour that emerged from the interactions is described in the paragraphs below.

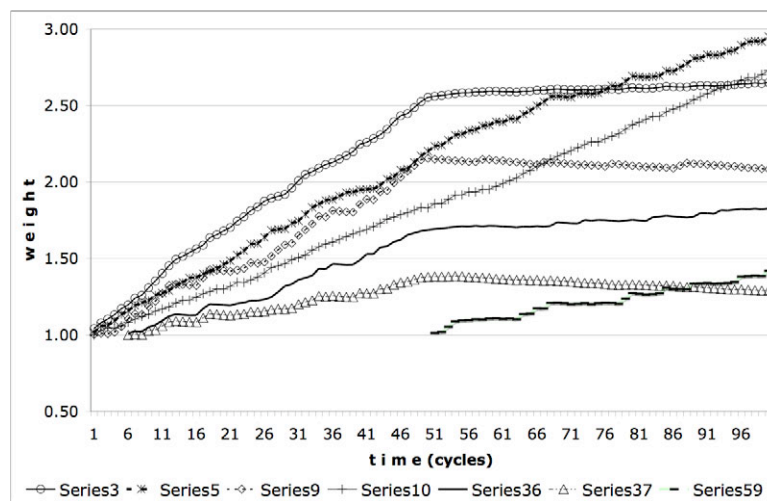


Figure 5. Memes curve of importance in time (selection)

In Figure 5, each “series” corresponds to the memes described in Table 5:

Series	Seq	dFL	dLL	nL	aW
3	11011101	1	50	288	2.650
5	10010101	1	100	329	2.952
9	11111111	1	88	331	2.082
10	10000000	1	100	173	2.729
36	01011111	6	93	123	1.826
37	11110000	6	43	4	1.293
59	00000001	51	100	12	1.420

Table 5. Description of memes

Memes 3, 5, 9 and 10 were all first listened to in the first interaction (dFL = date of first listening) with music ‘Arrojado’, by Nazareth. Memes 36 and 37 only appeared in time 6, after Agent ‘A’ listened to music ‘Ameno Reseda’, by the same composer. Meme 59 appeared in time 51 by listening to music ‘A Felicidade’, by Tom Jobim.

Observe that Agent ‘A’ only begun to learn from the rhythms by Jobim at time 50. From this moment on the behaviour of the curve of importance clearly changed for many of the memes (see Figure 5), affecting them positively in some cases and negatively in other cases. For some memes, on the contrary, the change in the Evaluation Matrix didn’t affect too much the previous behaviour.

For instance, although meme 37 begun to be listened to in time 6, its relative importance comparing to the other memes was never very high and even begun to fall after time 50. On the other hand, meme 59, which never appeared in the music by Nazareth, and was only listened to in time 51, at the end of simulation was victorious over meme 37.

Memes 3 and 36 showed a very strong increase during the time in which Agent ‘A’ was listening only to the music by Nazareth. After it begun to listen to the music by Jobim, these memes had only a light increase, meaning that there was a balanced ratio between the number of times they were listened to and the forgetting effect.

During the first half of the simulation the importance of meme 9 had a relative important increase but after time 50 it begun to be forgotten.

At the end of the simulation, some of the memes were definitely winners over others, as Table 6 shows:

Meme	Weight
10010101	2.952
01011101	2.858
10000000	2.729
11011101	2.650
01010101	2.491
10010111	2.310
11010101	2.293
00010101	2.146
00000101	2.101
11111111	2.082

Table 6: Winning memes

Notice that meme 5 (“series 5” in Figure 5), which was the winner meme above all the others, had a very strong increase from the beginning up to the end of the simulation. The next Figure shows its musical representation:



Figure 6: Musical representation of the winner meme

At last, this information will be used later on during the production phase in order to generate new rhythmic material, as mentioned above.

4. Conclusion

In this paper we introduced the learning stages of RGeme, an artificial intelligence system for the composition of rhythmic streams.

Besides the production stage that will be covered in a future paper, RGeme has already proved to be an efficient tool to evolve rhythmic worldviews in artificially inhabited environments. Through the description of a simulation we demonstrated how the exposure to different rhythmic material could ultimately shape the musical “knowledge” of an agent.

Experiments are being conducted with different sources of data according to musical genres and styles.

In the future, besides the rhythm information that is being currently employed, the system will deal with more complex musical structures that consider note information (pitches and vertical structures). A better parsing algorithm is being tested in order to extract memes of varied length and a new measure of distance is also being implemented.

5. Acknowledgements

This research is funded by the Brazilian Government’s Fundacao Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior (CAPES).

6. References

- Blackmore, S. *The Meme Machine*. Oxford: Oxford University Press. 1999.
- Cope, D. *Computers and Musical Style*. Oxford, UK: Oxford University Press, 1991.
- Cox, A. The mimetic hypothesis and embodied musical meaning. *MusicaeScientiae*, V(2):195–212. 2001.
- Dawkins, R. *The Blind Watchmaker*. Penguin Books, London, 1991.
- Dawkins, R. *The Selfish Gene*. Oxford University Press, 1989.
- Dodge, C. and Jerse, T. *Computer Music*. London, UK: Schirmer Books, 1985.
- Gabora, L. M. The origin and evolution of culture and creativity. *Journal of Memetics–Evolutionary Models of Information Transmission* [<http://www.cpm.mmu.ac.uk/jomemit/1997/vol1/gabora1.html>]. 1997.
- Horowitz, D., “Generating Rhythms with Genetic Algorithms”, *Proceedings of the 1994 International Computer Music Conference, San Francisco, ICMA*, 1994.
- Impett, J. “Interaction, simulation and invention: a model for interactive music,” in *Proceedings of ALMMA 2001 Workshop on Artificial Models for Musical Applications* (E. Bilotta, E. R. Miranda, P. Pantano, and P. M. Todd, eds.), (Cosenza, Italy), pp. 108–119, Editoriale Bios, 2001.
- Jan, S. “Replicating sonorities: towards a memetics of music”, *Journal of Memetics–Evolutionary Models of Information Transmission*, 4. 2000.
- Jan, S. Meme Hunting with the Humdrum Toolkit: Principles, Problems and Prospects. *Computer Music Journal* (2004) 28(4):68–84.

- Maes, P. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT Press, 1991.
- Manzolli, J., Moroni, A., von Zuben, F. and Gudwin, R. Vox Populi: An Interactive Evolutionary System for Algorithmic Music Composition. *Leonardo Music Journal*, vol. 10, p. 49-54. San Francisco, USA, 2000.
- Miranda, E. R. "Cellular automata music: An inter-disciplinary music project," *Interface (Journal of New Music Research)*, vol. 22, no. 1, pp. 03–21, 1993.
- Miranda, Eduardo R. "The artificial life route to the origins of music". *Scientia*, 10(1):5-33. 1999.
- Pachet, F. "Rhythms as emerging structures", *Proceedings of 2000 International Computer Music Conference, Berlin, ICMA*, 2000.
- Russell, S.J. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.
- Worral, D. "Studies in metamusical methods for sound image and composition," *Organised Sound*, vol. 1, no. 3, pp. 183–194, 2001.
- Xenakis, I. *Formalized Music: Thought and Mathematics in Composition*. Bloomington (IN), USA: Indiana University Press, 1971.

CInBalada: um Laboratório Multiagente de Geração de Ritmos de Percussão

Pablo Sampaio, Patrícia Tedesco, Geber Ramalho

Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50.732-970 – Recife – PE – Brasil

{pas,pcart,glr}@cin.ufpe.br

Abstract. *In this paper we present CInBalada, a multiagent system where intelligent percussionist agents participate in purely rhythmic performances. Each agent knows a limited set of rhythmic patterns for a percussion instrument. Thus, agents interact with their peers to decide which patterns they should play so that an interesting group performance emerges. Human users can try different ensembles and musical mixes with CInBalada, turning the system into a Rhythms Lab, which may be useful for composers and other musicians.*

Resumo. *Neste trabalho apresentamos o sistema CInBalada, onde agentes percussionistas inteligentes participam de uma performance puramente rítmica. Cada agente conhece um repertório limitado de padrões rítmicos de um instrumento de percussão e precisa interagir com os demais agentes para escolher qual padrão tocar com o grupo de maneira que a performance fique musicalmente boa. Usuários humanos podem usar o sistema para testar diferentes formações e diferentes misturas de estilos, fazendo o sistema funcionar como um laboratório rítmico, que pode vir a ser útil para músicos compositores e arranjadores.*

1. Introdução

Na intersecção entre as áreas de Inteligência Artificial e Computação Musical, vários sistemas foram criados aplicando o conceito de agentes inteligentes (Pachet, 2000) (Pachet, 2002) (Miranda, 2002) (Murray-Rust, 2003) (Wulfhorst, et.al 2003). Nesses sistemas, diversos aspectos da música são abordados, porém a ênfase maior costuma ser na geração de harmonia e/ou melodia. Poucos sistemas focam especificamente a parte rítmica, que é, entretanto, considerada fundamental para a “sensação” que a música causa em ouvintes humanos (Honning, 2002), além de ter grande importância cultural em algumas sociedades como algumas tribos da África (Ladzekpo, n.d).

Dentre os poucos sistemas inteligentes que abordam os instrumentos rítmicos, aparentemente a maior parte é voltada especificamente para bateria (Gannon, 2005) (Kragtwijk, et.al 2001) (Pearce, 2000), sendo ainda menor o número daqueles voltados para instrumentos de percussão em geral. Os poucos sistemas voltados para percussão não tratam de maneira homogênea da interação entre agentes inteligentes diversos e, quando o fizeram, consideraram um modelo de interação distante do que acontece com músicos reais. Além disso, não há sistemas inteligentes para a criação de arranjos puramente rítmicos, com possíveis misturas de estilos musicais.

Neste trabalho, apresentamos um sistema inovador voltado para percussão, o CInBalada, em que agentes percussionistas interagem para gerar uma performance puramente rítmica. Usuários do sistema poderão controlar as performances de cada um desses agentes de diversas maneiras: escolhendo os instrumentos presentes na performance, definindo um estilo musical (por agente) e alterando o andamento geral. Dessa forma, o CInBalada funciona como uma espécie de laboratório de ritmos, por meio do qual usuários (músicos, principalmente) poderão testar diferentes misturas rítmicas e criar arranjos de percussão variados. Além disso, tentamos abordar a questão da interação entre os agentes de uma maneira mais próxima ao que acontece em grupos musicais humanos de modo a permitir, futuramente, fazer do CInBalada um sistema interativo para uso profissional em shows ou como uma ferramenta de aprendizado para músicos iniciantes praticarem sozinhos.

Na próxima seção apresentamos os principais sistemas multiagentes existentes que tratam de performances rítmicas, bem como uma visão crítica desses sistemas. Na seção 3 apresentamos uma visão geral do sistema CInBalada. Na seção 4 apresentamos o agente responsável pela execução sonora do CInBalada: o agente *outputter*, e na seção 5 apresentamos detalhes dos agentes percussionistas do sistema. A seção 6 descreve a arquitetura e implementação do sistema. Por fim, a seção 7 apresenta conclusões e uma discussão sobre futuras extensões do CInBalada.

2. Percussionistas Virtuais

Nesta seção analisaremos dois sistemas multiagentes que tratam da criação de performances rítmicas: o Rhythm Band Editor e o VirtuaLatin.

2.1 Rhythm Band Editor (Pachet, 2000)

No Rhythm Band Editor, cada agente começa com um padrão rítmico inicial (possivelmente vazio) e uma base de regras de transformação. Uma partitura global é mantida de modo a permitir que cada agente tenha acesso ao que os demais agentes estão tocando. Um módulo de percepção presente nos agentes recebe como entrada a partitura global completa e gera um conjunto de informações de alto nível, tais como: estrutura de batidas e batidas acentuadas. Um outro módulo, chamado de módulo de produção, utiliza a base de regras fornecida na criação do agente para alterar o padrão corrente do agente. As regras têm suas pré-condições especificadas a partir das informações de alto-nível geradas pelo módulo de percepção e, quando disparadas (pré-condições satisfeitas), executam uma ação de edição no ritmo tocado pelo agente. Um exemplo de regra é: “se existe uma batida forte na qual o agente não toca, então adicione uma nota nessa batida”.

Os agentes do Rhythm Band Editor permanecem indefinidamente em ciclo, alternando, a execução de seus respectivos padrões com a aplicação das regras, de modo a evoluir os padrões rítmicos. Eventualmente, o sistema pode convergir para uma situação em que nenhuma regra é aplicável. Nesse caso, os agentes continuam a tocar seus padrões sem novas alterações.

2.2. VirtualLatin (Murray-Rust, 2003)

Outro sistema voltado para percussão é o VirtuaLatin, onde um agente inteligente, o timbaleiro, gera arranjos de timbales para salsa. O agente timbaleiro está

inserido num ambiente multiagente em que cada agente é um instrumentista de uma banda de salsa e produz saída musical de um compasso por vez. O timbaleiro, no entanto, é o único agente inteligente do sistema – os demais agentes apenas reproduzem trechos pré-gravados de uma música (do estilo salsa). Baseado no GTTM (Lerdahl & Jackendoff, 1983), o algoritmo do agente timbaleiro é capaz de extrair várias informações de alto nível da peça em execução por meio de várias etapas de análise (de atividade, de harmonia, rítmica e de paralelismo), executadas a cada compasso. Partindo dessa representação de alto nível, o sistema aplica um conjunto de regras para selecionar o padrão rítmico de salsa que melhor se adequa ao compasso, de acordo com o contexto musical em que ocorre. Por exemplo: o timbaleiro tende a executar viradas no final de cada seção da música.

2.3. Análise Crítica

Os dois trabalhos apresentados na seção anterior tratam da construção de agentes percussionistas, porém apresentam algumas limitações, que apresentamos nesta seção.

O Rhythm Band Editor, apesar de permitir interações entre múltiplos agentes percussionistas apresenta a desvantagem de usar um modelo de interações um pouco distante do que acontece em performance reais. O sistema foi criado com o propósito principal de estudar os ritmos enquanto estruturas evolutivas e não o de criar saídas musicais que possam ser aproveitadas por músicos humanos. Portanto, o foco principal desse sistema acaba sendo o conjunto de regras e não a música produzida. Com isso, é muito difícil controlar o rumo da performance musical nesse sistema, pois não é possível ter uma clara idéia da música que vai emergir como resultado das regras escolhidas.

Já o VirtuaLatin implementa um único agente de percussão e produz arranjos típicos apenas de um estilo musical específico – a salsa. O sistema oferece, portanto, opções limitadas para criação de arranjos de percussão, servindo apenas para gerar arranjos de timbales partindo de salsas pré-gravadas (sem os timbales). Além disso, o sistema não apresenta flexibilidade nem homogeneidade no modelo de interação entre os agentes, pois existe apenas um agente inteligente que simplesmente recebe as saídas musicais dos demais agentes e as usa como entrada para gerar sua própria saída (o arranjo de timbales).

3. O CInBalada

Neste contexto, o CInBalada oferece um ambiente inovador que oferece a possibilidade de experimentar as mais variadas misturas rítmicas, mesclando livremente estilos musicais usando as mais diversas formações de instrumentos de percussão. Desse modo, o sistema funciona como um laboratório rítmico, onde compositores e arranjadores podem criar e testar novos arranjos percussivos facilmente, sem ter que arcar com os custos de reunir músicos humanos para tocar em conjunto.

Inspirado em manifestações rítmicas populares (como o samba e o maracatu), o CInBalada dispõe de agentes inteligentes que simulam músicos percussionistas, que comunicam-se entre si para gerar uma performance rítmica polifônica. O uso de múltiplos agentes inteligentes é uma analogia natural para um grupo musical real tocando junto e permite a adaptação do sistema a cada nova situação imposta pelo

usuário (mudança de formação, mudança de estilo musical, etc.), além de permitir interatividade com músicos humanos numa possível extensão futura.

Para iniciar uma performance, o usuário **escolhe os instrumentos de percussão** que desejar. Para cada instrumento é associado um agente, possuidor de um conjunto (limitado) de padrões rítmicos do instrumento escolhido. Quando a performance é iniciada, os agentes tentarão escolher padrões rítmicos que se encaixem entre si (segundo uma função de avaliação), de modo a gerar uma performance musicalmente aceitável.

Esse processo de escolha dos padrões é realizado automaticamente pelos agentes, porém o usuário também tem a oportunidade de interferir nas escolhas ou de obrigar um agente a usar um padrão rítmico específico. Na atual versão, isso é feito por meio da **seleção de estilos musicais** de cada agente. O usuário pode selecionar um estilo musical específico para cada agente, de maneira a obrigar o agente a escolher apenas padrões rítmicos típicos dos estilos selecionados.

A figura 1 abaixo mostra a interface principal do sistema, onde podem ser acessadas as funcionalidades acima descritas. Nessa interface os usuários podem inserir e remover os agentes. Para inserir um agente, o usuário escolhe o instrumento do agente e um nome, que será usado para identificar o agente na interface. Atualmente, só pode haver um agente por instrumento de percussão. Na interface é possível visualizar uma tela de detalhes sobre cada agente. Na tela de detalhes é possível ver uma lista de estilos musicais que podem ser escolhidos, bem como o repertório completo de padrões rítmicos disponíveis para o dado agente.

A interface oferece, ainda, algumas opções básicas de **controle sobre a saída sonora** gerada. É possível parar (silenciar) e reiniciar a saída sonora, bem como alterar o andamento geral da performance a qualquer momento.

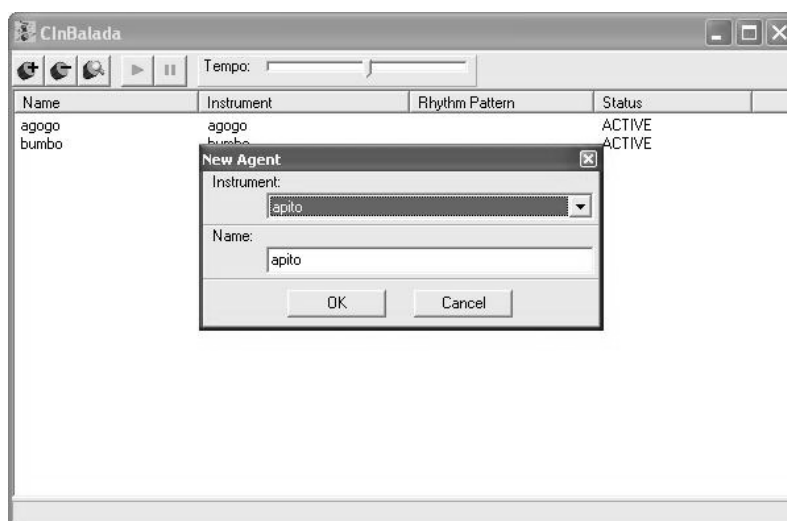


Figura 1. Interface principal do ClnBalada

3.1. Repertório de Padrões Rítmicos

Os padrões rítmicos usados pelos agentes do sistema são armazenados como arquivos MIDI e possuem de um a quatro compassos. No momento, o sistema conta com uma

base de cerca de duzentos padrões rítmicos MIDI para doze instrumentos de percussão.

Cada um desses padrões rítmicos está associado a um conjunto de informações de mais alto nível, que são utilizadas pelos agentes do CInBalada em suas negociações.

As principais informações de alto nível armazenadas para cada padrão são:

- **Instrumento** usado no padrão (cada arquivo do sistema é executado por apenas um instrumento de percussão).
- **Estilos musicais** aos quais o padrão é comumente associado, ou seja, estilos musicais onde o padrão rítmico costuma ser usado.
- **Tipo** do padrão rítmico. Indica o papel que ele tem em uma performance. Atualmente, inspirados no sistema VirtualLatin, consideramos que os padrões podem ser de três tipos: *basic* (ritmo básico, deve ser tocado a maior parte do tempo), *fill* (virada) ou *roll* (rufo).
- **Assinatura de tempo**. Indica se o compasso é binário, ternário ou quaternário.
- **Arquivo MIDI**, que contém o padrão rítmico propriamente dito.

Os padrões rítmicos, juntamente com as informações de alto nível associadas, são cadastrados no sistema por meio de uma interface auxiliar chamada CInBalada Repertoire Editor, que pode ser vista na figura 2.

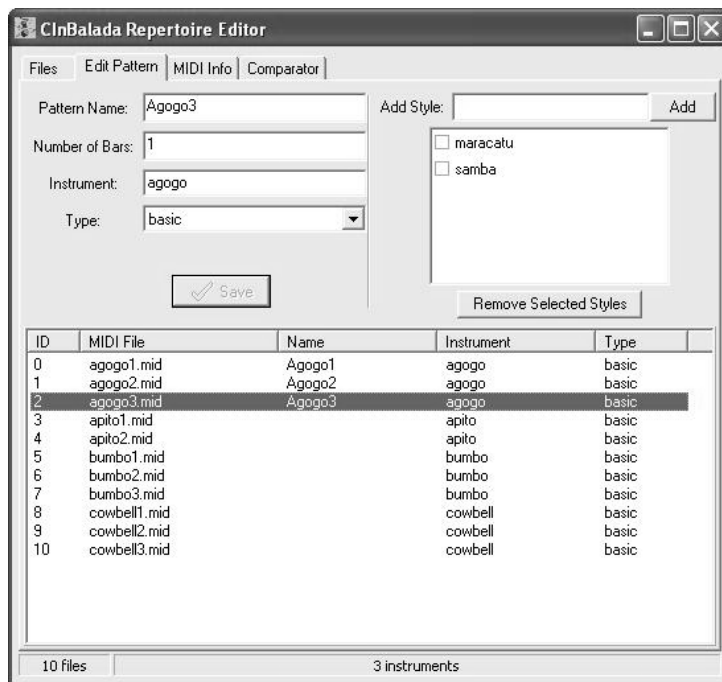


Figura 2. CInBalada Repertoire Editor.

4. Agente *Outputter*

O agente *outputter* é responsável por receber e mesclar todas os trechos musicais (padrões rítmicos) enviados pelos agentes músicos e enviá-los a alguma saída (placa de

som ou arquivo). Existe apenas um agente *outputter* por instância da aplicação. Ele é criado automaticamente quando a aplicação é iniciada.

Quando um agente percussionista é criado, ele se cadastra junto ao agente *outputter* como **produtor de música**. A partir disso, o *outputter* passa a mandar para o agente músico, periodicamente, mensagens requisitando frases musicais novas. As frases podem ser formadas por um número qualquer de compassos completos e são transmitidas em formato XML. Um exemplo de uma frase musical no modelo XML adotado pode ser visto na figura 3.

```
<?xml version="1.0" encoding="utf-8"?>
<Phrase resolution="400">
  <Note pitch="36" dynamic="80" startTick="12" endTick="18"/>
  <Note pitch="43" dynamic="80" startTick="30" endTick="45"/>
</Phrase>
```

Figura 3. Exemplo do modelo XML adotado para representar frases musicais.

Para permitir a interação entre os agentes, é necessário também que um agente saiba o que será tocado pelos demais agentes. Para isso, o agente *outputter* também encaminha frases de um agente para outros. Um agente interessado em receber as frases de outro agente precisa enviar uma mensagem para se cadastrar como **ouvinte**, indicando na mensagem qual agente deseja ouvir. O *outputter* enviará cada nova frase do agente ouvido para os seus respectivos ouvintes.

O uso de um agente que centraliza a saída musical tem várias vantagens (Murray-Rust, 2003):

- Ponto central de comunicação. Implica na troca de $\theta(n)$ mensagens, ao invés de $\theta(n^2)$, se cada agente se comunicasse com todos os demais.
- Separação entre criação e execução dos sons. Apenas o *outputter* precisa cuidar de seqüenciamento e metrificação das frases geradas pelos músicos.
- Saída centralizada. Simplifica o acesso à placa de som e possibilita gravar um arquivo MIDI central com toda a performance.

5. Agentes Percussionistas

Atualmente, o CInBalada é composto de um único tipo de agente percussionista, chamado **search-agent**. Este nome vem do mecanismo de negociação empregado para escolha dos padrões que serão tocados pelos agentes, que funciona como uma busca intensiva pela melhor combinação de padrões. Para julgar as combinações de padrões, o sistema usa uma função de avaliação rítmica (descrita em detalhes na seção 5.1).

Quanto à **organização** do sistema, os agentes se reúnem em uma estrutura horizontal chamada “Banda”. Apesar de não haver qualquer relação hierárquica entre os agentes, os agentes se comunicam por ordem de entrada na Banda, para facilitar as interações. Ou seja, durante uma rodada de negociação, o agente que primeiro ingressou na Banda toma uma decisão e repassa para o segundo que repassa para o terceiro, etc.

Quando um novo agente é inserido no sistema, ele inicia um processo de **negociação** com os agentes da Banda para tentar encontrar uma combinação entre os padrões rítmicos de todos os agentes de maneira que, dois a dois, todos os padrões “combinem”, segundo a função de avaliação. No decorrer da negociação, o agente novato sucessivamente apresenta cada um de seus padrões rítmicos à Banda (em formato XML). Os agentes da Banda interagem entre si tentando mudar seus padrões de maneira a manter a coerência, tanto internamente como com o agente novato. Se isso não for possível, o agente novato é rejeitado e encerrado.

Apresentamos na figura 4 abaixo um exemplo de uma rodada de negociação. Nesse exemplo, existem três agentes: o agente **NV**, que deseja entrar na Banda, e dois outros agentes, **AG1** e **AG2**, que já fazem parte da Banda, sendo AG1 o mais antigo deles.

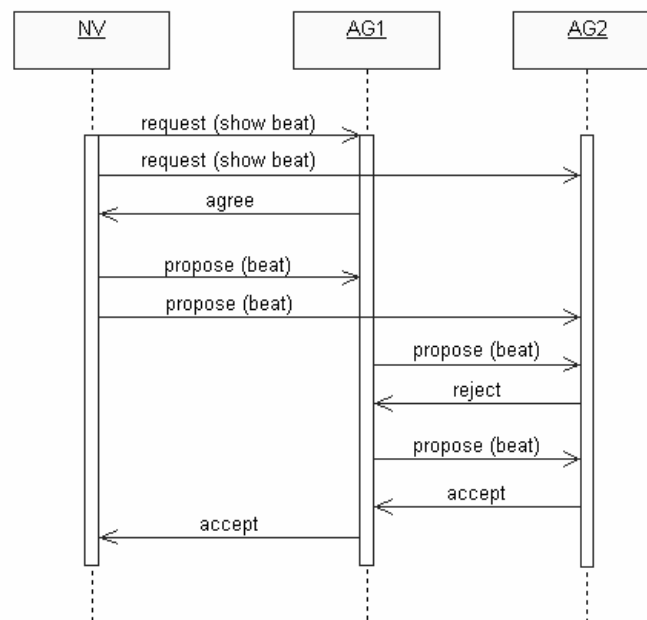


Figura 4. Exemplo de negociação entre os agentes percussionistas.

A figura acima mostra um exemplo de negociação que terminou em sucesso, ou seja, terminou com a entrada do agente NV na Banda. Esse exemplo começa com o agente NV enviando um pedido para se apresentar à Banda (as mensagens vindas de NV são sempre enviadas a todos). Após ter seu pedido aceito, o agente NV manda uma proposta de batida (padrão rítmico) à Banda. Ao receber a proposta, o agente AG1 analisa seu repertório e escolhe uma batida adequada à que foi apresentada por NV. O agente AG1, então, repassa esse padrão escolhido para AG2 (mensagem “propose”), que a recusa, por não encontrar em seu repertório um padrão rítmico que case com os dois já apresentados (o de NV e o de AG1). O agente AG1 propõe uma outra batida que também casa com a de NV e, dessa vez, o agente AG2 consegue encontrar um padrão que se adeque tanto ao de NV quanto ao de AG1. O agente AG1, representando toda a Banda, comunica a NV que sua batida foi aceita, permitindo-o ingressar na Banda.

5.1. Função de Avaliação

O objetivo da função de avaliação no CInBalada é medir a “**qualidade musical**” das combinações de padrões rítmicos escolhidas pelos agentes do sistema. Ela é usada durante o processo de negociação entre os agentes para definir quando um par de padrões é “aceitável”.

A escolha de uma função de avaliação é um dos pontos mais importantes do sistema. No entanto, essa é uma tarefa muito difícil, devido à ausência de trabalhos teóricos sobre esse tema. Para fazer um julgamento mais completo sobre a qualidade musical de um par de padrões tocados em conjunto, é necessário levar em conta vários critérios musicais, tais como: complexidade, paralelismo (auto-similaridade), nível de tensão, etc.

Aparentemente, não existem trabalhos que ofereçam medidas objetivas (funções) sobre a maior parte desses critérios. Portanto, desenvolvemos uma função de avaliação própria para julgamento das combinações de padrões. Essa função baseia-se na noção de que são necessárias tanto uma certa dose de afinidade (ou similaridade) entre os padrões, quanto um pouco de originalidade (diferenças) entre eles, para que soem bem em conjunto. Assim, a função consiste simplesmente em medir as quantidades de pontos de encontros e de pontos de desencontros entre ataques dos dois padrões. O valor da função será o módulo da diferença entre esses valores. Assim, quanto menor o valor, mais equilíbrio entre encontros e desencontros e, portanto, maior a coerência musical entre os dois padrões.

Para comparações com padrões com mais de um compasso (porém, de mesmo tamanho), é feita uma normalização, dividindo o valor da função pelo número de compassos. No caso de comparar padrões de diferentes tamanhos, calculamos o Mínimo Múltiplo Comum entre os tamanhos e concatenamos cada padrão consigo mesmo até que todos tenham esse mesmo tamanho. A partir daí, a comparação pode acontecer normalmente.

6. Implementação

O CInBalada está sendo desenvolvido em C++ e possui três módulos centrais: uma biblioteca musical, a plataforma multiagentes e os agentes do sistema. No decorrer dessa seção apresentamos sucintamente cada um dos módulos implementados, bem como os resultados obtidos com o sistema.

6.1. Biblioteca Musical

Criada para realizar as manipulações sonoras necessárias para a implementação dos agentes do CInBalada. O principal foco da biblioteca é a tecnologia MIDI. A biblioteca permite manipular arquivos MIDI e enviar eventos MIDI para a saída sonora. Além disso, há um conjunto de classes que mantém as informações de alto nível sobre os padrões rítmicos (mostradas na seção 3.1), armazenando-as no formato XML. Essas funcionalidades estão concentradas na classe `RhythmPattern` que é uma representação de alto-nível de um padrão rítmico de um instrumento de percussão.

6.2. Plataforma Multiagente

Para permitir a operação dos vários agentes do sistema foi implementada uma plataforma multiagentes para C++: o FAMA (*Framework* para Aplicações MultiAgentes). O FAMA é inspirado no JADE (Bellifemine, et.al 1999) e foi implementado devido à carência de plataformas multiagentes para C++. A plataforma oferece todos os serviços básicos recomendados pela FIPA: gerenciamento do ciclo de vida dos agentes, serviços de localização de agentes (páginas amarelas e páginas brancas) e transporte mensagens. No entanto, a plataforma do FAMA não é distribuída, uma vez que distribuição não é um requisito do projeto CInBalada.

6.3. Agentes do CInBalada

O agente *ouputter* é representado pela classe `OutputterAgent` e é criado automaticamente quando a plataforma FAMA é iniciada. Ao ser criado, ele cadastra o seu serviço como “*MIDI Ouput Service*” no serviço de páginas amarelas da plataforma e, então, inicia a sua execução, sendo responsável pelas tarefas de: receber requisições de cadastramento de agentes produtores de música e ouvintes, receber frases dos agentes, enviar requisições periódicas de frases e seqüenciar a saída sonora (placa de som e arquivo).

Os agentes percussionistas são instâncias da classe `SearchAgent` e guardam um conjunto de objetos `RhythmPattern` da biblioteca musical, que representam os padrões rítmicos conhecidos pelo agente. Os agentes percussionistas podem possuir apenas um de dois possíveis *comportamentos* (sub-classes de `Behaviour`): `JoinerBehaviour` e `PlayerBehaviour`.

O `JoinerBehaviour` implementa o papel do agente novato no protocolo de negociação dos agentes percussionistas. Inicialmente, ele faz uma requisição para entrar na Banda. Se algum padrão do agente for aceito, o `JoinerBehaviour` é encerrado e um `PlayerBehaviour` é instanciado para o mesmo agente. Se não for aceito, o agente é encerrado. Já a classe `PlayerBehaviour` implementa o comportamento de um agente que faz parte da Banda. Ele envia periodicamente seu padrão rítmico corrente ao *ouputter* sempre que requisitado e, quando surge uma requisição de um agente novato para entrar na Banda, ele inicia a negociação interna (com os demais agentes da Banda) para tentar se adaptar ao novo agente¹.

A figura 5 a seguir mostra o diagrama de classes da implementação dos agentes, mostrando também as principais relações com classes dos outros dois módulos. As classes mais escuras são as que formam os agentes, enquanto as classes claras são parte da biblioteca musical ou do *FAMA*.

¹ Conforme o mecanismo de negociação mostrado na seção 5.

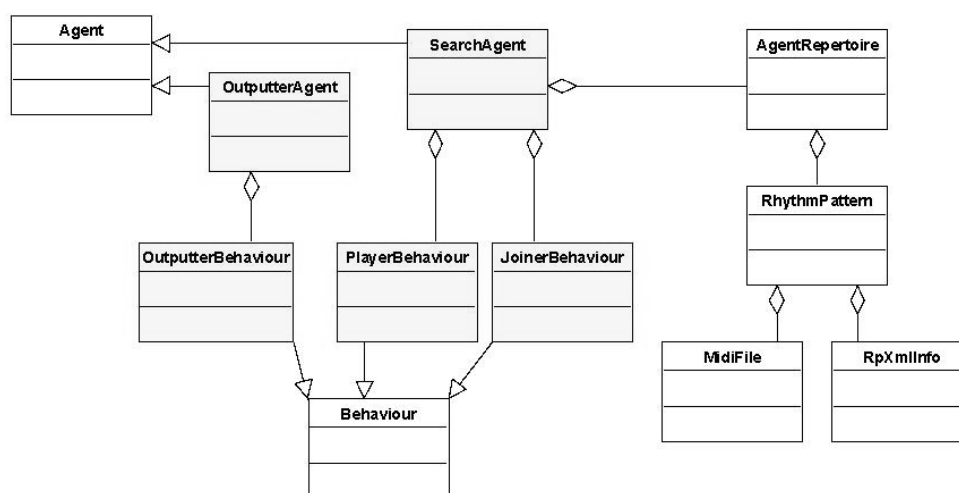


Figura 5. Diagrama de classes dos agentes do CInBalada.

6.4. Resultados

Os testes preliminares feitos com a presente versão do sistema mostraram resultados promissores. O sistema permite obter, facilmente, diversos arranjos de percussão com coerência musical entre os agentes aceitável, desde que sejam escolhidos poucos instrumentos (por volta de quatro). Com muitos instrumentos, o protocolo de negociação tende a rejeitar novos agentes e, quando os aceita, gera uma saída musical confusa, porque não há qualquer organização dos instrumentos dentro da performance.

7. Conclusão e Trabalhos Futuros

Neste artigo apresentamos o CInBalada, um sistema musical inteligente que permite a criação de arranjos puramente rítmicos variados. Para isso, o sistema dispõe de agentes percussionistas capazes de interagir e negociar entre si para manter a performance musicalmente aceitável, segundo uma função de avaliação interna. Por meio de uma interface gráfica, usuários humanos podem escolher diferentes formações e atribuir diferentes estilos a cada agente, ouvindo o resultado imediatamente. Desse modo, o CInBalada funciona como um laboratório de ritmos onde diferentes misturas podem ser experimentadas.

O CInBalada traz, basicamente, duas inovações em relação aos sistemas de Computação Musical atuais: a criação semi-automática de arranjos rítmicos polifônicos (múltiplos instrumentos) e a implementação de mecanismos de interação entre os agentes mais próximos do que acontece no mundo real, o que representa um passo importante para a criação de um sistema interativo no futuro. No entanto, no desenvolvimento do CInBalada ainda existem questões em aberto, que pretendemos abordar em um futuro próximo. Abaixo discutimos algumas dessas questões.

Em primeiro lugar, é muito importante pesquisar funções de avaliação que tenham significado musical claro, para melhorar a coerência das saídas musicais (e.g. medidas de complexidade, medidas de similaridade). Talvez seja deixado a critério do usuário escolher quais funções usar, ou qual a importância relativa de cada função de avaliação, para que as saídas do sistema se moldem ao gosto do usuário. Uma outra ex-

tensão interessante é a possibilidade de os agentes assumirem diversos papéis na Banda (e.g. solista, executor de virada e regente), de modo a organizar a função de cada instrumento dentro da performance. Esses papéis seriam escolhidos pelos usuários e dariam aos agentes diferentes graus de liberdade para a execução de suas performances, enriquecendo o realismo da simulação provida pelo CInBalada.

Também pretendemos enriquecer a interface com o usuário de modo a oferecer mais opções de personalização das performances. Porém essa é uma tarefa que ocorrerá a medida que novas características forem incorporadas ao sistema. Queremos, ainda, enriquecer a base de padrões rítmicos do sistema de modo a comportar uma maior variedade de instrumentos e de estilos musicais. Pretendemos aproveitar a riqueza rítmica da música brasileira adicionando padrões de estilos musicais típicos do Brasil, tais como: samba, baião e maracatu.

Por fim, estamos cientes da necessidade de realizar mais experimentos com o sistema e, por isso, planejamos a realização de experimentos com especialistas (músicos e/ou professores de música) no decorrer do desenvolvimento das próximas extensões. Eles terão a oportunidade de testar a ferramenta e, por meio de um questionário, julgá-la segundo critérios musicais e funcionais.

8. Referências

- Bellifemine, F. Rimassa, G. Poggi, A. (1999). JADE - A FIPA-compliant Agent Framework. Em: *Proceedings of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*. Londres.
- Ganascia, J.G. Ramalho, G. Rolland, P.Y. (1999). An Artificially Intelligent Jazz Performer. *Journal of New Music Research*, 28 (2), pp. 105-129. Amsterdã: Swets & Zeitlinger.
- FIPA: Foundation for Intelligent Physical Agents. (2003). Informações no site: <http://www.fipa.org/> (acesso em 16 de junho de 2005).
- Gannon, Peter. (2005). *Band-in-a-Box* [Programa de Computador]. Hamilton, Ontário: PG Music Inc.
- Honning, H. (2002). Structure and Interpretation of Rhythm and Timing. *Tijdschrift voor Muziektheorie* [Jornal de Teoria Musical], 7 (3), pp. 227-232.
- Kragtwijk, M. Nijholt, A. Zwiers, J. (2001). Implementation of a 3D Virtual Drummer. Em: *Computer Animation and Simulation 2001*, Editores: Magnenat-Thalmann, N., Thalmann, D. Viena: Springer Verlag Wien.
- Ladzekpo, C. K. (n.d.). *Rhythmic Principles*. Disponível em: <http://cnmat.cnmat.berkeley.edu/%7Eladzekpo/PrinciplesFr.html> (acesso 13 de junho de 2005).
- Lerdahl, F. Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press.
- Messick, Paul (1998). *Maximum MIDI: Music Applications in C++*. (S.l.): Manning Publications.
- Miranda, E. R. (2002). Emergent sound repertoires in virtual societies. *Computer Music Journal*, 26(2), pp. 77-90.

- Murray-Rust, D. (2003). *VirtuaLatin - Agent Based Percussive Accompaniment*. Dissertação (MSc in Informatics), School of Informatics, University of Edinburgh, Edimburgo.
- Papadopoulos, G. Wiggins, G. (1999). AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects. Em: *Proceedings of the AISB'99 Symposium on Musical Creativity*, pp. 110-117. Brighton, UK: SSAISB.
- Pachet, F. (2000). Rhythms as Emerging Structures. Em: *Proceedings of the 2000 International Computer Music Conference (ICMC)*. Berlim : ICMA.
- Pachet, F. (2002). Interacting with a Musical Learning System: The Continuator. Em: *Proceedings of the II International Conference on Music and Artificial Intelligence (ICMAI)*. Edimburgo.
- Pearce, M. (2000). *Generating Rhythmic Patterns: a Combined Neural and Evolutionary Approach*. Dissertação (MSc in Artificial Intelligence), School of Informatics, University of Edinburgh, Edimburgo.
- Salazar, M. (1991). *Batucadas de Samba*. Rio de Janeiro : Lumiar Editora.
- Wooldridge, M. J. (2002). *An Introduction to Multi-Agent Systems*. (S.l.): John Wiley and Sons.
- Wulfhorst, R. D. Nakayama, L. Vicari, R. M. (2003). A Multiagent Approach for Musical Interactive Systems. Em: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 584-591. Melbourne: ACM Press.

EV: Multilevel Music Knowledge Representation and Programming

Jesus L. Alvaro¹, Eduardo R. Miranda², and Beatriz Barros¹

¹ Departamento de Lenguajes y Sistemas Informáticos, UNED, Spain

JesusLAlvaro@gmail.com

bbarros@lsi.uned.es

² Computer Music Research, University of Plymouth, UK

eduardo.miranda@plymouth.ac.uk

Abstract. *This paper introduces EV Meta-Model, a new system for representing musical knowledge for computer-aided composition. It starts with a brief historical discussion on the fields of composition and sound synthesis. Then, the practice of musical composition is presented as a communication process from composers to listeners, where musical messages go through different representations: from the complex abstractions of composers and their compositional tools, to the performers and the perceptual representation of listeners.*

EV Meta-Model is proposed as a generic tool for representing any kind of time-based events that manifest themselves as coherent representations at different levels, including high-levels of musical abstractions. At the same time, it is intended to be a dynamic representation system, capable of handling each element as a "living" variable, and transmitting such dynamic character to the music that it represents. As examples of its applicability, the paper presents the Evscore Ontology, the implementation of EVcsound, a tool for the creation of detailed compositions with flexible temporal representations, and finally an example of algorithmic composition upon the EV Model.

1. The Representation of Musical Knowledge

A suitable knowledge representation tool (KR) is fundamental for a successful Artificial Intelligence system development. Brachman explains the KR concept with these words [Brachman, Levesque, 1985]: "It simply has to do with writing down, in some language or communicative medium, descriptions or pictures that correspond in some salient way to the world or a state of the world". The effectiveness of the intelligent system will depend, to a great extent, on how that KR fits the problem domain.

It can therefore be established that the first step for the design of an effective musical composition system is the definition of a musical KR that is appropriated to the domain and to the creative manipulations within the domain.

Conventional music notation allows for communication between the composer and the interpreter. But in this case, the compositional knowledge corresponding to the

mental abstractions of the composer is not represented explicitly. The compositional techniques, their experience, their creative processes and their intentions are not clearly represented by conventional music notation. In order to study what this compositional knowledge consists of, an analysis of the composition and the creative processes involved should be developed.

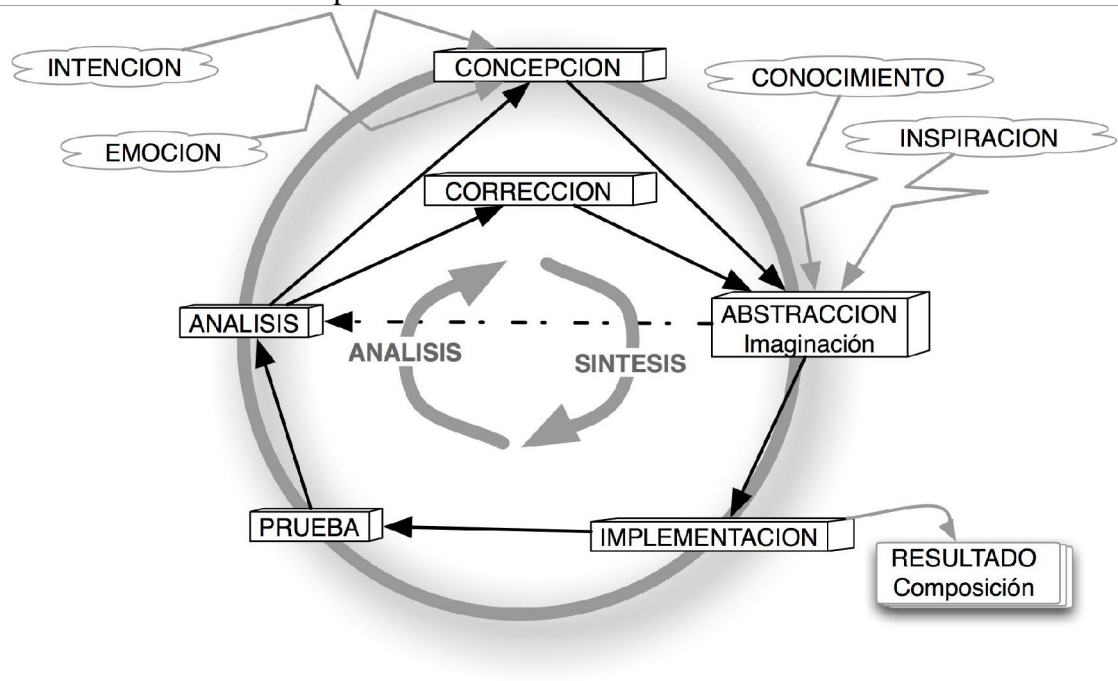


Figure 1. Subprocesses Cycle in Composition

Composition is comprised of subprocesses in the composer abstraction, such as conception, abstraction, imagination, implementation, analysis and correction. Figure 1 is a representation of an analysis model of composition processes. It is shown as a cyclic process of subprocesses. Starting from a voluntary intention or from an emotion, an element is conceived. Following it is imagined, and abstracted in the context from both experience and musical knowledge, and also, why not, from inspiration. Once imagined, the element is concretized, implemented, and tested. Often, the implementation is not explicit, but a mental visualization of the result is enough. Sometimes drawings and sketches are used for testing and analysis. Once evaluated, the affections to the rest of the composition are analyzed. As a consequence, new corrections are conceived starting a new cycle again. Every musical element or structure in the composition could be seen as a product of this kind of synthesis-analysis cycle.

In this analytic search of musical knowledge, several components have been identified. They include entities, relationships, procedures, strategies, and metaknowledge. They are explained in table 1. We can conclude that the musical knowledge comprises aspects, elements, procedures and strategies brought into play by the composer during the music creation process.

Table 1. Components of Musical Knowledge

<i>Elements and entities,</i>	Conceptualization and identification of elements and abstract objects arranged into ontological classes. These elements might be considered at different levels, from atomic elements such as notes, to more and more complex structures constructed by combinations of simpler elements. Objects and relationships
<i>Rules, patterns, constraints</i>	Relationships between the entities above, creating a definition of musical language and style.
<i>Intention driven procedures</i>	Procedures to develop the musical discourse, the temporal evolution, the narrative line.
<i>Rules breaks, originality</i>	Exceptions to the rules, patterns, creative innovations at all levels of relationships and structures. Providing liveliness, originality and freshness to the discourse.
<i>Strategies</i>	Based on experience, they constitute a whole heuristic of composition as a solution search
<i>General Criteria, Global Intentions, Finality</i>	Criteria above the composition process, as motivation, aesthetics and intention for composing. Some criteria as attention attraction, surprise, equilibrium. Meta-knowledge.

2. Paradigms in Music Representation

2.1. The score paradigm

The traditional musical score can be considered as a set of symbols arranged in a temporal succession, representing musical entities such as notes, durations and pitches. All annotations in the score are symbolic representations of instructions indicating what the interpreters should do in order to play the music. In this sense, the score is regarded as a good representation for the performance of music. One could hardly consider musical scores as a comprehensive KR method because information about the composition itself has to be deduced by means of an objective analysis of the score or obtained through subjective interpretation that is part of the listening process.

Being the performance instructions specific to different types of musical instrument, the composition is not completely defined without the specification of the instruments and the contribution of the interpreters. Therefore, attaching the corresponding orchestra completes the definition of a composition. Considered from the point of view of its function, an orchestra could be defined as a sound generator that has some knowledge about the interpretation of the score and its translation into acoustic waves. This *Orchestra-Score metaphor* is implemented in numerous sound synthesis systems, such as MusicV and its descendants Csound [Vercoe, 1994], CLM [Schottstaedt, 1994] and a few others. In Csound, for example, the composition is split into the score file (.sco) and the orchestra file (.orc).

2.2. The unit generator (UG) paradigm

The architecture of the first analog synthesizers has greatly influenced the development of sound synthesis systems. Those machines required the physical interconnection of small operating units called unit generators (UG). By means of the combination of UGs, multiple architectures with different performance could be assembled.

This philosophy has been traditionally applied in numerous systems for musical composition and sound synthesis. For example, in Csound instruments are programmed using variables to connect basic op-codes building instruments as more complex operating elements. In CLM and Nyquist [Dannenberg, 1997] the interconnection of UGs is defined using the LISP language, and in some other systems, such as PD [Puckette, 1997], these connections are done by means of a graphical interface.

From a point of view of KR, those structures could be represented as elements of the type ‘operating-unit-in the-time-domain’, defined as an interconnection of simpler units. It is interesting to note, however, that the UG paradigm allows for the definition of high-level structures starting from similar structures in a lower level.

2.3. The MIDI paradigm

MIDI standardization has unquestionably greatly influenced the development of computer music. In music representation, it considers that music is performed with a keyboard. This simplification provides the possibility of representing every musical note just with a number associated with the key being pressed, the pressing velocity, and when and for how long it is pressed. MIDI has been useful as a music representation for the last decades, but it cannot be considered as a complete representation system, as it does not include any information concerning the orchestra. Although some attempts have been made to standardize some orchestras (e.g., General MIDI), the impossibility of unifying MIDI players and its low resolution, prevent MIDI from being an efficient tool for representing musical sound. However, it provides the possibility of developing systems with low computational costs. Within algorithmic composition, systems such as Common Music (CM) [Taube, 2004] or Symbolic Composer (SCOM) [Stone, 1997] are examples of MIDI-based systems implemented in LISP.

3. From Composer To Listener

We consider the musical phenomenon as a communication process starting from the abstractions and emotions of the composer towards the ear of the listeners. Figure 2 shows the different representations that musical messages go through from the abstract world level, down to the acoustic level. The acoustic level can be reached through two possible ways.

In the traditional way, a human interpreter plays the conventional score found in the notation level. Composers write the score from mental abstractions. Usually, compilation does not take place directly, but sketches, scripts, plan drawings, drafts and many other methods are used. This is a long elaboration process where composers repeatedly go back to higher levels to perform adjustments and make finer conceptualizations of their abstractions. It is an iterative and feedback debugging

process, where many different paths are tried out and their results evaluated. From the point of view of Artificial Intelligence, this process could be considered as heuristic searches for the satisfaction of self-imposed constraints.

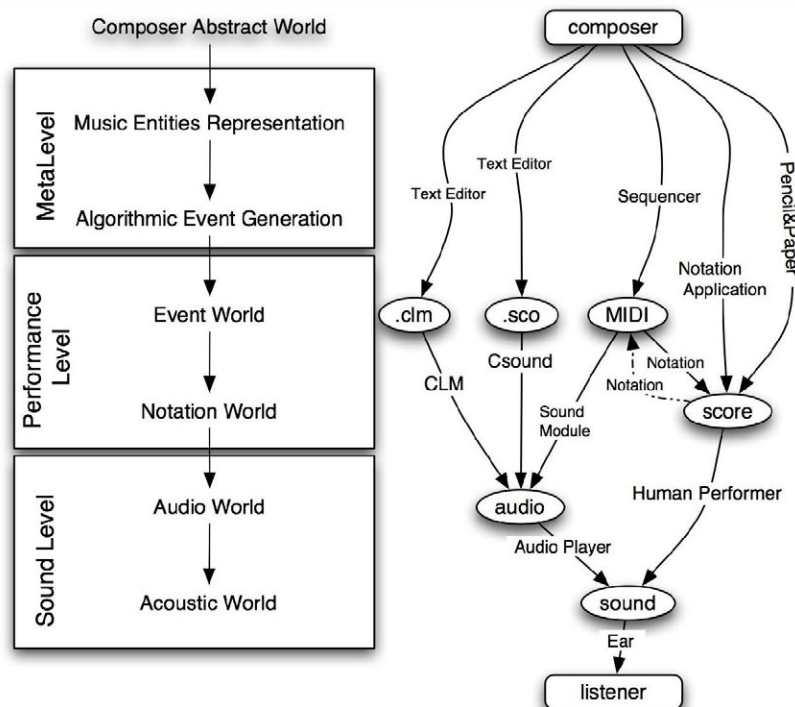


Figure 2. Levels of representation in musical communication

In conclusion, it is a time consuming work developed at levels above the level of the interpretation languages.

The other way of producing sound comes from a stratum immediately above, the audio world. A speaker, reproducing audio samples, now emits sound. Sound files are created based on performance events at the interpretation level like MIDI, *Csound*, *sco* or any other event list created by the composer.

3.1. The metalevel paradigm

The interactive composition process described for the creation of the score is similarly applicable in this case. The metalevel is not seen here just as a layer in the levels of structure, but as a stratified zone capable of supporting different architectures. In that area, we could place the following:

- Newell's "*knowledge level*" [Newell-1982], corresponding to the symbolic level of the score, where the musical entities of score are situated.
- Conceptualizations of musical elements in the dimensions of time and form, not directly represented in the score, such as motives, sequences, form structures and their relationships.
- Extra-score conceptualizations in the spectral or pitch dimension, such as intervals, chords, harmonic elements, tonalities and so on.
- New "*musical meta-objects*" at closer to composer levels.
- Elements and procedures of algorithmic musical generation.

The metalevel is an attractive area for the development of compositional tools. It is, however, necessary to use an efficient KR that is flexible and able to cope with new entities at several levels, and which can integrate new musical elements and structures created by the composer.

Music created with computers is often regarded by the general public as mechanical and lifeless. Note that in conventional music communication, both the composer and the interpreter contribute their human dimension to the music. In this case, the final sound carries creative contributions and “beautiful imperfections”, which manifest the richness and liveliness of its human origin. This is a target not to be forgotten when designing a representation system for musical composition.

4. EV Meta-Model: a Multilevel Music Representation

EV Meta-Model is proposed as a KR for elements in time at different levels. One of the key points of this approach is its simplicity. Figure 3 shows the core of our ontology, consisting on three main classes: the *event*, the *parameter* and the *dynamic object*. The definition of the classes is given as follows (in LISP notation):

```
(define-Class EVENT :is-a evclass
  :slots ((start-position :type real)
          (length :type real)
          (parameters :multiple list :type parameter)
          (events :multiple list :type event)
          (position-function :type function)))

(define-Class PARAMETER :is-a evclass
  :slots ((name :type string)
          (value :type dynamic-object)))

(define-Class DYNAMIC-OBJECT :is-a evclass
  :slots ((type :doc "Type of the out value")
          (out :type t)
          (dyn-function :type function)
          (dyn-arguments :multiple list :type DYNAMIC-OBJECT)
          (status-memory :type t :doc "State memory")
  ))
```

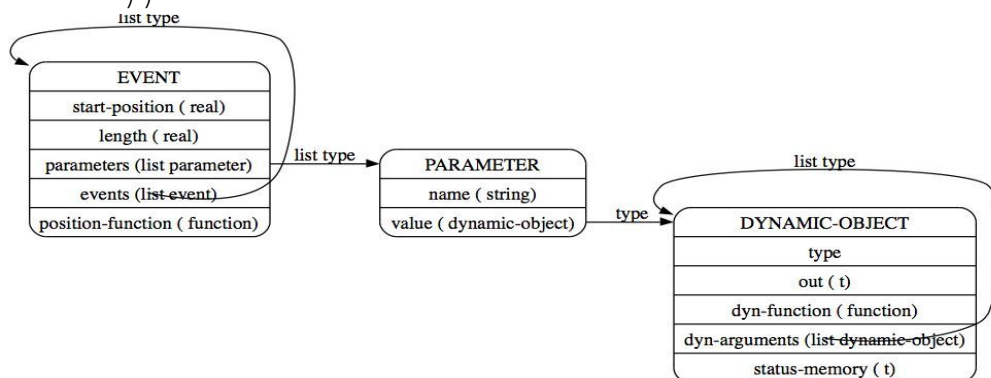


Figure 3. Ontological Core of EV Meta-Model.

4.1. The *event* class

This class comprises any kind of element that can be positioned in time. It is the central class of the ontology and it includes five main slots:

- *Start-position* is the time position of the event, the time position where it starts.
- *Length* indicates the active time, or the duration of the event.
- *Parameters* is a list of the properties of an object. Those properties (or parameters of a musical event) are instances of the class "parameter". Each parameter includes a parameter name and a dynamic value of type "*Dynamic-Object*", which is described in next paragraph.
- *Events* is the most important slot. It makes recursivity possible. Therefore it provides a multiple level representation capability. It contains a list of new events sharing the same object structure; it is a list of instances of its same class *event*. A "child-event" could itself contain new "child-events" and so on, thus creating a rich recurrent data tree from a single class. This concept drastically simplifies the definition of time structures at different levels and their relationships. A general behaviour could also be defined by generic functions. It is, therefore, a flexible ontological skeleton capable of representing almost any musical element as an instance of the class *event*. It keeps coherence from the notes level up to higher abstraction levels.
- *Position-Function* is another interesting feature that makes this approach unique. Each event could have its own time magnitude and scale with a dedicated handling. This means that there is an internal time inside the event, defined by its *Position-Function*, and a different time perception from the outside, which corresponds to the time organization of its "mother-event". That timing flexibility multiplies the creative possibilities and, at the same time, solves several practical problems. Let's see some examples:

1. Let us consider a practical example in film scoring. A film would be seen as an instance of the class *event*, using *SMPTE timecode* as time organization. A musical cue for the score is an *event* that begins at a specific timecode and ends at another, but it uses bars and meter from traditional notation for its internal time organization. All internal events in this cue, such as notes, are defined within this bar organization. In this example, the musical cue owns a *Position-Function* or time function called "tempo", which translates those time magnitudes. In live performance, the conductor will take the control of that Position-Function. The following code shows a possible implementation by subclass definitions:

```
(define-class FILM :is-a event
  :slots ((position-function :default #'smppte2real)
          (events :multiple list :type MUSIC-CUE)
        ))
(define-class MUSIC-CUE :is-a event
  :slots ((master-track :type master-track )
          (position-function :default #'meter2smppte)
          (events :multiple list :type MUSIC-SEGMENT)))
(define-class MUSIC-SEGMENT :is-a event)
```

2. A previously defined musical motive could be used at several moments of the piece at different speeds, by augmentation or diminution: a generalization of the traditional counterpoint procedure. For example, in order to decrease the speed by two, it is enough to provide the function *#double* as position-function. If we use the function *#half* we would get the counterpoint diminution.
3. Once the piece is completed, its timing could be reshaped, enlarging and shortening some sections by modifying the main position-function. In film scoring, it could be useful in many situations. Let us consider the last-minute director's change in the edition of an already orchestrated scene; some small adjustments in the position function could be enough to fit the new timing.

4.2. Dynamic Objects

The parameter value of the *event* is not static. It is considered to be a *dynamic object* with an evolving value. Data is dynamic during the event, so a parameter could be considered as "living" entities. Several possibilities are available for defining the evolution of the dynamic object. In order to simplify definitions, a set of elementary *dynamic objects* and an intuitive syntax to describe combinations was developed inspired by CYBIL [Piché, Burton, 1995]. Complex objects built from simpler ones will keep the dynamic character. The set of elementary units includes *lines* between an initial and a final value (the beginning and ending of the event), *logarithm* curves, *random* values within a range, *sequences* of values, *functions*, memory provided *generators*, etc. Recursivity is again present here because arguments included in the definition are also considered as instances of *dynamic objects*. In addition, dynamic objects are conceived with memory capability.

Both dynamic arguments and memory, provides the possibility of creating very complex dynamic objects, such as evolving systems, cellular automata and so forth.

4.3. Generator events

EV Meta-Model also provides the events with *behavior*. A subclass can be defined from the main class *event* with a specific behaviour. Generators are a special subclass type, which are useful for the abstraction of musical forms. A generator is an *event* that develops itself into new events of a particular subtype. From a musical point of view, generators provide the possibility of compiling a "musical piece" *event* instance into a score of interpretation events iteratively.

5. EVScore Ontology

Figure 4 shows the *EVScore Ontology*, an example based upon our proposal. It is a traditional notation compatible representation, which can integrate higher level music elements. Every class in this ontology is a descendant of the core class *event*, so they inherit all the properties of the Meta-Model.

Figure. 4. EVScore Ontology



EVcsound compiles a composition-event into a Csound .sco score. Every *csound-event* in the .sco file is associated with an instrument of the .orc file by an identifier and it consists of a list of numeric figures corresponding to parameters $p4$ to p_n for that instrument (p1 is the instrument ID, p2 the start position and p3 the duration of the event). The key point of the system is the *sampler-generator* type event, which samples the dynamic-objects along its development process and builds up every *csound-event* with such figures.

```
(evcsound '(0 20 superfm
            ( period (ran f .2 2)
              amplitud-db (ran f (li f 50 70)
                           (li f 60 80))
              duration (op #'/ 300 (pa amplitud-db))
              pan (ran f (lo f 0 .5) (lo f 0 .99))
              pitch (op #'* 55 (ran i 6 16))
              reverb-send (lo f .07 .7)
            0 25 reverb-st
            ( reverb-time 7))))
```

The first generator, lasting for 20 seconds, develops into multiple *csound-events* for an instrument named "*superfm*", with a random sampling period between 0.2 and 2 seconds, a random amplitude in the evolving range between 50-60 dB and 70-80 dB, a random pan evolving from *closed L* to a range between C and R, a duration inversely proportional to amplitude between 3 and 8 seconds and a randomly chosen harmonic pitch between the 6th and the 16th harmonics of A1 (55Hz). The second event, lasting

for 25 seconds, activates the reverb instrument. This example presents just one step down in the hierarchy; a compilation from such a high level can be easily achieved by defining recursive events.

7. An Algorithmic Composition Example

As a last example of applicability of the EV framework, a simple example of algorithmic composition of a melody is presented. In this example we also introduce three new elements of our system: tables, zones and maps.

Tables are used as a new kind of dynamic object. They can store the output of sequential objects and use them as a linear function. In our example, a random *brownian* shape is stored in a table to be used repeatedly.

Zone is a particular subclass of *generator event* inside which, a detailed time structure is arranged into zones. Categories for every zone in the structure are simultaneously defined, so it is easy to compare among them and set a desired evolution for that category across the zones. All category definitions are interpreted within the zone structure, so it is possible to address the desired zone by writing according structures in the definition. If just one dynamic object is provided, it is applied for the entire event. If a nested list is provided, each object is associated with the "same level and order" zone element. The zone-event generator can be developed into child events according with the defined parameters in zones.

Maps are dynamic objects that allow you to use symbol lists for representing recurrent structures of data in a simpler manner. Symbols are easily combined and manipulated to define a sequence of elements generating a musical form. This concept can be extended by the use of grammars to generate complex and rich forms. In the example below, they are defined by the dynamic function *mp*.

```
(deftable t1 256 (br 0 1 :seed .42723))

(define-zone-event my-melody
  :zone '((a b b c) ( a b c b c))
  :ryth (mp zone '("+++" "+--" "+---"))
  :shape (mp '(a b) (list t1 (retrograde t1)))
  :tessitura (tab t1 :min 0 :max 12)
  :ambitus (li i 5 8)
  :pitchclass 'glydian
  :step0pitch 'g4
)
```

The example code above defines a melody as a zone-event. In this subclass, the melody generation is developed by the algorithm represented in figure 5: A melody shape is obtained by placing a shapes structure within a defined tessitura and ambitus. That melody shape is then sampled by a rhythm and step quantified in some intelligent way. The resulting step values are then converted to pitches by mapping them into a given pitchclass and register.

8. Conclusions

The creative possibilities of a musical system are determined by the underlying knowledge representation and the paradigm in which it emerges.

Although the traditional score is a good representation for performance, actual musical knowledge extends far beyond the traditional musical score. Musical knowledge includes elements and entities, rules and constraints, intention driven procedures and creative breaks of patterns and rules. Musical representation could be stratified into levels spanning from the composer abstraction to the acoustic representation. We are interested in addressing the zone above performance or metalevel. A successful knowledge representation for composition should be flexible and simple, but still providing coherent representation at different levels.

EV Meta-Model emerges in this research enquire as an intuitive multilevel music representation system. Its design is based on both structure and function recursivity, making it appropriate in multiple levels and magnitudes. EV Meta-Model provides a dynamic data structure, being able to successfully implement musical ideas efficiently. This approach introduces innovative aspects to music computing research. One of the key points in its design is the unification of all musical entities in a single data type, designed specially for music.

9. References

- Brachman, R.J. and Levesque, H.J. (Editor) (1985). *"Readings in Knowledge Representation"*, San Mateo, CA: Morgan Kaufmann.
- Dannenberg, R.B. (1997). *"Machine tongues XIX: Nyquist, a language for composition and sound synthesis"*, *Computer Music Journal*, 21(3):50-60.
- Newell, A. (1982). *"The knowledge level"*, *Artificial Intelligence*, 18(1):82-127.
- Piché, J. and Burton, A. (1995) "CYBIL Language" Université de Montréal
<http://emu.music.ufl.edu/cecilia/cybil.html>
- Puckette, M. (1997). *"Pure Data"*, *Proceedings of ICMC*, Thessaloniki, Greece, pp 224-227.
- Schottstaedt, B. (1994). *"CLM: Music V Meets Common Lisp"*, *Computer Music Journal*, 18(2): pp 30-37.
- Stone, P. (1997) *"Symbolic Composer"* <http://www.symboliccomposer.com>
- Taube, H. K. (2004) *"Notes from the Metalevel"*, Andover UK: Swets & Zeitlinger.
- Vercoe, B.L. (1994). *Csound: A Manual for the Audio-Processing System*, Boston, MA: MIT Media Lab.

Classificação Automática de Gêneros Musicais Utilizando Métodos de *Bagging* e *Boosting*

Carlos N. Silla Jr. , Celso A. A. Kaestner , Alessandro L. Koerich

Pontifícia Universidade Católica do Paraná (PUC-PR)
Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Rua Imaculada Conceição 1155, 80215-901
Curitiba - PR - Brasil

{silla,kaestner,alekoe}@ppgia.pucpr.br

Abstract. *This paper presents a study that uses meta-learning techniques to the task of automatic music genre classification. The meta-learning techniques we used are Bagging and Boosting. In both cases the component classifiers used in both approaches are Decision Trees, k-NN (k nearest neighbors) and Naive Bayes. The experiments were performed on a dataset containing 1.000 songs with 10 different genres. The achieved results show that the Bagging approach is promising while the Boosting approach seems to be inadequate to the problem.*

Resumo. *Este artigo apresenta um estudo utilizando técnicas de meta-aprendizagem para o problema de classificação automática de gêneros musicais. As técnicas de meta-aprendizagem utilizadas foram Bagging e Boosting. Em ambos os casos usando como classificadores componentes os algoritmos de Árvores de Decisão, k-NN (k vizinhos mais próximos) e Naive Bayes. Os experimentos foram realizados utilizando uma base contendo 1.000 músicas de 10 gêneros diferentes. Os resultados obtidos com o algoritmo de Bagging foram positivos, enquanto o uso do método de Boosting apresentou resultados aquém dos esperados.*

1. Introdução

A quantidade de conteúdo multimídia disponível on-line criou uma necessidade de ferramentas capazes de organizar e gerenciar essa grande quantidade de informações [Fingerhut, 1999] [Pampalk et al., 2002]. No momento, a maior parte das informações sobre dados multimídia são classificadas e organizadas baseadas em meta-informações textuais que são associadas ao seu conteúdo, como é o caso dos tags ID3 nos arquivos de áudio com o formato .mp3. Apesar destas informações serem relevantes para as tarefas de indexação, busca e recuperação, elas são geradas manualmente e então associados com o arquivo multimídia.

A música digital é um dos mais importantes tipos de dados distribuídos na Internet. Como organizar e processar essa grande variedade e quantidade de dados de maneira eficiente para permitir indexar, buscar e recuperar é um grande desafio [Foote, 1999] [Guo and Li, 2003]. Existem muitos estudos e métodos sobre a análise de conteúdo de áudio usando diferentes características e métodos [Pampalk et al., 2002], [Guo and Li, 2003], [Zhang and Kuo, 2001], [Liu et al., 2003], [Aucouturier and Pachet, 2003].

Apesar dos esforços de vários pesquisadores, a classificação automática de áudio é realizada com relativa precisão somente em problemas simples, como diferenciar fala

e música [Carey et al., 1999]. Poucos trabalhos tratam com a tarefa de classificação automática de gêneros musicais [Pye, 2000], [Kosina, 2002], [Tzanetakis and Cook, 2002], [Liu et al., 2003], [Aucouturier and Pachet, 2003]. Um gênero musical é uma descrição importante que tem sido utilizado para classificar e caracterizar músicas digitais e para organizar grandes coleções disponíveis na Web [Tzanetakis and Cook, 2002] [Shao et al., 2003] [Liu et al., 2003], [Aucouturier and Pachet, 2003]. Os gêneros musicais são rótulos categóricos criados por especialistas humanos assim como por amadores para determinar títulos de músicas. Esses rótulos são relacionados com a instrumentalização utilizada, estrutura rítmica e conteúdo harmônico da música. Entretanto, um gênero musical é um conceito relativamente subjetivo, e mesmo a indústria musical muitas vezes é contraditória ao atribuir gêneros musicais para as músicas. Uma prática muito comum é que as músicas são categorizadas de acordo com o perfil do artista. Adicionalmente a classificação de músicas tem sido desenvolvida normalmente para álbuns, e não é aplicável diretamente para as faixas do álbum [Aucouturier and Pachet, 2003]. Dessa forma a classificação automática de gêneros musicais pode auxiliar ou substituir o usuário humano nesse processo, assim como prover um componente importante para um sistema de recuperação de informações para músicas.

A classificação automática de gêneros musicais é um tema relativamente novo e nos últimos anos tem atraído a atenção de diversos pesquisadores. Pye [Pye, 2000] usou os coeficientes cepstrais de frequência-Mel (*Mel-frequency cepstral coefficients* - MFCC) e um *Gaussian mixture model* para classificar músicas em cinco classes: blues, easy listening, classic, opera, dance e rock. Tzanetakis e Cook [Tzanetakis and Cook, 2002] propuseram um conjunto de características para modelar os sinais musicais que estão relacionadas a textura timbral (*timbral texture*), ritmo (*rhythm*) e variações da frequência da vibração (*pitch*). Nesse trabalho eles utilizaram dois algoritmos de classificação (Gaussian mixture model e k-NN) aplicando as características extraídas. Shao [Shao et al., 2003] usou um modelo de classificação não supervisionada baseada em Cadeias Escondidas de Markov (HMMs). Liu et al. [Liu et al., 2003] propôs um novo método de extrair características baseado nos histogramas computados utilizando os coeficientes da Daubechies Wavelet. Para classificação eles utilizaram diferentes algoritmos como: Support Vector Machines (SVM), k-NN, *Gaussian mixture models* e análise discriminante linear (*Linear Discriminant Analysis*). Uma breve revisão dos trabalhos anteriores na área de classificação automática de gêneros musicais mostra que o interesse atual é o desenvolvimento de novos conjuntos de características e a utilização de diferentes classificadores. Um ponto comum nos trabalhos anteriores é que eles utilizam sempre um único classificador.

Neste trabalho é proposta uma nova abordagem para o problema de classificação automática de gêneros musicais utilizando a combinação de classificadores componentes através de técnicas de meta-aprendizagem. Neste trabalho foram utilizadas as técnicas de *Bagging* [Breiman, 1996] e *Boosting* [Mitchell, 1997] com o algoritmo AdaBoostM1 [Freund and Schapire, 1996]. O restante deste trabalho está organizado da seguinte maneira: na seção 2 é apresentada uma visão geral do sistema; na seção 3 são apresentadas as características utilizadas; na seção 4 são apresentadas breves descrições dos métodos de meta-aprendizagem empregados; na seção 5 são apresentados os experimentos realizados e os resultados obtidos; e finalmente na seção 6 são apresentadas as conclusões deste trabalho.

2. Visão Geral do Sistema

O sistema de classificação de gêneros musicais utilizando técnicas de meta-aprendizagem é composto de dois módulos principais (Figura 1): um módulo para a extração de carac-

terísticas e um módulo de classificação de gêneros musicais. Inicialmente um conjunto de características é extraído a partir dos trinta primeiros segundos do sinal de áudio. Ao todo são extraídas trinta diferentes características que formarão um vetor de características.

Por se tratar de um sistema que utiliza algoritmos de classificação, o sistema opera em dois modos: treinamento e teste. No modo de treinamento os vetores de características são utilizados com seus respectivos rótulos (neste contexto o gênero da música) para treinar os algoritmos de classificação utilizados pelas técnicas de meta-aprendizagem *Bagging* e *Boosting*. Os rótulos consistem na informação textual representando o gênero musical atribuído àquela música por especialistas humanos. No modo de classificação, a música cujo gênero é desconhecido é fornecida ao sistema. Um vetor de características é gerado a partir dos trinta primeiros segundos da música e este vetor é passado ao algoritmo de classificação que atribui um gênero à música.

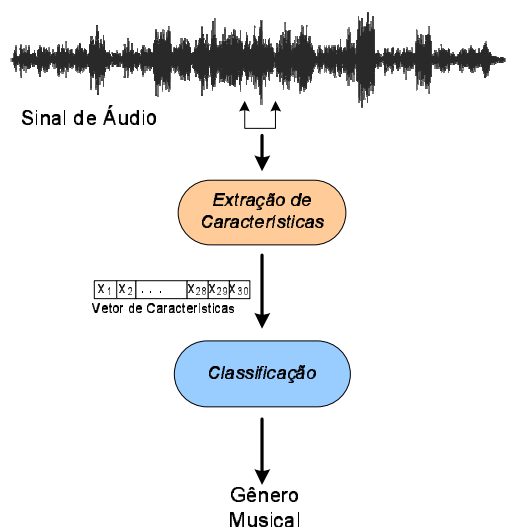


Figura 1: Visão Geral do Sistema

3. Extração de Características

O conjunto de características utilizadas neste trabalho foi originalmente proposto por [Tzanetakis and Cook, 2002] e utilizado em outros trabalhos [Kosina, 2002]. São considerados três tipos de características: textura timbral (*Timbral texture*), relacionadas à batida (*beat-related*) e relacionadas às variações da frequência da vibração (*pitch-related*). Características de textura timbral incluem a média e a variância do centróide espectral, do *rolloff* espectral, do fluxo espectral, das taxas de cruzamento zero, coeficientes cepstrais de frequência-Mel (MFCC), e da baixa energia. Características relacionadas à batida incluem as amplitudes relativas e as batidas por minuto. As características relacionadas ao pitch incluem os períodos máximos do pico do pitch nos histogramas. Estas características formam vetores de 30 dimensões (Textura Timbral: 9 FFT + 10 MFCC; Ritmo: 6; Pitch: 5) que posteriormente são utilizados no treinamento dos diferentes classificadores de maneira supervisionada. A seguir são descritas as características extraídas das músicas.

3.1. Textura Timbral

Centróide Espectral (*Spectral Centroid*) é o ponto balanceado do espectro. É uma medida da forma espectral e é associado frequentemente com a noção do brilho espectral. O centróide espectral pode ser calculado como apresentado na equação 1.

$$C_t = \frac{\sum_{n=1}^N M_t[n] * n}{\sum_{n=1}^N M_t[n]} \quad (1)$$

onde $M_t[n]$ é o valor da transformada de Fourier no quadro t e faixa de frequência n . O centróide espectral é um atributo perceptual importante na caracterização do timbre musical de instrumentos.

Rolloff Espectral (*Spectral Rolloff*) é outra medida da forma espectral que é definida como a frequência R_t apresentada na equação 2 na qual 85% da magnitude da distribuição está concentrada.

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 \sum_{n=1}^N M_t[n] \quad (2)$$

Fluxo Espectral (*Spectral Flux*) é uma medida da mudança espectral local e é definido como apresentado na equação 3.

$$F_t = \sum_{n=1}^N (N_t[n] - N_{t-1}[n])^2 \quad (3)$$

onde $N_t[n]$ é o valor normalizado da transformada de Fourier na janela t .

Taxas de Cruzamento Zero (*Time Domain Zero-Crossings*) é uma característica que ocorre quando as amostras sucessivas têm sinais diferentes. É calculada como apresentada na equação 4.

$$Z_t = \frac{1}{2} \sum_{n=1}^N |\text{sign}(x[n]) - \text{sign}(x[n-1])| \quad (4)$$

onde $x[n]$ é o sinal no domínio do tempo e a função sign é 1 ou 0 para os argumentos positivos e negativos respectivamente. Ao contrário do centróide espectral, do rolloff espectral, e do fluxo espectral, que são características no domínio da frequência, a taxa do cruzamento zero é uma característica no domínio do tempo.

Coefficientes Cepstrais da frequência Mel (*Mel-frequency cepstral coefficients*) são características perceptualmente motivadas que também são baseadas na transformada de Fourier (*Short Time Fourier Transform* - STFT). Após obter a amplitude logarítmica da magnitude do espectro, as faixas pré-determinadas são agrupadas e suavizadas (*smoothed*) de acordo com a motivação perceptual da escala da frequência Mel. Finalmente, para des-correlacionar os vetores de características resultantes, uma transformada discreta de cosseno é utilizada. Apesar de normalmente treze coeficientes serem utilizados para representar a fala, experimentos mostram que os cinco primeiros coeficientes levam a um melhor desempenho para a classificação de gêneros musicais [Tzanetakis and Cook, 2002].

Análise e Janela de Textura Em análise de áudio o sinal é quebrado em pequenos segmentos de tempo sobrepostos e cada segmento é processado separadamente. Esses segmentos são chamados de janela de análise e devem ser pequenos o suficiente para que

as características de frequência do espectro de magnitude sejam relativamente estáveis. Entretanto a sensação de textura do som surge como resultado de múltiplos espectros de tempo curto com diferentes características seguindo algum padrão no tempo. Por exemplo, a fala contém vogais e consoantes as quais tem diferentes características espectrais.

Logo, de forma a capturar a longa natureza da textura do som, as características computadas são médias e variâncias das características descritas anteriormente nessa seção, em um número de janelas de análise. O termo janela de textura é utilizado para descrever essa janela maior e idealmente deve corresponder ao mínimo de tempo de som que é necessário para identificar a textura de um som ou de uma música. Essencialmente, ao invés de usar os valores das características diretamente, são calculados os parâmetros de uma distribuição gaussiana multidimensional. Mais especificamente, os parâmetros (médias, variâncias) são calculados com base na janela de textura que consiste no vetor de características atual em adição a um número específico de vetores de características do passado.

Baixa Energia (*Low Energy*) é calculada sobre um número de janelas com a média e variação, e não separadas para cada janela como as outras características. A característica energia baixa é definida como a porcentagem das janelas que têm menos energia do que a energia média de todas as 40 janelas. Por exemplo, sinais musicais terão energia mais baixa que sinais de fala que normalmente contêm muitas janelas silenciosas.

Com as características apresentadas nesta seção, a textura timbral de uma música consiste nas seguintes características: médias e variâncias do centróide espectral, do rolloff espectral, do fluxo espectral, das taxas de cruzamento zero sobre a janela da textura (8), baixa energia (1) e as médias e variâncias dos cinco primeiros coeficientes MFCC sobre a janela de textura resultado assim em um vetor de características com dezenove dimensões.

3.2. Características Relacionadas à Batida (Beat-Related)

A batida e a estrutura rítmica de uma música é freqüentemente uma boa indicação do gênero. Por exemplo, dance music tende a ter uma batida principal muito forte e distintiva. A música clássica, geralmente não tem uma batida dominante e regular desobstruída, devido à complexidade do arranjo. A extração da característica da batida tenta encontrar a batida principal da música e de seu período em BPM (batidas por minuto). Além desta, é calculada também a batida mais forte, e um número de características relacionando a primeira e segunda batida.

Inicialmente o sinal é decomposto em um número de bandas de frequências usando uma transformada Wavelet discreta [Sweldens and Piessens, 1993]. Após essa decomposição, uma série de passos para a extração do envelope da amplitude no domínio do tempo é aplicada a cada banda: retificação de onda completa, filtragem passa-baixa, *downsampling* e remoção das médias [Kosina, 2002, Tzanetakis and Cook, 2002].

Após o passo da extração, os envelopes de cada banda são somados e a autocorrelação resultante é calculada. Esse resultado é uma função de autocorrelação onde os picos (*peaks*) dominantes correspondem ao tempo de lag (*time lags*) onde o sinal tem a auto-similaridade mais forte. Os primeiros três picos da função de autocorrelação são adicionados ao histograma de batida. Cada banda do histograma corresponde a um período da batida em BPM. Para cada um dos três picos selecionados, a amplitude do pico é adicionada ao histograma. Esse procedimento é repetido para cada janela de análise. Os picos mais fortes no final do histograma correspondem às batidas mais fortes do sinal. Seis características são calculadas usando o histograma de batidas:

- A amplitude relativa (i.e. a amplitude dividida pela soma de amplitudes) do primeiro e do segundo picos no histograma de batidas. Essa é uma medida de quão distintas são as batidas comparadas com o resto do sinal.
- A razão da amplitude do segundo pico dividido pela amplitude do segundo pico dividida pela amplitude do primeiro pico. Essa característica expressa a relação entre a batida principal e a primeira batida auxiliar.
- O período do primeiro e segundos picos em BPM, indicando quão rápida a é a música.
- A soma do histograma, a qual pode ser um indicador da força da batida. A soma das bandas do histograma é uma medida de força da auto-similaridade entre as batidas, a qual é um fator de quão rítmica uma música parece ser.

3.3. Características Relacionadas ao Pitch (Pitch-Related)

O conjunto de características de conteúdo pitch é baseado em múltiplas técnicas de detecção de pitch. Nesse algoritmo, o sinal é decomposto em duas bandas de frequência (abaixo e acima de 1.000 Hz) e envelopes de amplitude são extraídos para cada banda da frequência. A extração do envelope é realizada aplicando retificação de meia onda e filtro passa-baixa. Os envelopes são somados e uma função “aumentada” de autocorrelação é computada para que o efeito de múltiplos inteiros no pico das frequências para múltiplos pitch’s detectados sejam reduzidos.

Os picos proeminentes dessa função de autocorrelação “aumentada” correspondem aos principais pitches para aquele curto segmento de som. Esse método é similar a detecção da estrutura de batidas para curtos períodos correspondendo a percepção de pitch. Os três picos dominantes são acumulados em histogramas de pitch sobre todo o sinal de áudio. Para computar o histograma de pitch, é utilizada uma janela de análise de 512 amostras com taxa de amostragem de 22 050 Hz (aproximadamente 23 ms).

4. Classificação Utilizando Métodos de Meta-Aprendizagem

Com os vetores de características calculados na seção 3, é possível utilizar algoritmos padrões de aprendizado de máquina. Neste trabalho são utilizados os métodos de meta-aprendizagem de *Bagging* [Breiman, 1996] e *Boosting* [Mitchell, 1997] com os classificadores componentes de árvores de decisão [Quinlan, 1993] (versão Java J4.8), Naive Bayes [Mitchell, 1997] e k-NN [Aha et al., 1991].

O método de *Bagging* consiste em utilizar múltiplas versões de um conjunto de treinamento, cada versão é criada selecionando aleatoriamente $n' < n$ amostras do conjunto de treinamento D , com reposição. Cada um destas versões é utilizada para treinar diferentes “classificadores componentes” e a decisão da classificação final é baseada no voto de cada componente. Neste trabalho é utilizada uma abordagem tradicional, onde todos os empregam o mesmo método de classificação.

O método de *Boosting* tem como objetivo melhorar a precisão de qualquer algoritmo de aprendizagem. Para isso o procedimento utilizado é o seguinte: (1) é criado um classificador com precisão sobre o conjunto de treinamento; (2) cria-se um classificador com precisão sobre o conjunto de treinamento maior do que a média. Adicionalmente novos classificadores componentes são adicionados para formar um conjunto cuja regra de decisão tenha uma alta precisão arbitrária sobre o conjunto de treinamento. Neste trabalho foi utilizada uma variação do algoritmo de *Boosting* denominada AdaBoostM1 [Freund and Schapire, 1996].

Tabela 1: Resultados da Classificação Automática de Gêneros Musicais (base padrão)

Classificador	Individual	Pior Bag	Melhor Bag	Pior Boost	Melhor Boost
J4.8	52.1	57.4	66.9	59.9	68.1
k-NN (k=1)	49.3	47.1	52.8	49.3	49.5
k-NN (k=2)	45.4	48.8	54.3	46.4	47.2
k-NN (k=3)	49.9	50.2	55.0	46.3	46.3
k-NN (k=4)	49.8	51.0	54.7	49.8	49.8
k-NN (k=5)	50.2	50.4	54.7	50.2	50.2
Naive Bayes	57.0	57.2	59.4	56.6	56.6

Tabela 2: Resultados da Classificação Automática de Gêneros Musicais (base aleatória)

Classificador	Individual	Pior Bag	Melhor Bag	Pior Boost	Melhor Boost
J4.8	50.1	57.9	67.5	58.6	65.9
k-NN (k=1)	52.0	52.1	56.5	52.0	52.1
k-NN (k=2)	46.7	50.9	56.8	47.8	48.2
k-NN (k=3)	51.8	51.4	56.5	48.5	48.5
k-NN (k=4)	50.0	52.7	56.7	50.0	50.0
k-NN (k=5)	51.6	52.3	56.2	51.6	51.6
Naive Bayes	57.7	57.0	58.7	57.3	57.3

5. Experimentos

Uma base de dados contendo 10 classes (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae e rock) e 100 instâncias (ou músicas) de cada classe, totalizando 1.000 instâncias foi utilizada neste trabalho. Essa base foi a mesma utilizada nos experimentos de Tzanetakis e Cook [Tzanetakis and Cook, 2002]¹. Além dessa base padrão que contém as instâncias ordenadas por gênero, foi utilizada uma nova versão desta base com as instâncias de entrada ordenadas aleatoriamente no intuito de verificar a robustez dos métodos de meta-aprendizagem. Essa variação da base padrão foi chamada de base aleatória. É importante ressaltar que apesar da base ser utilizada ser a mesma que a utilizada em experimentos anteriores, uma comparação direta não é possível em virtude das diferenças nas condições experimentais.

Os experimentos foram realizados e avaliados com o método de validação-cruzada estratificada fator 10 (*Ten-Fold Stratified Cross-Validation*) e os resultados apresentados foram calculados utilizando a média das *F-measures* (calculada como a média harmônica dos valores de precisão e recobrimento) do resultado de cada classe. A precisão é o quociente entre o número de músicas que tiveram seu gênero classificado corretamente e o número de músicas que foram classificadas como sendo daquele gênero. O recobrimento é o quociente entre o número de músicas que tiveram seu gênero classificado corretamente e o número de músicas que possuem esse gênero.

No intuito de avaliar a eficácia de cada método de meta-aprendizagem, num primeiro momento, os experimentos foram realizados utilizando apenas um único classificador. Os resultados obtidos por cada classificador é apresentado na coluna *Individual* da tabela 1 para a base padrão e na tabela 2 para a base aleatória.

Os experimentos utilizando *Bagging* foram realizados visando determinar dois

¹Disponível em: <http://opihi.cs.uvic.ca/sound/genres>

parâmetros: o tamanho da cesta (*bag size*) que é a porcentagem de exemplos que devem ser selecionados aleatoriamente para compor cada um dos n' conjuntos de treinamentos e o número de classificadores componentes que deveriam ser utilizados. Para o tamanho da cesta foram utilizados os valores de 50%, 60%, 70%, 80% e 90%; e para avaliar o número de classificadores foram utilizados: 5, 10, 20, 30, 40 e 50 classificadores componentes em cada experimento. As colunas *Pior Bag* e *Melhor Bag* das tabelas 1 e 2 apresentam o pior e o melhor resultado obtido utilizando *Bagging* para a base padrão e base aleatória respectivamente. Nos experimentos utilizando *Boosting* foram utilizados o mesmo número de classificadores componentes utilizados nos experimentos de *Bagging*, ou seja: 5, 10, 20, 30, 40 e 50. As colunas *Pior Boost* e *Melhor Boost* das tabelas 1 e 2 apresentam o pior e o melhor resultado obtidos utilizando *Boosting* para a base padrão e base aleatória respectivamente.

Avaliação Geral dos Resultados

Os resultados obtidos utilizando apenas um único classificador para a tarefa de classificação automática de gêneros musicais mostram que em ambas as bases o classificador Naive Bayes apresenta uma taxa de classificação superior aos demais. Por outro lado, o método de *Bagging* apresentou um bom desempenho para os classificadores J4.8 e k-NN e resultados não tão bons para o classificador Naive Bayes.

No caso do J4.8 o método apresentou um desempenho surpreendentemente positiva. No pior caso teve um aumento na taxa de acerto de 5% (utilizando um *bag size* de 50% e 5 classificadores) e no melhor caso o aumento foi de 14.8% (utilizando um *bag size* de 90% e 50 classificadores) para a base padrão; e aumento de 7.8% no pior caso (utilizando um *bag size* de 50% e 5 classificadores) e de 17.4% no melhor caso (utilizando um *bag size* de 80% e 30 ou 40 classificadores) para a base aleatória.

Já os resultados obtidos pelo método de *Bagging* aplicado ao classificador k-NN apresentaram uma melhora razoável em relação ao uso de um único classificador k-NN. Os resultados obtidos mostram que o desempenho mínimo obtido utilizando a técnica de *Bagging* aplicada ao classificador k-NN obteve resultados equivalentes ao uso de um único classificador k-NN. As únicas exceções foram para 1-NN na base padrão e 3-NN na base aleatória, porém apresentando uma melhora de até 5% no melhor caso. Um fato interessante é que na base padrão, com exceção do classificador 2-NN, o menor resultado foi obtido utilizando o mesmo *bag size* (70%) e em todos os casos os pior resultado foi obtido utilizando o mesmo número de classificadores (5), assim como em quase todos os casos os melhores resultados foram obtidos utilizando um *bag size* de 50%. Já na base aleatória os melhores resultados sempre foram obtidos utilizando um *bag size* de 50% com pelo menos 30 classificadores enquanto os piores resultados foram sempre obtidos utilizando 5 classificadores e com um *bag size* de 80% ou 90%. Dessa forma é possível perceber uma relação no caso do k-NN que parece indicar que os melhores resultados são normalmente obtidos com um *bag size* de 50%, enquanto que os piores resultados sempre são obtidos com um *bag size* de 70% ou superior.

Os resultados apresentados nas tabelas 1 e 2 mostram que o uso do método de *Bagging* aplicado ao classificador Naive Bayes se mostrou praticamente ineficiente. No primeiro caso, utilizando a base padrão, o resultado mínimo foi de 57.2% utilizando um *bag size* de 60% e 5 classificadores. Esse valor é 0.2% maior que o resultado obtido por um único classificador Naive Bayes, que é de 57%. Já no melhor caso, o resultado máximo atingido foi de 59.4% utilizando um *bag size* de 50% e 40 classificadores. No segundo caso, utilizando a base aleatória, o resultado mínimo foi de 57% utilizando um *bag size* de 60% e 10 classificadores. Esse valor é menor do que o obtido utilizando

um único classificador Naive Bayes na mesma base, que é de 57.7%. E mesmo no melhor caso o desempenho obtido foi de 58.7% utilizando um *bag size* de 50% e 30 ou 40 classificadores.

O método de *Boosting* utilizando o algoritmo AdaBoostM1 apresentou ganhos significativos para o classificador J4.8 e se mostrou ineficiente para os classificadores k-NN e Naive Bayes. Utilizando o J4.8 o método de *Boosting* apresentou uma performance surpreendente. No pior caso teve um aumento na taxa de acerto de 7.8% (utilizando cinco classificadores) e no melhor caso o aumento foi de 16% (utilizando 30 classificadores) para a base padrão; e aumento de 8.5% (utilizando cinco classificadores) no pior caso e de 15.8% (utilizando 50 classificadores) no melhor caso para a base aleatória.

Para o k-NN o método de *Boosting* se mostrou ineficiente, apresentando resultados aquém do esperado, normalmente piorando o desempenho do classificador ou em alguns raros casos aumentando ligeiramente o desempenho em no máximo 2%. Os resultados obtidos tanto na base padrão quanto na base aleatória mostram que o 2-NN foi o único caso em que o desempenho do pior caso foi superior a do classificador Individual, sendo o desempenho normalmente igual ou inferior ao uso de um único classificador. Já no melhor caso apenas o classificadores de 1-NN e 2-NN apresentam alguma melhora, enquanto nos demais casos, os melhores resultados obtidos são piores do que os resultados obtidos utilizando um único classificador k-NN. Dessa forma, o método de *Boosting* utilizando o classificador k-NN teve um desempenho aquém da esperada, não apresentando nenhuma melhora significativa no desempenho do classificador e em alguns casos até mesmo piorando o desempenho do mesmo quando comparada a um único classificador sem o uso da técnica de *Boosting*.

Para o Naive Bayes, os resultados obtidos com o método de *Boosting* foram aquém do esperado, piorando o desempenho do classificador em todos os casos.

6. Conclusões

Neste trabalho foi proposta uma nova abordagem para o problema da classificação automática de gêneros musicais utilizando as técnicas de meta-aprendizagem conhecidas como *Bagging* e *Boosting*. O método de *Bagging* se mostrou eficiente para os classificadores J4.8 e k-NN e ineficiente para o Naive Bayes. No caso do J4.8 a performance aumentou significativamente, enquanto que no caso do k-NN o uso da técnica de *Bagging* obteve normalmente nos piores resultados uma performance similar ao uso de um único classificador. No caso do Naive Bayes o método não apresentou melhora significativa e chegou inclusive a obter resultados inferiores ao uso de um único classificador.

O método de *Boosting* se mostrou eficiente para o classificador J4.8 e ineficiente para os classificadores k-NN e Naive Bayes. No caso do J4.8 a performance aumentou significativamente (16% no melhor caso), enquanto que para o k-NN o método normalmente piorou a performance do classificador tendo alguns raros casos onde a performance foi ligeiramente aumentada em no máximo 2%. Os resultados obtidos com o classificador Naive Bayes foram aquém do esperado piorando o desempenho do classificador em todos os casos.

Sintetizando os resultados obtidos, no caso de um único classificador o melhor desempenho foi a do classificador Naive Bayes que teve um acerto de 57% na base padrão e 57.7% na base aleatória. Já no caso das técnicas de meta-aprendizagem, os melhores resultados na base padrão foram obtidos com o uso de *Boosting* com um conjunto de classificadores J4.8 tendo um acerto no melhor caso de 68.1% e na base aleatória com o uso de *Bagging* tendo um acerto no melhor caso de 67.5%.

Como trabalho futuro os autores pretendem verificar o uso das técnicas de *Bagging* e *Boosting* aliadas a outros classificadores, além de rodar experimentos em outras bases contendo outros gêneros musicais.

Referências

- Aha, D. W., Kidbler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.
- Aucouturier, J. J. and Pachet, F. (2003). Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):83–93.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.
- Carey, M. J., Parris, E. S., and Lloyd-Thomas, H. (1999). A comparison of features for speech, music discrimination. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 149–152.
- Fingerhut, M. (1999). The ircam multimedia library: A digital music library. In *IEEE Forum on Research and Technology Advances in Digital Libraries*, pages 19–21.
- Foote, J. T. A. (1999). An overview of audio information retrieval. *Multimedia Systems*, 7(1):42–51.
- Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156.
- Guo, G. and Li, S. Z. (2003). Content-based audio classification and retrieval by support vector machines. *IEEE Transactions on Neural Networks*, 14(1):209–215.
- Kosina, K. (2002). Music genre recognition. Technical report, Fachhochschul Hagenberg.
- Liu, T., Ogihara, M., and Li, Q. (2003). A comparative study on content-based music genre classification. In *Proc. of the 26th Annual Intl ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 282–289, Toronto, Canada.
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
- Pampalk, E., Rauber, A., and Merkl, D. (2002). Content-based organization and visualization of music archives. In *ACM Multimedia 2002*, pages 570–579, Juan-les-Pins, France.
- Pye, D. (2000). Content-based methods for the management of digital music. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2437–2440.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- Shao, X., Xu, C., and Kankanhalli, M. S. (2003). Applying neural network on the content-based audio classification. In *Fourth International Conference on Information, Communications and Signal Processing*, volume 3, pages 1821–1825.
- Sweldens, W. and Piessens, R. (1993). Wavelet sampling techniques. In *Proc. of the Statistical Computing Section*, pages 20–29.
- Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302.
- Zhang, T. and Kuo, C. C. J. (2001). Audio content analysis for online audiovisual data segmentation and classification. *IEEE Transactions on Speech and Audio Processing*, 9(4):441–457.

Fuzzy Granular Synthesis Controlled by Walsh Functions

Adolfo Maia Jr.^{1 2}, Eduardo R. Miranda¹

¹Computer Music Research, Faculty of Technology
University of Plymouth, Plymouth, Devon PL4 8AA
United Kingdom.

²Departamento de Matemática Aplicada, IMECC and
Núcleo Interdisciplinar de Comunicação Sonora (NICS), UNICAMP,
13.081-970, Campinas, SP, Brazil.

eduardo.miranda@plymouth.ac.uk

adolfo@nics.unicamp.br

Abstract. *In this paper we present a model for granular synthesis in which the internal content (spectrum) of grains are defined as Fuzzy Sets. Markov Chains modulated by Membership Matrices related to the internal structure of the sound grains are used to control the evolution of the sound in time. The final sequencing of grains (sound stream) is controlled by the so called Walsh Functions. Using a number of channels the outputs may have complex polyphonic sound structures. We also provide the mathematical foundations of the model.*

1. Introduction

Granular synthesis [Roads, 1988] is commonly known as a technique that works by generating a rapid succession of tiny sounds, metaphorically referred to as sound grains [Roads, 1996]. Granular synthesis is widely used by musicians to compose electronic or computer music because it can produce a wide range of different sounds, but it also has been used in speech synthesis [Miranda, 2002]. Clearly a discussion about musical aesthetics may arise from these developments. Although such discussion would be a very interesting topic on its own right, we will not deal with these matters in this paper. A good discussion on the aesthetics of microsound can be found in [Thomson, 2004].

Granular synthesis is largely based upon D. Gabor idea of representing a sound using hundreds or thousands of elementary sound particles [Gabor, 1947].

In this work we take C. Roads' definition of sound grain as a point of departure to develop a formal but flexible granular synthesis model. Our model uses stochastic processes, namely Markov Chains with Transition Probability Matrix modulated by Membership Functions of the grains with values in the interval $[0, 1]$, which give fuzzy characteristics to the grains. Thus, we propose a new method for controlling the grains by intertwining Stochastic Processes and Fuzzy Set Theory, where the content of the grains (or internal variables) can change their transition probabilities between states. For the sake of clarity, we have chosen a very simple *State Space* to introduce the model, where each grain is itself a state of a Grain Vector \mathbf{G} . Therefore, the membership functions in this case modulate the transition probabilities between states (i.e., grains), changing their ordering position in the time domain. In this paper we present just one of the several possible modes of interaction between internal and external control variables.

Walsh functions were used as tool for code transmission by electromagnetic signals [Beauchamp, 1975], [Hall Jr., 1986]. In addition in the 70s

they were used for real sound synthesis [Rozenberg, 1979],[Hutchins Jr., 1973], [Hutchins Jr., 1975],[Insam, 1974]. Our application differs from the last authors since we are most interested to use Walsh functions as a control for sound output of grains streams. As far as we know this is the first model in this direction.

In the next section we present the concept of Fuzzy Grains and their mathematical representation. In section 3 we describe the control of grain streams including halting criteria, a computer implementation, which we named Fuzkov 1.0, and the control of the sound output by Walsh Functions. In section 4 we conclude with some comments and some perspectives for future work. In addition we provided two appendices, namely, the first one is a very short review on Fuzzy Sets and the second one on Walsh Functions.

2. Markov Processes for Fuzzy Grains

Fuzzy Sets [Zadeh, 1965] are able for handling uncertainty, imprecisions or vagueness. Our aim here is to get a kind of Markov Process in which the Transition Matrix could be modified by the internal content of the grains. In order to weight the contribution of each Fourier component we have used a Membership Function for the grains from Fuzzy Sets theory. In Appendix A we present a short review of Fuzzy Sets.

In this work we are mainly interested in the output control of the material generated by Fuzzy Granular Synthesis. For the sake of completeness we explain shortly fuzzy grains and their generation.

Let us denote Ω the space of all possible oscillators, that is the *frequency \times amplitude* space of the ordered pair (ω, a) , where the variables ω and a varies in some suitable real intervals. Ω is referred to as a Parameters Space. In this work we define a grain g as a finite set of points $\{(\omega_i(t), a_i(t)), i = 1, 2, \dots, N\}$ in Ω . The sound in the macro scale, or in more technical words, the time ordering of grains and their subsequent sound output is generated as a Markov Chain. So a grain can be described by its Fourier Partial inside a real interval I . Clearly, this is suitable for producing grains with additive synthesis. Its spectral content can be written, without loss of generality, as

$$G(t) = \sum_{n=1}^N a_n \sin[2\pi\omega_n t + \delta_n], \quad (1)$$

where a_n, ω_n, δ_n reads for amplitude, frequency and a possible phase, respectively. In granular synthesis a sound can be viewed as a quick stream of grains which, from a geometrical point of view, describes a trajectory in the Ω space. Subsets of points in Ω do not have a natural well ordering for the space part. In our model, the internal content of a grain is coded in matrices. All operations on grains are represented as matrix operations. For the sake of completeness, we present an ordering of the elements of the grains, but this is a highly arbitrary choice. Below we just show the simplest ordering: that one that access the grain content (which is a two dimensional set of points) from the left to the right (that is, from low to high frequencies) and from the bottom to the top (that is, from low to large amplitudes). This is formally written as follows: let $x_i = (\omega_i, a_i)$ and $x_j = (\omega_j, a_j)$ be two arbitrary points in the Ω space. For $i \neq j$

$$x_i < x_j \Leftrightarrow \begin{cases} \omega_i < \omega_j \\ \omega_i = \omega_j \Rightarrow a_i < a_j \end{cases} \quad (2)$$

With this definition the matrix representation of a grain g_i with r components reads as a $2 \times r$ matrix:

$$g^i = \begin{bmatrix} \omega_1^i & a_1^i \\ \omega_2^i & a_2^i \\ \vdots & \vdots \\ \omega_r^i & a_r^i \end{bmatrix} \quad (3)$$

where the above defined order is implicit.

Now, a fuzzy grain can be represented as a three column matrix

$$G^i = \left[\begin{array}{cc|c} \omega_1^i & a_1^i & \alpha_1^i \\ \omega_2^i & a_2^i & \alpha_2^i \\ \vdots & \vdots & \vdots \\ \omega_r^i & a_r^i & \alpha_r^i \end{array} \right] \quad (4)$$

where we have introduced a third column with the membership frequency and amplitude values of each partial of the grain g^i . Note that g^i is a particular case of G^i for $\alpha_1^i = \alpha_2^i = \dots = \alpha_r^i = 1$. We can denote shortly this matrix by $G^i = [\omega^i, a^i, \alpha^i]$, where $\omega^i = [\omega_1^i, \omega_2^i, \dots, \omega_r^i]$, $a^i = [a_1^i, a_2^i, \dots, a_r^i]$, and $\alpha^i = [\alpha_1^i, \alpha_2^i, \dots, \alpha_r^i]$ are the frequency, amplitude and membership r -vectors of the grain. Also we define N -vectors of grains $\mathbf{g} = [g^1, g^2, \dots, g^N]$ and $\mathbf{G} = [G^1, G^2, \dots, G^N]$.

Below we show how the membership functions of fuzzy grains can modify the Markov Transition Matrix and so we get a fuzzy control for the Markov Chain. For a good account of Fuzzy Sets the reader is referred to [Diamond and Kloeden, 1994]. Let us consider a grain described by its Fourier-like equation (1). Each subset of points in Ω represents a grain with particular Fourier partials, that is, it is a sum of basic sinusoidal frequencies. With the above defined matrices G^i , it is possible to define an unambiguously time evolution of grains through out Markov Chains. This is usually accomplished through a Fuzzy Transition Table, constructed as follows: firstly, suppose that we have a transition matrix for ordinary grains, that is, with no membership vector yet defined. This can be written as follows:

$$\left[\begin{array}{c|cccc} & g^1 & g^2 & \dots & g^N \\ \hline g^1 & p^{11} & p^{12} & \dots & p^{1N} \\ g^2 & p^{21} & p^{22} & \dots & p^{2N} \\ \dots & \dots & \dots & \dots & \dots \\ g^N & p^{N1} & p^{N2} & \dots & p^{NN} \end{array} \right]$$

which can be viewed as a function

$$\begin{aligned} p : \quad \mathbf{g} \times \mathbf{g} &\longrightarrow [0, 1] \\ (g^i, g^j) &\longmapsto p(g^i, g^j) = p^{ij} \end{aligned}$$

Now, we define a Fuzzy Extended Probability Transition Matrix (or simply Fuzzy Transition Matrix) $Q : \mathbf{G} \times \mathbf{G} \longrightarrow [0, 1]$ as

$$Q^{ij} = Q(G^i, G^j) = \Phi^{ij} * p^{ij} \quad (5)$$

where the symbol $*$ means a matrix operation (e.g., a scalar product, a matrix product or any other well defined operation). The function Φ^{ij} is generated as a finite number of applications of the following basic operations of fuzzy sets: for $i, j = 1, 2, \dots, N$, we define

1.

$$\phi^{ij} = \max_{1 \leq k \leq r} \{ \alpha_k^i, \alpha_k^j \}, \quad (6)$$

where α^i and α^j are the membership vectors of the grains G^i and G^j respectively.

2.

$$\phi^{ij} = \min_{1 \leq k \leq r} \{ \alpha_k^i, \alpha_k^j \}, \quad (7)$$

where α^i and α^j are the membership vector of the grains G^i and G^j respectively.

3.

$$\alpha_c^i = 1 - \alpha^i. \quad (8)$$

These result in a product like $\Phi^{ij} = \phi_1^{ij} \phi_2^{ij} \dots \phi_l^{ij}$, where the third operation above can be performed on any product of α_i vectors. These are basic operations on Fuzzy Sets. See reference [Diamond and Kloeden, 1994] for a introduction to Fuzzy Sets and their metrics. Note that since the membership function modulates the probability values p^{ij} , the condition for the probability sum $\sum_{j=1}^N Q^{ij} = 1$ can be violated. In order to solve this problem we renormalize the matrix Q^{ij} as follows. Denoting $q_i = \sum_{k=1}^N Q_{ik}$ we define the elements of matrix \mathbf{P} as

$$P_{ij} = Q^{ij} / q^i \quad i, j = 1, 2, \dots, N \quad (9)$$

Now the probability property $\sum_{j=1}^N P_{ij} = 1$ is clearly satisfied. The above definition shows that the internal fuzzy content of the grains have a weight (through the function Φ^{ij}) for their transition to a next state of the Markov Chain.

The Fuzzy Transition Matrix (or Table) reads

$$\begin{array}{c|cccc} & G^1 & G^2 & \dots & G^N \\ \hline G^1 & P^{11} & P^{12} & \dots & P^{1N} \\ G^2 & P^{21} & P^{22} & \dots & P^{2N} \\ \dots & \dots & \dots & \dots & \dots \\ G^N & P^{N1} & P^{N2} & \dots & P^{NN} \end{array} \quad (10)$$

In this simple model a transition from one state to another corresponds to a jump from a particular grain to another in the grain vector \mathbf{G} . In addition the fuzzy content of a grain, that is, its membership vector, can have a significant weight on the probability transition. Since the process is finite, a criterium to halt the process is needed here. This will be discussed in the next section.

The above model is suitable for several kinds of matrix operations on internal as well external variables controlling the grains behaviour in time. There is plenty of room for the definition of a great number of different methods to generate and control the grains. We present our approach on the control of the streams below.

3. Control of Grain Streams

3.1. Halting Criteria

There exist many different ways (algorithms) to control the evolution of the grains in time. We show here one by using the so called Hausdorff Metric which is suitable to measure distance between sets (grains are finite and discrete subsets of Ω). Time evolution can be better controlled using a fuzzy metric that takes into account the degree of membership of the Fourier partials inside each grain. In other words, partials with low membership coefficients contribute little for the Hausdorff distance measure between the

grains. Membership vectors define the fuzzy character of the grains, or in a musical jargon, their weighted harmonic content. A metric control is closely related to the notions of approximation and/or the maximal time (or number of steps) available to run a process. Below we indicate three stop criteria we devised to halt a Markov Chain in our model of granular synthesis.

Halting Criteria

1. *Convergent Type*: If the distance between the last generated grain and a fixed grain (target) is smaller than a prefixed arbitrary number ϵ , the process halts.
2. *Cauchy Type*: If the distance between two states is smaller than ϵ the process halts.
3. *Maximal Number of Steps Type (MNS)*: Fix the maximum number of steps for the process to halt.

Any of the above criteria can be used to halt the process. Of course *Maximal Number of Steps Type* is the simplest one, since no metric is required. In our program *Fuzzkov 1.0* we have implemented fully the MNS and partially, the Cauchy type, at the Hausdorff Metric level, but not at the Fuzzy Metric level. We have implemented the Hausdorff Metric as an inequality, so that FuzzKov 1.0 runs in loops until it is satisfied. We obtained good results for both controls of the grains streams working together.

This procedure leads to a concentration of frequencies within a narrow bandwidth, but with a large bandwidth for the amplitudes. The *halting criterion* here can be taken as the *Cauchy type*. Given an arbitrary (but small) number ϵ , the process stops if $d_H(G_i, G_{i+1}) \leq \epsilon$, where the distance between two points used for defining the above Hausdorff Distance is given by, for example:

$$d((\omega_i, a_i), (\omega_j, a_j)) = \max_{1 \leq k \leq r} |\omega_i - \omega_j|. \quad (11)$$

If we fix a particular grain in the Ω space, such as \overline{G} , we can consider the *Convergent halt criterion*, that is the process stops if $d_H(G_i, \overline{G}) \leq \epsilon$.

We can also take the *mean frequency* only of the last m grains and so it reads as

$$\overline{\omega}^{(l)} = \sum_{k=1}^r \frac{\omega_k^{l-m} + \omega_k^{l-m+1} + \dots + \omega_k^{l-1}}{m} \quad (12)$$

and take the r closest frequencies to $\overline{\omega}^{(l)}$ from the set $U_l = \bigcup_{k=l-m}^{l-1} G^k$. Clearly, for $m = l$ we get the previous model.

3.2. Implementation

This section presents *Fuzzkov 1.0*, a prototype implementation of our model. In *Fuzzkov 1.0*, Membership Matrices modulate a Transition Probability Matrix of a Markov Chain, but the internal content of the grains are not changed during the generative process. Thus, *Fuzzkov 1.0* can be thought of as a system for *Coarse Grain Fuzzy Synthesis*. A block diagram of the program is shown in Figure 1. *Fuzzkov 1.0* was implemented in MATLAB. Input information and control parameters are as follows:

1. Control Parameters for the Markov Process

markov_type = type of generation of the Markov Matrix.

N = number of states (or number of grains).

n = number of steps of the Markov Process.

init_vect_type = type of the initial vector (range = 1-3).

2. Grain Parameters

fs = sample frequency.

dur = grain duration in seconds.

r = number of points in a grain, where each point is a Fourier Partial.

$grain_type$ = type of grain (range = 1-3).

3. Fuzzy Control Parameters

$memb_type$ = type of Membership Matrices (range = 1-4).

$alpha_type$ = type of vector to construct Membership Matrices (range = 1-4).

To begin with, the control parameters for Markov Processes generate a Markov Matrix p . After a manipulation with membership matrices as in Eq. (5) we get a fuzzyfied Markov Matrix as in Eq.(10). The grain parameters control the internal content of grains and their output is a Fourier sum of partials as in Eq.(1). The last group of parameters controls the fuzzy characteristic of the grains as described by Eq. (4) and Eqs. (6)-(8). There are a number of possibilities to generate Membership Matrices. In our implementation we have taken only four possibilities in Fuzzkov 1.0.

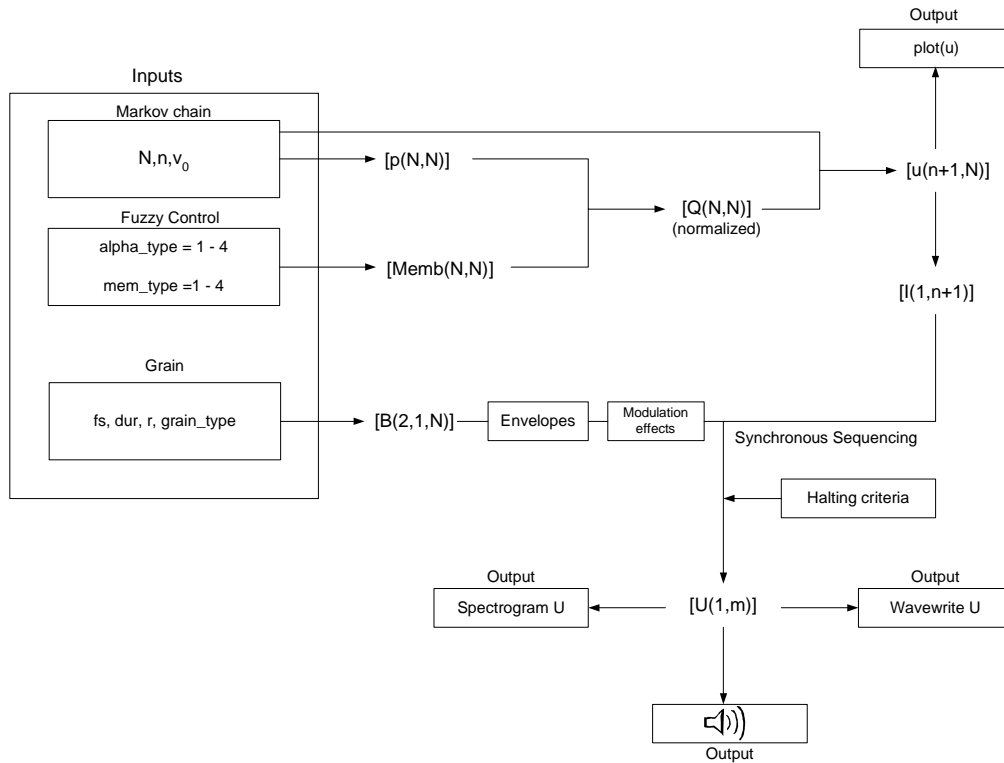


Figure 1: Diagram of FuzzKov 1.0

Sound grains are generated randomly by uniform and gaussian 3D matrices A with dimensions $2 \times r \times N$, which include r normalized frequencies and amplitudes for N grains (Fourier Partial). We have used uniform and Gaussian probability distributions to generate the sound grains. From this we obtain a Matrix $B(2, 1, N)$ with the sum of Fourier Partial for the N grains. A Markov transition Matrix $p(N, N)$ is generated and modified by a Membership Matrix $Memb(N, N)$. A number of different operations are available for this modification. We take a fuzzyfied Markov Matrix $Q(N, N)$, which operates on an array of probability vectors $u(n+1, N)$. Next, a particular filter selects the index of the maximal value of each probability vector $I(1, n+1)$. Finally, the program reorders the Grain matrix $B(2, 1, N)$ along the index vector $I(1, n+1)$ and produces the sound. We also generate spectral data for analysis of the results.

3.3. Control of Sound Streams by Walsh Functions

Walsh functions became important for representation of signals through the superposition of members of a set of simple functions which are easy to generate and define [Beauchamp, 1975]. They form an ordered set of rectangular waveforms taking only two amplitudes values $+1$ and -1 . Walsh Functions and the Hadamard Matrices which generate them are important tools in several areas such as electrical engineering and code theory. They are suitable to control time sequences and we have used this characteristic to drive our grains streams. We describe shortly, below, how we control the sound output using Walsh functions.

Each value of a Walsh function works as a trigger calling the operation associated with it. For the sake of simplicity, a Walsh function operates on a sound stream just deleting all grains of the stream which correspond to the value -1 . Also we made some experiments using the operation of time inversion of the grain when it has the value -1 associated to it. Triggering operations with Walsh functions can produce an endless number of outputs, depending only on the chosen operations. We can also use different Walsh functions associated to different operations and apply them independently on the sound streams. In this case the results from the triggering operations can not be easily predicted.

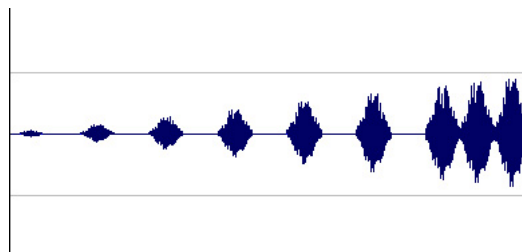


Figure 2: A typical Grains Stream with 15 steps showing the triggering by a Walsh Function. Silence between grains comes from the zero value of the Walsh Function.

An interesting experiment we have realized is to create several independent Walsh controlled sound streams (using a channel for each stream) and play them synchronously. Since the Walsh functions have independent actions, it resulted poliphonic streams with a kind of *grain counterpoint*. When combined with some special effects, such as amplitude modulation and others we get some very interesting sounds which could be used in computer music compositions. Walsh functions, as used in our model, are time symmetric. In practice, the symmetrical disposition of values of a Walsh function implies that operations are performed symmetrically in time. This property could be very interesting to music composition. However, one should not count on recognizing such a symmetry in the output, since it is unlikely for the input to share a similar symmetry. Nevertheless, given the scope of this work, the symmetrical properties of Walsh functions and their possible compositional uses deserve further attention in future work.

4. Conclusion and Perspectives

We have implemented a prototype of our model using Matlab in which Membership Matrices of Fuzzy grains modulate a Transition Probability Matrix of a Markov Chain which is a partial control of the time evolution. In addition sound outputs are controlled by Walsh Functions. This additional control leads the sound output to have a discontinuous sequency of grains. If a number of these outputs are played synchronously we get complex sound structures or, roughly speaking, a kind of *grain's counterpointistic structures*. Our model can be generalized to include other functions besides the Walsh ones. The next

step will be to use the so called *Sequency Functions* [Hall Jr., 1986],[Beauchamp, 1975] (an obvious generalization of the Walsh Functions) as triggers, and to incorporate other sound parameters, such as intensity, spacialization, etc. This will be accomplished elsewhere. As mentioned above an interesting aspect to be explored in the control of previous material by Walsh Functions is the fact that they present some symmetries which can also be better explored.

Appendix A: Fuzzy Sets

Fuzzy sets, first proposed by Zadeh [Zadeh, 1965] are able for handling uncertainty, imprecisions or vagueness. It is not a probabilistic approach in a sense that the membership function defined below can have value 1 for several, or even for all elements of the Fuzzy set. Below we present a short summary of Fuzzy Sets.

Let G be a subset of points of a Euclidian Space \mathbf{R}^n . Intuitively G is a Fuzzy set if for each of its elements we associate a membership degree. Formally a Fuzzy Subset G of Ω , is a non empty subset $(x, u(x)), x \in \Omega$ of $\Omega \times [0, 1]$ for some function $u : \mathbf{R} \rightarrow [0, 1]$. This function is named *membership function*. The subset of points in G with non zero membership value is named *support of G* . When the support of G is finite we can consider the membership function as a vector. So, in this way we can use indistinguishably the function u as a Fuzzy Set. Below we show three examples.

1. Let A be a finite subset of \mathbf{R} with m elements.

$$A = \{x_1, x_2, \dots, x_m\} \quad (13)$$

and the membership function defined by

$$u(x) = \begin{cases} 1/i, & \text{for } x = x_i, i = 1, 2, \dots, m; \\ 0 & \text{otherwise} \end{cases}$$

Clearly the support of A is the subset $\{x_1, x_2, \dots, x_N\}$.

2. Let A be a arbitrary set in \mathbf{R} with membership function given by

$$\chi_A = \begin{cases} 1 & \text{for } x \in A \\ 0 & \text{for } x \notin A \end{cases}$$

The above example is a extreme case for which the fuzzy set is an ordinary set (also named crisp set), that is, all of its elements have membership value equal 1.

3. Let $\mathbf{B}^n(R)$ the n -dimensional ball with radius R . Define the function

$$u(x) = \begin{cases} 1 - \frac{\|x\|}{R} & \text{for } x \in \mathbf{B}^n(R) \\ 0 & \text{elsewhere} \end{cases}$$

This is an example of a continuous fuzzy set. Observe that the membership is 1 only for the center of the ball and decrease to 0 as x gets closer to the ball boundary.

The Hausdorff Metric

Suppose that the space $\Omega = \mathbf{R}^N$ has a metric $d(x, y)$. Let x be a point in Ω and A a nonempty subset of Ω . We define the distance of the point x to the set A as:

$$\delta(x, A) = \inf \{d(x, y), y \in A\}. \quad (14)$$

The Hausdorff separation of a set B from a set A is defined by

$$\Delta(B, A) = \sup \{d(y, A), y \in B\} \quad (15)$$

In general, Δ is not symmetric, that is $\Delta(A, B) \neq \Delta(B, A)$. In order to get a symmetric one we define the so called Hausdorff distance by

$$d_H(A, B) = \max \{ \Delta(A, B), \Delta(B, A) \} \quad (16)$$

With this distance function (Ω, d_H) is a Metric Space. Nevertheless this metric do not take into account the fuzzy properties of the sets. Formally, we need to define another metric which will take into account the membership functions. In order to do this we firstly define some important subsets of a given fuzzy set.

Let $u : \mathbf{R}^n \rightarrow I = [0, 1]$ a membership function.

Definition: For each $\alpha \in [0, 1]$, the α -level set $[u]^\alpha$ of a fuzzy set u is the subset of points $x \in \mathbf{R}^n$ with membership grade $u(x)$ of a least α , that is

$$[u]^\alpha = \{x \in \mathbf{R}^n, u(x) \geq \alpha\}. \quad (17)$$

The support $[u]^0$ of a fuzzy set is then defined as the closure of the union of all its level sets, that is,

$$[u]^0 = \overline{\bigcup_{\alpha \in [0,1]} [u]^\alpha} \quad (18)$$

We consider here only the fuzzy sets which satisfy the property " u maps \mathbf{R}^n onto the real interval $[0, 1]$, or equivalently, $[u]^1 \neq \emptyset$. In addition we consider only membership functions so that $[u]^0$ is a bounded subset of \mathbf{R}^n . Below we present some properties of the α -level sets (see [Diamond and Kloeden, 1994] to a detailed presentation and proofs).

1. For all $0 \leq \alpha \leq \beta \leq 1$

$$[u]^\beta \subseteq [u]^\alpha \subseteq [u]^0 \quad (19)$$

2. $[u]^\alpha \neq \emptyset, \forall \alpha \in I$
3. $[u]^\alpha$ is a compact subset of \mathbf{R}^n for all $\alpha \in I$.

Now we are ready to define a fuzzy metric. We define the *supremum metric* d_∞ on \mathbf{F} by

$$d_\infty(u, v) = \sup \{ d_H([u]^\alpha, [v]^\alpha), \alpha \in I \} \quad (20)$$

for all $u, v \in \mathbf{F}$. It is worth to mention that there exist too many different metrics we can use. The above one was choose due to its simplicity and usefulness. In this work we have used the supremum metric in order to control stream of grains in $\bar{\Omega}$.

Appendix B: Walsh Functions and Hadamard Matrices

Walsh functions became important for representation of signals through the superposition of members of a set of simple functions which are easy to generate and define [Beauchamp, 1975]. They form an ordered set of rectangular waveforms taking only two amplitudes values $+1$ and -1 . A simple example of a set of rectangular waveforms are the Rademacher Functions which can be defined as

$$RAD(n, t) = \text{sign}[\sin(2^n \pi t)] \quad (21)$$

where $0 \leq t \leq 1$. Rademacher functions have two arguments n and t such that $RAD(n, t)$ has 2^{n-1} periods of square wave over a normalised time base, or interval $[0, 1]$.

The problem with Rademacher System is that it is not complete in the sense that any signal can be decomposed, like in a Fourier Series, as a sum (perhaps infinite) of

Rademacher Functions. The simplest complete set of rectangular functions (waveforms in the context of this work) is the Walsh Set. From the point of view of signal representation, Walsh functions consist of trains of square pulses (with the allowed states being -1 and 1) such that transitions may only occur at fixed intervals of a unit time step, the initial state is always 1. In general Walsh functions are defined in a Time Base interval T and periodically extended for intervals of length $kT, k \in \mathbf{Z}$. They are completely defined by two parameters, its *sequency order* n and its time variable t . It is denoted $WAL(n, t)$, with $n = 0, 1, 2, \dots, N - 1$, and N is the order of Walsh Functions defined below. Using a normalized time variable t/T the Walsh functions can be defined in the interval $[0, 1]$. In addition, they are symmetrical about the centre and so, when they are defined in the interval $[-1/2, 1/2]$ they are symmetrical. The even functions are collectively named *CAL* and the odd ones *SAL* which are in certain sense the counterparts of the cosine and sine trigonometric functions. so, we can write

$$WAL(2k, t) = CAL(k, t), k = 1, 2, \dots, N/2. \quad (22)$$

$$WAL(2k - 1, t) = SAL(k, t), k = 1, 2, \dots, N/2. \quad (23)$$

Both Rademacher and Walsh Sets are orthogonal systems in the same way as Fourier Systems of *sin* and *cos* functions. Other systems of rectangular functions do exist, such as Haar and Slant Functions. See reference [Beauchamp, 1975] for more information and bibliography on this subject.

Walsh functions can be ordered in a number of ways. One of them is the so called *sequency order*. The sequency k of a Walsh function is defined as half the number of zero crossings in one cycle of the time base. Walsh functions with nonidentical sequencies are orthogonal and the product of two Walsh functions is also a Walsh function. A way to generate Walsh functions is through the so called *Hadamard Matrix* when arranged in the *sequency order*. A Hadamard matrix of order N is a type of square matrix whose entries are only +1 and -1 and such that

$$\mathbf{H}\mathbf{H}^T = N\mathbf{I} \quad (24)$$

This equation implies that the rows of \mathbf{H} (or the Walsh functions of order N) are orthogonal. In the so called *normal form* the first row and the first column are formed only by +1. The lowest-order Hadamard matrix is the 2 dimensional matrix

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (25)$$

An important theorem on Hadamard Matrices is stated as:

Theorem: If \mathbf{H}_m and \mathbf{H}_n are matrices of orders m and n respectively, then their Direct Product is an \mathbf{H} matrix of order mn .

The proof of this theorem can be founded in [Hall Jr., 1986]. Most of constructions of Hadamard Matrices are based on Direct (Kronecker) Product of two matrices. The definition of Direct Product as follows. If $\mathbf{A} = (a_{ij})$ is an $m \times m$ matrix and $\mathbf{B} = (b_{rs})$ is an $n \times n$ matrix, then the Direct Product $\mathbf{A} \otimes \mathbf{B}$ is the $mn \times mn$ matrix given by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & \dots & a_{2m}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{i1}B & a_{i2}B & \dots & a_{im}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mm}B \end{bmatrix} \quad (26)$$

Using this theorem, higher-order matrices, with dimension 2^n are easily obtained by the recursive relationship

$$\mathbf{H}_N = \mathbf{H}_{N/2} \otimes \mathbf{H}_2 \quad (27)$$

where $N = 2^n$. Thus, for example,

$$\mathbf{H}_4 = \mathbf{H}_2 \otimes \mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad (28)$$

The ordering of Walsh Functions in a Hadamard Matrix is named *natural ordering* and in this case they are also named Hadamard Functions and denoted by $HAD(k, t)$, where $k = 0, 1, \dots, N$. For example, the Hadamard Functions of order 8 are given by

$$\mathbf{H}_8 = \mathbf{H}_4 \otimes \mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} HAD(0, t) \\ HAD(1, t) \\ HAD(2, t) \\ HAD(3, t) \\ HAD(4, t) \\ HAD(5, t) \\ HAD(6, t) \\ HAD(7, t) \end{bmatrix} \quad (29)$$

Hadamard remarked that a necessary condition for a Hadamard matrix to exist is that $n = 1, 2$, or a positive multiple of 4.

A. Acknowledgments

This work was supported by FAPESP, CAPES and FAEPEX/UNICAMP (Brazil). A.M.J is grateful to the Computer Music Group the University of Plymouth for the hospitality during the making of this work.

References

- Beauchamp, K. G. (1975). *Walsh Functions and Their Applications*. Academic Press, London.
- Diamond, P. and Kloeden, P. (1994). *Metrics of Fuzzy Sets: Theory and Applications*. World Scientific.
- Gabor, D. (1947). Acoustical quanta and the theory of hearing. *Nature*, 159(4044):591–594.
- Hall Jr., M. (1986). *Combinatorial Theory*. John Wiley and Sons Inc.
- Hutchins Jr., B. (1973). Experimental electronic music devices employing walsh functions. *J. Audio Eng. Soc.*, 21:640–645.
- Hutchins Jr., B. (1975). Application of a real-time hadamard transform network to sound synthesis. *J. Audio Eng. Soc.*, 23:558–562.
- Insam, E. (1974). Walsh functions in waveform synthesizers. *J. Audio Eng. Soc.*, 22:422–425.
- Miranda, E. (2002). Generating source streams for extralinguistic utterances. *Journal of the Audio Engineering Society*, 50(3):165–172.
- Roads, C. (1988). Introduction to granular synthesis. *Comp. Mus. J.*, 12(2):11–13.
- Roads, C. (1996). *Computer Music Tutorial*. MIT Press, Cambridge, MA.
- Rozenberg, M. (1979). Microcomputer-controlled sound processing using walsh functions. *Computer Music Journal*, 3(1):42–47.
- Thomson, P. (2004). Atoms and errors: towards a history and aesthetics of microsound. *Organized Sound*, 9(2):207–218.
- Zadeh, L. A. (1965). Fuzzy sets. *Informat. Control*, 8:338–353.

Bibliotecas Java Aplicadas a Computação Musical

Leandro L. Costalonga, Evandro M. Miletto, Luciano V. Flores, Rosa M. Vicari

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{l1lcostalonga,miletto,rosa}@inf.ufrgs.br, lflores@cpovo.net

Abstract. *Today, there are many programming languages and libraries available for the development of computer music applications. This paper presents a comparison of Java technologies for computer music, as a quick reference for interested developers. The main available musical libraries are listed, described and compared.*

Resumo. *Atualmente existem dezenas de linguagens de programação e bibliotecas destinadas ao desenvolvimento de aplicativos para computação musical. Este artigo apresenta um estudo comparativo sobre tecnologias Java para computação musical visando servir como referência rápida para desenvolvedores que queiram utilizar essa tecnologia. As principais bibliotecas musicais disponíveis são relacionadas, descritas e comparadas.*

1. Introdução

Um dos primeiros passos em um projeto de software é a escolha da linguagem que melhor possa codificar os requisitos identificados. Neste ponto, características da linguagem devem ser criteriosamente observadas e ponderadas.

No paradigma orientado a objetos, onde o reuso é fortemente favorecido, deve-se observar, além das características nativas da linguagem, as opções de expansão da mesma através de bibliotecas, APIs (*Application Programming Interfaces*) ou *add-ons* de terceiros que possam facilitar o desenvolvimento. Neste quesito Java está muito bem amparada e, em se tratando de bibliotecas e ferramentas de auxílio para softwares musicais, não é diferente.

O fato de Java estar sendo amplamente utilizada na programação de aplicações musicais não significa que é a única ou a melhor linguagem para este tipo de programação. Entretanto, alguns fatores que fazem com que Java se destaque como linguagem de programação para softwares musicais são os mesmos que fizeram de Java uma linguagem bem sucedida em áreas mais tradicionais, dentre os quais se cita a robustez, portabilidade, facilidade de aprendizado, bom suporte pela indústria, fácil depuração, bem projetada, etc. Há, contudo, algumas características que fazem de Java uma das linguagens mais usadas para desenvolvimento de softwares musicais como, por exemplo, o fato de incluir uma biblioteca para tratamento de som e mensagens MIDI (Java Sound).

As restrições do Java Sound (biblioteca nativa para manipulação de som que acompanha o Java) não desestimularam os programadores de aplicações musicais, pelo contrário, eles parecem ter ponderado as vantagens da linguagem e, em um esforço conjunto, começaram a desenvolver novas bibliotecas mais eficientes e completas. Esta

postura fez crescer o número de APIs disponíveis e, conseqüentemente, a dificuldade na escolha. Atualmente, muitos dos problemas iniciais do Java Sound já foram corrigidos.

Comparando Java com linguagens especialmente desenvolvidas para programação musical como Max/MSP, SuperCollider, Nyquist e KeyKit, pode-se afirmar que, por ser uma linguagem aberta, Java pode combinar música com outras funcionalidades da linguagem, como rede, gráficos e banco de dados.

Obviamente Java tem limitações, principalmente de performance, por ser uma linguagem tão completa e abrangente. Essas limitações motivam empresas e desenvolvedores independentes a criarem suas bibliotecas com soluções alternativas. Este trabalho pretende apresentar algumas das tecnologias disponíveis, facilitando a escolha e o reuso de componentes já criados por terceiros. Não é o objetivo deste trabalho ensinar ou esgotar todas as possibilidades de uso das ferramentas analisadas, e sim mostrar suas potencialidades.

O artigo está organizado como se segue: a seção 2 apresenta uma breve descrição de cada biblioteca analisada, ficando o estudo comparativo e classificatório para a seção 3. Uma breve conclusão é apresentada na seção 4.

2. Descrição das Bibliotecas

Foram analisados 14 bibliotecas que agregam ao Java a capacidade de trabalhar com áudio e dados musicais. São eles: Java Sound (Java Sound Resources, 2005), JMSL (Didkovsky e Burk, 2005), jMusic (Sorensen e Brown, 2005), Wire/Wire Provider (Wire, 2005; Gehnen, 2005), JavaMIDI (Marsanyi, 2005), NoSuch MIDI (NoSuch MIDI, 2005), MIDIShare (Grame, 2005), MIDI Kit (McNabb, 2005), jFugue (Koelle, 2005), Tritonus (Tritonus, 2005), JSyn (JSyn, 2005; JSynthLib, 2005), JScore (JScore, 2005), JASS (Doel, 2005) e Xemo (Project Xemo, 2003).

Procurou-se ordenar a apresentação das APIs por sua relevância (do ponto de vista dos autores), funcionalidade e data de criação. Todas as informações que constam na análise foram retiradas da documentação fornecida pelos fabricantes, o que nem sempre foi suficiente. Experimentos também foram realizados a fim de testar os códigos que constam nos exemplos. Infelizmente não há espaço suficiente no artigo para incluir código de exemplos e descrever em detalhes as APIs, mas tais informações são facilmente obtidas junto aos fabricantes.

Com base nos exemplos e na opinião própria dos autores (dada a subjetividade da questão), chegou-se à conclusão que as melhores APIs para trabalho com som (de maneira geral) são o JSyn, jMusic e Java Sound. Para trabalhar com MIDI, as melhores APIs são jMusic, JMSL e Java Sound. O destaque desta análise foi a jMusic que se mostrou completa, estável e de fácil uso em todas as categorias.

Este artigo é um resumo de um estudo muito mais extenso, realizado ao final de 2003, quando então as classificações e descrições básicas das bibliotecas foram elaboradas. As melhorias das novas versões vêm sendo acompanhadas e, quando significativas, são descritas neste artigo.

2.1. Java Sound (Versão 1.5)

Java Sound é uma API que provê suporte para operações de áudio e MIDI com alta qualidade, tais como: captura, mixagem, gravação, seqüenciamento e síntese MIDI. A Java Sound vem junto com o J2SE desde a versão 1.3 e por isso é considerada por alguns como a fonte do sucesso do Java para a programação musical. Na versão J2SE 1.5 teve grande avanço e correção e alguns de seus principais *bugs*, como a comunicação MIDI com dispositivos externos.

Em constante evolução e melhoria, são características nativas do Java Sound:

- Suporte aos formatos de arquivos de áudio: AIFF, AU e WAV;
- Suporte aos formatos de arquivos de música: MIDI Type 0, MIDI Type 1, e Rich Music Format (RMF);
- Formatos de som em 8/16 bits, mono e estéreo, com *sample rate* de 8 a 48 Hz;
- Dados de áudio codificados em linear, a-law e mu-law para qualquer um dos formatos de áudio suportados;
- Síntese e seqüenciamento MIDI por software, bem como acesso a qualquer dispositivo MIDI em hardware;
- Mixer com capacidade para mixar e renderizar mais de 64 canais de áudio digital e sons MIDI sintetizados.

A API Java Sound é o suporte a áudio de mais baixo nível na plataforma Java, permitindo aos programas um bom controle nas operações de som. A API não inclui editores de som sofisticados ou ferramentas gráficas, mas é extensível o suficiente para que isso seja construído a partir dela, como uma API de baixo nível deve ser.

Existem APIs de mais alto nível para desenvolvimento mais rápido e fácil de aplicações multimídia. A própria Sun, fabricante do Java Sound, distribui o Java Media Framework (JMF). O JMF especifica uma arquitetura unificada, protocolo para trocas de mensagem e interface de programação para captura, execução e sincronização de mídias baseadas no tempo, como som e vídeo.

Java Sound suporta tanto áudio digital como MIDI. Ainda, permite que terceiros criem e distribuam componentes de software customizados para estender suas capacidades.

Java Sound pode ser usado na Web via *applets*, entretanto deve-se ficar atento às restrições de segurança definidas na classe *AudioPermission*. O padrão é que uma *applet* executando sobre restrições de segurança pode tocar, mas não gravar sons. Isso pode ser modificado caso o usuário permita.

Apesar das restrições, o Java Sound permite que as aplicações escritas com suas classes funcionem da mesma forma nas diversas plataformas suportadas pelo Java.

2.2. JMSL – Java Music Specification Language (Versão 1.03)

JMSL é um *framework* desenvolvido em Java que auxilia no desenvolvimento de softwares musicais, em especial para composição de música computacional, performances interativas e construção de instrumentos virtuais.

Entre as principais vantagens do JMSL, pode-se citar:

- Total integração com a linguagem (Java) e recursos da mesma, incluindo conectividade com banco de dados, ferramentas de rede, *servlets*, etc.;
- Comunicação direta com outras bibliotecas e dispositivos implementados em Java, como o JSyn, JavaMIDI, MidiShare e Java Sound;
- Incorporação do pacote JScore que permite edição da notação musical;
- Permite ao compositor distribuir as aplicações localmente ou através de *applets*;
- Gratuito em uma versão mais restrita (*Lite*).

Baseada no HMSL – *Hierarchical Music Specification Language*, o JMSL mantém a idéia original de seu predecessor ao apoiar-se no conceito de hierarquias ao desenvolver aplicações musicais, que nada mais são do que relacionamentos pai-filho.

Hierarquias podem ser previamente definidas ou criadas durante a execução de uma peça musical. Dois tipos de hierarquias podem ser criadas no JMSL: seqüencial e paralela. Hierarquizar, no contexto do JMSL, não significa apenas posicionar os objetos em uma visão *top-down*, definindo assim a importância dos “objetos” (entidades) no contexto musical. No JMSL, posicionar um objeto na hierarquia significa definir o momento em que o mesmo será executado. Desta forma, o objeto mais alto na hierarquia solicita aos seus “filhos” (objetos ligados diretamente a ele) que sejam executados todos juntos (hierarquia paralela) ou seqüencialmente (hierarquia seqüencial).

2.3. JMusic (Versão 1.4.2)

jMusic é uma biblioteca de programação musical de alto nível escrita em Java, desenvolvida na Universidade de Tecnologia de Queensland (Brisbane – Austrália). Assim como algumas outras linguagens e ambientes de programação, o jMusic foi projetado para ser usado por músicos e não programadores, com a função de auxiliá-los no processo composicional. Entretanto, muitos programadores fazem uso do jMusic como uma poderosa API para desenvolver aplicações musicais, em especial instrumentos digitais, ambientes de educação e análise musical.

Pode-se citar como vantagens do jMusic:

- Por ter sido escrito em Java, o jMusic consegue manter as principais virtudes da linguagem como sua flexibilidade e portabilidade, além de usar conhecimento prévio de Java no aprendizado do jMusic;
- JMusic é gratuito e aberto distribuído sobre a GNU General Public License;
- Fácil aprendizado e uso, uma vez que foi construído de acordo com as convenções musicais tradicionais;
- Material musical construído em outros programas e interfaces musicais pode ser importado ou exportado com facilidade;
- Grande variedade de ferramentas utilitárias para visualização e audição da composição em construção.

As informações musicais são organizadas e armazenadas tal qual a pauta em papel, ou seja, a partitura é formada de partes, que por sua vez, é constituída de frases musicais onde estão contidas as notas. A nota é a estrutura básica usada no jMusic e traz consigo uma série de atributos, como: altura, volume, figura de tempo, controle de estéreo, duração e os acidentes (sustenido e bemol). Frase pode ser vista como vozes de uma parte – por exemplo, no piano cada mão tocaria uma voz. A frase só tem um atributo realmente importante, a lista de notas.

jMusic pode ser visto como um código para descrever música. A música pode ser representada em diversas notações diferentes levando-se em consideração aspectos como: conhecimento musical do usuário, quantidade de frases e partes musicais, objetivo do software em desenvolvimento, etc. A forma mais comum de representar a música é através da notação clássica, denominada no jMusic como *Common Practice Notation* (CPN). Atualmente, a implementação da CPN permite que somente frases musicais sejam exibidas e outras pequenas operações como: salvar em MIDI, modificar notas e formulas de compasso, tocar, etc. Apesar de bastante útil, a CPN possui restrições de implementação e possui difícil leitura para leigos em música. Uma das maiores desvantagens é fato de não conseguir exibir duas frases ao mesmo tempo em uma única janela. Uma notação que vem ajudar a transpor estas restrições é a *ShowScore*, um ponto intermediário entre a CPN e a *Piano Roll* (*Scratch*).

O jMusic possui um completo pacote de áudio que envolve síntese sonora, construção de instrumentos virtuais e uso de *samples* em composições.

Instrumentos virtuais são classes usadas para renderizar partituras do jMusic em arquivos de áudio (ou saída de áudio em tempo real) e são feitos a partir de objetos de áudio organizados em uma estrutura hierárquica denominada “corrente” (*chain*). Estas correntes de objetos de áudio definem todas as propriedades do som que se ouve e o tipo de síntese a ser utilizada para gerá-lo.

A organização hierárquica de objetos de áudio (e música) não é algo recente. Uma das primeiras linguagens de programação para música, Music V, escrita por Max Mathews nos Laboratórios da Bell – EUA, estabelecia a convenção de usar diagramas de fluxos de sinais. Outra opção dos instrumentos virtuais é usar amostras de som (*samples*). Nesta técnica, pequenos arquivos de som são gravados e definidos como notas que podem ter sua frequência (*pitch*) e duração modificadas e sequenciadas para gerar uma composição.

2.4. Wire/Wire Provider (Versão 0.97)

As primeiras versões do Java Sound não conseguiam trocar mensagens MIDI com dispositivos externos, o que só foi resolvido na versão JDK1.4.1 (em 2003). Para resolver este problema, alguns fabricantes e programadores disponibilizaram bibliotecas que permitem tal comunicação, porém muitas não eram escritas em Java, o que limita a portabilidade das aplicações que as usam.

Duas versões do Wire foram disponibilizadas. A primeira chama-se simplesmente Wire e foi escrita por uma pequena empresa alemã chamada Bonneville. É um exemplo típico de JNI (*Java Native Interface*), ou seja, classes escritas em C++ e aproveitadas pelo Java, no Windows. A outra versão foi escrita por Gerrit Gehnen e foi baseada na versão de Niel Gorisse (Bonneville) e no pacote para Linux da Tritonus.

2.5. JavaMIDI (Versão 5)

JavaMIDI é um conjunto de classes, escritas por Robert Marsanyi, que permite uso do MIDI sobre a plataforma Java assim como o Wire e o Wire Provider com a diferença de executar sobre Windows e Macintosh.

2.6. NoSuch MIDI (Versão Única)

Esta biblioteca é capaz de lidar com dispositivos MIDI externos, mensagens exclusivas, escalonamento em tempo real da saída, escrita e leitura de MIDI. Não precisa do Java Sound por ser uma implementação independente, o que a faz não executar em *browsers*. É gratuita para uso não comercial.

2.7. MIDI Share (Versão 1.9)

Feita a um tempo em que não existia suporte a MIDI no Java, mas sobreviveu ao Java Sound por não apresentar seus problemas com dispositivos MIDI externos e sobressaiu-se das demais bibliotecas similares por rodar em *applets*. Entretanto, é necessário instalar dois arquivos (bibliotecas nativas) no cliente JMidi e JPlayer.

2.8. MIDI Kit (Versão 1.0)

Esta biblioteca pode ser uma boa alternativa para quem precisa de portabilidade nas aplicações MIDI. Com ela é possível desenvolver com facilidade aplicações MIDI simples e locais, bem como servidores de processamento MIDI distribuído em rede que se configura dinamicamente para atender as necessidades de cada cliente.

Da mesma forma que outras bibliotecas que usam rotinas não escritas em Java, o MIDI Kit necessita que arquivos sejam postados no cliente para funcionar em *applets*.

Apesar do nome, o MIDI Kit também consegue processar áudio. Até o momento, somente o processamento de arquivos de som e execução dos mesmos estão disponíveis. O modo de funcionamento do MIDI Kit é baseado na arquitetura do NeXT Music Kit (Stanford – CCRMA, 1998).

2.9. JFugue (Versão 1.0)

JFugue é um conjunto de classes Java para programação musical. Esta biblioteca usa simples *strings* (cadeias de caracteres) para representar dados musicais, incluindo notas, acordes e mudanças de instrumentos. JFugue também permite a definição de música através de padrões que podem ser transformados para criar novos segmentos musicais derivados de peças musicais já existentes. Atualmente na versão 2.1, o JFugue já inclui suporte a controladores MIDI e melhorou o *parse* interno dos acordes, aumentando a gama de acordes trabalháveis.

2.10. Tritonus (Versão 1.x)

Tritonus é uma implementação com o código aberto do Java Sound 1.0 para Linux. É distribuída de acordo com os termos da GNU Library General Public License.

Tritonus é mais estável que o Java Sound, mas ainda não está completo e perfeito. Apesar de sua limitação de plataforma (somente Linux), o Tritonus é quase uma unanimidade entre aqueles que programam em Linux.

Existe uma relação unilateral do Java Sound em relação ao Tritonus, ou seja, o que for feito para o Java Sound, através do pacotes de SPI, deve funcionar no Tritonus.

Tendo em vista a estreita relação entre Tritonus e Java Sound, a seguir algumas funcionalidades são apresentadas:

- Suporta leitura e gravação de arquivos no formato .au, .aiff e .wav, assim como Java Sound, mas diferentemente deste não possui pacotes para *Service Providers* estenderem sua capacidades, tendo os mesmos que trabalhar direto no código fonte;
- Uma outra diferença do Java Sound é o fato do Tritonus não possuir um *mixer*, tendo que buscar algum disponível no sistema. Isto facilita a integração do código com a infra-estrutura de hardware da máquina mas também dificulta a criação de novos dispositivos de áudio, que têm que prover implementações para as interfaces Mixer, SourceDataLine, TargetDataLine e Clip;
- Standard linear: mono e estéreo, big e little endian, 8, 16, 24 e 32 bits, codecs A-law e mu-law;
- MP3 decoder (Javalayer 0.0.7 incorporado ao Tritonus, Java MP3 Player Project criação de MP3 usando a biblioteca LAME). O Java Sound não provém mais suporte a MP3 por problemas autorais.

As partes principais do suporte ao MIDI estão implementadas baseadas no sequenciador ALSA. Uma implementação baseada no MIDI Share está em desenvolvimento.

- Escrita e leitura de arquivos MIDI;
- Um sistema de síntese de software (TiMidity) não muito estável;
- Interface para sintetizadores em hardware (com restrições);
- MIDI IN/OUT (acesso a dispositivos MIDI externos).

2.11. JASS (Real-time Audio Synthesis - Versão 2.x)

JASS (Java Audio Synthesis System) é uma unidade geradora baseada em ambientes para programação de síntese de áudio. Escrita em Java puro, o ambiente baseia-se em um pequeno número de interfaces e classes abstratas que implementam a funcionalidade necessária pra criação de *patches*. *Patches* são criados juntando unidades geradoras em complexas estruturas e podem renderizar sons em tempo real. A comunicação com o hardware de áudio foi escrita com o Java Sound e, em algumas plataformas, JNI (*Java Native Interface*).

A biblioteca se propõe a ser uma alternativa ao Java Sound com baixa latência devido a métodos nativos. Atualmente possui implementação para Linux (ALSA e OSS), Macintosh (OS/X) e Windows (DirectX, ASIO *blocking* API, ASIO *callback* API). Todas as implementações, exceto ASIO *callback*, utilizam uma classe de entrada/saída de áudio em tempo real escrita em C++ por Gary P. Scavone. JASS tem o código fonte aberto e está disponível para uso não comercial.

Atualmente a biblioteca está na versão 2.012 e é atualizada constantemente. Em 2004 lançou um plug-in para IDE Eclipse, o que facilitou o desenvolvimento de

programadores terceiros. Entre as inovações da versão 2.x está o *Trace-Assertion* que ajuda a especificar o comportamento dinâmico do sistema em tempo de execução. Essa funcionalidade, antes obrigatória, passa a estar desabilitada por *default* na última versão lançada.

2.12. JSyn (Versão 14.2)

JSyn permite o desenvolvimento de programas Java para computação musical. Pode-se rodar como aplicações *stand-alone* ou como *applets* em *webpages* (usando o *plug-in*). Escrito em C para prover síntese em tempo real, o JSyn pode ser usado para gerar efeitos de som, ambientes de áudio ou música. JSyn se baseia no tradicional modelo de unidades geradoras que juntas podem gerar sons complexos. Seguem as principais características:

- Síntese em tempo real e de alta fidelidade usando a CPU;
- Biblioteca de unidades geradoras incluindo osciladores, filtros, envelopes, geradores de ruídos e efeitos;
- Todas as operações usam precisão de 32 bits;
- Pode-se combinar *samples* e sons sintetizados em tempo real;
- Fácil uso das classes Java para criar, conectar e controlar as unidades geradoras;
- Uso de *time-stamping* para programação de eventos no tempo;
- Uso de fila de *samples* e dados do envelope para programar repetição e colagem;
- Suporte a entrada de áudio para gravação e processamento de voz;
- Suporte para dispositivos multi-canal;
- Suporte para plataformas Windows, Macintosh e Linux;
- Editor gráfico (Wire) que permite gerar sons conectando unidades geradoras inteiramente, podendo exportar o código Java resultante;
- SDK e *plug-in* gratuitos, para uso não comercial.

2.13. JScore (Acompanha o JMSL 1.03)

JScore é um analisador sintático em Java para dados de música, que gera as partituras no formato XML e o respectivo código MIDI. O primeiro objetivo do JScore era demonstrar potencialidade do JLex (gerador de analisadores sintáticos), integrado ao JMusic que descreve a música no modo que o JLex deve entender. Atualmente o JScore integra o JMSL.

2.14. Xemo

Xemo é um projeto que visa desenvolver um *framework* para composição e notação musical. Em princípio o projeto não tem relação com qualquer tecnologia, ou seja, apenas define o que tem que ser feito. A API define interfaces para layout e renderização de símbolos musicais em alta resolução em dispositivos 2D, incluindo monitores e impressoras. Com isto, pode-se desenvolver programas de composição interativa, editores de notação musical, jogos e softwares educacionais.

Atualmente, a API é simplesmente um conjunto de classes gráficas vazias, ou seja, não possuem nenhuma funcionalidade para armazenar ou ler arquivos de áudio ou MIDI ou mesmo executar o que foi escrito.

A API contém funcionalidades específicas para representação musical, execução e composição interativa. Seus elementos base são pacotes para notação musical, representação de estruturas musicais e execução, e performance via MIDI. Apesar de terem sido lançados alguns pacotes, o projeto foi aparentemente descontinuado.

3. Comparativo

A seguir é apresentado um quadro comparativo entre as APIs analisadas dentro das classificações sugeridas.

3.1. Processamento de Áudio

Esta categoria caracteriza-se pela capacidade das APIs em processarem dados de áudio, de forma genérica. As funcionalidades desejáveis podem ser vistas na Tabela 1.

Tabela 1. Comparação entre APIs que processam áudio.

Critério/API	Java Sound	jMusic	MIDI Kit	JSyn	Tritonus
Gravação e reprodução de áudio	X	X	X	X	X
Conversão e suporte a diferentes formatos de áudio	X	X		X	X
Manipulação de <i>samples</i> (“ <i>samplear</i> ”)		X		X	
Manipulação e/ou processamento de dados de áudio	X	X			X

3.2. Síntese de Áudio

Esta categoria caracteriza-se pela capacidade das APIs em gerar/modificar sons baseando-se em algoritmos de síntese, de forma genérica. As funcionalidades desejáveis podem ser vistas na Tabela 2.

Tabela 2: Comparação entre APIs que sintetizam áudio.

Critério/API	JSyn	JASS	JMusic
Geração de áudio a partir da conexão de unidades geradoras (osciladores, envelopes, amplificadores etc.)	X	X	
Geração de áudio modificando parâmetros físicos das ondas sonoras		X	
Uso de instrumentos virtuais com parâmetros de áudio pré-estabelecidos e devidamente encapsulados	X		X
Suporte a diversos tipos de síntese, como <i>wavetable</i> , aditiva, subtrativa, FM, etc.	X	X	X

Vale lembrar que o JSyn integra o pacote do JMSL.

3.3. Seqüenciamento MIDI

Esta categoria caracteriza-se pela capacidade das APIs em escalonar eventos MIDI no tempo. As funcionalidades desejáveis podem ser vistas na Tabela 3.

Tabela 3: Comparação entre APIs que seqüenciam eventos MIDI.

Critério/API	Java Sound	JMusic	JMSL	MIDI Share	Tritonus
Possuir seqüenciador virtual	X			X	
Possuir mecanismo que permita programar/escalonar ações MIDI para serem executadas em determinado momento de tempo		X	X		X
Possuir mecanismo de sincronização	X		X	X	X
Leitura de gravação de arquivos MIDI	X	X	X	X	X
Controles da execução da música (andamento, posicionamento no tempo, dinâmica)	X	X	X	X	X
Controle dos canais (dinâmica, <i>mute</i> , solo, timbre)	X	X	X	X	X

3.4. Comunicação MIDI com Dispositivos Externos

Esta categoria caracteriza-se pela capacidade das APIs em enviar eventos MIDI a dispositivos externos. As funcionalidades desejáveis podem ser vistas na Tabela 4.

Tabela 4: Comparação entre APIs que trabalham com dispositivos externos.

Critério/API	Java MIDI	MIDI Share*	No Such	Wire	Wire Prov.	jMus.	MIDI Kit.
Capacidade para trocar mensagens MIDI com dispositivos de hardware externo	X	X	X	X	X	X	X
Portabilidade (pelo menos dois S.O.s)	X					X	
Suporte para <i>applets</i>		X	X			X	X
Comunicação via rede							X
Necessidade do Java Sound				X	X		

*O Midi Share também está incluído no pacote do JMSL.

3.5. Síntese a Partir de MIDI

Na classificação adotada por este trabalho, este item diferencia-se da síntese de áudio (Sub-seção 3.2), a qual permite também o uso de áudio como entrada, modificando-o.

Esta categoria caracteriza-se pela capacidade das APIs em gerar sons a partir de eventos MIDI. As funcionalidades desejáveis podem ser vistas na Tabela 5.

Tabela 5: Comparação entre APIs que geram sons a partir de eventos MIDI.

Critério/API	Java Sound	JMusic	Tritonus
Possuir sintetizador virtual	X	X	X
Possibilitar redirecionamento de mensagens MIDI para sintetizadores em hardware		X	X
Uso de <i>soundbanks</i> ou banco de timbres	X		X
Controle dos parâmetros de síntese via API (qualidade de áudio)		X	X
Persistência dos dados em MIDI e/ou áudio	X	X	X

3.6. Componentes Gráficos

Esta categoria caracteriza-se pela capacidade das APIs em exibir informações musicais, em diversas notações, através das interfaces gráficas. As funcionalidades desejáveis podem ser vistas na Tabela 6.

Tabela 6: APIs que exibem informações musicais em interfaces gráficas.

Critério/API	JScore	Xemo	JMusic
Exibição de dados MIDI em notação musical tradicional	X	X	X
Exibição de dados MIDI em notação musical alternativa			X
Funcionalidades de execução musical integradas	X		X

3.7. Representação Musical

Esta categoria caracteriza-se pela capacidade das APIs em trabalhar com outros formatos de arquivos usados para armazenar informações musicais. As funcionalidades desejáveis podem ser vistas na Tabela 7.

Tabela 7: APIs que trabalham com formatos de arquivos musicais.

Critério/API	JFugue	Xemo
Permite codificação de dados musicais em formatos ortodoxos, diferentes do convencional MIDI	X	X
MusicXML		X

3.8. Programação Musical (Composição)

Uma API musical pode ser usada em softwares destinados a diversos fins, entretanto possuem maior vocação para alguma atividade musical. Esta categoria é composta pela APIs criadas para uso em composição musical. As funcionalidades desejáveis podem ser vistas na Tabela 8.

Tabela 8: APIs criadas para uso em composição musical.

Critério/API	jMusic	JMSL	JSyn
Criação/edição gráfica de elementos musicais	X	X*	
Criação/edição por código de elementos musicais	X	X	X
Performances interativas	X	X	X
Gravação do material gerado	X	X	X
Suporte a MIDI	X	X	X
Síntese em tempo real	X	X	X

*MusicShape Editor

4. Conclusões

Este trabalho apresentou e analisou ferramentas que agregam ao Java a capacidade de trabalhar com áudio e dados musicais. Foram vistas características, arquiteturas e exemplos destas bibliotecas, que foram classificadas em 8 categorias distintas.

Chegou-se à conclusão que as melhores APIs para trabalho com som (de maneira geral) são o JSyn, jMusic e Java Sound. Para trabalhar com MIDI, as melhores APIs são jMusic, JMSL e Java Sound. O destaque deste trabalho foi a jMusic que se mostrou completa, estável e de fácil uso em todas as categorias. Certamente o usuário deve considerar a natureza da sua aplicação para optar por uma ou outra biblioteca.

Não se espera com este trabalho esgotar ou abranger todas as bibliotecas para programação musical em Java, somente apresentar as principais bibliotecas, visto o grande dinamismo com que são criadas diariamente.

Referências

- Didkovsky, N.; Burk, P. (2005) ‘Java Music Specification Language’, <http://www.algomusic.com/jmsl/>, ago. 2005.
- Doel, K. (2005) ‘Real-time Audio Synthesis using JASS’, <http://www.cs.ubc.ca/~kvdoel/jass/>, ago. 2005.
- Gehnen, G. (2005) ‘Java and Midi Programs by Gerrit Gehnen’, <http://www.geocities.com/ggehlen/>, ago. 2005.
- Grame (2005) ‘Recherche’, <http://www.grame.fr/Recherche/>, ago. 2005.
- Java Sound Resources (2005) <http://www.jsresources.org/>, ago. 2005.
- JScore (2005) <http://homepages.nyu.edu/~ray208/Jscoring/html/JScore.html>, ago. 2005.
- JSyn (2005) ‘Java Audio Synthesis’, <http://www.softsynth.com/jsyn/>, ago. 2005.
- JSynthLib (2005) ‘JSynthLib Home Page’, <http://www.jsynthlib.org/>, ago. 2005.
- Koelle, D. (2005) ‘JFugue - Java API for Music Programming’, <http://www.jfugue.org/>, ago. 2005.
- Marsanyi, R. (2005) ‘JavaMIDI’, <http://www.softsynth.com/javamidi/>, ago. 2005.
- McNabb, M. (2005) ‘Fantasia and The MIDI Kit’, <http://www.mcnabb.com/software/fantasia/>, ago. 2005.
- NoSuch MIDI. (2005) <http://www.nosuch.com/nosuchmidi/>, ago. 2005.
- Project Xemo (2003) <http://www.xemo.org/>, set. 2003.
- Sorensen, A.; Brown, A. (2005) ‘jMusic: Music Composition in Java’, <http://jmusic.ci.qut.edu.au/>, ago. 2005.
- Tritonus (2005) ‘Open Source Java Sound’, <http://tritonius.org/>, ago. 2005.
- Wire (2005) <http://www.bonneville.nl/software/Wire/>, ago. 2005.

Construindo Protótipos Musicais Cooperativamente na Web

Evandro M. Miletto, Marcelo S. Pimenta, Leandro Costalonga, Rosa Vicari,

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{miletto,mpimenta,llcostalonga,rosa}@inf.ufrgs.br,

Abstract. *This paper presents CODES - COoperative Music Prototype DESign, a Web-based environment for cooperative music prototyping. CODES aims to allow users (either interested lay people or experienced musicians) to make sound experiments and interact each other in order to create and refine simple musical pieces. Music Prototyping concepts are introduced followed by means a description of CODES architecture and behavior. The specific group awareness mechanisms of CODES are then described and illustrated by means some actual usage examples.*

Resumo. *Este artigo apresenta CODES - COoperative Music Prototype DESign, um ambiente para prototipação musical cooperativa baseado na web. CODES visa permitir que usuários (tanto leigos interessados como músicos) façam experimentos e interajam entre si para criar e refinar peças musicais simples. Os conceitos de prototipação musical são introduzidos e seguidos por uma descrição da arquitetura e comportamento de CODES. Os mecanismos específicos de Group Awareness de CODES são descritos e ilustrados por meio de exemplos de uso.*

1. Introdução

O crescente uso da Internet tem influenciado sobremaneira as mudanças pelas quais tem passado a tecnologia musical nas ultima década. Mesmo apresentando limitações ao tráfego de informação, a tecnologia da Internet torna o computador definitivamente uma ferramenta poderosa para propósitos musicais (Kon e Iazzeta 1998). Um exemplo prático é a *networked music*, que permite a artistas experimentais explorarem as implicações da interconexão de seus computadores (Barbosa 2003).

Neste artigo apresentamos CODES - “COoperative Music Prototyping DESign”, cuja idéia inicial foi apresentada em Miletto (2003). Seu propósito é permitir que usuários (tanto leigos quanto músicos experientes) façam experimentos musicais e interajam entre si na criação de peças musicais simples (aqui denominados “protótipos musicais” ou simplesmente protótipos). Prototipação não é uma palavra comum na literatura musical. Pessoas leigas, em princípio, não são compositores e o resultado das suas experiências criativas é aqui chamado deliberadamente de protótipo para realçar as diferenças. Estamos interessados em facilitar para qualquer usuário (leigo ou experiente) o acesso a experiências musicais envolventes e significativas, da mesma forma que Weinberg (2002).

O artigo está organizado como se segue. A seção 2 introduz o conceito de Prototipação Musical. Na seção 3, alguns trabalhos relacionados à composição musical cooperativa são resumidos. O sistema CODES, sua arquitetura, interface com usuários e aspectos de interação e outras características relevantes são apresentadas na seção 4. A seção 5 apresenta o suporte de CODES às atividades cooperativas e a seção 6 apresenta mecanismos de percepção do grupo, usando exemplos reais para ilustrar seu uso e comportamento. A seção 7 resume os resultados obtidos na avaliação de CODES, levando-se em conta métodos de avaliação subjetiva da área de IHC. Por fim, a seção 8 conclui o artigo.

2. Motivação para Prototipação Musical

Prototipação é um processo cíclico normalmente adotado pela indústria para criação da versão simplificada de um produto a fim de compreender suas características e processos de concepção e produção. Esse método objetiva criar sucessivas versões do produto de modo incremental, incluindo melhorias de uma versão para a próxima. Contudo, no campo da música, algumas peculiaridades tornam os processos de criação e concepção diferentes de como são realizados em outras áreas. A composição musical é uma atividade complexa onde não existe concordância sobre quais atividades devem ser executadas e em que ordem: cada pessoa tem seu estilo único e modo de trabalhar e, além disso, a maioria dos compositores ainda não tem a tradição de compartilhar suas idéias musicais e de colaborar durante as atividades composicionais. “Prototipação” não é uma expressão comum na literatura musical.

Em nossa opinião, música é um produto artístico que pode ser concebido pelo processo de prototipação. Uma idéia musical (nota, sequência de acordes, ritmo, estrutura ou pausa) é criada por alguém (tipicamente para execução em um instrumento musical) e a seguir cíclica e sucessivamente modificada e refinada de acordo com sua intenção inicial ou com idéias que surgem durante o processo de prototipação. Além dos músicos, os leigos em música provavelmente também possuem interesse em criar e participar de experiências musicais, mas carecem de ambientes orientados ao seu perfil de usuário. Na verdade nenhum conhecimento musical prévio deveria ser necessário para que qualquer usuário criasse seus protótipos musicais.

3. Trabalhos Relacionados

Esta seção resume as características de alguns ambientes para composição musical coletiva encontrados na literatura. Claramente deve-se notar que são ambientes de composição musical e não de prototipação musical.

Uma investigação a respeito de IMNs - Redes Musicais Interconexas (ou *Interconnected Musical Networks*) - propõe quatro níveis diferentes de interconectividade entre participantes, considerando o papel do computador como facilitador de suas relações sociais interdependentes: “Servidor”, “Ponte”, “Modelador” e “Ferramentas de Construção” (Weinberg 2002). A maior parte dos sistemas para composição musical baseados na Internet aqui descritos enquadra-se dentro do último nível (“Construction Kit”, em inglês). Nesse nível há alta interconectividade entre participantes, que normalmente são músicos experientes. É

permitido aos participantes contribuírem com seu próprio material e manipular (ouvir, alterar, refinar, etc.) a contribuição dos demais, normalmente de modo assíncrono e off-line.

O sistema PIWeCS (Whalley 2004) é um sistema de composição complexo baseado no diálogo entre agentes humanos/não-humanos com três objetivos principais: permitir ao leigo conhecer e explorar algumas das sonoridades dos tradicionais instrumentos Maori da Nova Zelândia em um espaço público; permitir igualmente que eles explorem combinações de alguns desses instrumentos em um contexto composicional e interativo; e possibilitar a extensão dos instrumentos em um contexto eletroacústico como um meio de exploração de sons híbridos. O PIWeCS integra agentes inteligentes com o software Max/MSP através de uma interface Web.

O sistema FMOL (Jordà 1999) relaciona-se a composição musical colaborativa em tempo real na Web. Utiliza um “plug-in” para permitir que vários usuários distribuídos trabalhem em conjunto sobre uma ou mais músicas. A colaboração é feita através de um modelo vertical de múltiplas trilhas.

O sistema EduMusical (Ficheman 2002) apóia a aprendizagem colaborativa e interativa à distância, tendo como objetivo ensinar música a crianças e adolescentes, sob orientação de educadores musicais de uma orquestra real - OSESP, a Orquestra Sinfônica do Estado de São Paulo. A composição coletiva é possível através da interação entre estudantes em salas de aula virtuais.

O sistema TransJam (Burk 2000) possibilita que músicos conectados ao seu sítio web toquem músicas em conjunto, selecionando os trechos (“loops”) dos instrumentos a serem tocados. Apesar do enfoque dado à execução, há algum apoio bastante simples à composição.

O Daisyphone (Bryan-Kins 2004) é um ambiente para improvisação musical em grupos remotos, apresentando um desenho inovador para ambientes musicais mais envolventes, sociais e de descoberta. Daisyphone tem seu foco na representação de música repetitiva e fornece suporte a colaboração remota e formulação de idéias.

PitchWeb (Duckworth 2003) é um instrumento musical multiusuário projetado especificamente para a Internet. A interação do usuário com o sistema permite escolher a ordem de execução de amostras sonoras, representadas pelas figuras. Existem outros sistemas que lidam com composição, como o Creating Music (Subotnick 2004) e o HyperScore (Farbood et al. 2004), que permitem que não-músicos componham (ou criem), de forma coletiva ou não.

O artigo de Rolf Woehrmann e Guillaume Ballet's (2002) examina arquiteturas cliente-servidor para computação musical, desenvolvidas em projeto de estúdio on-line do IRCAM. Os autores resumem aspectos de processamento de som distribuído e serviços de bancos de dados na World Wide Web.

Alguns aspectos, principalmente relativos a aspectos tecnológicos, comuns à maioria desses sistemas são: a) adoção de uma arquitetura tipo cliente-servidor, b) uso do formato MIDI, c) implementação em uma linguagem Java independente de plataforma, com exceção do FMOL, implementado em C++ e disponível apenas para Windows ou Linux e e) acesso

irrestrito a usuários comuns, ou seja, qualquer usuário pode conectar-se ou acessar o sistema sem custo adicional.

A maior parte deles pode ser adequadamente utilizada por usuários inexperientes; entretanto, na prática, usuários mais hábeis podem obter resultados melhores se possuírem conhecimento específico sobre síntese sonora (no caso do FMOL) ou se as atividades são supervisionadas por um tutor (no caso do EduMusical).

Além dessas características mais comuns e imediatas, propomos outros três aspectos importantes a serem considerados em um ambiente colaborativo para composição/prototipação musical:

1. Memória de Grupo: um mecanismo para armazenar e gerenciar o registro/ordenamento das ações e decisões dos membros de um grupo que manipula peças/protótipos musicais.
2. Apoio a sessões de prototipação longas. Um mecanismo importante em sessões de prototipação - na verdade em qualquer situação de projeto - é a capacidade para interromper a sessão e retomá-la para continuar o processo a partir do último ponto de parada.
3. Formatos de som para exportação/importação. Formatos disponíveis atualmente são os já conhecidos MIDI e Wave. Mas para nós o uso de algumas linguagens de marcação para música - como MusicXML, MML ("*Music Markup Language*"), MEI ("*Music Encoding Initiative*") e SMDL ("*Standard Music Description Language*") - é uma perspectiva interessante e um caminho necessário a explorar.

4. Visão Geral de CODES

Nesta seção apresentaremos uma breve descrição do sistema. O sistema CODES baseia-se na clássica arquitetura cliente-servidor (Figura 1). O *applet* Gerenciador de Manipulação Sonora, no lado cliente, tem a função de manipular o som (selecionar arquivos de som, mixar, tocar, parar, etc.), enviando os eventos do usuário aos gerenciadores CODES no lado servidor. As ações de convite, adição de comentário à peça musical e percepção de eventos ficam a cargo do Gerenciador de Cooperação, em conjunto com o Gerenciador de Usuário, para que haja autenticação dos usuários. Padrões sonoros são organizados no lado servidor pelo Gerenciador de Padrões Sonoros, que lê os diretórios, atualiza e busca os arquivos requisitados pelo Gerenciador de Manipulação Sonora. Todas as atividades relacionadas ao usuário (conexão, registro, autenticação) são executadas pelo Gerenciador de Usuário em comunicação com o Gerenciador de Cooperação para efetuar ações de cooperação entre usuários e grupos. O Gerenciador de Banco de Dados fornece o acesso à base de dados onde arquivos MIDI, dados da aplicação (mensagens, "logs", etc.) e dados dos usuários são armazenados.

Procura-se seguir o consenso existente na comunidade de IHC que diz: o desenvolvimento de GUIs começa com o entendimento dos usuários potenciais e as tarefas que eles deverão realizar no ambiente. Usando CODES, um usuário não necessita saber ler partitura musical para criar protótipos: ele pode simplesmente selecionar, executar e combinar arquivos sonoros (padrões sonoros) de forma interativa independente do formato de representação.

A interface com o usuário de CODES foi projetada objetivando considerar aspectos relativos à flexibilidade, robustez e facilidade de interação, bem como fornecer suporte adequado ao apresentar informação musical complexa, de modo a promover uma interação eficaz do usuário com o próprio ambiente. O ambiente foi projetado para chegar a um equilíbrio entre aquelas interfaces que são tão “fáceis” para o usuário que este acaba esgotando sua expressividade, e outras que são tão complicadas que desencorajam os iniciantes (D’Arcangelo 2002).

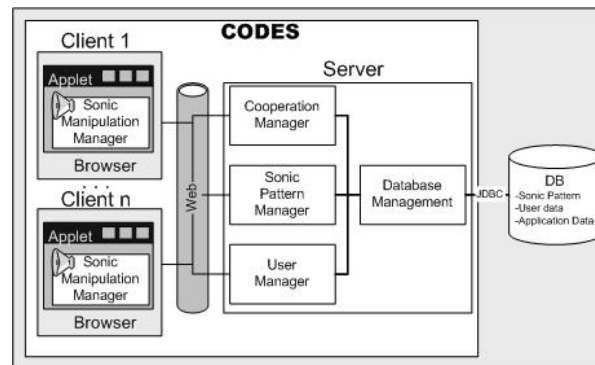


Fig. 1. Arquitetura do ambiente CODES

CODES considera que um protótipo musical é formado por linhas (de instrumentos, arranjos, tais como baixo, notas, percussão e base) que podem ser editadas. A edição é tipicamente feita pela seleção de padrões sonoros dentre os pré-definidos disponíveis em CODES. Padrões sonoros são estruturas musicais de alto nível (trechos de arquivos MIDI) que facilitam o processo de escolha e prototipação.

Um usuário pode criar mais de uma linha, ou seja, ter mais de um instrumento ou repeti-lo, se desejar (ver exemplo do usuário Alex na figura 2). Ao clicar no botão Play (>), o Gerenciador de Manipulação Sônica inicia a execução de todas as linhas que estão habilitadas para serem executadas. Todos os padrões agrupados verticalmente na mesma linha de tempo são mixados, executados e controlados por botões usuais (*play*, *stop*, *pause*, *forward*, *rewind*, *pause*)

Assim, a interação do usuário basicamente inclui ações como selecionar (através do clique) e tocar padrões sonoros, combinando-os com outros padrões selecionados pelos “parceiros” (demais usuários) do mesmo protótipo musical. Esta combinação pode se dar de diferentes maneiras: sobreposição (execução simultânea), justaposição (seqüenciamento), etc. (ver figura 2). Padrões sonoros são estruturas musicais de alto nível (pequenas partes de arquivos MIDI) que facilitam o processo de escolha de sons e prototipação. Os padrões que podem ser escolhidos para uma célula possuem pequenas diferenças entre si, mas mantêm o estilo e a duração, o que facilita sua adaptação à peça pelo usuário.

Clicando no botão Play, a execução dos padrões sonoros, mixados pelo gerenciador de manipulação sonora, é iniciada. Todas as linhas que tem a execução habilitada (ver coluna “status” figura 2) terão suas células mixadas e executadas ao mesmo tempo de acordo com a leitura da linha vertical de tempo, que fornece ao usuário um importante feedback de controle

de execução. A execução pode ser interrompida e reiniciada em qualquer tempo pelos botões usuais para controle de execução (*play, stop, forward, rewind, pause*).

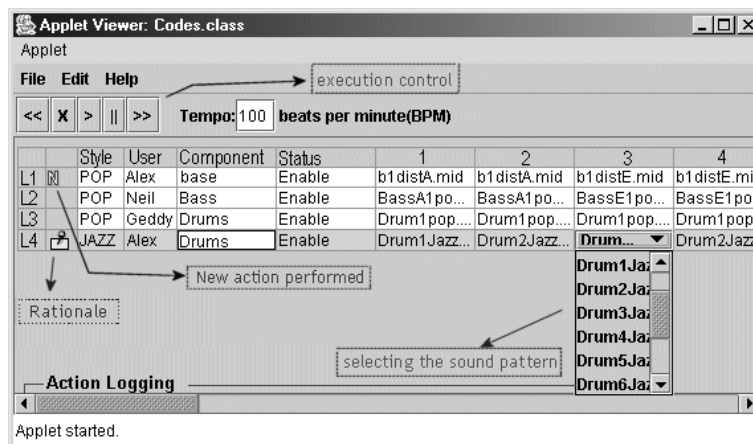


Fig. 2. Elements of CODES editing window

Outros detalhes da interface podem ser identificados na Figura 2. Três usuários (Alex, Neil e Geddy) são os proprietários das quatro linhas L1, L2, L3 e L4 existentes. O ícone do bilhete (vide *comentários* na figura) indica a presença de uma anotação ou de uma argumentação relacionada a alguma ação tomada na respectiva linha. A linha L3 pertence a outro usuário e uma possibilidade para o usuário ativo (user1) é desabilitá-la durante a execução, clicando em *Enable* e alterando para *Disable*.

A representação alternativa de estruturas musicais oferecida por CODES inclui conceitos como ritmo, tempo, melodia, harmonia e timbre, possibilitando ao usuário experimentar, escolher e combinar padrões sonoros de forma fácil e interativa e obtendo um resultado mais imediato neste processo, o que reduz possíveis dificuldades iniciais que poderiam ser apresentadas pelo uso de uma notação formal em um estágio inicial (Miletto 2003).

5. Atividades Cooperativas em CODES

Considerando o aspecto assíncrono de CODES, os usuários podem acessar o protótipo, fazer seus experimentos e escrever comentários em tempos diferentes. Autenticado no ambiente do CODES, um usuário poderá elaborar um protótipo musical inicial e solicitar a colaboração de outros “parceiros” através do envio de convites explícitos (normalmente usando recursos de correio eletrônico). Os parceiros que aceitam o convite podem participar da manipulação e do refinamento musical cooperativo.

A coordenação de todas as atividades no contexto musical pode acontecer naturalmente quando o grupo reconhece um participante como alguém que possui maiores habilidades musicais ou que seja mais experiente. Acreditamos que não seja necessária uma distinção explícita do papel do coordenador do grupo pois a hierarquização das ações do grupo não é nossa intenção. Por vezes, opiniões e ações de usuários reconhecidamente mais experientes em um grupo com um coordenador explícito pode vir a inibir a participação de outros usuários menos experientes.

O uso de CODES pode proporcionar alternativas interessantes para iniciantes em música. Por meio de interações e dicas com usuários mais experientes, o sistema oferece suporte para aprendizado, interdependência positiva, encorajamento de ações cooperativas, argumentação, discussão e aprendizado cooperativo durante o desenvolvimento de protótipo musical cooperativo.

Desta forma, o grupo de parceiros pode se transformar em uma comunidade virtual então CODES pode ser considerado como *communityware* (Liechti 2000). Tipicamente, uma *communityware* visa apoiar a formação de grupos informais de pessoas bem como as interações dessas comunidades. Assim, CODES é uma *communityware* para entretenimento. Com o aumento disponível de sistemas de comunicação públicos como no caso da Internet, assumimos um crescimento importante deste tipo de sistema.

6. Mecanismos de Percepção do Grupo em CODES

CODES proporciona 3 tipos de mecanismos de awareness: a) *Music Prototyping Rationale*: permite que usuários associem explicações às ações nos protótipos musicais, b) *Action Logging*: para manter explicitamente registrado o histórico dos passos e das decisões que conduziram o protótipo ao estado atual e c) *Modification marks*: para indicar a um usuário que o protótipo foi alterado por outros.

Os mecanismos de *awareness* têm muitas vantagens na prototipação musical:

- registrar e evolução das decisões;
- recuperar o progresso na prototipação musical e identificar conflitos, que podem iniciar um processo de negociação entre diversos pontos de vista.
- apoiar a construção de conhecimento cumulativo da prototipação;
- ajudar na integração de perspectivas de vários membros de um grupo.
- não há uma única resposta ou solução para um problema de prototipação musical.

A percepção (*awareness*) das ações dos membros de um grupo desempenha um papel crucial para o apoiar atividades cooperativas e multidisciplinares em CODES. Os principais aspectos desse mecanismo de percepção são discutidos a seguir.

A capacidade para associar argumentações a passos em um projeto é um processo pioneiro proposto na área de IHC e é chamado de *Design Rationale* (DR) (Lee & Lai 1991). *Design Rationale* é um mecanismo de comunicação da equipe de projeto para documentar as decisões críticas tomadas, quais alternativas foram investigadas e a justificativa para a alternativa escolhida. Há diversos modelos e notações para DR, como o IBIS (*“Issue-Based Information System”*) (Conklin 1998), e a Linguagem de Representação de Decisão DRL, ver (Wöhrmann 2002), um bom resumo das notações. Hoje em dia o DR é também adotado por outras disciplinas (como a Engenharia de Software, a Engenharia de Requisitos e a Engenharia de Sistemas) e reconhecido como um possível meio para auxiliar um membro de um grupo a entender melhor as decisões e ações dos outros membros do grupo. Ações e decisões musicais são em geral subjetivas, e daí a importância de se ter um mecanismo de comunicação específico para a argumentação das ações, de modo a informar as razões de

cada ação tomada aos demais membros do grupo, como selecionar um padrão sonoro, instrumento, pausa, etc. ou a decisão de fazer combinações ou excluir elementos.

Consideramos um processo de criação musical ou de fazer experimentos musicais também um processo que pode ser composto por decisões e escolhas. Quando os usuários estão prototipando em CODES, eles combinam suas peças musicais/sonoras nas suas linhas com outras linhas de outros usuários. Assim, escolhas, seleções, habilitar, desabilitar e executar (tocar) são tarefas realizadas constantemente em processo cíclico até que se atinja um consenso sobre o resultado da prototipação. Todas essas ações podem ser argumentadas no sistema por usuários para que possam ser informadas aos outros os motivos que levaram a estas ações. Esta é de fato uma maneira segura de garantir a percepção (“awareness”) em ambientes colaborativos assíncronos.

Os elementos básicos de *Music Prototyping Rationale* de CODES basicamente são Tópicos (ou assuntos) e Comentários. Tópicos correspondem a decisões, ações e estados alcançados durante a criação de um protótipo musical colaborativo e seu refinamento. Por exemplo, um tópico pode ser “trocar um instrumento de uma linha, inserir uma pausa, criar uma nova linha, misturar diferentes ritmos, etc.” (ver “*subject*” na figura 3(a)). Tópicos são motivados por escolhas consensuais e alternativas relacionadas das ações em curso.

Comentários são declarações feitas para apoiar a seleção do curso específico de uma ação (comentários a favor - pro) ou advertir o interesse de usuários através de uma expressão de objeção (comentários contra).

Além disso, comentários podem expressar sugestões, perguntas ou observações genéricas sobre um tópico. Não há, entretanto, um tipo específico de mensagem. Toda a decisão ou ação pode ser conectada (a favor ou contra) a comentários (ver figura 3(a)).

Um exemplo prático de *music prototyping rationale*, após uma sessão de experimento rítmico no ambiente de CODES, é descrita como se segue. Três usuários chamados Alex, Neil e Geddy participam do mesmo protótipo musical. Entretanto, Alex tem a idéia de misturar diferentes estilos de ritmos e decide criar outra linha de Jazz dentro do estilo Pop. Para obter a opinião dos outros participantes do protótipo Alex escreve uma explicação deste seu experimento.

Na janela de comentário, a edição de um comentário inclui o assunto e um corpo de texto livre para o usuário comentar. O usuário pode marcar uma característica opcional que é tornar o comentário privado (para controle próprio) ou público (para conhecimento dos demais).

CODES salva o comentário e o associa aos tópicos correspondentes do protótipo, informando aos outros usuários através de um ícone do tipo *post-it* (ver “*Rationale*” na figura 2) que algum comentário foi feito por algum usuário relacionado a alguma ação executada na linha. Clicando neste ícone, o usuário recupera o comentário. Conforme mostrado na figura 3(b), nossa abordagem de *music prototyping rationale* usa uma estrutura hierárquica para representar as razões dos usuários. Cada entrada em “*Description*” corresponde a um elemento de argumentação. Nesta janela cada elemento é acompanhado por três ícones, um (+ / -) que serve para propósitos de apresentação, outro (lupa) indica que o comentário foi lido e o último (o alto-falante) para indicar que o evento sonoro foi ouvido pelo usuário

correspondente. O Log de ações (*Action Logging*) - CODES tem um mecanismo de log para registrar informações (como data, autor, ação, elemento do protótipo afetado, etc) de todas as ações tornando-as disponíveis para todos os usuários para advertir o que foi realizado em qual sequência, bem como apresentar textualmente o histórico destas realizações, tendo como uma Memória de Grupo.

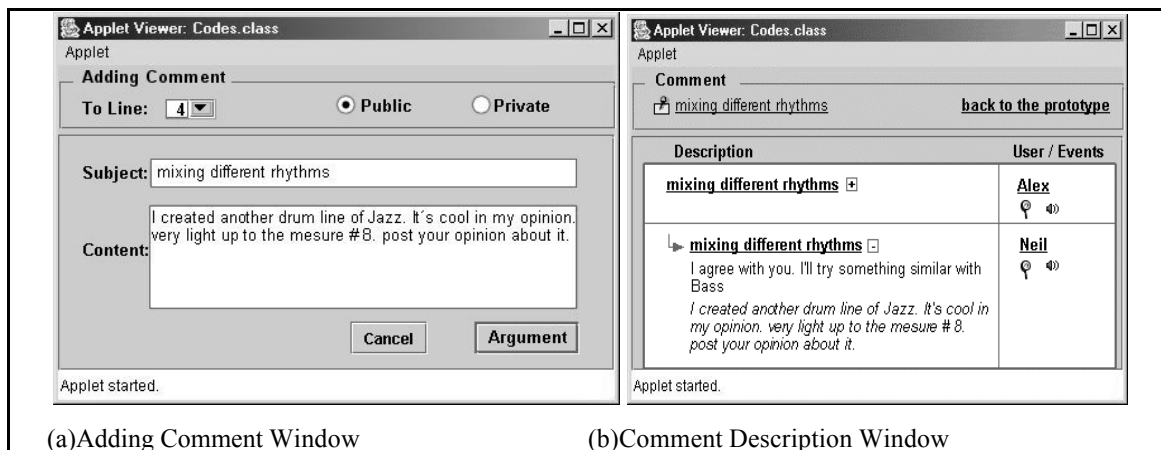


Fig. 3. Janela de comentários de CODES

A Figura 4 apresenta uma tela de CODES onde é possível verificar um trecho de um registro de ações organizadas em uma sequência cronológica. Conforme mostrado na Figura 4, o usuário pode filtrar o nível da informação registrada por eventos disparados pelo sistema. Navegando nesta janela, todos os usuários podem recuperar as etapas pelas quais passou o atual protótipo musical.

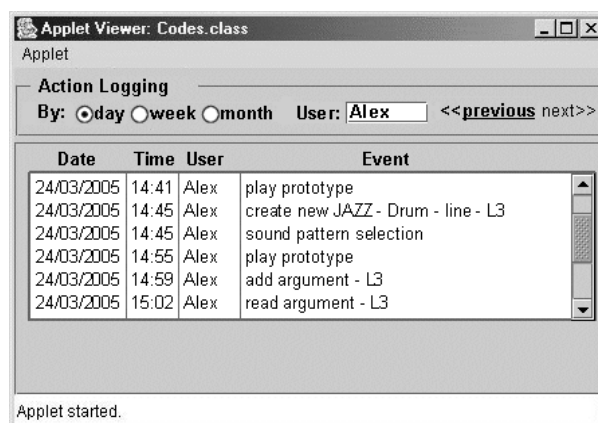


Fig. 4. Trecho da Tela de Registro de Log

Marcas de Modificação (*Modification Marks*) – CODES suporta sessões longas de prototipação e atividades cooperativas as modificações que podem perdurar de poucos dias a anos e as pessoas que cooperam em um protótipo musical devem ter acesso fácil às modificações. Assim, as persistências destas modificações devem ser mantidas através das seções e as notificações da sua existência devem ser explicitamente mostradas para todos os usuários. Neste sentido, o exemplo da figura 2 ilustra esta idéia, onde o ícone “N” (*New*) que aparece na linha indica que um protótipo foi modificado

7. Avaliando o Uso de CODES

CODES foi disponibilizado para uso em contexto acadêmico restrito. Levando-se em conta alguns métodos de avaliação da literatura de IHC e Ergonomia (Dix 1998), temos aplicado entrevistas orientadas à satisfação do usuário como procedimento de avaliação simples. Para capturar críticas e comentários relevantes (e também para evitar opiniões não-gravadas) os usuários são convidados a responder todas as questões *in loco*, imediatamente após o uso de CODES. As entrevistas contêm questões objetivas cujo processo de análise se torna rápido, facilitado e com baixo custo.

As perguntas foram concebidas para identificar conveniência da ordem e estrutura da informação para as atividades do usuário durante o processo de criação/edição do protótipo bem como da colaboração, incluindo itens específicos sobre usabilidade, acessibilidade, complexidade de navegação e também se o usuário está satisfeito com as funcionalidades implementadas. Além das respostas objetivas, a entrevista dá ao usuário oportunidade de fornecer comentários e para nós uma boa oportunidade para analisar seriamente os comentários.

Os resultados preliminares apontam para a necessidade de melhoras em relação a alguns aspectos na interface de CODES, tais como:

- implementar uma forma melhor de controle de tempo possibilitando o usuário parar em qualquer tempo e recomeçar deste último ponto;
- ouvir o padrão sonoro antes de selecioná-lo na linha;
- permitir ao usuário fazer o *upload* dos seus próprios arquivos musicais nas linhas de CODES;
- estabelecer uma padronização para representar as informações musicais de alto nível na interface gráfica do sistema.

Os resultados obtidos dos métodos de avaliação até agora têm mostrado um relativo sucesso do nosso trabalho, mas surpreendentemente um dos resultados mais interessantes é o conjunto de alternativas possíveis que os usuários encontram para o uso de CODES. De fato, além dos procedimentos tradicionais para os quais desenvolvemos as suas funcionalidades, alguns usuários encontraram outras aplicações para uso do ambiente, entre elas: a) suporte efetivo de aprendizado musical, b) ferramenta de entretenimento e c) sistema de acompanhamento para performance.

Em situações de aprendizado de música, CODES pode proporcionar alternativas interessantes para iniciantes em música. Um protótipo coletivo é criado por um grupo formado por estudantes supervisionados pelo professor de música (Miletto 2004) (Miletto 2005). O grupo, através das interações e orientações do professor, decide que estilo musical será estudado, bem como o número e o tipo de instrumentos a serem utilizados neste protótipo. A partir daí, é possível trabalhar na criação musical de forma coletiva, usando a metáfora de uma orquestra: cada aluno tem um papel definido no resultado final. Além disto, o professor pode habilitar diversos padrões relacionados ao mesmo instrumento para alunos diferentes e todos podem comparar suas diferentes contribuições, escolhendo ou combinando alternativas. Assim, a combinação entre prototipação musical e educação musical é promissora e

merecedora de pesquisas adicionais. Particularmente, é nossa intenção oferecer funcionalidades computacionais de apoio ao ensino de música seguindo o modelo (T)EC(L)A de Swanwick (1979).

Como ferramenta de entretenimento o CODES permite experiências atraentes àqueles que se interessam por atividades como a de DJ (que usa a metáfora de colagem musical). Escolhendo os padrões sonoros do sistema, os usuários têm a possibilidade de misturar diferentes ritmos, partes de músicas e estilos musicais apenas criando linhas paralelas destes e então as tocando juntas. Além disto, com um conjunto de linhas adicionadas com padrão sonoro selecionado, o usuário pode obter vários estilos de um mesmo protótipo musical, apenas selecionando ou desmarcando os grupos de linhas que deseja, experimentando diferentes combinações de sonoridades.

Outra possibilidade é usar o CODES como um sistema de acompanhamento à execução musical humana. Um usuário experiente pode criar algumas linhas no CODES de forma a simular o acompanhamento por uma banda. Supondo que ele toque guitarra, as linhas de baixo, ritmo e acordes (base) podem ser tocadas enquanto este usuário toca seu instrumento musical. Desta maneira, o CODES também pode ser visto como uma ferramenta auxiliar ao estudo individual de prática de instrumento.

8. Considerações Finais

Apresentamos CODES, um ambiente para cooperação entre usuários para criar protótipos musicais coletivos. A abordagem CODES para cooperação entre usuários na criação de protótipos musicais coletivos é um exemplo de ferramenta bastante promissora, pois permite compartilhar conhecimento através de uma interação rica e de mecanismos de argumentação associados a cada modificação no protótipo. Conseqüentemente, cada participante pode compreender os princípios e regras envolvidas no complexo processo da experimentação e criação musical. Evidentemente, não está em discussão a qualidade musical do trabalho final, e sim a possibilidade de “criá-lo”.

As ferramentas de apoio a esse processo devem empregar tecnologias e abordagens mais eficientes para adequar-se a sua complexidade e diversidade crescente. Um exemplo é a necessidade de lidar com prototipação musical cooperativa, que é um tópico de importância crescente, em especial, mecanismos de *awareness*. Nosso objetivo inicial foi desenvolver um mecanismo útil e ativo que não apenas estruturasse a informação envolvida no processo de prototipação musical, mas também ajudasse usuários durante este processo. Acreditamos que a integração do mecanismo de *awareness* aqui discutido é uma forma razoável para facilitar a cooperação entre usuários e também facilitar interações não planejadas, e conseqüentemente, melhorar as possibilidades de prototipação musical.

9. Referências

- Barbosa, A. Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation. Leonardo Music Journal 13. MIT Press, (2003) Cambridge MA.
- Bryan-Kinns, N. 2004. Daisysphone: The Design and Impact of a Novel Environment for Remote Group Music Improvisation" in Proceedings of DIS 2004 – ACM Symposium on Designing Interactive Systems 2004, 135 – 144.

- Burk, P. "Jammin'on the Web – a new Client/Server Architecture for Multi-User Musical Performance.Visual". In Proceedings of the International Computer Music Association Conference 2000, Berlin, 117-120.
- Conklin, J. 1998. The IBIS Manual: A Short Course in IBIS Methodology. Viewed at: <http://www.gdss.com/IBIS.htm>
- D'arcangelo, G. 2002. Creating a Context for Musical Innovation: A NIME Curriculum. Proceedings of the 2002 Conference on New Instrument for Musical Expression - NIME2002, Ireland.
- Dix, A., Finlay, Janet, Abowd, Gregory, Beale, Russell. Human-Computer Interaction, Second Edition. London, Prentice Hall Europe. (1998)
- Duckworth, W. "Making Music on the Web". Leonardo Music Journal, Vol. 9, (2003). 13 – 18, MIT Press.
- Farbood, M.; Pasztor, E.; Jennings, K. "Hyperscore: A Graphical Sketchpad for Novice Composers". IEEE Computer Graphics and Applications, Volume: 24, Issue: 1, Year: Jan.-Feb. (2004).
- Ficheman, I. K. Collaborative Distance Learning Supported by Interactive Electronic Media: A case study. Master Thesis . Escola Politécnica da Universidade de São Paulo. São Paulo, 2002. (In Portuguese)..
- Jordà, S. (1999) Faust Music On Line: An approach to real-time collective composition on the Internet. Leonardo Music Journal, Vol 9, 5-12., (1999)
- Kon, F. e Iazzetta, F. Internet Music: Dream or (Virtual) Reality? In: V Simpósio Brasileiro de Computação e Música, 1998, Belo Horizonte. Anais... Belo Horizonte: Escola de Música / UFMG, 1998. p.69-81
- Liechti, O. Awareness and the WWW: an overview ACM SIGGROUP Bulletin, V. 21 , Issue 3 (Dec 2000) P: 3 – 12, ACM Press.
- Lee, J. & Lai, K.-Y.. "What's in Design Rationale?" Human-Computer Interaction Special Issue on Design Rationale 6(3-4) pp. 251-280. 1991.
- Miletto, E. M.; Pimenta, M. S. Towards a Web-based Environment for Cooperative Musical Composition. In: Proceedings of the IX Brazilian Symposium on Computer Music, Campinas, Brazil. 2003
- Miletto, E. M.; Pimenta, M. S.; Costalonga, L. L. Using the Web-Based Environment for Cooperative Music Prototyping CODES in Learning Situations. In: 7th International Conference on Intelligent Tutoring Systems, 2004, Maceió. Proceedings of the International Conference on Intelligent Tutoring Systems. Springer-Verlag, 2004. p. 835-837.
- Miletto, E. M.; Pimenta, M. S.; Vicari, R.; Using Codes: Cooperative Music Prototyping and its Educational Perspectives. International Computer Music Conference, ICMC2005. Barcelona. A ser apresentado.
- Subotnick, "M. Creating Music". Available in the web at <http://creatingmusic.com/>, accessed in March (2005).
- Swanwick, Keith. "Music, mind, and education". Music Education in a Pluralist Society 1988, International Journal of Music Education, No. 12. 1988.
- Weinberg, G. "The aesthetics, history, and future challenges of interconnected music networks." Proceedings of the International Computer Music Association Conference, Göteborg, Sweden, (2002)
- Whalley, I. 2004. PIWeCS: enhancing human/machine agency in an interactive composition system. Organised Sound 9(2):167-174,.
- Wöhrmann, R. Ballet, G. "Design and Architecture of Distributed Sound Processing Systems for Web-Based Computer Music Applications". Computer Music Journal 23, 73-84. 2002.

Self-Organizing Topological Timbre Design Methodology Using a Kohonen Neural Network

Marcelo Caetano^{1,2}, César Costa², Jônatas Manzoli², and Fernando Von Zuben¹

¹Laboratory of Bioinformatics and Bio-inspired Computing (LBiC)

²Interdisciplinary Nucleus for Sound Studies (NICS)

University of Campinas (Unicamp), PO Box 6101 - 13083-970, Brazil

{caetano,vonzuben}@dca.fee.unicamp.br; {jonatas,cesar}@nics.unicamp.br

Abstract. *Generating sounds for music composition with the desired timbral characteristics has been a challenge ever since the dawn of electroacoustic music. Timbre is a remarkably complex phenomenon that has puzzled researchers for a long time. Actually, the nature of musical signals is not fully understood yet. In this paper, we present a sound synthesis technique that uses Kohonen's one-dimensional self-organizing map to generate neuronal-sounds to respond to a fixed and predefined set of stimulus-sounds, producing timbral variants with the desired characteristics. The self-organizing algorithm provides maintenance of topology so that the intended aesthetical result is properly achieved by avoiding the formal definition of the timbral attributes. To evaluate the obtained results we propose crossing a mathematical/subjective spectral distance from the neuronal-sounds to the stimulus-sounds with the method of timbral classification using Kohonen's two-dimensional self-organizing map.*

1. Introduction

Computer music is an ever-growing field partly because it allows the composer such great flexibility in sound manipulation when searching for the desired result. In the particular case of music composition, once the search space and the goals are defined, a technique for achieving the final product is required. Within the frame of this work, when we consider music improvisation, there is no goal and no such thing as a final result. It is the actual path through the search space that is of interest. Many different approaches have been proposed to meet the requirements of the process, i.e. creating interesting music, with results that vary from the unexpected to the undesired, depending upon a vast number of factors and on the methodology itself. Traditional sound synthesis techniques present limitations especially due to the fact that they do not take into consideration the subjective and/or the dynamic nature of music, by using processes that are either too simple or not specifically designed to handle musical sounds [Caetano et al. 2005 a].

Musical timbre is the characteristic tone quality of a particular class of sounds. Timbre is much more difficult to characterize than either loudness or pitch. No one-dimensional scale – such as the loud/soft of intensity or the high/low of pitch – has been postulated for timbre, because there exists no simple pair of opposites between which a scale can be made. Because timbre has so many facets, computer techniques for multidimensional scaling [Grey 1975; Grey and Moorer 1977] have constituted the first major progress in

quantitative description of timbre, since the pioneering work of Hermann von Helmholtz (1885) in the nineteenth century. Since then, researchers have determined a more accurate model of natural sound. Digital recording has enabled the contemporary researcher to show that the waveform (and hence the spectrum) can change drastically during the course of a tone. Risset and Wessel (1982) observed that complex sounds have dynamic spectra and the evolution in time of the sound's spectrum plays an important part in the perception of timbre [Grey and Moorer 1977]. Timbre variations are perceived, for example, as clusters of sounds played by a particular musical instrument, or said by a particular person, even though these sounds might be very distinct among themselves, depending upon its pitch, intensity or duration. In fact, the concept of timbre has always been related to sounds of musical instruments or speech, and it is in this scope that the majority of researches on timbre have been developed. These works identified innumerable factors that form what is called timbre perception.

Many researchers have recently suggested the creation of Bio-Inspired and Artificial Intelligence (AI) based systems for music composition and improvisation. Applications of Bio-Inspiration and AI in music composition involve artificial neural networks [Chen and Miiikulainen 2001], cellular automata [Burraston et al. 2004], artificial immune systems (AIS) [Caetano et al. 2005 a], particle swarms [Blackwell and Young 2004] and evolutionary computation (EC) [Biles 1994; Horowitz 1994; Caetano et al. 2005 b]. Refer to the work of Santos et al. (2000) for a detailed review of the application of EC in music systems. As a preliminary step toward the current proposal, Caetano et al. (2005 a,b) suggested the use of AI to pursue stationary/fixed target sounds that are considered the user's desired timbral outcome. The reported results can be interpreted as a sort of spectral blend between the initial and target sounds.

In this work, we are focusing primarily on the production of sounds that present complex spectral dynamic features for musical applications taking self-organization as paradigm. We propose the use of a one-dimensional Self-Organizing Map (SOM) in our approach to timbre design, founded on unsupervised learning and on the ability of SOM to orderly arrange the original soundspace in a cyclical fashion, proposing a tentative timbral improvisational scale.

SOMs are the most commonly used strategy in Artificial Neural Networks (ANNs) for unsupervised learning [Lippman 1987]. During the training process the neurons tend to represent statistical properties of the input data, preserving the topology of the input space (soundspace), even though it is unknown. It is a handy tool for feature analysis of high-dimensional data, allowing visualization in a low-dimensional neuron layer space.

The main application of SOM is data clustering using two-dimensional mapping [Kohonen 1984 b]. One-dimensional SOM has been applied to the Travelling Salesman Problem (TSP) [Gomes and Von Zuben 2003] and as a topological ordering method of multidimensional data. We found several different proposals of timbre taxonomical classification making use of SOM as a feature extraction tool [Damiani et al. 1995; Cosi et al. 1994 a,b; De Poli and Tonella 1993; De Poli and Prandoni 1997; Loureiro et al. 2004; Feiten and Gunzel 1994]. The authors did not find any applications of one-dimensional SOMs in timbre design in the literature.

The subsequent sections describe the fundamentals of SOMs and the way they are related to the development of our timbre design technique. Experimental results are then

described and analyzed. Finally, concluding remarks and perspectives for further research are considered.

2. Kohonen's Self-Organizing Feature Maps

One important organizing principle of sensory pathways in the brain is that the placement of neurons is orderly and often reflects some physical characteristic of the external stimulus being sensed [Kandel and Schwartz 1985]. For example, at each level of the auditory pathway, nerve cells and fibers are arranged anatomically in relation to the frequency which elicits the greatest response in each neuron. This tonotopic organization in the auditory pathway extends up to the auditory cortex [Moller 1983]. Although much of the low-level organization is genetically pre-determined, it is likely that some of the organization at higher levels is created during learning by algorithms which promote self-organization. Kohonen (2000) presents one such algorithm which produces what he calls self-organizing feature maps (SOMs) similar to those that occur in the brain.

Kohonen's algorithm creates a mapping of high-dimensional input data into output nodes arranged in a low-dimensional grid, characterizing a vector quantizer [Lippmann 1987]. Output nodes are extensively interconnected with many local connections. During training, continuous-valued input vectors are presented either sequentially in time or in batch without specifying the desired output. This is called unsupervised learning. In addition, the weights will be organized such that topologically close nodes are sensitive to inputs that are physically similar. Output nodes will thus be ordered in a natural manner. This may be important in complex systems with many layers of processing because it can reduce lengths of inter-layer connections. After enough input vectors have been presented, weights will specify clusters or vector centers that sample the input space such that the point density function of the vector centers tend to approximate the probability density function of the input vectors [Kohonen 1984 b]. Kohonen demonstrates how SOMs can be used in a speech recognizer as a vector quantizer [Kohonen 1984 a].

2.1 SOM's algorithm

After the synaptic weights initialization, the learning procedure enters upon an episodic loop that just stops when a defined final condition is achieved. Each epoch corresponds to a learning procedure whereby every input data is presented to the network. For each data input the procedure is divided into three processes. In the competitive process, the output node with the shortest distance to the input data, called Best Matching Unit (BMU), is selected to learn the input. Euclidean distance is normally used as a distance metric [Kohonen 2000]. In the cooperative process, the nodes that support the BMU's victory are also selected to learn, but in a lower level related to the nodes' help effort. The degree of cooperativeness of a node is defined by a neighborhood function that is monotonically decreasing with the nodes' distance to the BMU [Kohonen 1984 b]. At last, it's in the adaptive process that the learning takes place. The weights of each node are updated by the learning procedure shown in equation (1). At each epoch (n), the weight vector (ω) of each node (j) is changed in the direction of the input data vector (x). The learning degree is obtained by the product of the current global learning rate (η) and the neighborhood function (h) of the BMU ($i(x)$), considering the node being updated.

$$\omega_j(n+1) = \omega_j(n) + \eta(n) \times h_{j,i(x)}(n) \times (x - \omega_j(n)) \quad (1)$$

The global learning rate and the dispersion of the neighborhood function decreases exponentially in time [Gomes and Von Zuben 2003]. This policy grants two different stages on the map's development: a rough and fast convergence with initially high learning values; a fine tune with the decrease of the learning values. The initial values for rate and dispersion and their time constants act as control parameters to the dynamics of SOM's generation and to the quality of the final mapping.

2.2 Clustering

A class is defined as a data group with similar properties, as illustrated in Figure 1a. Different classes have non related properties. The topology preserving feature implies that correlated data are mapped into close regions in the array of neurons. Therefore, an output node will be closer to nodes related to data from the same class than to output nodes that represent data from other classes. A cluster can be identified as a group of output nodes nearly located in terms of their weight vectors produced by the learning phase.

2.3 U-Matrix

The U-Matrix is a useful tool for clustering visualization in SOMs. It represents an average picture of the distance profile between the weight vector of each neuron and the weight vector of its immediate neighbors. High values in the U-matrix indicate neighbor neurons with distant weight vectors, and low values indicate neighbor neurons with high-correlated weight vectors, so that they will be stimulated by similar input patterns [Ultsch 2003]. As seen in Figures 1b and 1c, valleys denote neurons with similar behavior, being an indication of a cluster. High areas indicate that neighbor neurons have dissimilar weight vectors, revealing transition between two distinct clusters.

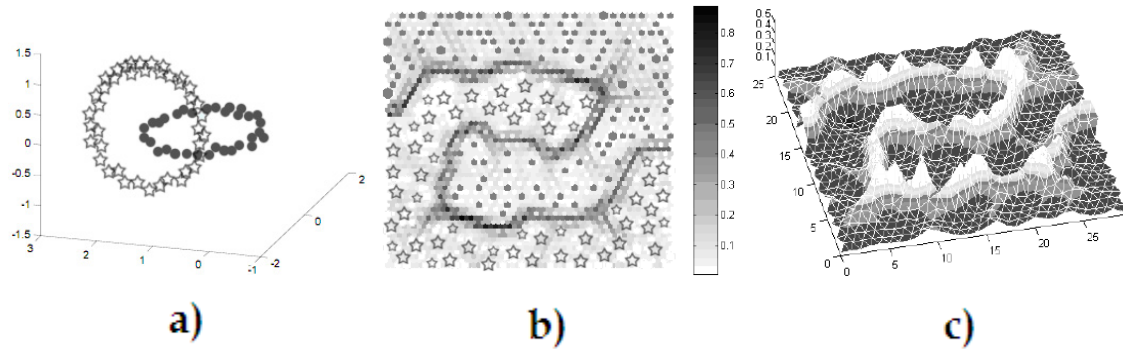


Figure 1 U-Matrix for a two-dimensional map. (a) 3D two-class data set; (b) U-Matrix visualization after clustering (gray levels indicate height and data hit marks identifies winning neurons); (c) 3D visualization of U-Matrix, with two contiguous valleys (dark areas) and peaks (light areas) characterizing the frontier between the two valleys. After Zuchini (2003).

3. Neural Network Timbral Improvisation

Here, we present a timbre design method that allows the composer to express a certain degree of subjectivity by simply choosing the number of neurons and setting the parameters adequately, according to aesthetical preferences. The user is enabled to find

candidate solutions that meet certain musical requirements by using a set of waveforms (stimuli) as examples of the desired timbres (Figure 2 a). Instead of describing the sounds using numerical parameters or any other linguistic tool, we used a set of waveforms to characterize timbre. Smalley (1990) declared that the information contained in the frequency spectrum cannot be separated from the time domain, because “*spectrum is perceived through time and time is perceived as spectral motion*”. Thus, by specifying the target waveforms (stimuli), the user is also specifying the spectral contents and the timbral characteristics of the tones. Grey (1975) discusses the advantages of time domain representation. We aim at sound design by means of the specification of the spectral contents using time-domain representation and manipulation.

Timbre soundspace is unknown and there is no consensus ordering or classification (Figure 2 a). Due to the self-organizing feature of SOM, it is possible to propose timbral arrangements that respect the original topology (Figure 2 b). The key feature of SOM that allows this process is that stimuli with similar characteristics trigger neurons in close regions of the one-dimensional mapping that represents the topological neighborhood in the original soundspace. In our application, self-organization gives rise to two different musically profitable phenomena. Firstly, one stimulus might trigger more than one neuron, causing the result to represent timbral variations of the original (stimulus) sounds (Figure 2 c). Secondly, more than one stimulus-sound might trigger the same neuron (zoomed-in areas in Figure 2 b). The expected result is a timbral merger of the stimuli corresponding to the neuronal-sounds that responded to these inputs. The resultant one-dimensional arrangement corresponds to a cyclic ordering of the stimulus-sounds that can be regarded as a proposal of a sort of timbral scale. The concept of timbral improvisation emerges from the possibility of following this orderly self-organizing path provided by the method in much the same way scales are used in traditional music improvisation. Moreover, the very dynamic convergence process of the neuronal sounds from the initialization to the final result can be sequentially played. This process would reveal the timbral neurological-induced transformation resulting from the path followed by each neuron during the self-organizing process.

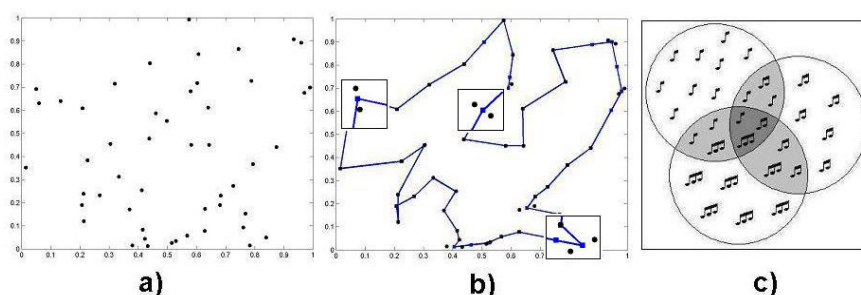


Figure 2 Depiction of the vector quantization ability of SOMs. Part (a) shows the original data topology; part (b) shows the resultant one-dimensional SOM representing the original data topologically arranged; and part (c) illustrates the common timbral features of three variants of sounds.

3.1. Representation

The input parameters of the present implementation are shown in Table 1. Each individual is codified as a vector composed of L samples of a given waveform at a

sampling frequency of SF samples per second. The individuals are, thus, represented in time domain, as vectors in \mathbb{R}^L . The individuals are arranged as a one-dimensional circular SOM with the desired number of output nodes (neuronal-sounds). After training, the weight vectors represent the input data in the same vector space (with the same number of dimensions). The procedure used is similar to a TSP solution obtained via SOMs [Gomes and Von Zuben 2003].

Table 1. Input parameters that can be controlled by the user

Parameter	Description
L	Number of samples per individual
SF	Sampling rate
G	Number of Stimuli
N	Number of Neurons (weights)
Lr	Initial Learning Rate
$Radius$	Initial Radius
Tlr	Exponential decrease learning rate time constant
Tr	Exponential radius decrease rate time constant
$Epoch$	Number of epochs

4. Experimental Results

The objective of the experiments described is to analyze input/output correlation to verify the vector quantization capability of our proposal. We simulated the method using a set of seven different natural sounds as stimuli and an output layer of fourteen neurons (output data). The neurons outnumber the stimuli so as to force some neurons to never be BMU and simply be pushed around by their neighbors during the self-organizing process. In other words, this procedure tends to maximize the timbral merger and timbre variation cases. Table 2 shows the chosen stimulus-sounds. They were selected to try and maximize dissimilarity in order to explicit individual features of the stimulus-sounds blended in different resultant neuronal-sounds. The weights were initialized with a random Gaussian variable (white noise) and were expected to converge to sounds correlated to the stimulus-sound set. The parameters used are shown in Table 3. The stopping criterion was achieved either by reaching the maximum number of epochs or by a learning rate lower than 0.01.

Table 2 Instruments adopted as stimulus-sounds

Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7
Alto sax	Electric bass	Guitar	Piano chord	Harmonica	Voice	Whistle

We aim at showing that each neuronal-sound is correlated with at least one stimulus-sound, representing a variant. In cases when neuronal-sounds are correlated to more than one stimulus-sound, we wish to show that it represents a timbral merger of the related stimuli by means of blending their features. The classification procedure consists of evaluating the correlation of the neuronal-sounds with the stimuli by means of the estimation of the distance between stimulus-neuron pairs in the timbral soundspace. Short distances are to be interpreted as showing that the particular stimulus-neuron pairs are highly correlated. Caetano et al. (2005 a) define a spectral metric that was applied to each input/output (stimulus/neuron) pair. A similarity table was constructed using this metric (Table 4), associating dissimilarity to spectral distance.

Then, the same procedure was done using a subjective similarity criterion. Five musically untrained subjects were presented to all the stimulus/neuronal-sound pairs and were asked to define a distance value between 1 and 5, 5 being maximum similarity. For each pair, the stimulus-sound was played first, followed by one of the neuronal-sounds. Table 5 shows the average and standard deviation values for this evaluation.

Finally, a two-dimensional SOM was used to generate a topological representation of the input/output resultant soundspace [Damiani et al. 1995; Cosi et al. 1994 a,b; De Poli and Tonella 1993; De Poli and Prandoni 1997; Loureiro et al. 2004; Feiten and Gunzel 1994]. A U-Matrix topological map similar to Figure 1b was generated. All data was presented to the map during learning. Sounds with similar properties were expected to be mapped into the same region, while others should be mapped in a different class valley. This way, neuronal-sounds mapped near stimulus-sounds may represent a similarity relation (Figure 4).

Table 3 Parameters for the experiments

<i>L</i>	<i>SF</i>	<i>G</i>	<i>N</i>	<i>Lr</i>	<i>epoch</i>	<i>Radius</i>	<i>Tlr</i>	<i>Tr</i>
4096	44100	7	14	0.2	451	3	150	150

The convergence dynamics of one specific neuron is shown in Figure 3. Snapshots of one stimulus-sound and the corresponding BMU (neuronal-sound) are plotted at different stages of the self-organizing process. The goal is to highlight the rapid convergence early in the process (exploration of soundspace), followed by a fine tuning stage due to the decreasing of the neighborhood along the learning dynamics (exploitation of promising areas). Notice how the noisy, highly uncorrelated neuron learns to respond to the stimulus, representing its timbral features.

The results of the mathematical and subjective distance evaluations are shown, respectively, in Tables 4 and 5. Values in bold represent a high correlation between the respective input-output pair, i.e. stimulus-neuron. In Table 4, the distances must be interpreted relatively to all the values in the same column, once it is not normalized. Low values mean small distance and thus high correlation. Table 5 shows the average value and standard deviation estimated by the subjects. Here, the subjects estimated the similarity between the input-output pairs. Therefore, high values imply high correlation.

Finally, Figure 4 shows the resultant mapping using a two-dimensional SOM of all the stimulus-sounds and neuronal-sounds together. The gray scale represents topological distance, white being the shortest. The input data (stimulus-sounds) are plotted as black spots and output data (neurons) are plotted as white spots. All data is labeled. Here we expected the same patterns that emerged from Tables 4 and 5 to reveal in this two-dimensional clustering. Highly correlated input/output sounds should be mapped in the same region. Thus, the relations made explicit in Table 6 were also expected to arise in this analysis. The neuronal-sounds that were considered very close to a given stimulus-sound should have been mapped in the same light region.

The results of the experiment discussed here, as well as other significant results, can be found in <http://www.dca.fee.unicamp.br/~caetano/SBCM.html>.

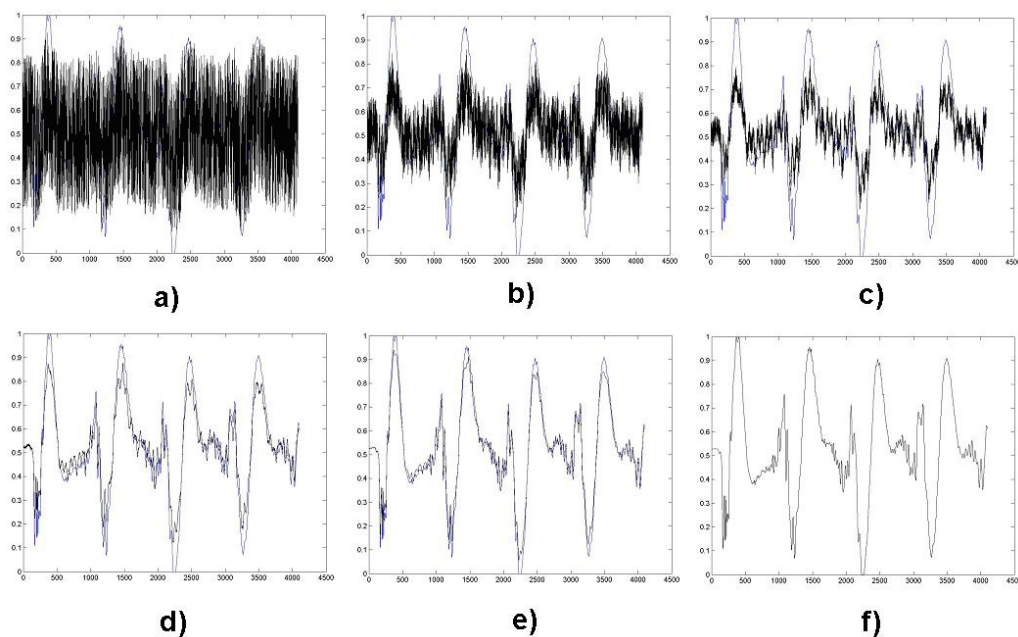


Figure 3 Depiction of the convergence dynamics after different epochs: Stimulus-sound \times Neuronal-sound. Part a) 1 epoch; b) 3 epochs; c) 5 epochs; d) 10 epochs; e) 50 epochs; f) 100 epochs;

5. Discussion

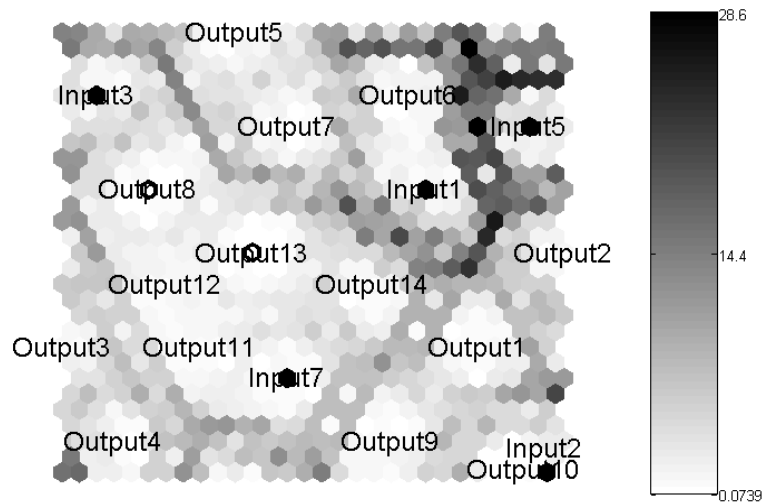
Crossing the result of the subjective and mathematical similarity evaluations (Tables 4 and 5) it is possible to infer a relation between the input and output data. Table 6 shows the outputs (neurons) with maximum similarity to each of the inputs extracted from the values in bold in Tables 4 and 5. Interestingly enough, both the spectral distance and subjective similarity values vary, revealing different degrees of correlation between the pairs. Notice that although the subjective estimation presents maximum similarity evaluations, no spectral distance resulted in zero (refer to [Caetano et al. 2005 a] for the definition of the spectral distance metric).

Table 4 Spectral distance: Input x Output evaluation

	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7
Output 1	34.7606	140.7773	37.6053	117.6477	34.4166	33.3402	44.7520
Output 2	66.3417	144.5577	7.1025	123.9408	53.3262	53.3984	60.8768
Output 3	65.7337	141.9019	8.3761	123.6183	52.4717	52.5984	60.1636
Output 4	127.6824	18.9327	129.2990	165.1742	119.4274	120.5729	123.7715
Output 5	129.9995	16.4692	131.9691	166.9731	121.8598	123.0056	126.1364
Output 6	127.6733	19.0678	129.7479	163.7705	119.3749	120.5557	123.6851
Output 7	107.3686	167.9962	113.6441	15.6206	97.3370	99.4349	99.5542
Output 8	6.7819	143.9433	68.7541	119.4962	38.6160	36.9926	47.6565
Output 9	70.1802	149.4947	80.6534	61.0166	55.0873	58.3408	53.5189
Output 10	50.4559	141.3996	65.7026	113.7396	28.8807	33.9405	3.4725
Output 11	45.2253	139.1221	61.4320	112.2853	15.1187	25.3820	17.0501
Output 12	43.9312	138.1801	60.1144	112.5711	1.4608	22.8388	31.5043
Output 13	41.3641	138.9457	59.6131	114.1093	21.3422	2.5079	35.2750
Output 14	5.1640	144.0565	69.1358	119.7276	39.2329	37.2416	48.0991

Table 5 Result of subjective similarity Input x Output evaluation

	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7
Output 1	3.0±1.58	1.6±0.89	3.8±0.45	2.2±0.84	1.8±0.84	2.4±1.34	1.6±0.89
Output 2	2.0±1.41	1.4±0.89	4.8±0.45	2.0±0.71	1.6±0.89	1.6±0.89	1.4±0.89
Output 3	2.0±1.41	1.6±0.89	4.6±0.55	2.0±0.71	1.2±0.45	1.8±0.84	1.4±0.89
Output 4	1.2±0.45	4.2±1.30	1.8±0.84	1.4±0.55	1.0±0.00	1.0±0.00	1.2±0.44
Output 5	1.2±0.45	4.8±0.45	1.8±0.84	1.2±0.45	11.0±0.00	1.6±0.89	1.2±0.44
Output 6	1.2±0.45	4.8±0.45	1.6±0.89	1.2±0.45	1.0±0.00	1.2±0.45	1.2±0.44
Output 7	1.2±0.45	1.0±0.00	1.4±0.55	5.0±0.00	1.6±0.89	1.0±0.00	1.2±0.44
Output 8	4.8±0.44	1.2±0.45	1.2±0.45	1.0±0.00	1.8±0.84	1.4±0.89	2.0±0.71
Output 9	1.2±0.45	1.0±0.00	1.6±0.89	4.6±0.55	1.4±0.55	1.0±0.00	1.4±0.89
Output 10	2.0±1.00	1.2±0.45	1.0±0.00	1.0±0.00	1.8±0.84	1.2±0.45	5.0±0.00
Output 11	1.4±0.55	1.2±0.45	1.2±0.45	1.0±0.00	3.0±2.00	1.2±0.45	4.0±0.71
Output 12	1.6±0.89	1.0±0.00	1.8±0.84	1.0±0.00	5.0±0.00	1.0±0.00	1.6±0.55
Output 13	1.4±0.89	1.4±0.55	1.4±0.55	1.0±0.00	1.2±0.45	5.0±0.00	1.2±0.45
Output 14	4.4±0.89	1.4±0.89	1.6±0.89	1.2±0.45	1.8±1.30	1.8±1.30	1.6±0.89

**Figure 4 U-Matrix visualization. Light areas represent clusters associated with similar sounds. Dark areas represent cluster borders. The distance scale is shown beside the map.**

One can conclude that, despite the subjective evaluation estimations imply in some cases that some outputs are exactly the same as the inputs, the distance metric reveals that it actually represents a variant. Intermediate distance (similarity) values can also be interpreted as resulting from the blending of timbral features present in one or more stimuli. The spectral distance metric revealed very adequate, matching the subjective estimation at every instance.

Table 6 Input-Output relation inferred from subject data

Input	1	2	3	4	5	6	7
Related Output	8,14	4,5,6	1,2,3	1,7,9	12	13	10,11

On the other hand, a direct comparison of the U-Matrix visualization (see Figure 4) does not have the same effect. The similarity found by the above mentioned analysis, explicated by Table 6, ceases to exist here. Outputs that were mapped together by the U-Matrix are shown in the same valley (white regions). Dark regions represent cluster

borders. The map confirms some relations but it fails to relate a representative number of others. In fact it does suggest new ones. This may be due to two distinct factors. Either different classes were found in Figure 4, implying that the inputs and outputs can be clustered in different ways than the distance evaluations results show, according to different criteria; or it is simply impregnated with topological violations, probably due to the great effort of maintaining the topology of such high-dimensional vector space (\mathbb{R}^{4096} !) represented by the stimuli, projecting it into a two-dimensional space. Differently from the results reported in the literature using two-dimensional SOM to classify sounds according to timbre [De Poli and Tonella 1993], we found the method inappropriate for the purpose of timbral classification in high-dimensional timbral soundspaces.

Here we should stress the important fact that this is hardly the first proposal for a measure of timbral topological relations or classification. Many other techniques are available, including multidimensional scaling [Grey 1975; Grey and Moorer 1977] and subjective analyses [Caetano et al. 2005 b], among others.

The experiments show that SOM is capable of producing sounds that have the desired spectral content with flexibility and robustness. The method makes possible to avoid the burden of trying to describe the desired result in terms of timbral attributes or to exhaustively search the entire soundspace for the desired result interactively, as is the case for Interactive Genetic Algorithms [Biles 1994].

6. Conclusion

A novel method of timbre design was presented, which utilizes SOM, a connectionist clustering technique, in the task of obtaining sounds that are topologically arranged using self-organization. These sounds possess a set of desired timbral characteristics that are inherent to musical sounds and that cannot be precisely described due to the intrinsic multidimensional nature of timbre and the subjective characteristics involved. There is no consensus on how many or what these dimensions are, let alone their subjective relation to the spectral contents of the tone.

The input-output similarity was tentatively measured to base the resultant arranged timbral improvisation cycle respecting topology of the original timbral soundspace. A spectral measure of distance was crossed with a subjective similarity analysis to classify the outputs as being most closely related to one input, representing a variant. Posteriorly, this result was compared with a two-dimensional SOM clustering technique well documented in the literature for timbre classification [Cosi et al. 1994 a,b]. Crossing the results of both evaluations, we found that SOM fails to properly represent the topological relations of the sounds, incurring in topological violations probably due to the high dimensionality of the vector space the sounds were represented in.

The method presented is adjustable according to the input parameters and leads to interesting variations and mixtures of the stimulus-sounds (inputs). The characteristics of maintenance of topology and unsupervised learning provided by SOMs are essential in the results.

Many extensions can be envisaged and tested. It can be used to compose soundscapes, as a timbre design/improvisation tool or in live electroacoustic music where a neurological timbre is generated, which evolves in real time along with other music

materials. Future trends might include using the technique in AI-based musical systems and adapting the method for dynamic environments, i.e. using time-varying stimulus-sounds.

7. Acknowledgments

The authors wish to thank FAPESP (process no. 03/11122-8 and 04/11742-9) and CNPq (process no. 306672/2004-9 and 308765/2003-6) for their financial support.

8. References

- Biles, J. A. (1994) GenJam: A Genetic Algorithm for Generating Jazz Solos, Proceedings of the 1994 International Computer Music Conference, (ICMC'94), pp. 131-137.
- Blackwell, T. and Young, M. (2004) Swarm Granulator. In G. R. Raidl et al. (Eds): EvoWorkshops, Lecture Notes in Computer Science 3005, pp 399-408.
- Burraston, D., Edmonds, E. A., Livingstone, D. and Miranda, E. (2004) Cellular Automata in MIDI based Computer Music. Proceedings of the International Computer Music Conference, pp. 71-78.
- Caetano, M., Manzolli, J. and Von Zuben, F. J. (2005 a) Application of an Artificial Immune System in a Compositional Timbre Design Technique. In C. Jacob et al. (Eds): ICARIS 2005, Lecture Notes in Computer Science 3627, pp 389-403.
- Caetano, M., Manzolli, J. and Von Zuben, F. J. (2005 b) Interactive Control of Evolution Applied to Sound Synthesis. in Markov, Z., Russel, I. (eds.) Proceedings of the 18th International Florida Artificial Intelligence Research Society (FLAIRS), Clearwater Beach, Florida, EUA, pp. 51-56.
- Chen, C. J. and Miikkulainen, R. (2001) Creating Melodies with Evolving Recurrent Neural Networks, Proceedings of the International Joint Conference on Neural Networks (IJCNN-01), 2241-2246.
- Cosi, P., De Poli, G. and Lauzzana, G. (1994a) Auditory modelling and self-organizing neural networks for timbre classification. Journal of New Music Research 23, 71-98.
- Cosi, P., De Poli, G. and Lauzzana, G. (1994b) Timbre classification by NN and auditory modeling. In Marinaro, M. and Morasso, P. G., editors, Proc. ICANN'94, International.
- Damiani, F., Pérez, M. J., Fornari, J. E. (1995) Reconhecimento de timbres musicais através da rede neural auto-organizável de Kohonen . XV Congresso da Sociedade Brasileira de Computação, II Simpósio Brasileiro de Computação e Música. Pag. 107-113.
- De Poli, G. and Prandoni, P. (1997) Sonological models for timbre characterization. Journal of New Music Research 26, 170-197.
- De Poli, G. and Tonella, P. (1993) Self-organizing neural networks and Grey's timbre space. In Proceedings of the 1993 International Computer Music Conference, pp. 441-444.

- Feiten, B. and Gunzel, S. (1994) Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal* 18(3), 53-65.
- Gomes, L. C. T. and Von Zuben, F. J. (2003) Multiple criteria optimization based on unsupervised learning and fuzzy inference applied to the vehicle routing problem. *Journal of Intelligent & Fuzzy Systems*, IOS Press, vol. 13, no. 2-4, pp. 143-154.
- Grey, J. M. (1975) An Exploration of Musical Timbre. Doctoral dissertation, Stanford Univ.
- Grey, J. M. and Moorer, J. A. (1977) Perceptual Evaluations of Synthesized Musical Instrument Tones. *Journ. Ac. Soc. Am.*, 62, 2, pp 454-462.
- Helmholtz, H. (1885) *On the Sensations of Tone*. London, Longman.
- Horowitz, D. (1994) Generating Rhythms with Genetic Algorithms. *Proceedings of the 1994 International Computer Music Conference (ICMC'94)*, pp. 142-143.
- Kandel, E. R. and Schwartz, J. H. (1985) *Principles of Neural Science*, Elsevier.
- Kohonen, T. (1984 a) Phonotopic Maps – Insightful Representation of Phonological Features for Speech Representation. *Proceedings IEEE 7th Inter. Conf. On Pattern Recognition*, Montreal, Canada.
- Kohonen, T. (1984 b) *Self-Organization and Associative Memory*. Springer-Verlag.
- Kohonen, T. (2000) *Self-Organizing Maps*. Springer.
- Lippmann, R. P. (1987) An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*.
- Loureiro, M. A., de Paula, H. B. and Yehia, H. C. (2004). Timbre classification of a single musical instrument. *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, Barcelona, Spain.
- Moller, A. R. (1983) *Auditory Physiology*. Academic Press.
- Risset, J. C. and Wessel, D. L. (1982) Exploration of timbre by analysis and synthesis. In D. Deutsch (ed.) *The Psychology of Music* (pp. 26-58). New York: Academic.
- Santos, A., Arcay, B., Dorado, J., Romero, J. and Rodríguez, J. (2000) Evolutionary Computation Systems for Musical Composition. *International Conference on Acoustic and Music: Theory and Applications (AMTA 2000)*. vol 1. pp 97-102.
- Smalley, D. (1990) Spectro-morphology and Structuring Processes. In *the Language of Electroacoustic Music*, ed. Emmerson, pg. 61-93.
- Ultsch, A. (2003) U*-Matrix: a tool to visualize clusters in high dimensional data. Technical report, Departement of Mathematics and Computer Science, Philipps-University Marburg.
- Zuchini, M.H., (2003) Aplicações de mapas auto-organizáveis em mineração de dados e recuperação de informação. *Dissertação Mestrado*, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas.

AUDIENCE – Audio Immersion Experiences in the CAVERNA Digital

Regis Rossi A. Faria¹, Leandro F. Thomaz¹, Luciano Soares¹, Breno T. Santos¹,
Marcelo K. Zuffo¹, João Antônio Zuffo¹

¹LSI – Laboratório de Sistemas Integráveis – Universidade de São Paulo
Av. Luciano Gualberto, 158 tv.3 – 05508-900 – São Paulo – SP – Brasil

{regis,lfthomaz,lsoares,brsantos,mkzuffo,jazuffo}@lsi.usp.br

Abstract. *In this paper we introduce the AUDIENCE project undergoing in the CAVERNA Digital of the University of São Paulo, whose main purpose is to implement flexible and scalable multichannel spatial audio solutions for this CAVE environment, to permit navigation through a 2D/3D audiovisual scene with both visual and auditory immersion. An architecture for spatial audio production has been proposed to build auralizers, and a whole infrastructure has been designed and installed in the CAVE, so to support several speaker array setups. We present our activities towards the construction of an Ambisonics auralizer, outline some details and challenges of the implementation. We also cover recent achievements of the project and future directions of investigations.*

1. Introduction

Most previous audio systems in immersive virtual reality environments (such as CAVE systems) were more concerned about having some sonification or accompanying sound than with realistic and accurate spatial sound production and auralization. Very few works have addressed spatial audio for CAVE's [Ogi, 2003], [Eckel, 1998]. Frequently these relied upon amplitude panning techniques, and did not go for real sound field rendering, such as with Ambisonics [Gerzon, 1973] or Wave Field Synthesis (WFS) techniques [Berkhout, 1993]. Very often sound was approached as a secondary or complimentary task, addressing stereo audio support only.

As a consequence of the popularization of the multichannel systems due to 5.1 standard issued by ITU-T, there was a reborn of interest in sound field rendering in the last decade, and many groups are working on multichannel approaches addressing large and stable sweet spots for larger audiences and environments, such as cinemas theaters and auditoriums. Take for recent examples the IOSONO [IOSONO, 2005] and CARROUSO [Carrouso, 2005] project approaches, both relying on WFS techniques.

In this paper we introduce the project AUDIENCE – Audio Immersion Experience by Computer Emulation [Faria, 2004], which is undergoing in the CAVERNA Digital of the University of São Paulo, Brazil [Zuffo, 2001]. The project aims the implementation of a flexible and scalable system for 2D/3D audio production and displaying through multichannel techniques.

Objectives include since the implementation of decoding systems for the traditional formats and surround configurations (such as 5.1, 7.1, DTS[®] and Dolby[®]) up to the

development and deployment of more sophisticated formats for 2D/3D audio generation and reproduction, such as Ambisonics and WFS. In the AUDIENCE project we are interested in giving, for more than one user inside the CAVE, the auditory immersion experience similar to the one achieved in the visual domain with stereoscopy techniques. The possible universe of applications is infinite, but we can stress some interesting examples. Imagine for instance the experience of navigating through a symphony orchestra playing on one of your favorite theaters. Or a more modern approach to enjoy alternative bands, primarily setting up a stage, focusing timbres and gestures and locating them in space (associating them to certain positions) and assigning different acoustics properties to instruments or regions in space.

The idea surpasses the situation where users can set up a desired stage and a position in the audience, but goes much further, making it possible to be within the orchestra, e.g. closer to instruments, to experience the conductor position, to create acoustic dimensions and locate timbres in space. One ultimate degree of freedom in this way is to make available such an edition power for the multichannel sound track engineer and also for the final consumer. To investigate such advanced scenarios we are addressing mainly sound field auralization techniques in immersive audiovisual environments.

The following sections of this paper will address activities of the 1st and 2nd phases of the AUDIENCE project, mainly about the sonorization infrastructure and software-based auralizer proposals for sound field generation in CAVE's. These reproduce novel results in the area, since construction of audio infrastructures and multichannel auralization in CAVE's are not well covered in literature. Our approach in audio infrastructure relies upon high-quality commodity equipment found in the marketplace. For the auralization engine we bet on software builds, which are remarkably more flexible and do not show computational disadvantages when cluster parallel computers are available to host processes, as it is the case at the CAVERNA Digital.

2. Sonorization infrastructure for AUDIENCE project

Sound field projection techniques are more difficult to set up in auditory restrictive environments such as CAVE's, and this is one of the main reasons this has not been addressed before in immersive cubes. However, there are several ways to systematic attack some traditional impeachments which render almost impracticable stable sound field projection in these environments. One important thing is to have a proper audio hardware and flexible patch bays.

In the AUDIENCE project we have designed and constructed a complete audio setup, from the soundcard selection up to the speaker array mounting. There was no previous system in marketplace adequate for the purposes of the AUDIENCE project, although Lake Huron (www.lake.com.au) and AuSIM (www.ausim3d.com) have some sonorization systems for CAVE's.

The patch hardware consists of 3 levels: (1) the *rack bay*, where digital audio interface and amplifiers are mounted, (2) the *multicable distribution* section, where balanced electric routes are sent from rack to the CAVE backstage, and (3) the *terminal panels*, from where terminal cables are connected via TRS plugs to the speakers.

We are using in the 1st phase of the project 8 to 16 analog hi-fi LANDO speakers (www.lando.com.br), high quality and low noise Sankya multichannel amplifiers

(www.sankya.com.br), and “Cabos Golden” cables and distribution panels (www.cabosgolden.com.br).

Sonorization of CAVE's are somehow complicated due to extensive forbidden areas where loudspeakers cannot be positioned due to back visual projection. Speakers should preferably be invisible to users, lay behind screens, and irradiate properly towards the centre of the CAVE but covering a large listening volume inside, around the center.

3. Auralization architecture

Spatial audio in interactive electronic media have roughly three mainstreams, one addressing the game industry, one addressing the multichannel surround market, and finally one addressing high precision/realistic rendering of acoustical phenomena. A lack of a reference architecture linking these segments is an important cause of several individual proposals in the field, without cross-references.

Faria (2005) proposed an architecture for referencing spatial audio production, based on four functional layers, from scene composition up to rendering and sonorization processes. Figure 1 shows the layers reference model of the architecture.

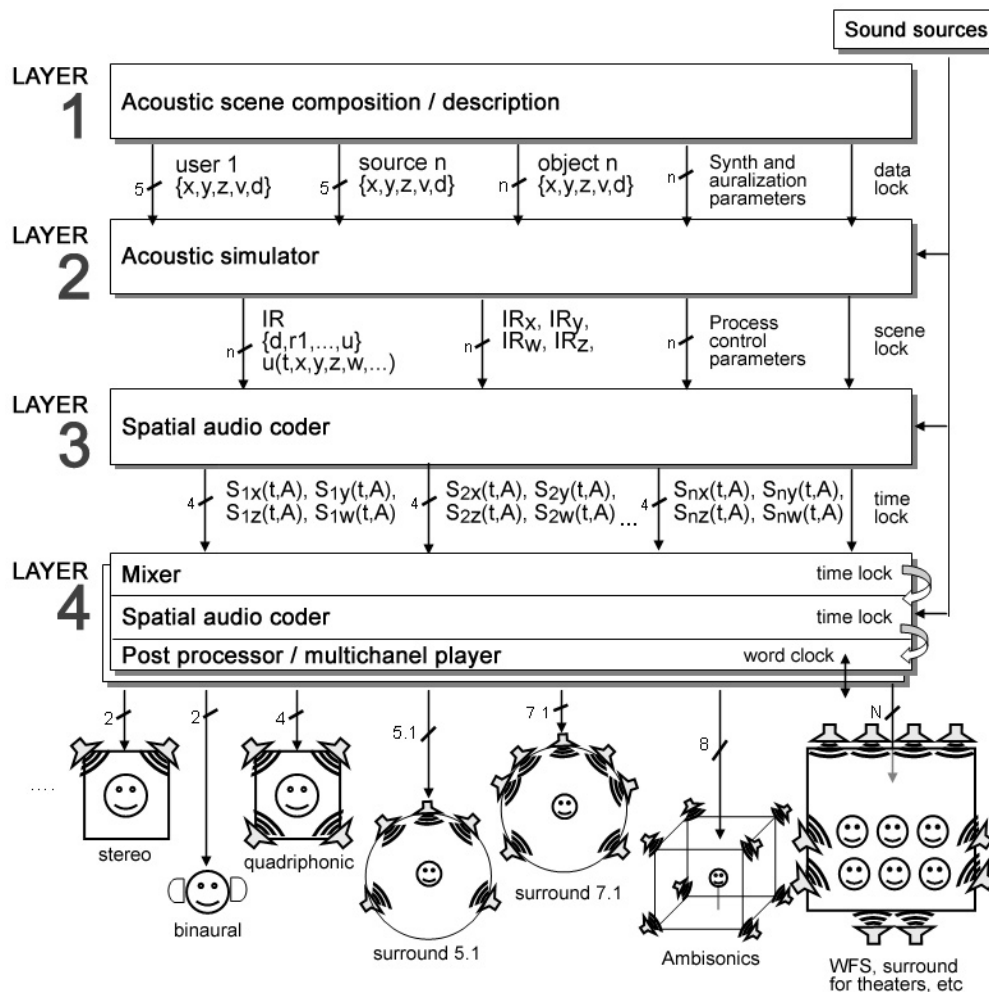


Figure 1 – Layers reference model

Usually tools for programming spatial audio in electronic platforms rely on proprietary or non-complete API's which do not offer flexibility for developers to adopt one or another component from different vendors. For example, they offer tools for selecting positions and attributes of sources, environment and receptors, but not tools for selecting acoustic propagation/simulation techniques and spatial audio coding formats.

This architecture aims a clear identification of signals in between functional layers, so that tools from different developers can still interoperate. As we pursue the highest possible interconnection and interchange of tools avoiding the need for a whole system re-write due to a change in one function, we are building auralization engines following this strategy. Several possible output configurations and speakers arrays are possible for final sonorization, as seen in the figure 1.

4. Building an auralizer

Based on the architecture above, in the first phase of the project we are building an auralizer using image-source techniques for modeling acoustic simulation and the Ambisonics technique for spatial coding the sounds.

Figure 3 shows a functional block diagram of the auralization engine. Three major blocks are detached: (1) the (audiovisual) VR (virtual reality) application, (2) the auralizer (which comprises the acoustic simulator and spatial format coder) and (3) the sonorizator node, where decoding, post-processing and multichannel reproduction take place.

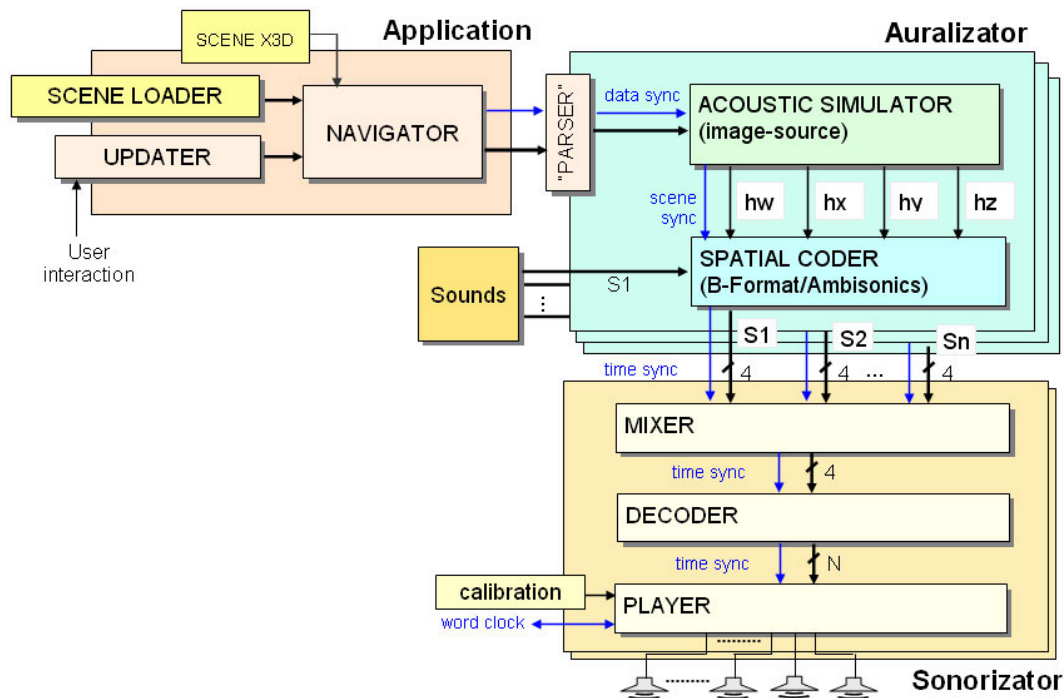


Figure 3 – Auralization engine block diagram

We are using the Pure Data (PD) software as framework to build auralizer blocks and make the necessary interconnections [Puckette, 2005]. PD is a real time graphical

programming environment for audio and music applications, widely used by related communities [Noisternig, 2003], [Fraunberger, 2003].

Having a flexible and real-time capable tool for the audio subsystem and having proper software components for the *glue-logic* with visual and CAVE management subsystems are two major issues. These are being addressed as two concurrent task forces. A remarkable tool contributing to this interconnection is a shared format for audiovisual scene description, and a synchronization hierarchy similar to that used in the visual domain.

Blocks that perform specific tasks are connected to implement a function: patches are created producing chains of several processing algorithms through a chain of connections. When PD is told to compute audio, it starts to pass audio signal chunks through the blocks.

PD is used in the AUDIENCE project as the tool that binds together all the different modules and renders the audio. It also permits these modules to communicate with the VR navigator application and the operating system by built-in network blocks: *netsend* and *netreceive*. These blocks permit also exchange audio information between the navigator and the PD path through TCP sockets.

4.1. An auralizer patch in PD

Basically, four processing layers are shown in the patch presented in Figure 4 below.

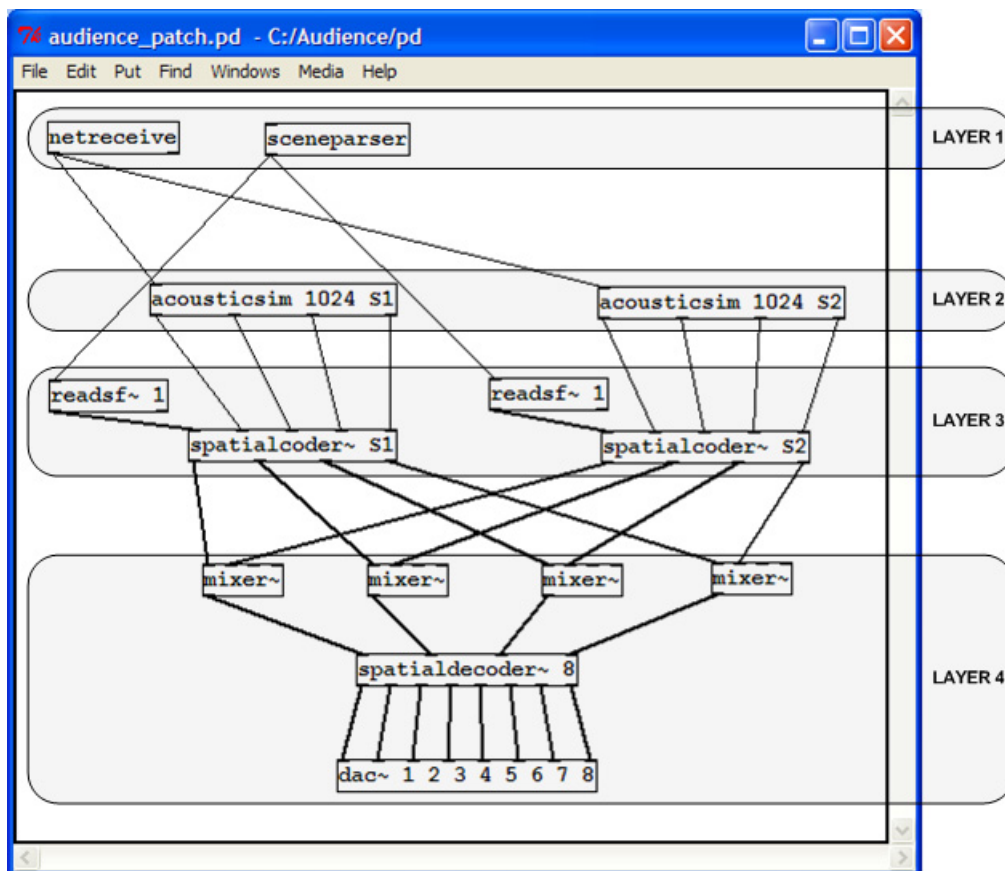


Figure 4 – An example of auralization patch for Ambisonics

The first one, *sceneparser*, receives and parses acoustic attributes from audio nodes within the scene graph, generated by the 3D VR navigator application. Next section will introduce this application, and cover the scene description layer with more detail.

In a second layer, *acousticsim* receives the acoustical attributes, environment dimensions, and listener and source's positions, and render the acoustic simulation using an image-source algorithm modified from Allen's (1979), producing as outputs multidimensional impulse responses (IR_W , IR_X , IR_Y and IR_Z) actually coded in B-format. In other words, we have developed an image-source to B-format acoustic renderer.

Next layer is formed by the *spatialcoder*~ block, which basically convolves the sound sources (S1 and S2 in Figure 4) with a B-format set of impulse responses, producing an Ambisonic-coded sound source.

If more than one sound source is being rendered at a time, their outputs in B-format may be combined to generate a single 4-channel set. This is done in a mixing layer. Final B-format signals are then routed to the *spatialdecoder*~ block, which then produces the speakers' outputs. Item 4.5 ahead describes this layer with more details.

4.2. Acoustic scene parsing

In the CAVERNA Digital we have developed a 3D virtual reality browser called Jinx, which permits to navigate in a virtual scene described in the X3D format [X3D, 2005], using commodity clusters [Soares, 2004]. For the acoustic scene, the Jinx parses acoustic data from the scene graph, and sends it to the PD auralization patch.

Jinx runs in each node of the cluster. Some nodes can take care of graphics, I/O devices etc. At least one node must take care of sound, and host the auralization tree. The digital sound interface must also be installed in this server. The node running the sound server can share resources with other processes (like video) or it can run in a dedicate fashion. Auralization processes share data with visual scene graph, and may also actually run distributed in more than one node. Figure 5 shows how processes communicate inside Jinx, including sound. The current AUDIENCE auralization patch is designed to be called by Jinx.

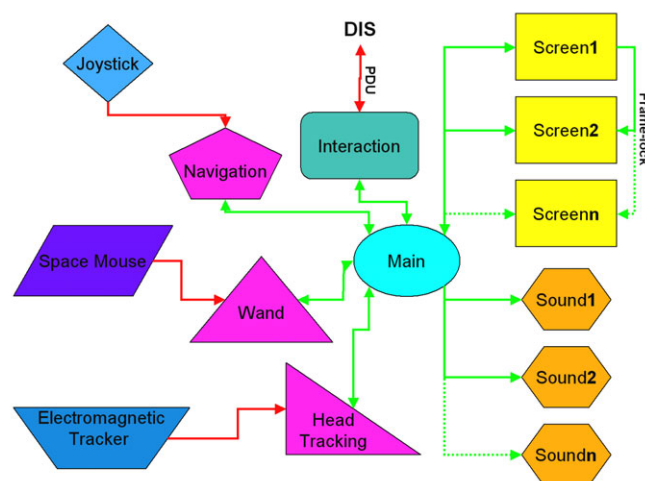


Figure 5 – Jinx Hierarchy

If Jinx is not able to find AUDIENCE auralizator, it is going to use Fmod, an open source sound library [Fmod, 2005]. Fmod supports some features for 3D sound, but with hidden implementations for layer 2 and 3. It may also be used as multichannel player in layer 4. Jinx loads Fmod with the sound files (like wave) with objects' position and orientation.

If AUDIENCE auralizator is available, Jinx parses the file and sends the sound configuration to the *sceneparser* block in PD thought network. *Sceneparser* is an external dynamic library block built in PD, performing layer-1 functions. During normal operation Jinx updates the user position and orientation for each frame in the *sceneparser*. This is locked with graphical output by means of data-lock/scene-lock sync signals, using *netreceive* block (see Figure 4). At the end of navigation, Jinx sends and end tag to finish sound processes.

For the acoustic scene we found the current version of X3D standard not capable of conveying all the information we needed to transmit to subjacent levels of processing, such as material acoustic attributes. To hold sound information for the environment it was then proposed an extension to X3D scene graph. Figure 6 presents an idea of a scene graph tree with a special sound node called *AcousticScene*, with has information as environment dimensions, objects' acoustic attributes, etc.

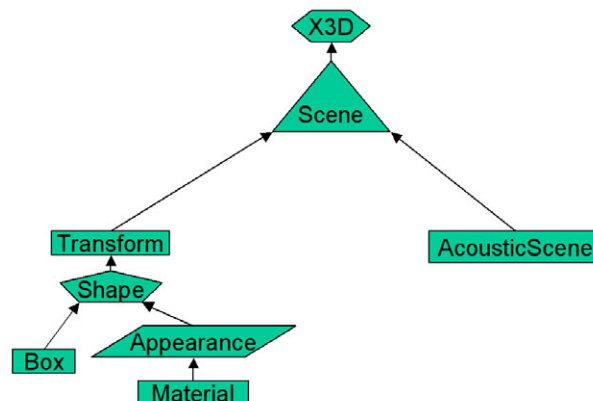


Figure 6 – Scene Graph with Acoustic Information

We also found necessary to add additional children nodes, e.g. the *AcousticMaterial* node. For example take the following code passage:

```

<Transform DEF="floor" center="7.5 0.05 10">
  <Shape>
    <Box size="15 0.1 20"/>
    <Appearance>
      <Material diffuseColor="1 1 1"/>
    </Appearance>
  </Shape>
  <Sound>
    <AcousticMaterial coefref=".8" freqref="1000"/>
  </Sound>
</Transform>

```

In the above passage we define for the floor not only its dimensions but also its acoustic reflection coefficient, which are both needed in the next processing layer. The *AcousticMaterial* node was added to permit our parser block to extract these parameters for the acoustic simulator (the *acousticsim* block).

The MPEG-4 Advanced AudioBIFS (Binary formats for Scenes) is a more comprehensive tool for describing acoustic and sound relevant parameters [Väänänen, 2002], and this is one focus for future works. Spatial coding for several applications do require metadata to be transmitted aside the media itself, which can describe scene, set up decoding and mastering/editing parameters for final channel production in the terminal gear. This is, for example, being explored in the DAB (Digital Audio Broadcast) and SAC (Spatial Audio Coding) initiatives for standardizing multichannel and spatial audio coding/transmission schemes.

4.3. Acoustic simulator

The acoustical simulation is maybe the most important task in the auralization process. Spatial perception and quality are directly associated with this. There are several methodologies for modeling acoustical propagation, often relying on two different approaches for modeling sound: ray-based or wave-based.

For validating the strategy for building and integrating acoustic simulators within our architecture, our first approach was to consider a simple geometry and a precise technique to calculate reflections and obtain artificial impulse responses, such as the image-source method, which is a ray-based technique.

We have developed an acoustic simulator based on Allen's image-source technique for rectangular spaces [Allen, 1979]. A reflection in this technique is supposed to come from a virtual image source, which is located behind the reflective wall, tracked as in optical geometry laws. Figure 7 shows this concept.

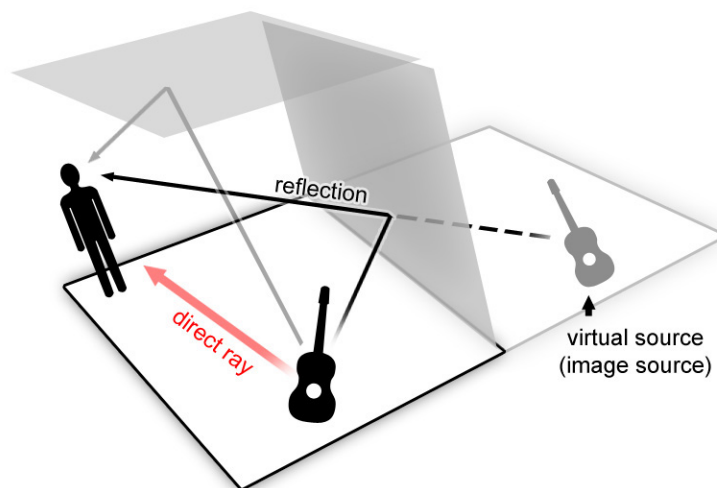


Figure 7 – Image Source ray tracing concept

4.4. Spatial audio format coder

Several formats can be exploited to produce final audio channels comprising both temporal and spatial sound attributes. One of the most elegant, however, is the Ambisonics B-Format. Originally developed in the 1970's by Gerzon and others, its functional mechanism to register a 2D/3D sound field may be explained either via mathematical approach or via an extension of Blumlein's stereo techniques for a 2D and 3D microphone setup. Initially aimed at the record industry, Ambisonics emerged in a

time when quadraphonic systems were in decline. This fact, and also a lack of a proper multichannel media to record and transmit all necessary channels at that time, contributed to its low popularity.

We have designed an Ambisonics 1st order coder, which takes the outputs of the acoustic simulator coded in B-Format (4 channels) and convolve them with the anechoic sound source, so to produce a final B-format spatial encoded audio signal. A B-Format set of signals for 3D audio coding is comprised of four channels, W, X, Y and Z, which register the temporal and spatial properties of sound.

4.5. Spatial audio decoder and multichannel player

The spatial audio decoder developed in the current auralization engine is a basic 1st order Ambisonics decoder. Figure 8 shows its block diagram. In the first stage, four shelf filters, one for each B-format channel (W, X, Y, Z) are applied, with the purpose of adjusting the psychoacoustic quality of the sound for enhancing auditory localization cues [Gerzon, 1974]. Next, there is an amplitude matrix stage. Depending on the number of speakers and their positions, it applies different gains to each one of the channels and produces as output a number of channels corresponding to the number of speakers.

A very important component of the decoder is an inverse filter to minimize the effects of the CAVE's projection screens. They are reflective, and have a diffuser effect on transmitting sound from the speakers, located behind them, to the center of the CAVE. This filter is usually designed using the impulse response of the speaker-screen system, measured with a flat microphone inside the CAVE. This convolution is also made within the implemented decoder block.

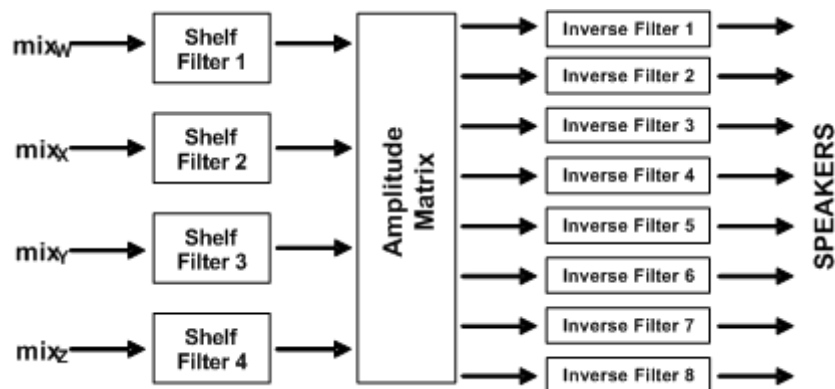


Figure 8 – Block diagram of *spatialdecoder~*

The Ambisonics decoder was implemented as a PD block, named *spatialdecoder~*. The gains were obtained from previous calculations made by Richard Furse [Furse, 2005]. It takes as input a 4-channel mixdown (from a previous mixing stage, where all B-format encoded sound sources available were mixed). It also receives parameters, as the intended type of speaker configuration (cube, octagon etc).

5. Conclusions

The AUDIENCE project 1st phase objectives were achieved with a very flexible and high-quality audio infrastructure setup. In the 2nd phase, very good initial results in perceived spatial quality showed that our current approach to build auralization engines shall lead to progressive refinements. The open/layered architecture permits us to invest independently in any phase of the spatial sound production.

In the scene layer, for instance, the standardization of more powerful nodes for X3D seems a prospective and important task, for a better sound description in virtual environments. The integration of multiple acoustic simulation methods, such as geometrical-based and wave-based techniques, is expected to resolve acoustical phenomena in a large bandwidth and in a complementary way: one technique covering the weakness of the other.

Several spatial audio formats may also be addressed, such as WFS/MPEG-4 AAC (Advanced Audio Coding) and commercial formats, as 5.1/6.1/7.1/10.2 etc. Finally, many decoders, including commercial ones, may be employed in different situations, depending on operating system, application requirements, etc.

We have been testing two speaker configurations: a horizontal (2D) octagonal array, and a cubic (3D) array, both with eight speakers surrounding the CAVE. For the octagonal rig, we have tested it outside the CAVE and surrounding the CAVE, in an experiment that showed the challenges in treating the acoustic paths and interferences due to screen and backstage superimposed acoustics.

A simple experiment was performed where a single source and the listener were positioned in a virtual room with 20x15x8m dimensions. Two timbers were tested, a real flute and a synthetic battery set, both playing a small passage with several important musical gestures for spatial quality evaluation. Figure 9 shows a picture of a testing octagon rig mounted outside the CAVERNA and later mounted surrounding it.

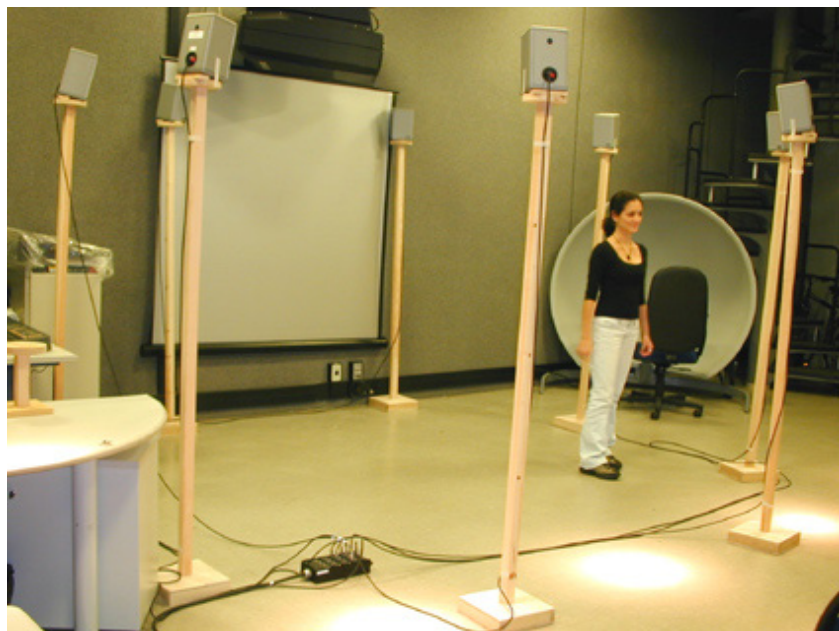


Figure 9 – Octagonal (2D) horizontal speaker array for Ambisonics auralization

Excellent localization was achieved with the auralizer rig outside the CAVE. Inside the CAVE, without any special active or passive treatment, the perception of position was less precise, although the direction cue was always preserved. The cubic array was only tested in the CAVE. Direction was also preserved in this rig, although the elevation difference between source and listener was not easy to perceive. Better speaker positioning however will be soon possible, due to a new elevated support to hold speakers in the upper part of the cube.

Faria discusses some results of these experiments in [Faria, 2005] and also lists future directions and works. Several other experiments are being prepared, for example an audiovisual scene with 4 instruments playing on stage (a flute, a violin, a trumpet and a cello) where we intend to explore complete real-time audiovisual virtual navigation in the environment.

A future phase of the PD Ambisonics decoder is to implement an auto-configuring decoding matrix in a way that, given an arbitrary array of speakers and their positions, the software configures automatically the gains. To accomplish this feature, a numerical solution of the cylindrical Bessel wave equations/functions will be required, which is the basic mathematical concept behind the Ambisonics technique in reconstructing plane waves.

For the WFS phase of the project, the higher number of speakers and other requirements of the technology pose a lot of additional challenges beyond the Ambisonics system. Different speaker arrays and audio distribution will be proposed for this phase, to be addressed in future papers.

References

- Ogi, T. et al. "Immersive sound field simulation in multi-screen projection displays". In: International Immersive Projection Technologies Workshop, 7. Eurographics Workshop On Virtual Environments, 9. Zurich, 2003. Proceedings. Zurich: Eurographics, 2003. p.135-142.
- Eckel, G. A spatial auditory display for the CyberStage. In: ICAD'98, 5th International Conference on Auditory Display, Glasgow, 1998. Proceedings. Glasgow: ICAD, 1998.
- Gerzon, M. "Periphony: With-height sound reproduction", JAES Jan./Feb. 1973, Vol.21, No.1
- Berkhout, A. J. et al. A wave field extrapolation approach to acoustical modeling in enclosed spaces. Journal of the Acoustical Society of America, v.93, n.5, p.2764-2778, May 1993.
- IOSONO Wave Field Synthesis System. In: <http://www.iosono-sound.com/> Access in: 20 May 2005.
- CARROUSO Project: Creating, assessing and rendering in real time of high quality audio-visual environments in MPEG-4 context: system specification and functional architecture. In: <http://www.idmt.de/projects/carrouso/index.html>. Access in: 20 May 2005.

- Faria, R. R. A. "AUDIENCE – *Audio Immersion Experience by Computer Emulation*". In: <http://www.lsi.usp.br/interativos/nem/audience/>. 2004. Accessed in: 20 May 2005.
- Zuffo, J. A et al. "CAVERNA Digital – Sistema de Multiprojeção Estereoscópico Baseado em Aglomerados de PCs para Aplicações Imersivas em Realidade Virtual. In: 4th Symposium of Virtual Reality, Florianópolis, 2001. Proceedings.
- Faria, R. R. A. "Auralização em ambientes audiovisuais imersivos". Thesis (Ph.D.). Electronic Engineering. Polytechnic School, University of Sao Paulo. 2005.
- Puckette, M. "Pd Documentation". In: http://crca.ucsd.edu/~msp/Pd_documentation/. Accessed in: 20 May 2005.
- Noisternig, M., Sontacchi, A., Musil, T. and Höldrich, R. "A 3D Ambisonic Based Binaural Sound Reproduction System". In: Audio AES 24th Conference, Banff, 2003.
- Frauenberger C., Ritsch, W., Höldrich, R., "Internet Archive For Electronic Music IAEM-IARS (Internet Audio Rendering System)" In: Audio AES 24th Conference, Banff, 2003.
- Allen, J. B.; Berkley, D. A. "Image method for efficiently simulating small-room acoustics". Journal of the Acoustical Society of America, v.65, n.4, Apr. 1979, p.943-950.
- "X3D". In: <http://www.web3d.org/>. Accessed in: 20 May 2005.
- Soares, L. P. and Zuffo, M. K. "JINX: an X3D browser for VR immersive simulation based on clusters of commodity computers". In Proceedings of the ninth international conference on 3D Web technology, Monterey, California, USA, p.79-86. 2004.
- "Fmod music & sound effects system". In: <http://www.fmod.org>. Accessed in: 28 Aug. 2005.
- Väänänen, R., Huopaniemi, J. "SNHC audio and audio composition". In: Pereira, F.C.N, Ebrahimi, T. "The MPEG-4 Book". New Jersey, IMSC Press/Prentice Hall, 2002.
- Gerzon, M. "Surround-sound psychoacoustics". In: Wireless World, Dec. 1974.
- Furse, R. "First and Second Order Ambisonic Decoding Equations". In: <http://www.muse.demon.co.uk/ref/speakers.html>. Accessed in: 20 May 2005.

A User-Friendly Graphical System for Room Acoustics Measurement and Analysis*

Leo Kazuhiro Ueda¹, Fábio Leão Figueiredo², Fernando Iazzetta², Fabio Kon¹

¹Department of Computer Science – University of São Paulo

²Department of Music – University of São Paulo

lku@ime.usp.br, fabioflf@hotmail.com, iazzetta@usp.br, kon@ime.usp.br
<http://gsd.ime.usp.br/acmus>

Abstract. *AcMus is an integrated software platform for musical room acoustics. A preliminary version of the software was developed as a MATLAB prototype of acoustical analysis functions. The prototype processes measurements recorded from a room and outputs calculations that help us assessing the quality of the room for musical performance. We have measured a number of musical rooms ourselves and are currently analyzing the data. Meanwhile, we are working on the final implementation of the AcMus integrated platform, which is based on a flexible and extensible Java plug-in framework. This new, open source version of the system incorporates our experiences with the preliminary prototype and tries to maximize the usability and effectiveness of the user interface.*

1. Introduction

In 2002, we started the AcMus project [Iazzetta et al., 2004, Yili et al., 2003], an effort to consolidate a research community in São Paulo, Brazil, devoted to issues related to musical room acoustics. One of the main goals is to build an open-source extensible software for estimation, measurement, analysis, and simulation of rooms especially designed for musical performance. The software is divided in three modules.

1. **Measurement Module:** helps the user to perform and analyze acoustical measurements.
2. **Utilities Module:** offers tools that can be useful for the design of rooms, acoustical measurements, and audio processing.
3. **Simulation and Optimization Module:** helps the design of rooms by performing computer simulations and applying optimization techniques.

This paper focuses on the results obtained so far concerning the Measurement Module. We first built a MATLAB prototype [Masiero, 2004], which we describe in Section 2.1, that allowed us to analyze measurements taken from various concert halls located in the city of São Paulo. This data has been used to carry out an investigation on how to correlate objective and subjective acoustic parameters [Figueiredo et al., 2004], we talk about this in Section 2.

The prototype consists of a number of MATLAB DSP functions with no special user interface other than the MATLAB's own graphical development environment. During the work on the field, we also used some general-purpose audio commercial tools. With this experience, we confirmed the necessity of an efficient, integrated, and easy-to-use computer system that would help us take and analyze the measurements. So, in

*Supported by FAPESP, Brazil, proc. numbers 02/02678-0, 04/05396-0, and 02/10660-3.

parallel to this work, we are applying what we learned from this practical experience to build a graphical user interface for the Measurement Module in Java. We have described the ongoing work of this interface in a previous paper [Ueda et al., 2005], so in Section 4 we show how it will be useful.

2. Measurement Module Prototype

With our MATLAB prototype, we are able to process the data from the rooms we measured. This analysis gives us acoustical parameters that tells us something about the quality of the room. Although many acoustical parameters may be determined by specific measurements and calculations, our interest lays in those that shows correlation with sensitivity and perception of subjects in a certain environment, particularly in rooms designed for musical execution and listening.

Such parameters play a crucial role in the artistic quality of a musical event. Each of the subjective parameters [Beranek, 2004] (Liveness, Clarity, Definition, Spatial Impression, among others) is related to specific physical phenomena that are responsible for the acoustical impressions that define the characteristics of a music room. The most important parameters are represented by mathematical expressions that generate objective values, so we may see them as measurable physical quantities [Beranek, 2004]. For example, RT60 for reverberation, C80 for clarity, and BR for bass ratio.

The measurement process consists of playing specific generated signals inside of the room and then record the room response to this signal in different positions. The recorded signal is mathematically compared with the original generated one, from which we obtain the room impulse response (IR) [Kuttruff, 1991]. Figure 1 shows two impulse responses obtained from the main floor of the *Teatro Municipal de São Paulo*.

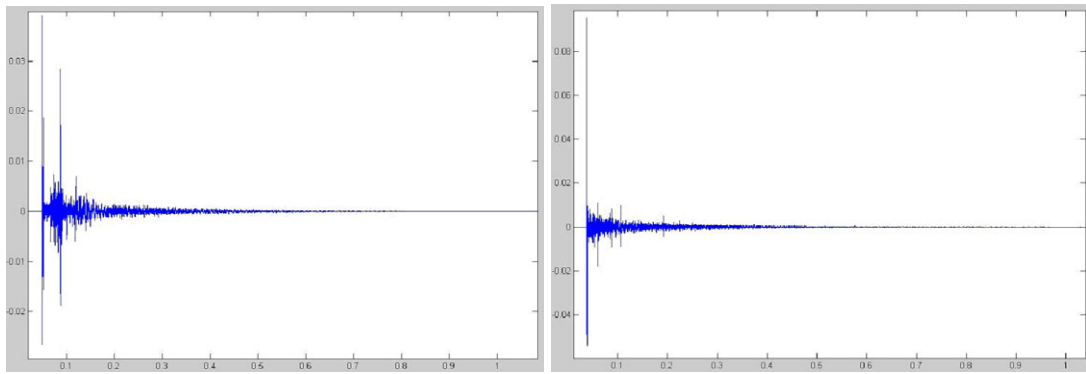


Figure 1: Impulse responses from the *Teatro Municipal de São Paulo*

The most efficient method to calculate the impulse response is the one that uses a sine sweep. The sine sweep is a sinusoid in which its instant frequency varies in time. This variation may be linear or logarithmic. We chose to use the logarithmic sweep, which has a pink spectrum, that is, its amplitude decays 3dB per octave. This means that each octave of the signal contains the same amount of energy. The frequency doubles at a fixed rate. The impulse response is obtained by deconvolving [Müller and Massarani, 2001] the generated signal with the recorded one.

From the impulse response, the functions we implemented give us the desired parameters. We calculate the energy decay curve from the impulse response by using the Schroeder Integration method [ISO 3382, 1997]:

$$E(t) = \int_t^\infty p^2(\tau) d\tau = \int_0^\infty p^2(\tau) d\tau - \int_0^t p^2(\tau) d\tau \quad (1)$$

where p is the impulse response. By manipulating this integration, we can calculate the main acoustical parameters. Figure 2 shows a graphical output example of this manipulation. The measurement was taken from the *Teatro São Pedro*.

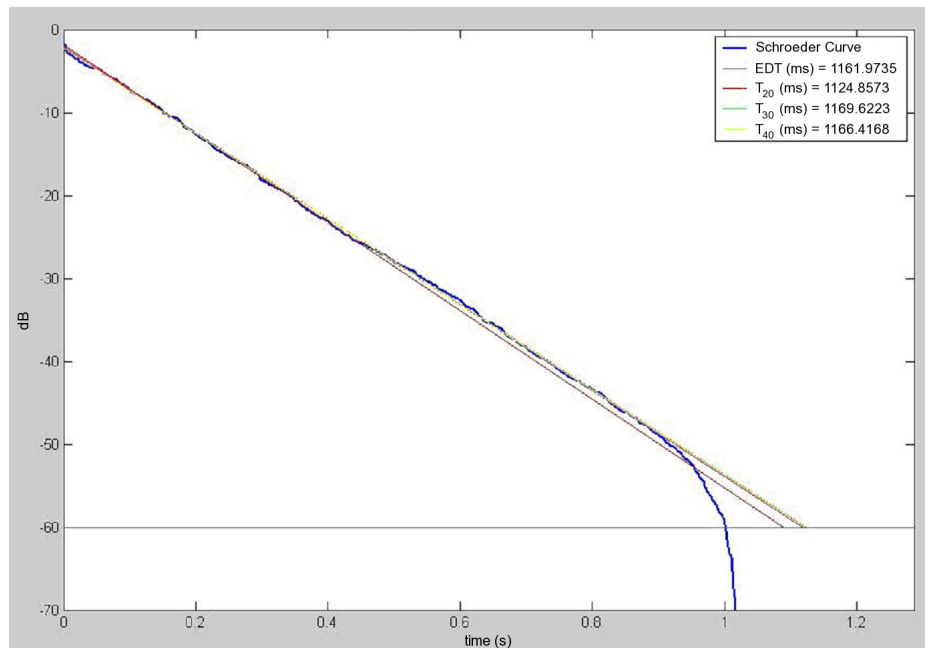


Figure 2: Decay curve (*Teatro São Pedro*)

We measured and compared some concert rooms in São Paulo where stable symphonic groups perform regularly. Figure 3 shows some of the parameters obtained from the *Anfiteatro Camargo Guarnieri* (CG), *Teatro de Diadema* (TD), *Teatro do Memorial da América Latina* (ME), *Teatro Municipal* (TM), *Teatro São Pedro* (SP), and *Teatro Sérgio Cardoso* (SC).

The theaters showed quite distinct acoustical behaviors due to different architectural characteristics and the acoustical treatment. Although we are still processing the measurement data, preliminary analysis leads to important conclusions about the acoustical performance of the rooms.

For instance, we observed that the *Teatro do Memorial da América Latina* has too much absorption material on the lateral surfaces. This resulted in lower values of the Brilliance and Lateral Fraction parameters. We detected acoustical distortions (echo and excessive bass) at the back of the main floor. The analysis of the impulse response suggests that this distortions are due to the peculiar shape of the ceiling: a parabolic curve that reflects and confines the sound waves in the distortion areas.

The *Teatro Sérgio Cardoso* has a very large scenic space, resulting in relatively long reverberation times. This is certainly a negative aspect for the orchestra performance.

The parameters for the *Teatro Municipal* indicate that it is suitable for operistical presentations. Its various possible positions for the audience (main floor, balcony, gallery) provides diversified listening conditions. The measurements show the stage is excessively dry, confirming the opinion of the musicians we interviewed. The superior floors present higher Clarity levels than the main floor. However, in the gallery we observe excessive diffusion and low Clarity, this may be a problem for certain repertoires.

Considering now the Brilliance (TR) and Bass Ratio (BR) parameters, the *Anfiteatro Camargo Guarnieri* and the *Teatro São Pedro* are more balanced than the others. The *Teatro de Diadema* and the *Teatro Sérgio Cardoso*, on the other hand, presented ex-

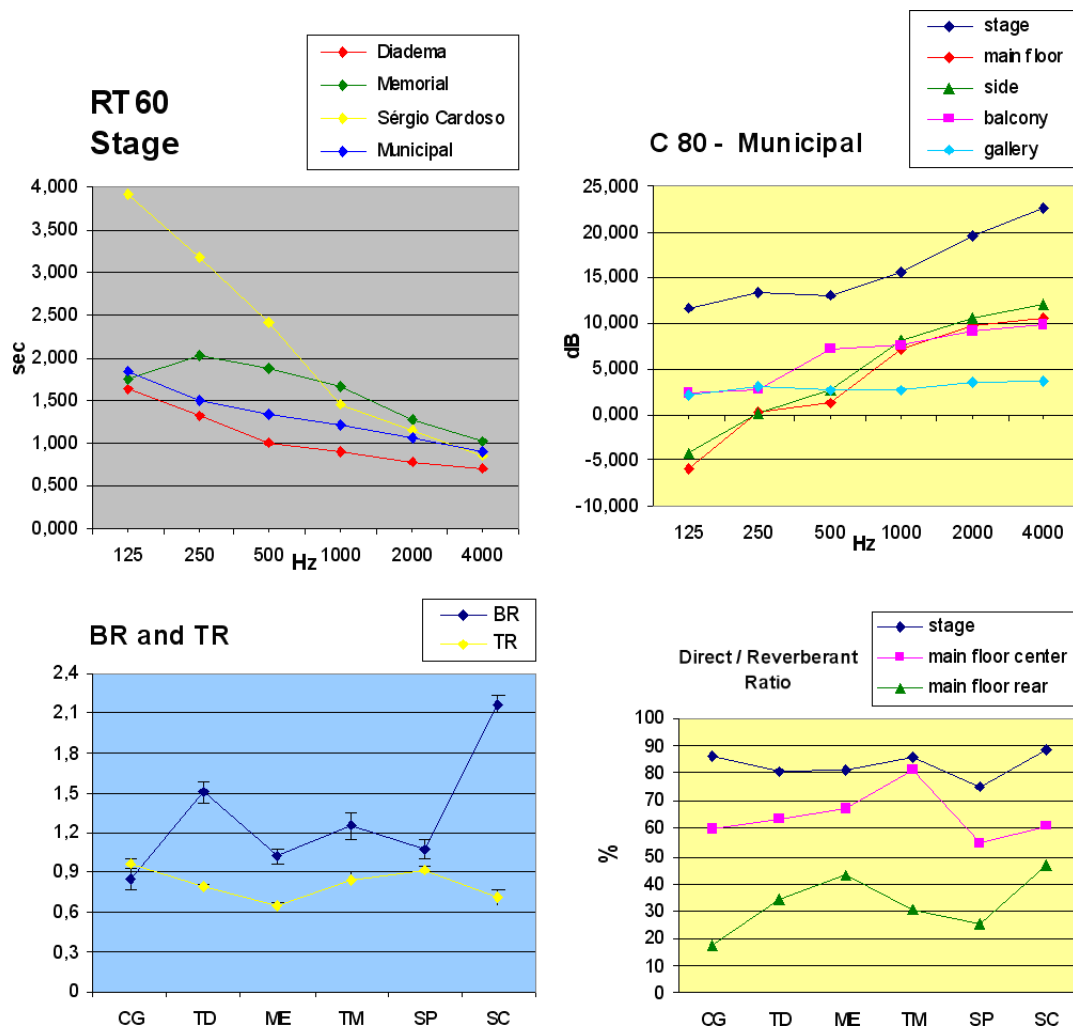


Figure 3: Some acoustical parameters of rooms in São Paulo

cessive Bass Ratio. A possible explanation for this behavior in the *Teatro de Diadema* is that it has wide openings at the stage sides, making an additional space in the area where the sound is produced. The *Teatro Sérgio Cardoso* has a huge reverberant chamber behind the orchestra position and the panels placed on the stage are not efficient enough to solve this problem.

When we finish processing all the data, we will be able to diagnose precisely the acoustical behavior of the rooms and to suggest possible solutions to their acoustical problems.

2.1. MATLAB Implementation

MATLAB (<http://www.mathworks.com/products/matlab>) is a commercial package that includes a high-level interpreted programming language and a development environment. Unlike the operators in the more common programming languages, which usually only deal with scalars, MATLAB built-in functions and operators let the programmer work with matrices and complex numbers in a quick and easy way. Along with its development environment and its comprehensive DSP and math libraries, this makes it a good prototyping language, suitable for our needs to test and validate the audio processing algorithms we studied.

The prototype we implemented is a set of MATLAB functions that provides most of the analysis functionality of the Measurement Module for two methods: Maximum-Length Sequence (MLS) and Log Sweep FFT (LSF). Let us take a look at them.

Order	Length	Class	Tap	Order	Length	Class	Tap
2	3	a	2, 1	15	32767	c	15, 11
3	7	a	3, 1			d	15, 8
		b	3, 2	16	65535	a	16, 5, 3, 2
4	15	a	4, 1			b	16, 15, 13, 4
		b	4, 3	17	131071	a	17, 3
5	31	a	5, 2			b	17, 14
		b	5, 3			c	17, 14, 13, 9
6	63	a	6, 1	18	262143	a	18, 7
		b	6, 5			b	18, 11
7	127	a	7, 1	19	524287	a	19, 6, 5, 1
		b	7, 6			b	19, 18, 17, 14
8	255	a	8, 6, 5, 1	20	1048575	a	20, 3
		b	8, 5, 3, 2			b	20, 17
9	511	a	9, 4	21	2097151	a	21, 2
		b	9, 5			b	21, 19
10	1023	a	10, 3	22	4194303	a	22, 1
		b	10, 7			b	22, 21
11	2047	a	11, 2	23	8388607	a	23, 5
		b	11, 9			b	23, 18
12	4095	a	12, 7, 4, 3	24	16777215	a	24, 4, 3, 1
		b	12, 11, 8, 6			b	24, 23, 22, 17
		c	12, 11, 10, 2	25	33554431	a	25, 3
13	8191	a	13, 4, 3, 1	26	67108863	a	26, 8, 7, 1
		b	13, 12, 10, 9	27	134217727	a	27, 8, 7, 1
14	16383	a	14, 12, 11, 1	28	268435455	a	28, 3
		b	14, 13, 8, 4	29	536870911	a	29, 2
		c	14, 13, 12, 2	30	1073741823	a	30, 16, 15, 1
15	32767	a	15, 1	31	2147483647	a	31, 3
		b	15, 14	32	4294967295	a	32, 28, 27, 1

Table 1: Feedback taps for the MLSs of order 2 to 32

Signal generation

In order to perform a measurement, we first need to generate the input signal to be played inside the room. The user should use the right function for the chosen method.

<pre>[mls,row,col] = mls2tap(N,tap0,tap1)</pre>	
<pre>[mls,row,col] = mls4tap(N,tap0,tap1,tap2,tap3)</pre>	
Generates the N-order MLS with two or four feedback taps.	
Receives: N MLS order. tap* feedback taps, according to Table 1.	Returns: mls the MLS. row, col permutation arrays to be used in the IR calculations.

<pre>[c,B,A] = varredura(fs,t,f0,f1)</pre>	
Generates the logarithmic sine sweep with sample frequency fs, duration t, start frequency f0, and end frequency f1.	
Receives: fs sample frequency. t duration. f0 start frequency. f1 end frequency.	Returns: c the logarithmic sine sweep. B, A butterworth bandpass filter coefficients to be used in the IR calculations.

From the vectors `mls` or `c`, we may use the function `wavwrite` to output a Microsoft WAVE sound file so that they can be played by an external program. The MLS signal should be generated with at least two repetitions of `mls`.

Impulse Response Calculation

The generated signal should be played inside the room and the response recorded simultaneously. Figure 4 shows the equipment setup we used. It produces a stereo recording

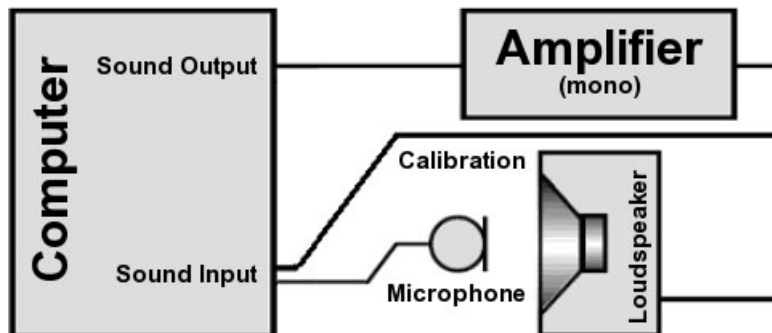


Figure 4: Diagram of the equipment setup

where the first channel is the room response and the second is the calibration. This configuration is important for the processing functions below.

Before we proceed, we must read the recorded samples to a MATLAB matrix. This may be done by the `wavread` function, which reads a Microsoft WAVE file. By now we should have a 2-column matrix where the first column contains the samples of the room response channel and the second column contains the samples of the calibration channel. We can now calculate the impulse response.

<code>ir = demls(signal,row,col, reps)</code>	
Calculates the average of the <code>reps</code> repetitions of the generated MLS in <code>signal</code> , excluding the first one.	
Receives: <code>signal</code> the recorded response. <code>row,col</code> permutation arrays returned by <code>mls2tap</code> or <code>mls4tap</code> . <code>reps</code> number of repetitions of the generated MLS played.	Returns: <code>ir</code> the impulse response.

<code>ir = dechirp(signal,B,A,N)</code>	
Deconvolves the sine sweep signal with the room response to it using the FFT.	
Receives: <code>signal</code> the recorded response. <code>B,A</code> filter coefficients returned by <code>varredura</code> . <code>N</code> desired length for the resulting IR.	Returns: <code>ir</code> the impulse response.

Parameters Calculation

The last step is the calculation of the acoustical parameters. Before being processed, the impulse response needs a treatment to minimize the signal disturbance. We implemented three different methods in the following functions.

<code>[s] = ldbparam(ir,fs,flag)</code>	
Implements the Lundeby method.	
Receives: <code>ir</code> the impulse response. <code>fs</code> sample rate. <code>flag</code> if equals 1, displays the Schroeder curves.	Returns: <code>[s]</code> the acoustical parameters.

<code>[s] = chuparam(ir,fs,flag)</code>	
Implements the Chu method.	
Receives: <code>ir</code> the impulse response. <code>fs</code> sample rate. <code>flag</code> if equals 1, displays the Schroeder curves.	Returns: <code>[s]</code> the acoustical parameters.

<code>[s] = hrtparam(ir1,ir2,fs,flag)</code>	
Implements the Hirata method.	
Receives: <code>ir1,ir2</code> two impulse responses obtained under the same conditions. <code>fs</code> sample rate. <code>flag</code> if equals 1, displays the Schroeder curves.	Returns: <code>[s]</code> the acoustical parameters.

3. Usability Requirements

The task of taking measurements from a room demands the maximum of agility and organization. Considering the high number of signal takes and measurement positions, we want the software to support the work on the field. During the measurements, we dealt with the manipulation, conversion, and storage of the data without the use of any special software, we only used our MATLAB prototype and a few audio tools. That experience showed us the importance of a integrated system designed for these tasks that offers an agile and flexible interface. We then enumerated some usability requirements for the software we are building.

1. **Organization of the data:** when we measure a room, we must perform the signal takes in a lot of different positions. On top of that, each take must be repeated several times. This generates a great number of recordings, each one with its set of calculated parameters associated with it, so it is important to label and store them properly.
2. **Display of results:** we also need an efficient way to display this great amount of data. This should be done in the form of graphs and tables generated by the software.
3. **Audio tools:** tools for audio visualization may be useful when dealing with the recordings.
4. **Automated repetitions:** given that we need to repeat the same take a few times for each position, the software should be able to do it automatically.
5. **Integrated environment:** all these tools and features should be integrated in a single environment for convenience.

4. AcMus Measurement Module

We are working on the final Java implementation of the AcMus Measurement Module taking into account the experience we gained from the measurements. We plan to implement the three modules of the AcMus software in the same environment, satisfying Requirement 5. We are doing this with the help of the Eclipse Platform.

4.1. Eclipse

The AcMus software is being implemented as a set of Eclipse plug-ins. Eclipse (<http://www.eclipse.org>) is a powerful, generic, extensible, open set of computer tools for developing programs. It is actually a general-purpose Integrated Development Environment (IDE) that can be largely extended with contributed plug-ins written in Java. It provides a foundation for constructing and running these plug-ins, allowing extensions built by different people to integrate seamlessly and become part of the Eclipse Platform. It also counts with a great number of high quality open-source plug-ins offered by the Eclipse community.

The Eclipse Platform is then used by both the plug-in programmers (who have the role of the platform developers) and the application programmers (who have the role of the platform users). Its architecture is based on the concept of *plug-in*. Plug-ins are pieces of programs with a defined structure that add functionality to the platform. A plug-in may define *extension points* to let other plug-ins extend it. Almost the entire platform is built upon this model, even its main subsystems are structured as sets of plug-ins, resulting in a highly extensible system.

When started, the Eclipse Platform opens the *workbench*, shown in Figure 5. Besides the menus and the toolbar, the workbench also contains panels, known as *views*,

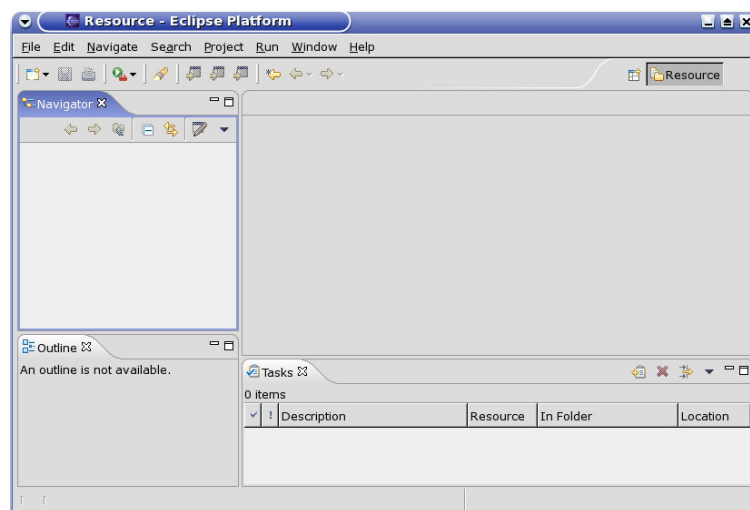


Figure 5: The Eclipse Workbench with the default perspective.

and an *editor* area. These elements compose a graphical interface for the development and management of *projects*. A project is mapped to a folder in a file system. Files inside this folder represent the project resources. A *perspective* defines a collection of view, menus, and buttons in the toolbar. In a way, the perspective determines the kind of tasks for which the workbench will be useful. The default perspective offers generic views for project management and basic file editing. For example, the Navigator View lets the user create, select, and remove projects. To its right is the editor area. When a document is selected in the Navigator View, the appropriate editor windows opens there. The Java perspective, accordingly, offers the functionality of a full-featured Java IDE.

Each of these GUI elements is a plug-in or a set of plug-ins that can itself be extended. The AcMus platform is being built upon this model. The Measurement Module described in this paper implements a perspective that offers customized views and editors.

4.2. AcMus Perspective

The *AcMus Perspective* is shown in Figure 6. The view on the left hand side of the window

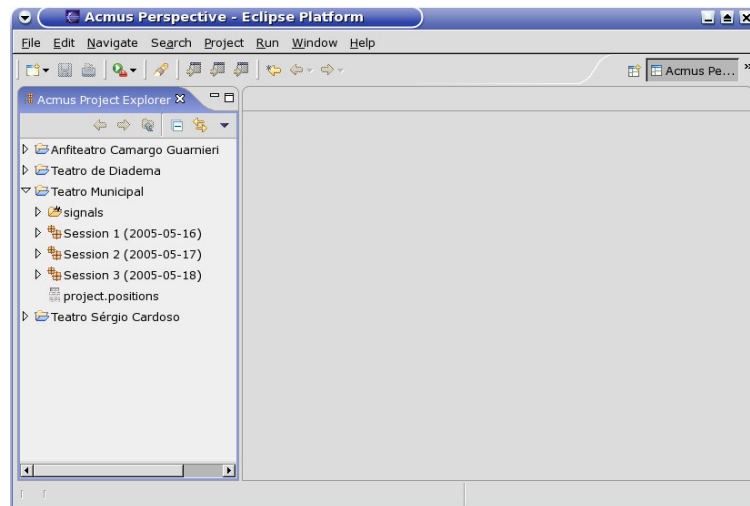


Figure 6: The AcMus Perspective.

is the *AcMus Project Explorer*, which extends the Eclipse Navigator View. It has the same purpose of managing projects. In order to create new projects and resources, we provide wizards that were built using the wizard framework offered by the Eclipse Platform. The plug-in also defines different types of resources, some of them having their own editor. For example, the resources that represent audio files may be viewed with the *Audio Player* (which is actually an editor), as shown in Figure 7. This player draws the waveform and

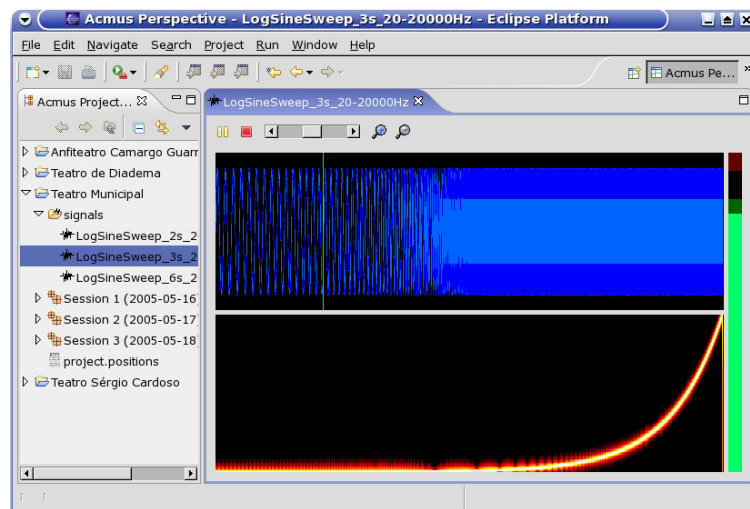


Figure 7: The Audio Player.

the spectrum of an audio file, conforming with Requirement 3.

In the next sections we show more of these elements and how they work together.

Measurement Management

The Project Explorer view lets the user store measurements taken from different rooms in hierarchical folders. There are four main kinds of folders.

- **Project folder:** represents a room.
- **Session folder:** groups measurements taken at a specific period in time.
- **Set folder:** stores the repetitions of the same measurement.
- **Measurement folder:** stores the audio file of the room's response and the output of the response analysis.

The wizards for the creation of these folders let the user input additional information about the folder such as date, time, equipment, comments, and so on. This structure, designed to comply with Requirement 1, induces a certain organization. Figure 8 shows an actual example.

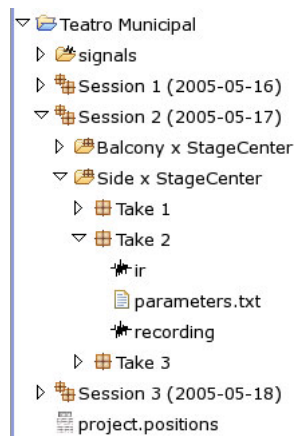



Figure 8: An example of the organization of the folders.

The special folder  **signals** stores the generated signals to be played in the room. The current implementation provides a wizard for the creation of logarithmic sine sweeps.

Measurement Interface

A double-click on a measurement folder opens an interface for the management of this measurement, the Measurement editor. The interface, shown in Figure 9, allows the user to perform a measurement, that is, to play the chosen input signal and record the room response. After that, by clicking on the “Calculate IR” button, the user asks the software to

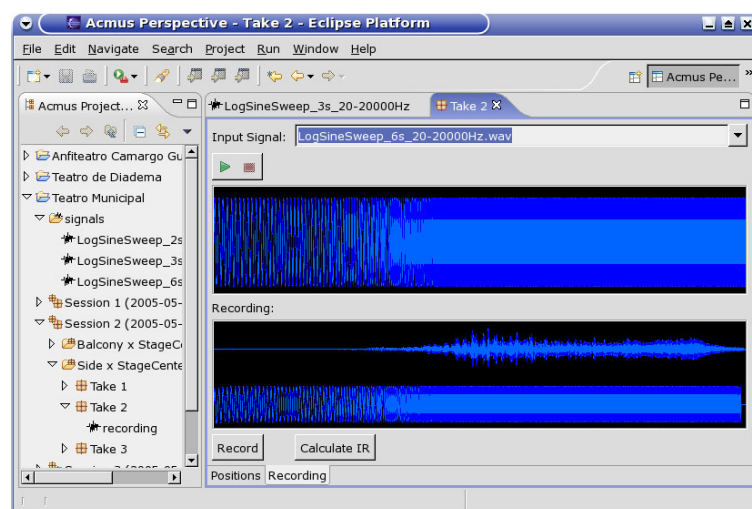


Figure 9: The Measurement Interface.

calculate the impulse response from the captured sound. The result is shown in Figure 10. Finally, the button “Calculate parameters” shows the acoustical parameters obtained from the impulse response. Figure 11 shows the parameters that the current implementation is able to calculate.

The code for this part of the software is a reimplementation of the MATLAB prototype (Section 2.1). For this we are also implementing a DSP library that currently

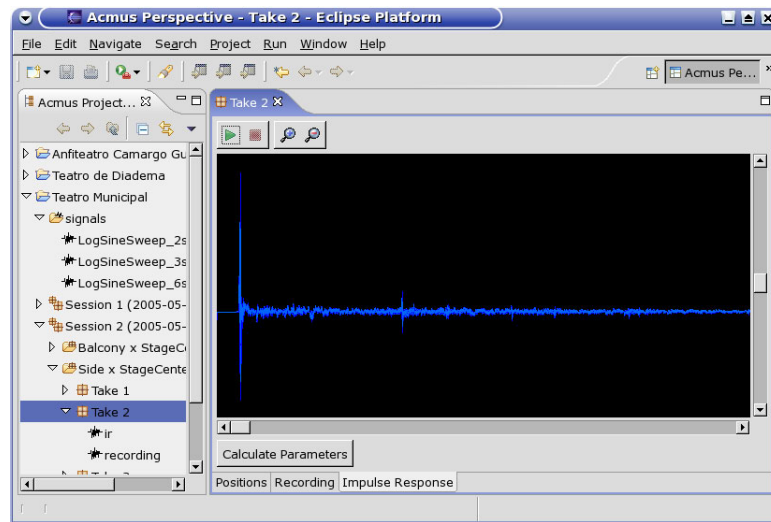


Figure 10: The calculated impulse response.

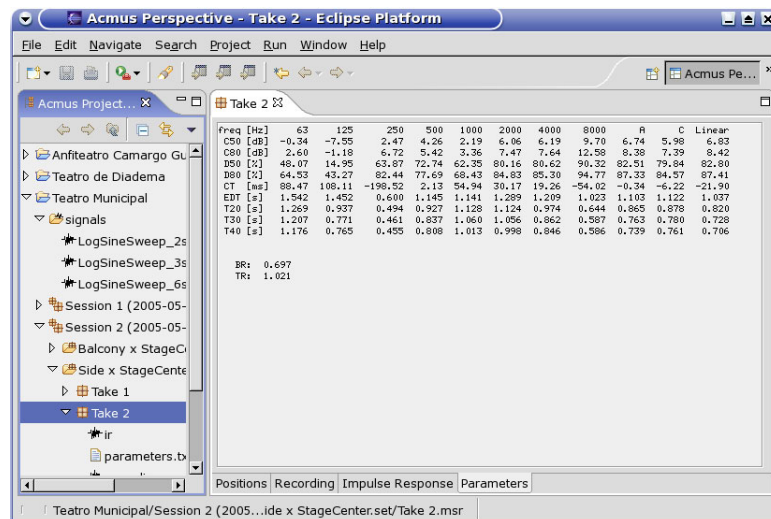


Figure 11: The calculated parameters.

has a few FFT and filtering functions. At this early stage of development, we do not know yet if the Java performance will be an issue, since we have been doing a straightforward translation from the MATLAB functions. If this is the case, we may have to implement some of these functions in C++ and integrate them with the JNI [Liang, 1999] at the cost of the full portability.

5. Conclusions

The AcMus software offers the main acoustic calculations and processing tools available in similar commercial systems such as WinMLS (<http://www.winmls.com>), Aurora (<http://pcfarina.eng.unipr.it/aurora/home.htm>), ETF5 (<http://www.etfacoustic.com>), and Sample Champion (<http://purebits.com/>). However, some additional tools are not implemented in AcMus, for instance, signal convolution and deconvolution, and STI (Speech Transmission Index) and ST1 (support) calculations, but they may be added in future releases.

Besides, in order to share the results of our research with other groups more easily, the AcMus software is an open-source project. All the source code and papers we produced are available at our Web site: <http://gsd.ime.usp.br/acmus>. None of the

mentioned systems are completely freely available and open-source.

5.1. Next Steps

In addition to the analysis of the theaters we measured, we have work to be done on the final implementation of the Measurement Module. There is a lot of room for enhancements, especially in the measurement interface, so that we could properly meet Requirements 4 and 2. Also, the parameter calculation functionality is not complete yet.

All this work is of great importance for the last stage of the project: the final implementation of the Simulation and Optimization Module. We have been working on this module since the beginning of the project [de Queiroz, 2003, de Avelar Gomes et al., 2004], so now we are getting ready to incorporate this research to the AcMus integrated platform.

References

- Beranek, L. L. (2004). *Concert halls and opera houses: music, acoustics, and architecture*. Springer-Verlag, New York.
- de Avelar Gomes, M. H., Vorländer, M., and Gerges, S. N. Y. (2004). Measurement and use of scattering coefficients in room acoustic computer simulations. In *Proceedings of the European Acoustics Symposium*, Portugal.
- de Queiroz, M. G. (2003). Some optimization models for listening room design. In *Proceedings of the 9th Brazilian Symposium on Computer Music*, pages 149–154, Campinas, Brazil.
- Figueiredo, F. L., Masiero, B. S., and Iazzetta, F. (2004). Análise de parâmetros acústicos subjetivos: Critérios para avaliação acústica de salas de música. In *Anais da 4ta. Reunion Anual de la Sociedad Argentina para las Ciencias Cognitivas de la Música*, Tucumã, Argentina.
- Iazzetta, F., Kon, F., de Queiroz, M. G., da Silva, F. S. C., and de Avelar Gomes, M. (2004). AcMus: Computational tools for measurement, analysis and simulation of room acoustics. In *Proceedings of the European Acoustics Symposium*, Portugal.
- ISO 3382 (1997). Acoustics – measurement of the reverberation time of rooms with reference to other acoustical parameters.
- Kuttruff, H. (1991). *Room Acoustics*. Elsevier Applied Science, London.
- Liang, S. (1999). *Java Native Interface: Programmer's Guide and Reference*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Masiero, B. S. (2004). Estudo e implementação de métodos de mediação e resposta impulsiva em salas de pequeno porte. Technical report, Fapesp. Available at http://gsd.ime.usp.br/acmus/publi/relat_medicao.pdf.
- Müller, S. and Massarani, P. (2001). Transfer function measurements with sweeps. *J. Audio Eng. Soc.*, 49:443.
- Ueda, L. K., Kon, F., and Iazzetta, F. (2005). An open-source platform for musical room acoustics research. In *Proceedings of the 2005 International Congress and Exposition on Noise Control Engineering*, Rio de Janeiro, Brazil.
- Yili, Y., Silva, F., Iazzetta, F., and Kon, F. (2003). Estimadores de qualidade para pequenas salas destinadas a atividades musicais. In *Anais do IX Simpósio Brasileiro de Computação Musical*, pages 163–170, Campinas, Brazil.

Localização de Fontes e Ouvintes em Salas de Escuta*

Luciana Dias¹, Marcelo Queiroz¹

¹Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Abstract. *This work consists in the study and implementation of geometrical acoustics and optimization techniques for the automatic search of an optimal placement of sound sources and listeners in listening rooms, aiming at a minimal harmonic distortion. The implementation is part of the ACMUS toolkit which provides computational tools for measurement of acoustical parameters as well as other simulation packages. Original contributions are the use of global optimization strategies and the release of open-source implementations.*

Resumo. *Este trabalho consiste no estudo e implementação de técnicas de acústica geométrica e otimização para a busca automática de localizações de fontes e ouvinte em salas de escuta, visando obter o mínimo de distorção harmônica possível. Esta implementação faz parte do conjunto de ferramentas de simulação do projeto ACMUS, que possui também ferramentas para a medição de parâmetros acústicos e outras ferramentas de simulação. São contribuições originais o uso de estratégias de otimização global e a disponibilização de implementações em código-aberto.*

1. Introdução

Uma sala de escuta é um ambiente caracterizado por possuir superfícies com diferentes características de reflexão e absorção do som, sendo portando um recinto reverberante. Quando uma fonte sonora é situada em um ambiente como esse, o volume de ar encerrado entre as superfícies limites da sala é excitado não somente pelas ondas sonoras provenientes diretamente da fonte, mas também pelas ondas que são refletidas pelas superfícies [Beranek, 1993, López, 2001].

A forma mais apropriada para obtenção de parâmetros acústicos de uma sala é a execução de medições no próprio local; quando isto não é possível, pode-se construir uma réplica em miniatura e utilizar as medições na réplica para obter aproximações dos parâmetros referentes ao ambiente original. A utilização de réplicas é viável quando não se pretende fazer modificações na sala de escuta, ou quando o número de modificações é muito pequeno. Ao considerar o problema de projeto de salas de escuta observa-se a dificuldade em utilizar medições no local ou em réplicas, pois durante a fase de projeto se deseja testar uma enorme quantidade de combinações dos materiais e de posições de fontes ou ouvintes, ou mesmo as dimensões da sala, entre outros parâmetros.

*Projeto financiado pela FAPESP, bolsa 04/01184-9 e projeto temático 02/02678-0.

Para simular computacionalmente o comportamento acústico de uma sala e quantizar o nível de distorção harmônica na posição de escuta, é necessário obter uma aproximação de sua *resposta em frequência* a partir de uma quantidade mínima de informações que correspondem ao modelo da sala. A resposta em frequência está associada à distorção harmônica pois quantifica a atenuação ou amplificação sofrida por cada frequência, para uma certa disposição de fonte sonora e ouvinte. Essas diferenças no comportamento de cada frequência são consequência da ressonância do recinto, das diferentes velocidades de amortecimento dos sinais e dos modos normais de vibração. Dois dos principais modelos geométricos teóricos empregados para a simulação acústica computacional são o *modelo de traçado de raios* e o *modelo de fontes virtuais* [Rindel, 1997, Rindel, 2000, López, 2001]. Estes modelos permitem a obtenção de forma aproximada da resposta impulsiva da sala, que é convertida em resposta em frequência através da transformada rápida de Fourier (FFT). Tais simulações computacionais são imprescindíveis para uma busca automática de localizações ótimas para fontes sonoras e ouvinte.

Uma das alternativas para efetuar uma busca automática de localizações ótimas de fonte sonora e ouvinte é utilizar técnicas de otimização [Warusfel, 1995, D'Antonio and Cox, 1997, Queiroz, 2003]. Para tal, é necessário definir matematicamente qual é o objetivo de tal processo de busca. Neste trabalho considera-se a minimização da distorção harmônica, que pode ser expressa através de uma medida de distância entre a resposta em frequência simulada e uma resposta em frequência ideal; tal resposta ideal poderia ser uma resposta plana ou outra função pré-definida. A cada passo do algoritmo de busca, calcula-se através de uma simulação por acústica geométrica a resposta em frequência associada a uma certa localização da fonte sonora e do ouvinte e se obtém a medida de distorção correspondente, que é a função objetivo a ser minimizada. Ao final do algoritmo são produzidas as localizações mais adequadas para o posicionamento da fonte sonora e ouvinte, isto é, as posições que minimizam a distorção da resposta em frequência dentre as posições visitadas. O uso de técnicas de otimização global, não mencionadas em trabalhos relacionados, é de grande relevância pois a medida de distorção em função da localização das fontes e ouvinte possui muitos mínimos locais com valores bastante diversos.

É objetivo deste trabalho contribuir com as pesquisas do projeto *ACMUS*, que visa o desenvolvimento de um software para cálculo, análise e simulação de acústica de salas para prática musical [F. H. Iazzetta and Silva, 2001, AcMus, 2005]. Uma das fortes contribuições deste projeto consiste em disponibilizar os códigos-fonte de todas as ferramentas implementadas. Utilizou-se a linguagem *Java*, por sua robustez, versatilidade e portabilidade. As seções a seguir detalham as técnicas de acústica geométrica e otimização global utilizadas na implementação, e apresentam os resultados obtidos.

2. Modelo de Acústica Geométrica

O Modelo de Fontes Virtuais

Segundo o modelo de fontes virtuais, o sinal da fonte sonora refletido por uma parede é idêntico ao de uma onda direta criada por uma fonte virtual situada do outro lado

da parede, como uma imagem especular da fonte sonora real. Ou seja, de acordo com este modelo, as fontes virtuais irradiam ondas sonoras sincronizadas com a fonte real, e apenas caminhos diretos entre fontes e ouvinte são considerados. Seguindo o mesmo processo para todas as fontes virtuais nas salas virtuais adjacentes, pode-se encontrar todas as reflexões de primeira ordem que passam por um determinado ponto do ambiente. Esse processo se repete para reflexões de segunda ordem, terceira, etc.

Este modelo é aplicado a ambientes de superfícies planas e usa o modelo de reflexões especulares, que é o caso dos ambientes tratados neste projeto, e é mais indicado pra salas de geometrias simples, como as de forma cubóide, já que para geometrias mais complexas é muito difícil identificar as posições das fontes virtuais. Modelos geométricos que tratam a propagação do som por raios são bastante acurados em altas frequências, mas sua qualidade cai consideravelmente nas frequências mais baixas. Por outro lado estes modelos têm como principal vantagem a sua eficiência computacional, visto que toda informação processada diz respeito à localização do ouvinte e às reflexões que atingem aquele ponto. Outras técnicas de simulação, como a solução da equação de onda por elementos finitos, são computacionalmente inviáveis por necessitarem da informação da variação de pressão em uma grande quantidade de pontos (que fornecem uma aproximação de todo o espaço da sala).

Na modelagem para o problema em questão, cada fonte virtual está localizada em uma sala virtual, identificada por um índice (i, j, k) , que indica a distância da sala virtual em relação à sala real em número de salas, sobre um eixo tridimensional. Por exemplo, a sala virtual imediatamente à direita da sala real na direção do eixo x possui índice $(1, 0, 0)$. Para efeito de cálculo computacional, considera-se apenas as salas virtuais situadas a uma distância menor do que um certo raio máximo da sala real, onde a medida de distância é dada por $|i| + |j| + |k|$.

Obtenção da Resposta Impulsiva

A resposta impulsiva de um ambiente de escuta musical corresponde à resposta produzida por ele para um impulso produzido pela fonte sonora. De acordo com o modelo geométrico adotado e considerando-se a simulação de um pulso ideal (delta de Dirac), a resposta impulsiva fornece um *mapa temporal* das reflexões, registrando o instante e a intensidade com que cada reflexão chega ao ponto onde está situado o ouvinte.

Para obter o *retardo* com que a reflexão da sala de índice (i, j, k) chega ao ouvinte, calcula-se a distância da fonte sonora real (x_F, y_F, z_F) à fonte sonora virtual

$$(2X[i/2] + (-1)^{|i|}x_F, 2Y[j/2] + (-1)^{|j|}y_F, 2Z[k/2] + (-1)^{|k|}z_F)$$

onde (X, Y, Z) são as dimensões da sala. A partir da posição de cada fonte virtual e da posição do ouvinte, é fácil calcular o tempo gasto pelas reflexões até atingir o ouvinte, dividindo-se a distância percorrida por cada reflexão pela velocidade do som no ar.

Para calcular a absorção sofrida pelo sinal sonoro saído da fonte localizada na sala virtual de índice (i, j, k) deve-se considerar todos os coeficientes de absorção das superfícies intersectadas pelo raio que liga a fonte virtual ao ouvinte. Se os

coeficientes das superfícies são $\alpha_0, \alpha_1, \beta_0, \beta_1, \gamma_0, \gamma_1$ então a absorção total devido às superfícies será

$$\alpha_0^{\lfloor |i/2| \rfloor} * \alpha_1^{\lceil |i/2| \rceil} * \beta_0^{\lfloor |j/2| \rfloor} * \beta_1^{\lceil |j/2| \rceil} * \gamma_0^{\lfloor |k/2| \rfloor} * \gamma_1^{\lceil |k/2| \rceil}$$

onde os expoentes indicam justamente quantas vezes o sinal proveniente da sala (i, j, k) é refletido em cada uma das superfícies. Além disto há também a atenuação devido à distância percorrida d , que corresponde a um fator de atenuação $\mathcal{O}(1/d)$.

Obtenção da Resposta em Frequência

A resposta em frequência de uma sala fornece uma quantificação da atenuação ou amplificação sofrida por cada frequência, para uma certa localização de fonte sonora e ouvinte. Através da Transformada de Fourier, pode-se obter a resposta em frequência através do espectro da resposta impulsiva. Este espectro é calculado através do algoritmo conhecido como *Transformada Rápida de Fourier (FFT)*, que permite obter N amostras da resposta em frequência em tempo $\mathcal{O}(N \log N)$.

Estimativa de Distorção na Resposta em Frequência

Tendo a resposta em frequência amostrada para uma certa posição de fonte sonora e ouvinte, pode-se comparar a quantidade de distorção presente nessa resposta em relação a uma resposta em frequência “neutra”, como por exemplo a *resposta plana*, ou outro modelo adequado.

Uma resposta plana corresponde à situação hipotética em que nenhuma frequência sofreria qualquer atenuação ou amplificação, a não ser aquela devida à absorção do ar. Esta situação hipotética só pode ser obtida num ambiente sem superfícies refletoras (ao ar livre ou em um ambiente com 100% de absorção), e não é sequer uma situação ideal, visto que a reverberação é de fato desejável em um ambiente de escuta musical.

Por outro lado a resposta plana fornece um bom parâmetro de comparação entre duas respostas em frequência “reais”, pois aquela que resultar em maior distorção estará também mais distante da resposta plana.

A distorção é calculada a partir da resposta logarítmica em frequência, pois esta corresponde melhor à percepção de distorção: intuitivamente, pode-se pensar que um fator s (uma amplificação de s vezes) e um fator $1/s$ (uma atenuação de s vezes) são percebidos como distorções igualmente “ruins”. Para uma dada resposta logarítmica em frequência armazenada em um vetor F com N amostras, define-se a medida de distorção pela expressão

$$d(F) = \sqrt{\sum_{i=0}^{N-1} (|F[i] - \mu|)^2}$$

onde $\mu = \frac{1}{N} \sum_{i=0}^{N-1} F[i]$ é a média dos fatores de atenuação/amplificação, e normalmente está próximo de zero (que é o logaritmo do valor 1, que corresponde a nenhuma atenuação ou amplificação). Observe que esta expressão corresponde à distância entre o gráfico de F e o gráfico de resposta constante e igual a μ , utilizando a norma Euclidiana.

3. Técnicas de Otimização Global

Em geral um problema de otimização global corresponde a minimizar uma função $f(x)$ (chamada de função objetivo) onde $x \in S$ e S é o conjunto de soluções viáveis. Muitos problemas de otimização contínua possuem várias soluções ótimas locais. Para esses casos, principalmente quando as condições do escopo clássico de otimização contínua (diferenciabilidade e convexidade) não são satisfeitas ou verificáveis, usa-se estratégias de otimização global. Em princípio, o objetivo é encontrar um ótimo global, ou seja, um $x^* \in S$ tal que $f(x^*) \leq f(x), \forall x \in S$. Quando S é finito, poder-se-ia fazer uma busca exaustiva, embora isso seja computacionalmente inviável na maioria dos casos. Quando S não é finito, em geral são utilizados métodos iterativos, ou seja, métodos que geram uma sequência de soluções que aproximam-se da solução ótima.

Métodos de otimização global podem ser classificados como *determinísticos* ou *estocásticos*, bem como *heurísticos* ou *exatos*. Nos métodos determinísticos, a partir dos mesmos dados iniciais, chega-se à mesma solução; já os métodos estocásticos possuem, em geral, algum componente de busca global aleatória, mas podem e normalmente são utilizados em conjunto com estratégias determinísticas. A distinção entre métodos exatos e heurísticos tem relação com a existência de garantias teóricas de convergência a uma solução global. Poucos problemas de otimização global são suficientemente bem estruturados a ponto de admitirem um método de solução exata; muitos problemas reais e relevantes só podem ser resolvidos aproximadamente.

Uma característica fundamental do problema, no que concerne a escolha de um algoritmo específico, é o alto custo computacional envolvido no cálculo da função objetivo, que depende da simulação geométrica e do cálculo da FFT; deve-se, assim, garantir que o algoritmo não calcule a função objetivo em um número excessivamente grande de pontos, ou de forma desnecessária.

Para comparar duas abordagens heurísticas para o problema, foram implementados o método estocástico *Density Clustering* [Horst and Pardalos, 1995] e o método determinístico baseado em grades [Coope and Price, 2001].

Método *Density Clustering*

Este é um método em duas fases: a primeira fase (fase global) gera uma amostragem aleatória de pontos viáveis usando a distribuição uniforme sobre o conjunto viável, pontos estes que são avaliados de acordo com a função objetivo; a segunda fase (fase local) aplica um procedimento de otimização local L a partir de um subconjunto dessa amostragem, obtendo vários mínimos locais, dos quais o melhor é selecionado.

Duas técnicas para o pré-processamento da amostra são *Redução*, que consiste em tomar uma fração da amostra apenas com os pontos com melhor valor da função objetivo, e *Concentração*, que consiste em modificar a amostra aplicando alguns passos de Cauchy para cada ponto. Antes de aplicar L a cada ponto, são criados grupos (*clusters*) com pontos relativamente próximos, e aplica-se L apenas uma vez para cada grupo. O método básico para identificação de clusters é tomar o melhor ponto da amostra e adicionar outros pontos usando uma *regra de clustering*.

Uma destas regras baseia-se na hipótese de que a região de atração de um mínimo local pode ser aproximada por uma elipsóide, isto é, a função objetivo pode ser localmente aproximada por uma função quadrática. O algoritmo consiste basicamente no seguinte:

- 0 $k \leftarrow 1, X^* \leftarrow \emptyset, i \leftarrow 1, j \leftarrow 0$
- 1 Gere N pontos usando uma distribuição uniforme sobre o conjunto viável S e selecione os γkN melhores pontos, para um dado $\gamma \in (0, 1)$ (método da redução).
- 2 Se todos os pontos selecionados estão em algum *cluster*, vá para o passo 4.
 Se $j \leq |X^*|$, escolha o j -ésimo mínimo local $x_* \in X^*$ como o próximo “ponto semente” e vá para o passo 3.
 Se $j > |X^*|$, escolha o melhor ponto \bar{x} que não estiver em nenhum *cluster* e aplique o método de otimização local L a partir de \bar{x} . Se o mínimo local resultante x_* for um elemento de X^* , inclua \bar{x} no *cluster* de x_* e volte para o passo 2. Se $x_* \notin X^*$, adicione x_* a X^* e faça x_* ser o próximo “ponto semente”.
- 3 Adicione todos os pontos sem *cluster* da amostra reduzida que estejam dentro de uma distância $r_i(x_*)$ do ponto semente ao *cluster* iniciado por x_* . Se nenhum ponto tiver sido adicionado, faça $j \leftarrow j + 1$ e volte para o passo 2; senão, faça $i \leftarrow i + 1$ e volte para o passo 3.
- 4 Faça $k \leftarrow k + 1$ e volte para o passo 1.

A distância crítica $r_i(\cdot)$ sugerida em [Rinnoykan and Timmer, 1987] é dada pela equação

$$r_i(x) = \frac{1}{\pi} \left(i \Gamma \left(1 + \frac{d}{2} \right) \cdot \sqrt{-\det H(x)} \cdot m(S) \cdot \frac{\zeta \ln(kN)}{kN} \right)^{1/d}$$

onde d é a dimensão do problema, $m(S)$ é a medida de Lebesgue, $H(x)$ é a Hessiana no ponto x , $\zeta > 0$ é um parâmetro positivo do algoritmo e $\Gamma(\cdot) = \int_{t=0}^{\infty} t^{z-1} e^{-t} dt$.

Método baseado em grades

O método de otimização baseado em grades é um método determinístico, que busca um minimizador da função objetivo examinando-a em uma seqüência de grades sucessivamente refinadas. Cada grade $G^{(m)}$ é definida por um conjunto de n vetores-base linearmente independentes $V^{(m)}$ onde $V^{(m)} = \{v_i^{(m)} \in \mathbb{R}^n : i = 1, \dots, n\}$. Os pontos da grade $G^{(m)}$ são $G^{(m)} = \{x \in \mathbb{R}^n : x = x_o^{(m)} + h^{(m)} \sum_{i=1}^n \eta_i v_i^{(m)}\}$ onde η é um inteiro e $h^{(m)}$ é um número positivo que representa a espessura da malha. Este parâmetro é ajustado a medida que m cresce para assegurar o refinamento das malhas. A base $V^{(m)}$ é usada para formar uma base positiva $V_+^{(m)}$ tal que todo vetor em \mathbb{R}^n é combinação linear positiva dos vetores em V_+ e nenhum elemento de V_+ é combinação linear dos outros. Se $V^{(m)}$ é a base canônica, pode-se construir uma base positiva fazendo $V_+^{(m)} = \{V_1^{(m)}, \dots, V_n^{(m)}, -\sum_{i=1}^n V_i^{(m)}\}$; outra possibilidade é fazer $V_+^{(m)} = \{V_1^{(m)}, \dots, V_n^{(m)}, -V_1^{(m)}, \dots, -V_n^{(m)}\}$.

Por definição, tem-se que $x \in G^{(m)}$ é um *minimizador local de grades* com respeito à base positiva $V_+^{(m)}$ se e somente se $f(x + h^{(m)}v_i) \geq f(x)$, $\forall v_i \in V_+^{(m)}$. Se x não é um *minimizador local de grades* então existe uma direção de descida $s^{(k)} \in V_+^{(m)}$, isto é, uma direção que satisfaz $f(x^{(k)} + h^{(m)}s^{(k)}) < f(x^{(k)})$. Como esta direção promove uma melhoria no valor da função objetivo, é interessante continuar explorando esta direção através de uma busca em profundidade: a função objetivo f é avaliada na seqüência de pontos $x^{(k)} + \alpha_i h^{(m)}s^{(k)}$, onde $\{\alpha_i\}$ é uma seqüência crescente de inteiros, enquanto $f(x^{(k)} + \alpha_{i+1}h^{(m)}s^{(k)}) \leq f(x^{(k)} + \alpha_i h^{(m)}s^{(k)})$. A busca termina na primeira violação desta última condição.

Algoritmo de Minimização em Grades

- 0 $m \leftarrow 1$, $k \leftarrow 1$ e $x^{(1)} \in G^{(m)}$.
- 1 Enquanto as condições de parada forem falsas,
 - 1.A Faça $i \leftarrow 1$ e $r \leftarrow 0$.
 - 1.B Enquanto $x^{(k)}$ não for um minimizador local da grade,
 - Calcule uma direção de descida $s^{(k)} \in V_+^{(m)}$; se não houver direção de descida, vá para o passo [1.C].
 - Escolha uma seqüência crescente de inteiros $\alpha_0 = 1 < \alpha_1 < \alpha_2 \dots$ e determine o menor índice l que satisfaz $f(x^{(k)} + \alpha_{l+1}h^{(m)}s^{(k)}) \geq f(x^{(k)} + \alpha_l h^{(m)}s^{(k)})$.
 - Faça $x^{(k+1)} \leftarrow x^{(k)} + \alpha_l h^{(m)}s^{(k)}$.
 - Faça $k \leftarrow k + 1$.
 - 1.C Faça $m \leftarrow m + 1$.

As condições de parada envolvem algum teste para detectar convergência dos iterados, bem como limitações no tempo de execução ou número de iterações. É possível provar que, sob certas hipóteses técnicas, o algoritmo acima gera uma seqüência de pontos que converge para um ponto estacionário de f . Para mais detalhes, veja o artigo [Coope and Price, 2001].

4. Implementação e Resultados

Resposta Impulsiva

O cálculo da resposta impulsiva da sala é feito no método *generateImpulseResponse*, que se encontra no arquivo *ImpulseResponse.java*. Seus dados de entrada consistem nas características de uma sala de escuta cubóide (cuja descrição é dada pelas dimensões e coeficientes de absorção de cada uma de suas superfícies) e posições fixadas para a fonte sonora e o ouvinte. O número máximo de fontes virtuais gerado é também passado como parâmetro, e é representado por um “raio máximo” de salas em torno da sala real. Em outras palavras, para um raio máximo igual a R , serão geradas todas as fontes virtuais contidas em salas cujos índices (i, j, k) , indicadores de suas posições em relação à sala real, satisfaçam a condição $|i| + |j| + |k| \leq R$.

A lista das intensidades das reflexões que chegam ao ouvinte a cada instante é gerada e o vetor correspondente à resposta impulsiva é gerado pelo método *generateFFTInputArray*, que está no arquivo *FrequencyResponse.java*. Como os instantes de chegada de cada reflexão, para salas pequenas, são menores que 1, foram tomadas as x casas decimais seguintes como sendo os índices daquela reflexão em um vetor adequado de tamanho igual a uma potência de 2. Aqui, x é igual a $-\log_{10} \epsilon$, onde ϵ é uma constante real próxima de 0. Os valores para o raio máximo de salas e para o erro máximo permitido estão definidos no arquivo *Defs.java*, nas variáveis *maxSum* e *epsilon*, respectivamente.

A resposta impulsiva assim simulada é uma função com muitas descontinuidades, o que gera no espectro uma quantidade muito grande de ruído, especialmente em altas frequências. Uma maneira de compensar este efeito é aplicar um filtro passa-baixa [Moore, 1990] à resposta impulsiva produzida. O método *applyFilter*, também em *FrequencyResponse.java*, implementa e aplica à resposta impulsiva um filtro FIR.

Resposta em Frequência e Cálculo da Distorção

Unindo os dois procedimentos descritos nas seções anteriores (FFT e cálculo da resposta impulsiva), torna-se simples obter a resposta em frequência da sala. A resposta logarítmica em frequência é utilizada a seguir para calcular a distorção, de acordo com a definição apresentada. O método *stdDeviation*, no arquivo *FrequencyResponse.java*, é responsável por realizar o cálculo da resposta impulsiva, gerar a partir deste resultado a entrada adequada para o algoritmo da FFT e obter a resposta em frequência, tudo isso através de métodos auxiliares. Este método foi encapsulado na classe *ObjectiveFunctionPAOSE* para que pudesse ser chamado a partir dos algoritmos de otimização.

Algoritmos de Otimização

Como mencionado anteriormente, foram implementados dois algoritmos de otimização: o *Density Clustering* e o *Grid-based method*. Ambos foram escritos de forma que pudessem ser reutilizados para a resolução de outros problemas, permanecendo independentes desta aplicação específica. Foi criada também uma classe *Cache*, responsável por armazenar os valores da função objetivo em todos os pontos já percorridos pelo algoritmo. Essa estratégia foi escolhida porque cada cálculo do valor da função objetivo é relativamente caro; por isso, é necessário aproveitar todos os cálculos já efetuados.

O algoritmo *Grid*, implementado no método *executeAlgorithm* do arquivo *GridMethod.java*, depende dos seguintes parâmetros: a dimensão d do problema, o ponto de partida do algoritmo *xInit*, uma matriz contendo os limites de cada uma das d coordenadas dos pontos a serem sorteados, um objeto do tipo *ObjectiveFunction*, que deve encapsular a chamada da função objetivo desejada, o número máximo de iterações permitido, o tamanho da malha inicial, e o objeto *Cache* que fará o armazenamento dos pontos visitados.

O algoritmo *Density Clustering* foi implementado no método *executeAlgorithm* do arquivo *DensityClustering.java*. Ele executa uma série de iterações bus-

cando o ótimo global de um problema. Para a execução deste método, é necessário o conhecimento dos seguintes valores: a dimensão d do problema, o número N de pontos a serem sorteados a cada iteração, a fração de redução γ , o número máximo de iterações desejado para o algoritmo, o número máximo de iterações para o método de otimização local, uma matriz contendo os limites de cada uma das d coordenadas dos pontos a serem sorteados, um objeto do tipo *ObjectiveFunction*, que deve encapsular a chamada da função objetivo desejada, a medida de Lebesgue do conjunto viável (que para este problema de otimização consiste simplesmente no produto das dimensões da sala) e o objeto *Cache* que fará o armazenamento dos pontos visitados. O procedimento de minimização local usado nesta implementação do *Density Clustering* foi o próprio *Grid-based method*.

O Programa

Em <http://gsd.ime.usp.br/acmus/> encontra-se disponível para download o programa a que se refere este artigo. O método principal do programa do Projeto Acústico Ótimo de Salas de Escuta (PAOSE) encontra-se no arquivo *MainPAOSE.java*. A interface (em linha de comando) do programa é

```
java MainPAOSE arquivo.txt [OPÇÃO]
```

O arquivo.txt deve conter os dados da sala (dimensões e coeficientes de absorção) no seguinte formato:

```
X Y Z
alpha0 alpha1 beta0 beta1 gamma0 gamma1
```

onde X, Y e Z são as dimensões da sala e alpha0, ..., gamma1 são os coeficientes de absorção de suas superfícies. O segundo argumento consiste em um código que define o procedimento de otimização a ser utilizado no programa. As opções são -d para o método *Density Clustering* com o algoritmo baseado em grades como método de otimização local e -g para o método baseado em grades como método de otimização global.

O programa imprime na saída padrão as seguintes informações: tempo de execução até a obtenção da resposta; localizações ótimas encontradas para fonte e ouvinte; quantidade de distorção gerada nas localizações ótimas encontradas; maior quantidade de distorção dentre as geradas nos pontos visitados pelo método de otimização; este valor é utilizado para comparação com o ótimo. Ainda, definindo o valor da variável booleana *debug* do arquivo *Defs.java* como *true*, o programa imprime na saída padrão o caminho percorrido pelo algoritmo de otimização.

Testes Computacionais

Os testes realizados até o momento são preliminares e sugerem algumas modificações, discutidas na seção a seguir. Considerou-se inicialmente o problema de localizar duas fontes e um ouvinte, dispostos simetricamente em uma sala de escuta de dimensões $X = 3m \times Y = 4m \times Z = 5m$. Uma das fontes ocupa a posição (x_F, y_F, z_F) e a outra $(X - x_F, y_F, z_F)$, onde (x_F, y_F, z_F) satisfaz $0 \leq x_F \leq X/2$, $Y/2 \leq y_F \leq Y$ e $0 \leq z_F \leq Z$. O ouvinte deve ser localizado em (x_O, y_O, z_O) onde $x_O = X/2$ e $0 \leq y_O \leq Y/2$. Este é um problema que pode ser resolvido considerando-se apenas

a primeira fonte e o ouvinte, visto que a resposta impulsiva produzida pela segunda fonte é idêntica à produzida pela primeira, o que é consequência direta da simetria, e portanto a distorção harmônica produzida pela segunda fonte é também idêntica.

O método *Grid* aplicado aos dados acima produziu a seguinte resposta

```
Número de avaliações da função no Grid: 75
Grid Method - tempo de resposta: 87.509 s
Valor do pior ponto encontrado: 1.4202278753437534
- Localizações ótimas encontradas -
Fonte: (0.2109375, 3.5, 4.84375)
Ouvinte: (1.5, 0.5, 0.0)
Quantidade de distorção: 0.5617007729474769
```

Neste exemplo observa-se uma variação de 252.84% entre a pior localização e a melhor localização encontrada. Vela a pena mencionar também que a localização inicial do método foi ($x_F = 0.0, y_F = 2.0, z_F = 0.0$) e ($x_O = 1.5, y_O = 0.0, z_O = 0.0$), com uma distorção de 1.339057806330562 (238.38% acima do ótimo).

O método *Density Clustering* produziu a resposta

```
Density Clustering - tempo de resposta: 1378.040 s
Valor do pior ponto sorteado: 0.9838447204945652
- Localizações ótimas encontradas -
Fonte: (1.499908447265625, 3.5316162109375, 4.10125732421875)
Ouvinte: (1.5, 0.8748779296875, 0.00152587890625)
Quantidade de distorção: 0.5761921592717776
```

Observe que o tempo de resposta é bem mais alto do que o Grid, o que é esperado, visto que o Grid é uma subrotina do Density Clustering, utilizada um grande número de vezes. Neste teste em particular a solução final encontrada não era melhor do que a do Grid, o que pode ser explicado como “sorte” do método Grid em encontrar um mínimo local tão bom em apenas uma tentativa. Outras considerações sobre a comparação dos métodos é dada na seção seguinte.

Os mesmos dados de descrição da sala foram fornecidos aos dois métodos sem a exigência de simetria das localizações; neste caso apenas uma fonte foi considerada, que poderia ser localizada em qualquer ponto da sala, o mesmo acontecendo com o ouvinte. Os dados obtidos foram

```
Número de avaliações da função no Grid: 57
Grid Method - tempo de resposta: 61.775 s
Valor do pior ponto encontrado: 0.7797158869018019
- Localizações ótimas encontradas -
Fonte: (2.25, 0.0, 0.0)
Ouvinte: (2.25, 2.0, 2.5)
Quantidade de distorção: 0.3958269857678017

Density Clustering - tempo de resposta: 850.816 s
Valor do pior ponto sorteado: 1.7976931348623157E308
- Localizações ótimas encontradas -
```

Fonte: (1.6875, 1.75, 2.5)

Ouvinte: (1.6875, 1.75, 2.490234375)

Quantidade de distorção: 0.36589052199761113

No último exemplo observa-se o comportamento típico dos dois métodos. O *Density Clustering* encontrou uma solução melhor do que a do *Grid*, em tempo também maior. Outras observações sobre as implementações são feitas a seguir.

5. Conclusão e Trabalhos Futuros

Os resultados da seção anterior são ainda preliminares, pois várias melhorias ainda podem ser feitas no método levando-se em consideração apenas o modelo de sala atual. No entanto, pode-se afirmar que são inovadoras a proposta de usar estratégias de otimização global e a disponibilização de código-aberto para a localização de fontes e ouvinte em salas de escuta.

Considerando que o método baseado em *Grade* busca apenas um minimizador local da função, é natural que seu tempo de execução seja muito menor que o método *Density Clustering*, que em particular efetua várias chamadas do *Grid*. É natural supor que um usuário escolherá o método de busca global apenas quando tiver tempo de esperar a resposta. Assim sendo, algumas variantes do método devem ser exploradas. Uma destas variantes corresponde a substituir a base positiva do método *Grid* $V_+^{(m)} = \{V_1^{(m)}, \dots, V_n^{(m)}, -\sum_{i=1}^n V_i^{(m)}\}$, escolhida para minimizar o número de chamadas da função objetivo, pela base $V_+^{(m)} = \{V_1^{(m)}, \dots, V_n^{(m)}, -V_1^{(m)}, \dots, -V_n^{(m)}\}$, que fornece mais liberdade de movimentação para o método e aumenta a chance de se encontrar uma direção de descida a partir do ponto atual.

Outras extensões do trabalho correspondem a melhorias no modelo da sala e no resultado da simulação por acústica geométrica. A consideração com maior prioridade de inclusão é a representação dos coeficientes de absorção como funções da frequência. Neste respeito podem ser utilizadas tabelas de materiais com valores de coeficiente de absorção por bandas de frequência (por exemplo terços de oitava), ou ainda aproximações baseadas em modelos matemáticos das curvas de absorção, com preferência para o que representar o menor *overhead* computacional para o método. Coeficientes de absorção variam também em função do ângulo de incidência; tal dependência poderia também ser levada em consideração através de valores tabulados ou de modelos analíticos.

A consideração de outras geometrias além da cubóide apresenta grande interesse prático e também grande dificuldade teórica e computacional. Ao considerar-se salas de geometria poliédrica geral, o modelo das fontes virtuais deixa de possuir fórmulas fechadas para a localização das fontes e passa a depender de técnicas de álgebra linear; também não é desprezível o fato de que em geral o número de fontes virtuais de ordem n cresce exponencialmente em função de n (para geometria cubóide esse crescimento é polinomial). Uma alternativa é utilizar o modelo de traçado de raios, que gera uma grande quantidade de raios saindo da fonte real em todas as direções, e acompanha o caminho geométrico destes raios até atingirem o ouvinte; ou ainda utilizar uma estratégia híbrida de fontes virtuais e traçado de raios.

Finalmente, outros fenômenos acústicos tais como difusão, difração e sombreamento acústico (referente à visibilidade de fontes virtuais) poderiam ser considerados, às custas de um aumento considerável tanto de complexidade no modelo matemático como de custo computacional. Embora seja esperado que estas considerações tornem o modelo mais robusto e condizente com a realidade, no contexto do projeto automático de salas tais inclusões devem ser avaliadas em relação à melhoria real da solução obtida pelo método de otimização e o acréscimo de custo computacional envolvido.

Referências

- AcMus (2005). <http://gsd.ime.usp.br/acmus/projeto.html>.
- Beranek, L. L. (1993). *Acoustics*. New York: Acoustical Society of America.
- Coope, I. D. and Price, C. J. (2001). On the convergence of grid-based methods for unconstrained optimization. In *SIAM J. Optim.* 11(4), pages 859–869.
- D’Antonio, P. and Cox, T. J. (1997). Room optimizer: A computer program to optimize the placement of listener, loudspeakers, acoustical surface treatment and room dimensions in critical listening rooms. In *103rd Convention of the Audio Engineering Society*, pages Preprint 4555, Paper H-6. New York.
- F. H. Iazzetta, F. K. and Silva, F. (2001). Acmus: Design and simulation of musical listening environments. In *Proceedings of the 8th Brazilian Symposium on Computer Music*. Fortaleza.
- Horst, R. and Pardalos, P. M. (1995). *Handbook of Global Optimization*. Kluwer Academic Publishers.
- López, M. R. (2001). *Acondicionamiento Acústico*. Paraninfo.
- Moore, F. R. (1990). *Elements of Computer Music*. Prentice-Hall.
- Morrison, N. (1994). *Introduction to Fourier Analysis*. Wiley-Interscience.
- Queiroz, M. (2003). Some optimization models for listening room design. In *Proceedings of the 9th Brazilian Symposium on Computer Music*. Campinas.
- Rindel, J. H. (1997). Computer simulation techniques for the acoustical design of rooms - how to treat reflections in sound field simulation. In *Proceedings of ASVA '97*, pages 201–208. Tokyo.
- Rindel, J. H. (2000). The use of computer modeling in room acoustics. In *Journal of Vibroengineering*, pages 41–72.
- Rinnoykan, A. H. G. and Timmer, G. T. (1987). Stochastic global optimization methods. part i: clustering methods. In *Mathematical Programming* 39, pages 27–56.
- Warusfel, O. (1995). Predictive acoustics software and computer aided optimization in room acoustics. In *15th Intl. Congress on Acoustics*, pages 693–696. Trondheim, Norway.

Tararira: Sistema de búsqueda de música por melodía cantada

Ernesto López , Martín Rocamora*

Instituto de Ingeniería Eléctrica – Facultad de Ingeniería de la Universidad de la República
Julio Herrera y Reissig 565 – (598) (2) 711 09 74, Montevideo, Uruguay

elopez@fing.edu.uy, rocamora@fing.edu.uy

Abstract. *The problem of music retrieval by sung query consists of building a machine capable of simulating the cognitive process of identifying a musical piece from a few sung notes of its melody. In this paper, the algorithms of pitch tracking, onset detection and melody matching used in the system Tararira are described. Much effort has been put on automatic transcription of singing voice as it is a key factor in the overall performance. A novel way of combining note by note matching with a recent approach based on pitch time series matching is introduced.*

Resumen. *El problema de búsqueda de música por tarareo consiste en construir un sistema capaz de simular el proceso cognitivo de identificar una pieza musical a partir de unas pocas notas cantadas de su melodía. En este artículo se describen los algoritmos de detección de altura, segmentación de audio en notas y comparación de melodías utilizados en el sistema Tararira. Se concentran esfuerzos en la transcripción automática de la voz cantada ya que es determinante en el desempeño del sistema. Para la comparación de melodías se propone una forma de combinar los enfoques basados en notas y series temporales, considerados antagónicos hasta el momento.*

1. Introducción

El desarrollo tecnológico de los sistemas de almacenamiento y reproducción de audio permiten disponer de grandes colecciones de música, incluso en dispositivos muy pequeños. En este escenario se plantean nuevas dificultades. Al aumentar la cantidad de material disponible, crece la dificultad de organizar y buscar esta información. Asimismo, el tamaño de algunos dispositivos (por ejemplo, los reproductores portátiles de audio comprimido) impone una interfaz con el usuario muy reducida que resulta limitada para acceder a la gran cantidad de datos que almacenan. Se hace necesario entonces, el desarrollo de nuevas formas de interacción hombre-máquina para el acceso práctico y eficiente a bases de datos de música. En los últimos años, la búsqueda y recuperación de música se ha convertido en un campo muy activo de investigación.

En la música occidental, una de las características mas representativas y recordables es la melodía. La mayoría de las personas son capaces de identificar con facilidad una pieza musical a partir de un fragmento de melodía. Surge así el interés por desarrollar sistemas de búsqueda de música por contenido que permitan el acceso por melodía usando la voz cantada para formular la consulta. Estos sistemas reciben el nombre de sistemas de búsqueda de música por tarareo (QBH, Query By Humming). Sin embargo, simular de forma automática la habilidad de los humanos de reconocer melodías (así como otros procesos cognitivos) es una tarea desafiante.

* Ambos financiados por la Comisión Sectorial de Investigación Científica (CSIC).

Los sistemas de búsqueda de música por tarareo buscan identificar la melodía cantada por el usuario en una base de datos de melodías. Desde el sistema propuesto en [Ghias et al., 1995], a lo largo de la última década se han considerado distintos enfoques para enfrentar este problema. En todas las propuestas, tanto por razones prácticas como técnicas, la base de datos se compone de música codificada en alguna notación simbólica, generalmente MIDI, en lugar de audio crudo (wav, aiff) o comprimido (ogg, mp3). La razón técnica más importante es que no existen formas automáticas de extraer la melodía de una grabación para compararla con la melodía cantada por el usuario. La principal razón práctica es la de reducir la información para un menor costo de procesamiento.

Los sistemas propuestos pueden dividirse respecto a la representación y técnica de búsqueda básicamente en dos enfoques. El enfoque tradicional es el basado en comparación de notas (Event Based Search) [Ghias et al., 1995] [McNab et al., 1996], mientras que un enfoque más reciente utiliza la comparación de series temporales de frecuencia fundamental (Frame Based Search) [Mazzoni y Dannenberg, 2001] [Dannenberg y Hu, 2002] [Shasha y Zhu, 2003]. El primer enfoque se fundamenta en que una forma natural de comparar melodías es a través de las notas que las componen y consiste en transcribir la señal de voz a una secuencia de notas y buscar las mejores ocurrencias de ese patrón en una base de datos de melodías. Debido a que los errores en la transcripción automática deterioran el desempeño del sistema, el otro enfoque busca evitarla comparando melodías a través de series temporales de frecuencia fundamental (F0). Desafortunadamente, al trabajar con secuencias largas (mucho más que las secuencias de notas) el tiempo de procesamiento requerido se torna intolerable. Adicionalmente, es necesario imponer que el usuario cante un fragmento de la melodía previamente definido [Dannenberg y Hu, 2002] [Shasha y Zhu, 2003]. En este trabajo se propone una forma novedosa de combinar ambos enfoques aprovechando las ventajas de cada uno.

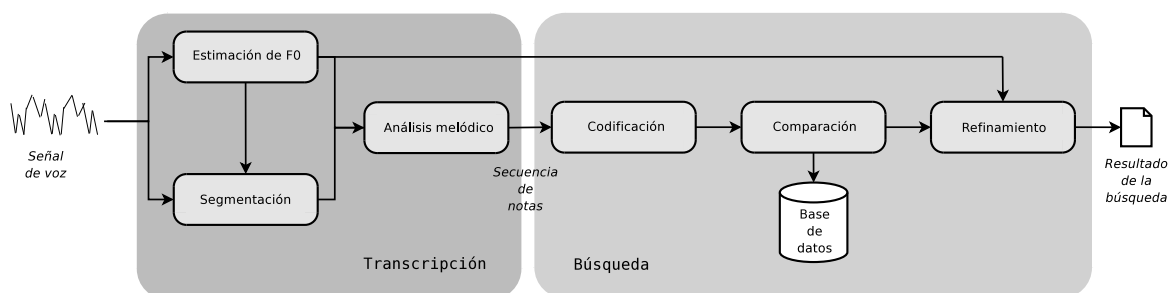


Figura 1: Diagrama de bloques del sistema.

El sistema selecciona inicialmente un grupo reducido de elementos de la base de datos a través de la comparación de secuencias de notas. Luego se refina la selección mediante la comparación de series temporales de F0. La arquitectura del sistema entonces consta básicamente de dos etapas, como se muestra en la figura 1. La primera consiste en la transcripción de la consulta a una secuencia de notas. En la siguiente etapa se compara esta secuencia con las melodías almacenadas en la base de datos y se devuelve una lista de piezas musicales ordenadas según su similitud con la consulta. La etapa de transcripción involucra las siguientes tareas:

- Estimar el contorno de F0 de la voz para determinar la altura de las notas.
- Segmentar la señal para establecer el tiempo de comienzo y fin de cada nota.
- Realizar un análisis melódico para ajustar la altura de las notas a la escala temperada.

Las tareas que constituyen la etapa de búsqueda son:

- Codificar la secuencia de notas para independizar la comparación de melodías del tempo y altura.
- Establecer criterios de similitud flexibles en la comparación para contemplar adornos y errores en la consulta, así como errores en la transcripción automática.
- Refinar la selección de candidatos comparando series temporales de frecuencia fundamental, de forma de eludir los errores en la transcripción automática.

2. Transcripción de voz cantada

Dada la forma de onda digitalizada de una señal de audio producida por la voz cantada, el objetivo de la transcripción automática es extraer la secuencia de notas que mejor representa la melodía cantada. Para ello, se identifican en la señal de audio los eventos con mayor probabilidad de corresponder a notas. Cada evento se caracteriza por tres valores: altura, tiempo de inicio y duración. Otro tipo de información, como características expresivas (intensidad, vibrato), no es de interés debido a que no forma parte de la notación simbólica tradicional.

En este trabajo se hace hincapié en la transcripción de la voz cantada ya que es un problema para el cual no existe aún una solución completamente satisfactoria. La voz cantada es uno de los instrumentos musicales más difíciles de tratar. Las grandes variaciones tímbricas, los recursos expresivos, la microentonación, la entonación inexacta y errores de altura y duración son algunas de las características que dificultan su análisis.

2.1. Detección de altura

La altura de un sonido es un concepto subjetivo determinado a partir de la percepción, si bien se relaciona en general con la frecuencia fundamental de la onda sonora. Esto tiene excepciones, como señales no periódicas que producen una sensación de altura (ruido de banda limitada, sonidos percutivos). Por esta razón, para establecer la altura de las notas cantadas se debe estimar la evolución de la frecuencia fundamental (F0) de la señal de voz, para lo cual existen técnicas bien conocidas. La frecuencia fundamental de la voz cantada usualmente varía entre 60 y 1000 Hz¹.

El algoritmo implementado utiliza la función diferencia normalizada [de Cheveigné y Kawahara, 2002], una variante de la función de autocorrelación que presenta algunas ventajas. La función diferencia consiste en la diferencia entre la señal y una versión retardada de la misma, cuya variable es el retardo,

$$d(\tau) = \sum_{j=1}^W (x_j - x_{j+\tau})^2 \quad (1)$$

donde W corresponde al largo del bloque de análisis. Si la señal es periódica, esta función se anula para valores del retardo múltiplos del período, mientras que para señales que no son perfectamente periódicas (como los tramos sonoros de la voz), no se anula pero presenta mínimos cercanos a cero. El valor del retardo correspondiente al primer mínimo coincide con el período de la señal. Por otro lado, los mínimos de la función diferencia de una señal no periódica (tramo de voz sordo), no son cercanos a cero. Es posible discriminar entre tramos sonoros y tramos sordos a partir del valor del mínimo de la función diferencia. Esto puede hacerse estableciendo un umbral de decisión, para lo cual es necesario normalizar la función diferencia. La normalización adoptada consiste en dividir los valores de la función entre la media acumulada,

$$d_{norm}(\tau) = \begin{cases} 1, & \text{si } \tau = 0 \\ \frac{d(\tau)}{\frac{1}{\tau} \sum_{j=1}^{\tau} d(j)}, & \text{en otro caso} \end{cases} \quad (2)$$

¹Sin embargo una cantante soprano puede cantar frecuencias fundamentales de hasta 1400 Hz.

La discriminación entre tramos sonoros y sordos colabora a la segmentación de la señal en notas, además de evitar la estimación de F_0 para tramos de señal que no presentan periodicidad (en los tramos sordos y silencios, se establece $F_0 = 0$).

El algoritmo calcula la función diferencia normalizada cada 10 ms para bloques solapados de señal y estima el valor de F_0 detectando el primer mínimo. Debido a que se trabaja con señales muestreadas a 8 kHz, el error en la estimación es mayor al semitono para cierto rango de frecuencias. Para aumentar la resolución de la estimación se aproxima por una parábola cada valle de la función diferencia donde se presenta un mínimo. Los errores típicos de un algoritmo de estimación de F_0 en el dominio del tiempo corresponden a la detección de un subarmónico de la frecuencia fundamental. Para evitar estos errores se suma una recta de pendiente positiva a la función diferencia, favoreciendo la elección del mínimo de menor orden. La aproximación por parábolas también colabora a la elección del mínimo correcto.

2.2. Segmentación en notas

Una vez obtenido el contorno de F_0 de la señal de voz para establecer la altura de las notas, debe determinarse su tiempo de inicio y duración. Este problema se denomina segmentación automática de audio en notas y constituye la etapa más difícil de la transcripción automática. La voz cantada contiene un conjunto de rasgos que hacen que los límites entre notas sean a veces difusos y por lo tanto difíciles de resolver. Es importante entonces poder discriminar comienzos genuinos de cambios graduales y modulaciones que tienen lugar durante el transcurso de una nota. Es posible distinguir distintos tipos de comienzo de nota en una señal de voz. En el caso de notas cantadas con sílabas que comienzan con fonemas oclusivos (por ejemplo, /ta/), la repentina liberación de energía posterior a la restricción del pasaje de aire produce inicios marcados. El inicio de estas notas resulta sencillo de determinar dado el gran cambio de energía presente en la señal. Cuando la nota comienza con un aumento gradual de energía (por ejemplo, una consonante nasal), el inicio es más suave y por lo tanto más difícil de establecer. En señales de voz cantada existen inicios suaves e inicios marcados por lo que un algoritmo de segmentación automática debe manejar adecuadamente ambos casos. Desafortunadamente, no ha sido desarrollado hasta el momento un sistema capaz de detectar el amplio espectro de clases de inicio de nota existentes en las distintas formas de cantar. El peor escenario lo constituye la voz cantada con letra. En este caso el contorno de frecuencia fundamental y la forma de onda de la señal de voz presentan una infinidad de particularidades difíciles de caracterizar que entorpecen el análisis [Pollastri y Haus, 2001]. Esto es especialmente notorio en cantantes no experimentados.

El algoritmo de segmentación implementado, para lograr una detección robusta, busca indicios de eventos tanto en la envolvente de amplitud de la señal como en el contorno de F_0 . En una primera etapa se detectan eventos asociados a cambios de energía. Los eventos de mayor intensidad se asumen como comienzos de nota genuinos. En la segunda etapa se validan los eventos de menor intensidad si están acompañados de un cambio de altura. Finalmente se identifican los inicios de nota asociados a cambios evidentes de altura que no presentan un incremento de energía (por ejemplo, ligados).

Detección de eventos por cambios de energía El incremento de energía en la señal de audio cuando se produce un nuevo evento se manifiesta en un aumento de amplitud de la envolvente de la forma de onda. Tomando la derivada de la envolvente es posible construir una función de detección que presente picos donde hay cambios bruscos de amplitud [Dixon, 2001] [Schloss, 1985]. Esto funciona adecuadamente solo para casos particulares, como sonidos percutivos. Para segmentar señales más complejas, se propone trabajar con envolventes de amplitud de distintas bandas de frecuencia [Klapuri, 1999].

En [Scheirer, 1998] se señala que es posible mantener la información rítmica de una señal de audio solo a través de envolventes de amplitud de distintas bandas de frecuencia. Este mismo enfoque puede aplicarse a la segmentación de una señal de audio, dividiéndola en bandas de frecuencia y detectando eventos en las envolventes de amplitud de cada banda. En general, los eventos aparecen más claramente en alguna de las bandas que en la envolvente de la señal (ver figura 2, izquierda). Por ejemplo, cuando un nuevo evento está asociado a cambios de altura o cambios tímbricos, puede producir la aparición repentina de información en alguna banda de frecuencia.

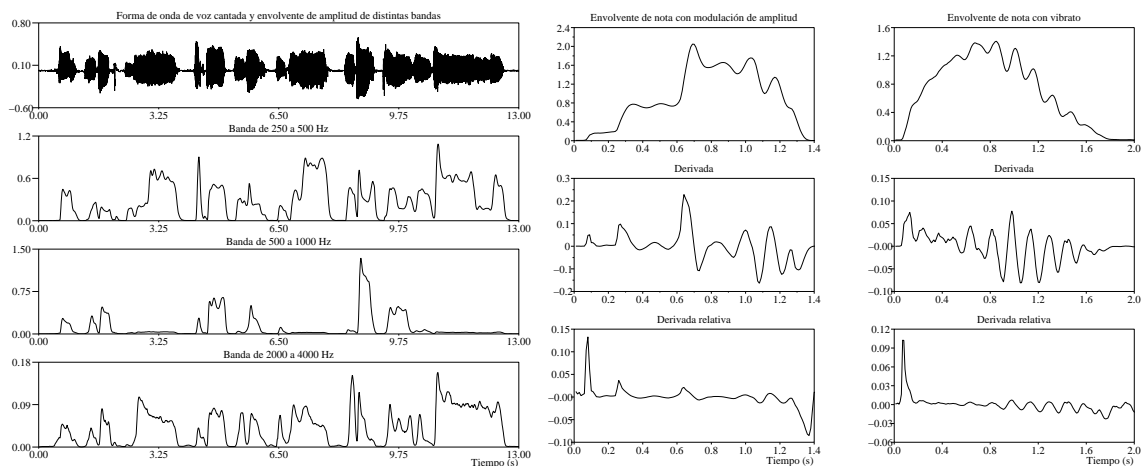


Figura 2: Envolventes de amplitud de distintas bandas de frecuencia (izquierda). Los eventos son más evidentes en alguna banda que en la señal original. Envolvente de amplitud, derivada y derivada relativa (derecha). El máximo de la derivada relativa coincide con el inicio físico de las notas.

En el algoritmo implementado la señal de audio se divide en bandas de frecuencia, se extrae la envolvente de cada banda y se obtiene su derivada y derivada relativa. Estas señales son utilizadas para determinar candidatos de comienzo de nota, a los cuales se les asigna un valor de intensidad. Finalmente se combina la información de las distintas bandas y se seleccionan los candidatos más prominentes. A continuación se describen cada una de las etapas.

La señal de audio es normalizada en amplitud y filtrada en 6 bandas de octava por un banco de filtros de frecuencias de corte 125, 250, 500, 1000 y 2000 Hz. Las señales de cada banda se rectifican onda completa y se deciman a una frecuencia de muestreo de 100 Hz. Las envolventes se obtienen convolucionando cada señal con una ventana mitad Hanning (coseno elevado) de 100 ms, que produce una integración de potencia que mantiene cambios repentinos pero enmascara modulaciones rápidas.

Tradicionalmente los algoritmos que trabajan con envolventes de amplitud calculan su derivada de primer orden y asocian los puntos de máxima pendiente al comienzo de notas. Sin embargo, la derivada de primer orden es apropiada para reflejar la sonoridad de las notas, pero sus máximos no indican adecuadamente el instante de su inicio. Esto se debe principalmente a que la envolvente correspondiente al inicio de un sonido no es monótonamente creciente en la mayoría de los casos, dando lugar a varios máximos locales en la derivada de primer orden. Asimismo, muchos sonidos (en particular los graves) tardan cierto tiempo en alcanzar el punto donde su amplitud crece con pendiente máxima, y éste es posterior al inicio físico del sonido. Por esta razón se propone utilizar la derivada relativa $D_r(t) = \frac{\frac{d}{dt}A(t)}{A(t)}$, donde $A(t)$ es la envolvente de amplitud [Klapuri, 1999]. De esta forma, las oscilaciones de amplitud que ocurran durante el inicio de una nota serán atenuadas. Por otra parte el máximo de la derivada relativa se ubica antes que el máximo de la derivada, coincidiendo con el inicio físico de la nota (ver figura 2, derecha). El uso de

la derivada relativa tiene sentido desde el punto de vista psicoacústico, ya que el incremento de intensidad percibido por el sistema auditivo está relacionado con la amplitud del sonido, siendo un mismo incremento de amplitud más relevante para sonidos de menor amplitud². En el algoritmo implementado la derivada relativa de la envolvente se calcula como la derivada de la envolvente comprimida según la ley μ , con un valor de $\mu = 100$.

La selección de candidatos de comienzo de nota se realiza detectando los picos de la derivada relativa de cada banda que superan cierto umbral. El valor de intensidad se establece como el primer máximo de la derivada de amplitud a partir del instante del pico de la derivada relativa. Para evitar la detección de inicios duplicados debido a variaciones de la pendiente durante el ataque se eliminan candidatos de baja intensidad cercanos a otro más prominente. De esta forma se toma en cuenta por ejemplo, que no es posible un cambio instantáneo de fortissimo a pianissimo.

La información de las distintas bandas se combina asumiendo que candidatos ubicados a menos de 50 ms corresponden al mismo evento. La elección definitiva de los candidatos se realiza usando un umbral fijo y un umbral dinámico similar al anterior. Debido a que los candidatos de menor intensidad podrían corresponder a modulaciones de amplitud provenientes de recursos expresivos o respiración, el algoritmo clasifica los candidatos según su intensidad y devuelve un grupo de eventos fuertes y un grupo de eventos débiles. Los primeros se consideran inicios de nota genuinos, mientras que los más débiles se validan en una etapa posterior.

Validación de eventos débiles por cambios de altura La mayoría de los eventos débiles provienen de modulaciones de amplitud que tienen lugar durante el transcurso de una nota, mientras que otros se deben a transiciones suaves entre notas. Un evento débil se valida como inicio de nota si va acompañado de un cambio de altura significativo (ver figura 3, izquierda).

Para validar el evento débil se obtiene la mediana del contorno de frecuencia durante cierto intervalo de tiempo antes y después del mismo. Luego se comparan estos valores determinando si su diferencia supera un umbral establecido a partir del intervalo de semitono (6 %). El umbral utilizado es menor al semitono ya que el contorno de F0 presenta transiciones suaves entre notas sucesivas.

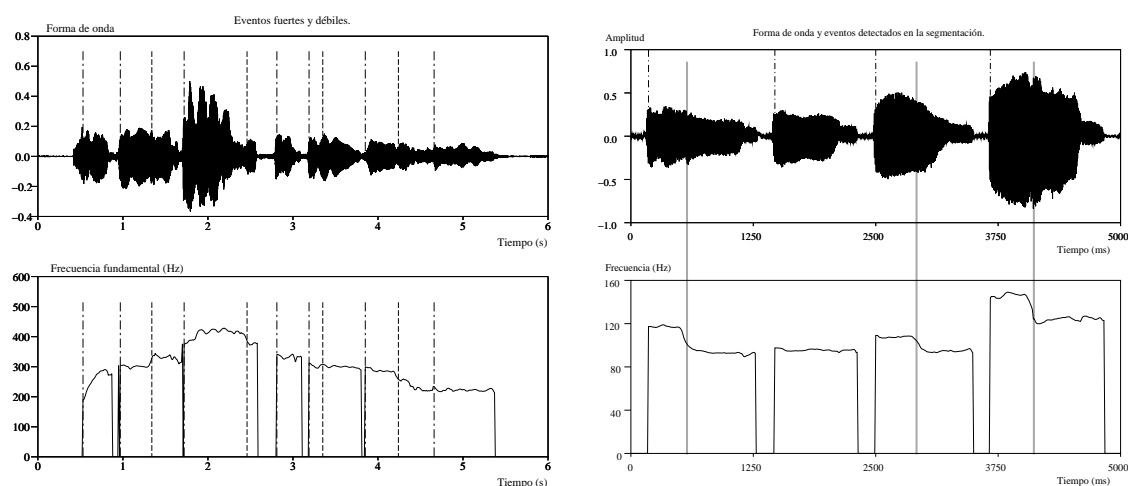


Figura 3: Se indican los eventos fuertes (raya y punto) y los eventos débiles (línea punteada) de la segmentación (izquierda). Los eventos débiles asociados a un cambio de altura se confirman. Las notas ligadas no se pueden deducir a partir de la forma de onda (derecha).

²Si ΔI es el incremento de intensidad mínimo perceptible, la relación $\Delta I/I$ (fracción de Weber) es constante.

Detección de eventos por cambios de altura La segmentación basada en cambios de energía no es capaz de detectar el inicio de algunas notas. Un ejemplo son las notas ligadas, en donde existe un cambio de altura que no va acompañado de un aumento de amplitud (ver figura 3, derecha). Otro caso son las notas de ataque muy suave, por ejemplo con fonemas sonoros nasales.

Las notas omitidas, en muchos casos, pueden ser agregadas observando los cambios de altura en el contorno de F_0 . Esto sin embargo no es una tarea sencilla debido a que la expresividad en la interpretación y la falta de entrenamiento en cantantes inexpertos introducen un conjunto de rasgos en el contorno de F_0 [Pollastri, 2003] que pueden ser considerados por error como notas adicionales (transiciones suaves, picos, inestabilidades, vibrato).

A partir del contorno de F_0 y de los eventos detectados hasta el momento se construye un conjunto de alturas y tiempos de inicio y fin de nota. El tiempo de fin está dado por el inicio de una nueva nota o cuando se anula el contorno de F_0 . La altura se calcula como la mediana del contorno de F_0 dentro del intervalo. Se procesa cada intervalo identificando tramos del contorno que se desvían más de un semitono del valor de altura asignado. Si el tramo cumple ciertas condiciones de estabilidad y duración mínima es considerado una nota.

2.3. Ajuste a la escala temperada

Para asignar una altura a cada nota es necesario en primer lugar aproximar el contorno de F_0 de la nota a un único valor de frecuencia y luego asociar este valor a una altura de un sistema de afinación (por ejemplo, escala temperada con $A_4 = 440$ Hz).

Como ya se mencionó, el contorno de F_0 de una nota puede ser muy variable y resulta difícil establecer un criterio para asignarle un único valor de frecuencia. El valor de frecuencia que mejor representa la altura de la nota es aquel que toma el contorno cuando alcanza la estabilidad, lo que implica ignorar las transiciones suaves, inestabilidades, picos espurios. En este trabajo se adopta un criterio sencillo y efectivo: el valor de frecuencia de la nota es la mediana del contorno. Típicamente la región de transición y los picos espurios son de corta duración respecto a la zona de estabilidad, por lo que la mediana generalmente se aproxima al valor de estabilidad.

La música de diversas culturas utiliza distintos sistemas de afinación. Una hipótesis necesaria para la transcripción de una melodía es el sistema de afinación a usar. Este trabajo se concentra en música occidental, por lo que la transcripción se hará a la escala temperada. Sin embargo, las personas al cantar no tienen la capacidad de ajustarse a un sistema de afinación sin escuchar una referencia, salvo raras excepciones (oído absoluto). La interpretación entonces no respeta exactamente los intervalos ni la referencia de la escala temperada. Se hace necesario corregir el desajuste natural entre la melodía cantada y el sistema de afinación. Una alternativa es asignar la nota mas cercana de la escala temperada, pero de esta forma las notas pueden ajustarse en diferentes sentidos distorsionando los intervalos cantados.

Se han propuesto métodos mas apropiados de ajuste a la escala temperada [McNab et al., 1996] [Pollastri y Haus, 2001] [Viitaniemi et al., 2003]. Los resultados de evaluar las distintas técnicas indican que el método más apropiado es el propuesto por Pollastri. Este se basa en la hipótesis de que al cantar se mantiene un tono de referencia en mente y las notas cantadas pertenecen a una escala temperada referida a este tono. El método consiste en determinar la desviación más frecuente para estimar el tono de referencia y así ajustar las notas cantadas a la escala temperada absoluta. Para ello se divide el semitono en 10 intervalos iguales de 0.2 semitonos solapados 0.1 semitono, se calcula

la desviación de cada nota como la parte fraccionaria de la nota MIDI equivalente³ y se asocia al intervalo correspondiente. La desviación más frecuente corresponde a la media de las desviaciones del intervalo con más notas. A cada nota se le resta esta desviación y se redondea a la nota MIDI más cercana. Al estimar el tono de referencia a partir de la desviación más frecuente se considera que además de la desviación respecto a la escala absoluta, en algunos intervalos puede existir una desviación adicional pequeña que se relaciona con la dificultad de cantarlos.

3. Comparación de melodías

Los principales desafíos que presenta la etapa de búsqueda son la codificación de la información y los criterios de similitud. La melodía es información de más alto nivel que la secuencia de notas específica que la compone. Una melodía puede identificarse a pesar de que se interprete en diferentes alturas, a distinto tempo (dentro de límites razonables) y con adornos o rasgos expresivos. Estas consideraciones son esenciales en el diseño de la etapa de búsqueda. La independencia respecto a la altura y al tempo puede lograrse en la codificación de las notas (codificación invariante a la transposición y al tempo). Mediante criterios de similitud flexibles durante la búsqueda es posible establecer tolerancia a las alteraciones provenientes de adornos, así como a errores en la interpretación y en la transcripción automática.

Como ya se mencionó, existen básicamente dos enfoques para la comparación de melodías: la comparación de notas y la comparación de series temporales de frecuencia. Ambos enfoques presentan sus desventajas. El primero requiere la transcripción automática de la consulta, lo que inevitablemente introduce errores que deterioran el desempeño del sistema. El segundo, si bien evita la transcripción, involucra un costo computacional alto y permite buscar solo fragmentos de melodía definidos previamente.

Usualmente los sistemas de comparación de melodías retornan una larga lista como resultado de la búsqueda. En el sistema desarrollado se busca retornar únicamente la pieza musical buscada. Por esta razón, se aumenta la eficacia del sistema agregando a la búsqueda por comparación de notas una etapa de refinamiento basada en comparación de series temporales de F0.

3.1. Comparación basada en secuencias de notas

Al trabajar con secuencias de notas, la búsqueda de melodías es básicamente un problema de búsqueda aproximada de cadena de caracteres (Approximate String Matching). Es necesario derivar de la secuencia de notas una codificación que sea invariante tanto a la transposición de altura como al tempo. La cuantización de los intervalos de altura y duración permite ajustar la tolerancia a los errores en la consulta. Asimismo, cuantizar los intervalos a un alfabeto discreto es necesario para utilizar técnicas de búsqueda de cadena de caracteres.

Codificación La secuencia de notas que devuelve la transcripción automática se representa por una secuencia de alturas y una secuencia de tiempos de inicio y duración. La invarianza a la transposición de altura se obtiene codificando la secuencia de alturas $A = (a_1, a_2, \dots, a_n)$ en la secuencia de intervalos $\bar{A} = (a_2 - a_1, a_3 - a_2, \dots, a_n - a_{n-1})$. Resulta evidente que una secuencia A' transposición de A , es decir $a'_i = a_i + c$, tiene la misma representación en intervalos. Una forma de lograr invarianza al tempo es normalizar las duraciones respecto a una duración de referencia d_{ref} invariante al tempo.

³La nota MIDI equivalente se obtiene a partir de la frecuencia como $n_{MIDI} = 69 + 12 \frac{\log(\frac{f}{440})}{\log(2)}$.

Un valor apropiado es el período de pulso, como en la notación musical. Lamentablemente, en el caso de una melodía cantada, no siempre es posible estimar el pulso, por lo que este criterio no es aplicable. Sin conocer el tempo, no existe una normalización de las duraciones completamente aceptable. Una alternativa sencilla utilizada frecuentemente es considerar d_{ref} como la duración de la nota previa [Pardo y Birmingham, 2002]. Dada la secuencia de duraciones $D = (d_1, d_2, \dots, d_n)$, la representación invariante al tempo utilizada es la secuencia de duraciones relativas $\bar{D} = (\frac{d_2}{d_1}, \frac{d_3}{d_2}, \dots, \frac{d_n}{d_{n-1}})$. Esta secuencia se cuantiza a un alfabeto discreto de enteros por medio de la función logarítmica $r(i) = \text{round} \left(10 \log_{10} \left(\frac{d_i}{d_{i-1}} \right) \right)$ [Pollastri, 2003]. El logaritmo suaviza la relación de duraciones de forma de atenuar las grandes aproximaciones de duración que se cometen al cantar despreocupadamente. Un error común de los cantantes inexpertos es finalizar prematuramente las notas, por lo que se considera el intervalo entre inicios de nota sucesivos (IOI, Inter Onset Interval) como una representación más consistente de las duraciones. La secuencia de intervalos de altura no se cuantiza (codificación en intervalos exactos), pero en la etapa de comparación se utilizan criterios flexibles.

Comparación La etapa de comparación consiste en encontrar buenas ocurrencias de la secuencia de notas codificada en la base de datos. Para ello es necesario definir una medida de distancia. Se ha demostrado que la Distancia de Edición (Edit Distance) es la mejor medida de similitud para la comparación de melodías [Uitdenborgerd y Zobel, 1999]. La distancia de edición consiste en el número mínimo de alteraciones necesarias para transformar una cadena de caracteres en otra y se determina a través del algoritmo conocido como Programación Dinámica (DP, Dynamic Programming) [Lemström, 2000].

El algoritmo de comparación calcula la distancia de edición combinando la información de duración y altura. Al realizar la combinación se prioriza la información de altura ya que permite mayor discriminación que la duración. Adicionalmente la información de duración es menos confiable debido a las grandes aproximaciones que se cometen al cantar y a que la secuencia de duraciones es más sensible a los errores de la transcripción automática⁴.

Sean a_i^a y b_j^a los intervalos exactos de altura y a_i^d y b_j^d los intervalos de duración codificados de las secuencias a comparar. La distancia de edición se calcula llenando recursivamente una matriz D_{DP} en la que cada elemento d_{ij} se obtiene como,

$$d_{ij} = \min \begin{cases} d_{i-1,j} + i \\ d_{i,j-1} + o \\ d_{i-1,j-1} + s \\ d_{i-1,j-1} + c & |a_i^a - b_j^a| < 2 \text{ y } |a_i^d - b_j^d| < 2 \\ d_{i-1,j-1} + s_d & |a_i^a - b_j^a| < 2 \end{cases}$$

Costo

i	o	s	c	s_d
1	1	1	-1	0

donde i es el costo de una inserción, o el de una omisión, s el de una sustitución, c el de una coincidencia de duración y altura y s_d el de una sustitución de duración. Cabe señalar que a pesar de usar intervalos exactos de altura, dos notas que difieren en un semitono se consideran iguales en el cálculo de la distancia de edición. Se introduce también tolerancia en los intervalos de duración.

A partir de la distancia de edición se define una medida de similitud S que toma valores entre 0 y 100 como $S = 100 \frac{(m-1)-E}{2(m-1)}$, con E la distancia de edición y m el número de notas de la consulta. Se comparan todos los elementos de la base de datos con la consulta codificada y se selecciona un conjunto de las mejores ocurrencias en función del valor de similitud.

⁴Por ejemplo, si no se segmentan dos notas consecutivas de la misma altura, se afecta solo un intervalo de altura y tres intervalos de duración.

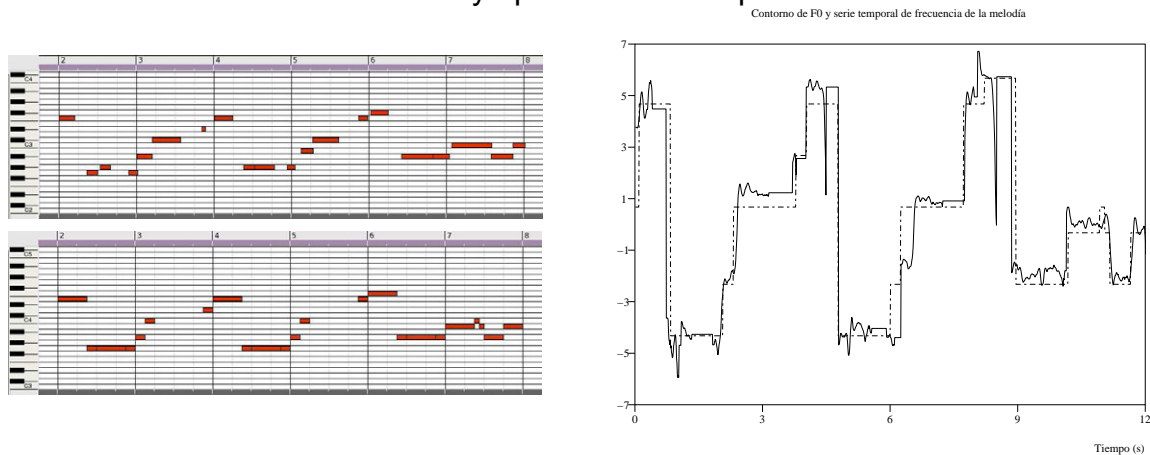


Figura 4: Transcripción de la consulta y ocurrencia en la base de datos (izquierda) y las correspondientes series temporales normalizadas y alineadas por el sistema (derecha).

3.2. Comparación basada en series temporales

Como alternativa al enfoque basado en comparación de notas, recientemente se estudia la posibilidad de identificar melodías directamente a partir de características derivadas de la señal de voz. Este enfoque consiste en comparar el contorno de F0 de la consulta con melodías codificadas como series temporales de frecuencia. Intuitivamente la forma de comparar dos series temporales de distinto largo es, en primer lugar, ajustarlas al mismo largo y luego compararlas punto a punto permitiendo cierta deformación temporal. La técnica de Deformación Temporal Dinámica Local (LDTW, Local Dynamic Time Warping) permite comparar series temporales de esta forma.

Además del alto costo computacional, una restricción del uso de LDTW es que algún elemento de la base de datos debe corresponder exactamente a la consulta, ya que no se puede buscar subsecuencias dentro de secuencias manteniendo invarianza a la trasposición de altura y al tempo [Dannenberg y Hu, 2002] [Shasha y Zhu, 2003]. Por esta razón el proceso de construcción de la base de datos es complejo ya que es necesario identificar dentro de la melodía original los fragmentos más probables de ser cantados.

En el sistema implementado se seleccionan las mejores ocurrencias del patrón buscado a través de la comparación de notas. Durante este proceso se identifican los tramos que se asemejan a la consulta dentro de las melodías seleccionadas. Luego se construyen series temporales de frecuencia de esos tramos que se comparan con el contorno de F0 de la consulta. De esta forma se aplica LDTW sobre un conjunto reducido de candidatos sin imponer restricciones sobre la consulta (ver figura 4).

El primer paso para aplicar LDTW es ubicar el comienzo y fin de las mejores ocurrencias del patrón en las melodías de la base de datos. Esto corresponde a encontrar el alineamiento entre las secuencias, lo que puede hacerse a través del camino mínimo⁵ de la matriz D_{DP} . Luego se normalizan las duraciones de las ocurrencias para igualarlas a la de la consulta. Esto es necesario debido a que para aplicar LDTW se requiere que las series a comparar sean del mismo largo. En esta etapa se descartan candidatos de tiempo excesivamente distinto al de la consulta. A partir de la secuencia de notas de la ocurrencia normalizada en el tiempo se construye una serie temporal de valores de altura. El siguiente paso es llevar las series a forma normal para poder compararlas, es decir transformarlas en series de varianza uno y media nula. Finalmente se calcula la distancia LDTW para cada una de las ocurrencias con un factor de deformación máximo de un segundo.

⁵El camino mínimo es el camino de menor costo.

El sistema devuelve un único elemento de la base de datos como resultado de la consulta si es capaz de diferenciarlo del conjunto de los candidatos en función de los valores de similitud S y distancia LDTW. En caso contrario se retorna la lista de candidatos ordenada según la distancia LDTW.

4. Evaluación

El sistema desarrollado denominado Tararira fue implementado en C++⁶ y se construyó una base de datos de melodías MIDI monofónicas con la colección completa de The Beatles (208 temas). Se condujo una evaluación en la que participaron más de 30 personas sin entrenamiento musical. En la tabla de la figura 5 se presentan los resultados. Si bien el tamaño de la base de datos es acotado, el desempeño obtenido es alentador.

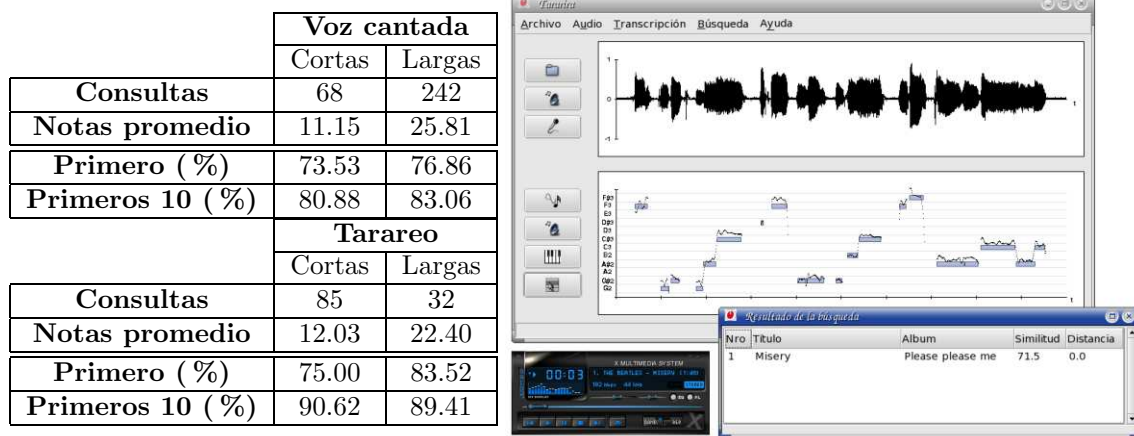


Figura 5: Resultados de la evaluación y aspecto de la aplicación.

La evaluación confirma algunas hipótesis sobre el problema. Las características de la voz cantada con letra dificultan su transcripción lo que se refleja en los resultados. Se confirma que al aumentar el largo de la consulta mejora el desempeño, ya que la melodía se torna más identificable.

5. Conclusiones

Un sistema ideal de búsqueda de música por melodía además de ser rápido y eficaz, debe tolerar errores en la consulta, no restringir al usuario en la forma de cantar y retornar únicamente la pieza buscada. Otra característica importante es que el agregado de piezas musicales a la base de datos sea sencillo. Si bien aún no existe un sistema que cumpla con estos requerimientos, Tararira fue diseñado tomándolos en cuenta.

A los efectos de contemplar las distintas formas de cantar se desarrolló un sistema de transcripción robusto, integrando y adaptando técnicas que representan el estado del arte. Se buscó perfeccionar la segmentación integrando de manera efectiva la información de energía y altura de la señal de voz.

Los enfoques de búsqueda de música basados en comparación de notas y comparación de series temporales son considerados antagónicos y no se conocen antecedentes de usarlos en forma conjunta. Con el objetivo de aumentar la capacidad de discriminación del sistema de búsqueda se combinó de forma novedosa ambos enfoques aprovechando las

⁶El programa compilado para Linux puede obtenerse de la página web <http://iie.fing.edu.uy/investigacion/grupos/gmm/proyectos/tararira/>

ventajas de cada uno. De esta forma, la construcción y extensión de la base de datos del sistema es relativamente sencilla ya que no es necesario extraer los fragmentos más representativos de la melodía, como en los sistemas de comparación de series temporales.

Se desarrolló un sistema de búsqueda completo que funciona correctamente. En trabajos futuros se abordará la validación exhaustiva del sistema y su extensión a aplicaciones de mayor escala.

Referencias

- Dannenberg, R. B. y Hu, N. (2002). A comparison of melodic database retrieval techniques using sung queries. *JCDL*, pages 301–307.
- de Cheveigné, A. y Kawahara, H. (2002). Yin, a fundamental frequency estimator for speech and music. *JASA*, 111:1917–1930.
- Dixon, S. (2001). Learning to detect onsets of acoustic piano tones. *Proc. of the Workshop on Current Directions in Computer Music Research*.
- Ghias, A., Logan, J., Chamberlin, D., y Smith, B. C. (1995). Query by humming: Musical information retrieval in an audio database. *Proc. ACM Multimedia*, pages 231–236.
- Klapuri, A. P. (1999). Sound onset detection by applying psychoacoustic knowledge. *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*.
- Lemström, K. (2000). *String Matching Techniques for Music Retrieval*. PhD thesis, Department of Computer Science, University of Helsinki.
- Mazzoni, D. y Dannenberg, R. B. (2001). Melody matching directly from audio. *Proc. of ISMIR*.
- McNab, R. J., Smith, L. A., Witten, I. H., Henderson, C. L., y Cunningham, S. J. (1996). Towards the digital music library: Tune retrieval from acoustic input. *Proc. of the ACM Digital Libraries*, pages 11–18.
- Pardo, B. y Birmingham, W. (2002). Encoding timing information for musical query matching. *ISMIR*, pages 267–268.
- Pollastri, E. (2003). *Processing Singing Voice for Music Retrieval*. PhD thesis, Università Degli Studi Di Milano.
- Pollastri, E. y Haus, G. (2001). An audio front end for query-by-humming systems. *Proc. of ISMIR*.
- Scheirer, E. D. (1998). Tempo and beat analysis of acoustic signals. *JASA*, pages 588–601.
- Schloss, W. A. (1985). *On the Automatic Transcription of Percussive Music: From Acoustic Signal to High Level Analysis*. PhD thesis, Stanford University, CCRMA.
- Shasha, D. y Zhu, Y. (2003). Warping indexes with envelope transforms for query by humming. *Proc. of the 2003 ACM SIGMOD Conference on Management of Data*, pages 181–192.
- Uitdenborgerd, A. y Zobel, J. (1999). Melodic matching techniques for large music databases. *Proc. of the ACM Multimedia*, pages 57–66.
- Viitaniemi, T., Klapuri, A., y Eronen, A. (2003). A probabilistic model for the estimation of single-voice melodies. *Finnish Signal Processing Symposium, Tampere University of Technology*.

Extracting Patterns from Guitar Accompaniment Data: Some Experimental Results

Ernesto Trajano de Lima¹, Søren Tjagvad Madsen^{2,3}, Márcio Dahia¹,
Gerhard Widmer^{2,3}, Geber Ramalho¹

¹Centro de Informática (CIn) – Universidade Federal de Pernambuco
Caixa Postal 7851– 50732-970 – Recife – Pernambuco – Brazil

²Austrian Research Institute for Artificial Intelligence (ÖFAI)
Freyung 6/6, A-1010, Vienna, Austria

³Department of Computational Perception – University of Linz, Austria

{etl,mlmd,glr}@cin.ufpe.br, soren@ofai.at, gerhard.widmer@jku.at

Abstract. *The analysis of expressive performance, an important research topic in Computer Music, is almost exclusively devoted to the study of western classical piano music. Instruments like the acoustic guitar and styles like Bossa Nova and Samba have almost never been studied, despite their harmonic and rhythmic richness. This paper describes some experimental results obtained with the extraction of rhythmic patterns from the guitar accompaniment of Bossa Nova songs. The performances, recorded with a MIDI guitar, were represented as strings and processed by three different string matchers. Results obtained showed that parts of a previously acquired catalogue of patterns could be found in the data set. Surprising results, such as very long patterns, also showed up.*

1. Introduction

It is common sense that playing music in the exact way as written in the score results in a mechanical and uninteresting succession of sounds. To make the written music interesting, the musician is required to make variations on various musical parameters, such as: local tempo (*accelerandi, ritardandi, rubato*); dynamics; notes articulation (*staccati*, ligatures, etc.); micro-silences between the notes, etc. [Widmer, 1998]. Several researchers stress the importance, as well as the difficulties, of studying this phenomenon, also known as *expressive performance* [Sundberg et al., 1991, Desain et al., 2001, Widmer et al., 2003].

Studying the expressive performance phenomenon usually implies discovering some sort of systematic relationship within a piece and/or among pieces. These relationships can be described in many ways, from different points of view or levels of abstraction, and including various musical parameters. Examples of such relationships are rules like “lengthen a note if it is followed by a longer note and if it is in a metrically weak position” or “stress a note by playing it louder if it is preceded by an upward melodic leap larger than a perfect fourth” (for more examples see [Widmer, 2002]).

The research on expressive performance today is almost exclusively devoted to the western classical music composed for the piano. We are interested in the study of the *Música Popular Brasileira* (Brazilian Popular Music)—MPB, represented by artists like João Gilberto, Tom Jobim, Caetano Veloso, Gilberto Gil, etc., in particular the guitar music. There is, however, a fundamental difference between these two genres: While in western classical music there is one “official” notated version of musical pieces (the

score), in MPB it does not exist. What is usually available is the description of the chord grid only, and, sometimes, the score for the melody. So, in this genre (MPB), the rhythmic accompaniment must be determined by the performer himself.

It is known that the guitar accompaniment in styles like *Bossa Nova* and *Samba* is built by the concatenation of certain rhythmical patterns [Garcia, 1999, Sandroni, 2001]. There are, however, several aspects of the accompaniment construction that are only known by practitioners of these styles. Moreover, the knowledge about the accompaniment construction is mainly subjective. Due to this lack of formalized knowledge, there are many open questions such as:

- Are there rhythmic patterns that are preferred by a certain performer or required for a certain musical style? In which situations and in which frequency do they show up?
- Are there variations of these patterns? Is it possible to group these variations in meaningful way? Which variations (timing, dynamics, etc.) are acceptable within a pattern?
- Is it really the case that everything is a pattern, i.e., are there parts that are not recurrent?
- Is it possible to justify the choice of a pattern in terms of other musical features (melody, harmony, tempo, musical structure, style, etc.)?
- Is it possible to build a dictionary of patterns for a given player? Does this dictionary changes when the style changes (*Bossa Nova* and *Samba*, for instance)? Do different players have different dictionaries?
- Is it possible to build a grammar or a set of rules that is able to describe formally how the patterns are chained and/or the rhythmical transformations done by a given performer? If so, what are the relations between grammars from performers p_1 and p_2 ?

This paper presents some experiments that deal with some of these questions, focusing on the discovery of rhythmic patterns in *Bossa Nova* music. For this, two different performers played several songs on a MIDI guitar, which were processed in the form of strings. Based on previous work (the acquisition of a catalogue of *Bossa Nova* patterns [Dahia et al., 2004]), we tried to identify how much of this catalogue could be automatically found in the data we collected¹.

The remainder of the paper is organized as follows: In Section 2, we describe how the data was acquired and the representation we used. In Section 3, we introduce the algorithms we used. In Section 4, we present the experiment itself, as well as the results we obtained. Finally, in Section 5, we present some conclusions and future directions for this work.

2. Data Acquisition and Representation

For this experiment, two different players (from now on referred to as Player1 and Player2) recorded the accompaniment of some *Bossa Nova* songs on a MIDI guitar. Player1 performed the following songs: *Bim Bom*, *O Barquinho*, *Insensatez* (How Insensitive), *Garota de Ipanema* (Girl from Ipanema), *Só Danço Samba*, and *Wave*. From Player2 we recorded *A Felicidade*, *Chega de Saudade*, *Corcovado*, *Desafinado*, *Eu Sei Que Vou Te Amar*, *Samba de uma Nota Só*, *Garota de Ipanema*, *Só Danço Samba*, *Insensatez*, *Tarde em Itapoã*, and *Wave*. In the total, we collected 16 recordings (ca. 30 minutes

¹The catalogue reflects the rhythmic patterns used by João Gilberto, the “inventor” of this style.

of music). It was requested the performers to play the songs according to a provided notation (the chord grid as notated by Chediak [Chediak, 1990]).

The acquired data was, however, not ready for usage. Probably due to technological restrictions, the resulting MIDI files were noisy (they contained several extra notes, i.e., notes the player did not play). So, we had to manually correct them, removing these extra events. After correction, the data was beat tracked at the eight note level using *BeatRoot* [Dixon, 2001], an interactive system that outputs the MIDI file beat tracked.

As we are interested in the discovery of rhythmic patterns, the exact pitches played (i.e., A, B, C \sharp , etc.) are not that relevant. Apart from the durations, a much more relevant abstraction is the right hand finger used by the player to pluck the string². It becomes even more relevant when you find in the literature (e.g., in [Garcia, 1999]) that musicians usually describe the rhythmic patterns in terms of “baixo” or *bass* (events played with the thumb only) and “puxada” or *chord* (events played with some combinations of two or more fingers). The right hand fingering for each song was determined automatically. Given a MIDI file (beat tracked or not), the algorithm (as described in [Trajano et al., 2004]) outputs the fingering as depicted in Figure 1. Letters *T*, *F*, *M* and *R* represent, respectively, the thumb, fore, middle and ring fingers, crosses (+) represent the beats, and pipes (|) represent the measure bars. Each beat was equally divided by four, so each letter, cross or minus (−) represents the duration of a 32nd. Except for the last line, that represents exclusively the beats, each of the remaining lines represents one guitar string, ordered from higher to lower (i.e., first line represents the high E string, second line the B string, and so on until the low E string).

	-----		-----		--	
	R---R-----R-----		R---R-----R-----		R-	
	M---M-----M-----		M---M-----M-----		M-	
	F---F-----F-----		F---F-----F-----		F-	[. . .]
	-----		-----		--	
	T-----T-----		T-----T-----		T-	
	+---+---+---+---+		+---+---+---+---+		++	

Figure 1: Right hand fingering for song *Insensatez*, as played by Player2

In order to reduce the complexity of the pattern extraction, this initial representation for the fingering was reduced to a one-dimensional string. This simplified string is formed by the alphabet $\Sigma = \{b, B, p, P, l, a, A, s, S, -, +, |\}$. The meaning of these symbols is the following:

- Uppercase letters stand for events that occur on-beat, while lowercase letters for off-beat events;
- Letter *b* stands for “bass”, i.e., events played with the thumb only;
- Letter *p* stands for “chord” (sic), i.e., events that are played with some combinations of two or more of fingers *F*, *M* and *R*³;
- Letter *l* also stands for “chord”, but a chord whose duration goes beyond the measure it was played (it means that we make a difference between a chord that is completely within a single measure and a chord that starts in one measure and ends in the next one);
- Letter *a* stands for “all”, i.e., *b* and *p* played together;
- Letter *s* stands for “single note”, i.e., events that are played with only one of fingers *F*, *M* and *R*; and

²We assume the player is right handed.

³The terms “baixo” and “puxada” may explain more clearly the origin of letters *b* and *p*!

- Note that this reduction is also done by the musicians themselves, when they describe the rhythmic patterns they play as sequences of basses and chords (“baixos” and “puxadas”). Figure 2 depicts part of the fingering for song *Insensatez*. Above the thick black line is the fingering as output by the right hand fingering algorithm. Under it is the resulting simplified string.

	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-		
	R	-	-	-	R	-	-	-	-	-	R	-	-	-	-	-		R	-	-	-	R	-	-	-	-	-	R	-	-	-	-	-		R	-
	M	-	-	-	M	-	-	-	-	-	M	-	-	-	-	-		M	-	-	-	M	-	-	-	-	-	M	-	-	-	-	-		M	-
	F	-	-	-	F	-	-	-	-	-	F	-	-	-	F	-		F	-	-	-	F	-	-	-	-	-	F	-	-	-	-	-		F	-
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-
	T	-	-	-	-	-	-	T	-	-	-	-	-	-	-	-		T	-	-	-	-	-	-	-	-	T	-	-	-	-	-	-		T	-
	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-		+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-		+	-
	A	-	-	-	P	-	-	-	B	-	p	-	+	-	s	-		A	-	-	-	P	-	-	-	B	-	p	-	+	-	-	-		A	-

3. Algorithms

FlExPat is an inexact string matching algorithm that was inspired by algorithms from the Computational Biology field, but that also incorporates results from previous research on musical similarity [Mongeau and Sankoff, 1990]. Given an input (here the simplified string previously described) and using the edit distance as its similarity measure⁴, the algorithm outputs a collection of patterns, organized in classes. Each class has a prototype, that is the most representative pattern of the class, and several occurrences, possibly inexact, of this prototype. It is also possible for the user to provide the algorithm with some constraints, such as the maximal and minimal length of patterns, the similarity threshold, the maximum difference between two candidates to be compared, etc.

⁴Dynamic programming is used to compute it efficiently.

4. Experiment

In a previous work [Dahia et al., 2004], we used a catalogue containing 21 rhythmic patterns of Bossa Nova guitar music, acquired manually from João Gilberto's performances, to automatically generate the guitar accompaniment of some songs. These songs were, thereafter, judged by some specialists as being stylistically correct. Figure 3 depicts some of these patterns.

Figure 3 displays three examples of rhythmic patterns from a catalogue, labeled Pattern P1, Pattern P15, and Pattern P17. Each pattern is shown as a musical staff with a 'Chord' staff and a 'Bass' staff. The time signature is 2/4. The 'Chord' staffs show a sequence of chords, while the 'Bass' staffs show a sequence of bass notes. The patterns are:

- Pattern P1:** Chord staff has a sequence of four chords (quarter, eighth, quarter, eighth). Bass staff has a sequence of four quarter notes.
- Pattern P15:** Chord staff has a sequence of four chords (quarter, eighth, quarter, eighth). Bass staff has a sequence of four quarter notes.
- Pattern P17:** Chord staff has a sequence of four chords (quarter, eighth, quarter, eighth). Bass staff has a sequence of four quarter notes.

Figure 3: Examples of rhythmic patterns from the catalogue

As one can imagine, the manual acquisition of such a catalogue is tiresome. It can even be said that, in practice, it can not be done if you want to have the catalogues of many players⁵. So, the acquisition of catalogues of this kind must be done automatically (or semi-automatically, at least). The main objective of our experiment is to identify at which extent this Bossa Nova catalogue could be automatically found in the data we collected. This will give us hints on algorithms that could be applied to the acquisition of rhythmic catalogues for other styles, as Samba, for instance.

We rewrote the catalogue in the form of the simplified string previously described and used the Boyer–Moore algorithm to test occurrences of each pattern in the data set. From the 21 patterns in the catalogue, the algorithm could only find one pattern (P1) in the whole data set, appearing only in three of the songs. Table 1 summarizes the results. Each entry in the table represents the number of occurrences of P1.

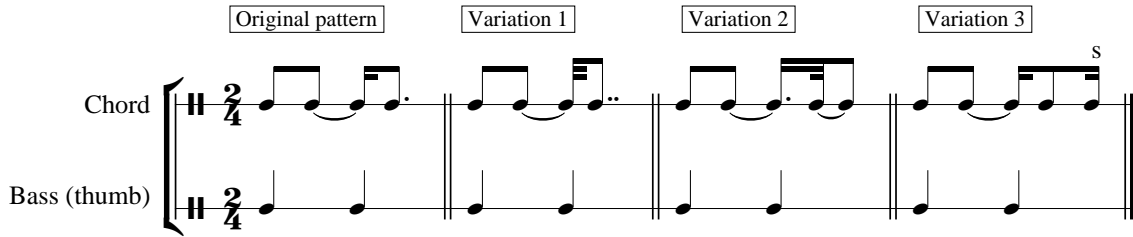
The main question now is the following: is there any flaw in the methodology we used? Since João Gilberto is the inventor of the Bossa Nova guitar style, we expected to find a more representative number of the patterns in the data set. Besides, by hearing the songs it is possible to notice that more patterns from the catalogue are used. The problem confirms that an exact matcher like the Boyer–Moore is not able to cope with any sort of deviations that are usually done by the performers (as in any expressive performance, in

⁵See, for instance, the work of Owens [Owens, 1974]. It took him 16 years to build a similar catalogue for Charlie Parker *only*.

Song	Player1	Player2
Desafinado	0	1
Garota de Ipanema	0	6
Insensatez	4	29

Table 1: Number of occurrences of pattern P1

fact). The most usual of these deviations are small anticipations and delays of events. As an example, part of a pattern that is transcribed as A---P---B-p+---- can be played slightly different from notated (and it usually is!) such as in A---P---Bp---+---- (an anticipation of the last *p*) or as in A---P---B--p+---- (a delay of the same *p* event). Another possible modification is playing the same pattern as A---P---B-p-+-s-, i.e., adding a single note at the end. Figure 4 depicts these variations (the “s” above the last 16th represents a single note).

**Figure 4: Some variations of part of a pattern**

The main problem here is that an exact matcher, such as the Boyer–Moore algorithm, isn’t suitable for the task, due to these kind of variations. In order to discover patterns, we must use algorithms that can perform the so called *inexact string matching*. For that, we used FIEsPat and SimilaritySegmenter.

The results were in general good. If we take into account the small modifications done by the performers, this time more patterns from the catalogue were found in the data set⁶.

The configuration we used for FIEsPat was the following:

- $m_{min} = 17$ (minimal pattern length);
- $m_{max} = 34$ (maximal pattern length);
- similarity threshold: 0.75 (normalized values);

The lengths m_{min} and m_{max} correspond to patterns varying from one to two measures. It does not mean, however, that the patterns necessarily start at the beginning of the measure. There is no way to specify such a constraint in FIEsPat. On average, each song has 48 classes of patterns (smallest number of classes was 40 and greatest number was 78), which is an acceptable number.

In song Garota de Ipanema, FIEsPat identified the following pattern for Player2: | A---P---B-p+---- | P---P---Bp---+1. The corresponding pattern in the catalogue is | A---P---B-p+---- | A---P---B-p-+-1-. Looking closer at the modifications done by the performer, it is possible to notice that he usually substitutes the A by P in the second measure of the pattern, and that he anticipates both final chords in this same measure. Figure 5 depicts these variations.

⁶Note that now, with the inexact matching, there is a difference in our methodology: while with the Boyer–Moore algorithm we tested the existence of the patterns of the catalogue directly in the data set, now we must first induce patterns and only then look and see if the induced patterns have some resemblance with the patterns in the catalogue.

Original pattern

Pattern found by FIEsPat

Figure 5: Pattern found by FIEsPat

In song Wave, as played by Player1, FIEsPat identified as the prototype of class 18 the following pattern: | A---P---B-p++l-- | B---P---Bp---+p-. The corresponding pattern in the catalogue is | A---P---B-p++l- | B---P---B-p++l-. As in the previous case, the modifications done by the performer are mainly anticipations.

Table 2 summarizes the results we obtained for FIEsPat.

Pattern	Player1	Player2	Pattern	Player1	Player2
P1	yes	yes	P12	no	no
P2	yes	no	P13	no	no
P3	no	no	P14	no	no
P4	yes	no	P15	yes	no
P5	no	no	P16	no	no
P6	no	no	P17	no	no
P7	no	no	P18	no	no
P8	no	no	P19	no	no
P9	yes	yes	P20	no	no
P10	yes	yes	P21	no	no
P11	no	no	/	/	/

Table 2: Occurrences of patterns in data set (FIEsPat)

Although FIEsPat found several interesting patterns in the data set, there were some problems with the results we obtained. The main problem we found was that the extracted patterns, many times, start from the middle of a measure, such as in --B---B---+--- | P---B---B---+--- | P (FIEsPat also found the expected pattern, | P---B---B---+--- | P---B---B---+---). This brings some problems to the evaluation of the results: due to the great number of patterns with this structural malformation, it becomes difficult to validate the patterns. This problem is even bigger when most of a class is formed by such patterns (it happened many times, unfortunately).

For the SimilaritySegmenter, we allowed a maximal number of 2 mismatches per measure. We also specified some structural constraints, namely that each pattern should start at the beginning of the measure (first character should be a |) or that it should start at the first l before a measure bar. That is, patterns should have one of the following two structures:

- | < pattern > or
- $l * |$ < pattern >, where $*$ is a sequence, possibly empty, of only minuses (-).

Table 3 summarizes the results the obtained for SimilaritySegmenter.

Due to the possibility of specifying structural constraints, the results were much easier to evaluate (as compared to FIEsPat). One surprising result was that the algorithm

Pattern	Player1	Player2	Pattern	Player1	Player2
P1	yes	yes	P12	no	no
P2	no	no	P13	no	no
P3	no	no	P14	yes	no
P4	no	no	P15	no	yes
P5	no	no	P16	no	no
P6	no	no	P17	no	no
P7	no	no	P18	no	no
P8	no	no	P19	no	no
P9	yes	yes	P20	no	no
P10	no	yes	P21	no	no
P11	no	no	/	/	/

Table 3: Occurrences of patterns in data set (SimilaritySegmenter)

was able to find rather long patterns. In song *Insensatez* performed by Player2, for instance, the algorithm found three occurrences of the following pattern (136 characters long, i.e., 8 measures):

```
| A---P---B-p-+--- | A---P---B-p-+--- | A---P---B-p-+---
| A---P---B-p-+--- | A---P---B-p-+--- | A---P---B-p-+---
| A---P---B-p-+--- | A---P---B-p-+---
```

Although it is formed by the same cell (`| A---P---B-p-+---`), it is very significant that the algorithm could find such a long pattern. It may have structural implications: Player2 may have used this sequence during some specific part of the song. Looking closer at the data, we notice that, in this specific case, Player2 used this pattern as the accompaniment of most part of the theme. Other example of a long pattern is the following one:

```
| B---P---B-p-+-s- | A---P---B-s-+-l- | B-p-+-p-B-p-+-l-
| B-p-+-p-B---P---
```

This pattern is 68 characters long, occurred twice, and was also played by Player2, this time in song *Tarde em Itapoã*,

The algorithm also found some garbage, such as, `| B-` or `| A---+-`, but the main question that arose during the experiment was due to the algorithm's inherent non-determinism: if a pattern in the catalogue wasn't found is it because it was really not played or is it because the algorithm didn't run long enough? This seems to be case of P10 for Player1, which was found several times by FIEXPath.

5. Final Remarks and Future Work

This paper described an experiment that dealt with extraction of rhythmic patterns from Bossa Nova songs. Sixteen beat tracked MIDI files, representing the recording of several songs by two different players, were represented as a string and thereafter processed by three different string matching algorithms. The objective was to identify in the data set patterns from a previously acquired catalogue.

First results showed that, as expected, an exact matcher was not able to cope with the variations of the patterns done by the players, as well as, with some single notes they play between chords. With the use of inexact matchers the results were much more

interesting: several patterns from the catalogue could be found in the data set. We also obtained some surprising results, such as the long patterns found by SimilaritySegmenter.

There are, however, several points for improvement. The first one, and may be most important one, is the representation. The representation we used is, indeed, simple. Attributes like tempo, structural or harmonic information are not represented. The more attentive reader may have noticed that the actual duration of the events is not represented. We just used the onset information, which turned out to work for this experiment. To further investigate the particularities of the patterns, however, we surely need to represent appropriately the duration of each event.

The evaluation of the algorithms' results was done in an *ad hoc* manner: results were compared, one by one, to the patterns in the catalogue. This procedure takes too much time, is error prone, and, therefore, must be improved. We plan to implement a tool to help us with this task.

FLEXPAT's problem (structural malformation) must be examined more carefully. Instead of a problem, it can mean another thing. Several patterns of the catalogue have a common substructure (i.e., subparts of these patterns are equal). It may be the case that the second measure of patterns whose second measure are equal, are frequently concatenated with patterns whose first measure are similar. So, it may be the case that these concatenations are so typical that they are "promoted" to patterns by the algorithm. This must be further investigated.

Another point for further investigation is the possibility of combining the strengths of the algorithms into a single procedure. Whereas FLEXPAT performs a complete search for patterns, SimilaritySegmenter is able to satisfy some quite specific structural constraints about them. It may be interesting to try to combine such particularities into a single method.

Finally, we already have collected some other songs, namely some Sambas recorded by three different players. We plan, as soon as we have prepared the data, to perform a similar experiment on this data set.

6. Acknowledgments

The first author is supported by a research grant from CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico). The research by the Austrian Research Institute for Artificial Intelligence was supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. Y99-START. The Austrian Research Institute for Artificial Intelligence acknowledges financial support by the Austrian Federal Ministries for Education, Science and Culture (BMBWK) and for Transport, Innovation and Technology (BMVIT).

References

- Boyer, R. S. and Moore, J. S. (1977). A fast string searching algorithm. *Commun. ACM*, 20(10):762–772.
- Chediak, A., editor (1990). *Songbook: Bossa Nova*, volume 1–5. Lumiar Editora, Rio de Janeiro.
- Dahia, M., Santana, H., Trajano, E., Sandroni, C., Ramalho, G., and Cabral, G. (2004). Using patterns to generate rhythmic accompaniment for guitar. In *Proc. of Sound and Music Computing (SMC'04)*, pages 111–115.

- Desain, P., Honing, H., and Timmers, R. (2001). Music performance panel: NICI/MMM position statement. In *MOSART Workshop on Current Research Directions in Computer Music*, Barcelona, Spain.
- Dixon, S. (2001). Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58.
- Garcia, W. (1999). *Bim Bom: A Contradição sem Conflitos de João Gilberto*. Editora Guerra e Paz.
- Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press.
- Madsen, S. T. and Widmer, G. (2005). Evolutionary search for musical parallelism. In *Proc. of the EvoWorkshops2005*, Lecture Notes in Computer Science 3449, pages 488–497. Springer Verlag.
- Mongeau, M. and Sankoff, D. (1990). Comparison of musical sequences. *Computer and the Humanities*, 24:161–175.
- Owens, T. (1974). *Charlie Parker: Techniques of Improvisation, 2 vols.* PhD thesis, Univeristy of Los Angeles California.
- Rolland, P.-Y. (2001). FIEXPath: Flexible extraction of sequential patterns. In Cercone, N., Lin, T. Y., and Wu, X., editors, *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, pages 481–488. IEEE Computer Society.
- Sandroni, C. (2001). *Feitiço Decente: Transformações do Samba no Rio de Janeiro (1917–1933)*. Jorge Zahar Editor, Rio de Janeiro.
- Sundberg, J., Friberg, A., and Frydén, L. (1991). Common secrets of musicians and listeners: an analysis-by-synthesis study of musical performance. In Howell, P., West, R., and Cross, I., editors, *Representing Musical Structure*, pages 161–197. Academic Press.
- Trajano, E., Dahia, M., Santana, H., and Ramalho, G. (2004). Automatic discovery of right hand fingering in guitar accompaniment. In *Proceedings of the International Computer Music Conference (ICMC'04)*, pages 722–725.
- Widmer, G. (1998). Applications of machine learning to music research: Empirical investigations into the phenomenon of musical expression. In Michalski, R. S., Bratko, I., and Kubat, M., editors, *Machine Learning, Data Mining and Knowledge Discovery: Methods and Applications*. Wiley & Sons, Chichester (UK).
- Widmer, G. (2002). Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research*, 31(1):27–50.
- Widmer, G., Dixon, S., Goebel, W., Pampalk, E., and Tobudic, A. (2003). In search of the Horowitz factor. *AI Magazine*, 24(3):111–130.

Similarity Measures for Rhythmic Sequences

João M. Martins¹, Marcelo Gimenes¹, Jônatas Manzolli², Adolfo Maia Jr.²

¹Computer Music Research, Faculty of Technology
University of Plymouth, Plymouth, Devon PL4 8AA
United Kingdom.

²Departamento de Matemática Aplicada, IMECC and
Núcleo Interdisciplinar de Comunicação Sonora (NICS), UNICAMP,
13.081-970, Campinas, SP, Brazil.

joao.martins@plymouth.ac.uk, marcelo.gimenes@plymouth.ac.uk

jonatas@nics.unicamp.br, adolfo@nics.unicamp.br

Abstract. *This paper presents a new model for measuring similarity in a general Rhythm Space. Similarity is measured by establishing a comparison between subsequences of a given rhythm. We introduce the hierarchical subdivision of rhythm sequences in several levels, and compute a Distance Matrix for each level using the “block distance”. The information about the similarity of the rhythmic substructures is retrieved from the matrices and coded into a Similarity Coefficient Vector (SCV). We also present possibilities for the reduction to single values of similarity derived from the SCV. In addition, two applications of the formal model are presented, showing the potential for development using this approach.*

1. Introduction

”To study rhythm is to study all of music. Rhythm both organises, and is itself organised by, all the elements which create and shape musical processes” and this is how Meyer and Cooper [Cooper G., 1963] emphasize the importance of rhythm in the overall structure of music. In the 20th century, rhythm in music, has finally been put in the focus of the attention of composers, such as Igor Stravinsky, Olivier Messiaen, Iannis Xenakis. Since then, an increasing interest to automatically compare music styles and composing techniques. Rhythm, as the most fundamental aspect of music plays a decisive role in this task.

The rhythm organization of music, as well as speech sounds and environmental events, are highly dependent on the perception of human beings. Even when subjects are presented with equal pulses at equally spaced intervals, the pulses are perceived as being grouped in a regular metric structure [Handel, 1989]. It is argued that this implicit metrical organization improves attention and memorisation of sequential tasks.

Perception influenced a great number of researchers in music [Gabrielsson, 1973, Povel and Essens, 1985]. Cooper and Meyer [Cooper G., 1963], developed an auditory theory based on *Gestalt* theories of perception, where rhythm groups in the basic level are seen as units that are categorised according to the position of the accentuated notes. Also according to them, one strong cue in rhythm organization is the one of pattern repetition. When a rhythmic motive is repeated, the brain integrates it creating a unit that is memorised and categorised accordingly.

In our approach we provide a measure of the occurrence of these repetitive patterns in a stream of rhythmic events. We leave out of this study the implications of accentuation, melody, harmony, timbre, and articulation to the perception of rhythm, and we do so for two reasons: Firstly, we are able to extract interesting and meaningful information solely from the position where the events take place, and secondly, we can find repertoire for percussion that does not contemplate any of the former musical characteristics apart from accentuation. We strongly believe, though, that our measure can be extended to incorporate some of these characteristics. Furthermore, we can have rhythms which do not obey the marks of bars or any metrical structure. This enables to compare and distinguish rhythmic sequences with different subdivisions, and possibly to provide some insight on situations that metric is difficult to extract.

Computers find it simple to discriminate if something is equal or different, but the problem rises when there is the need to evaluate if something is similar [Minsky, 1988]. The necessity of similarity measures concerns many areas of music research, specially music information retrieval systems [Hewlett and Selfridge-Field, 2005], automatic rhythm transcription of human-performed music to MIDI protocol [Takeda et al., 2003], evaluation of copyright issues, and evolutionary music [Miranda, 2004].

On the side of the abstract models, interesting results were achieved using the Levenshtein distance, also called edit distance. This is a popular method for measuring similarity between strings of text of arbitrary length. This method counts the number of insertions, deletions and substitutions necessary to change one string into another one, being this number the measure of similarity between the sequences. Orpen and Huron have applied this distance to measure melodic, rhythmic and harmonic similarity in Bach chorales [Orpen and Huron, 1992]. Mongeau and Sankoff provided a method which can be seen as an extension of the previous [Mongeau and Sankoff, 1990]. Instead of considering that each transformation to the sequence contributes with the value of one to the distance, each transformation contributes with a weighted value sensitive to the kind of musical differences who are to be measured.

In this work we are most interested in constructing an abstract and formal model which can be able to compare rhythm patterns in the most general way, capturing information in several layers of detail. In addition we intend our model to be able to manipulate rhythm sequences in order to create new ones, which could be used in music composition. In the future we will extend this work by comparing it with the existing formal models and we hope to establish a closer relation between our model and perception by testing rhythmic similarity with human subjects.

In the next section we formally introduce the concepts of Rhythm Space and Similarity Measure. In section 3 we describe our algorithm implementation. In section 4 we present two applications of our model. In the last section we conclude with some comments about the model and list some interesting topics for further research.

2. Rhythm Space and Similarity Measure

In this work, rhythm sequences are thought as elements (or vectors) of a finite dimension vector space. Formally we have coded rhythms as sequences of numbers (b_1, b_2, \dots, b_r) , where the entries b_i can be any number of the set $B = \{-1, 0, 1, 2, \dots, J\}$ which we named *Beat Set*. A positive number in a sequence indicates first that one have a beat and its magnitude indicates the level of accentuation, such as strong beat, weak beat, half strong beat, etc. The number of sequential 0s indicates the duration of the beat. The number -1 indicates pauses or, in MIDI protocol, a note off. We associate the positive numbers to accentuation. For example, taking $J = 3$ we get the Beat Set $B = \{-1, 0, 1, 2, 3\}$, the accen-

tuation should be read as: 1 means a *weak beat*, 2 a *half strong* and 3 a *strong beat*. Then we can construct rhythm sequences like, for example, $(3, 0, 1, 0, -1, 0, 0, 0, 2, 0, 0, 0, 3)$. Clearly we can have as many accentuation we wish, just extending the Beat Set. However we must introduce a prescription in order to avoid some ambiguities.

Rule: *If in a rhythm sequence a value -1 occurs, it can only occur again after a positive number had occurred first.*

This rule avoids ambiguities, for example, if we compare the sequences like $a = (1, -1, -1, 0, 0)$ and $b = (1, -1, 0, 0, 0)$. Since they are different one from another, the distance (see below) between them is positive. Nevertheless they represent the same events (in this case, pause) which intuitively suggests the distance must be zero. Our rule says that only the second sequence b is a valid one, that is, it is an element of our Rhythm Space R defined below.

Now, given a Beat Set B , we define its associated n -rhythm vector space $R_n(B)$ as the set of all n -vectors $\mathbf{v} = (v_1, v_2, \dots, v_n)$ in which each entry is an element of the Beat Set B . On $R_n(B)$ we can define a distance. Let $\mathbf{v} = (v_1, v_2, \dots, v_n)$ and $\mathbf{w} = (w_1, w_2, \dots, w_n)$ be two vectors in $R_n(B)$. The p -distance between them is defined as

$$d_p(\mathbf{v}, \mathbf{w}) = \left(\sum_{i=1}^n |v_i - w_i|^p \right)^{1/p}. \quad (1)$$

The value of p can be chosen according to the application or the kind of music considered. In our examples and applications we take, for the sake of simplicity, $p = 1$, the so called *block distance*.

The above p -distance is defined only for rhythm sequences which have the same length. Obviously, in most of applications, we must compare rhythm sequences of different lengths. Although it is possible to define a distance between vectors with different sizes (the so called Hausdorff Distance) we prefer to use for comparison of arbitrary rhythm sequences the concept of similarity in a particular way. So, the next logical step is to put together all the possible rhythm sequences into a same *Rhythm Space* and define a *Similarity Measure* on it. This is as follows.

Firstly, we define the **Rhythm Space**, denoted here by R , as the union of all $R_n(B)$, that is, $R = \bigcup_{i=0}^{\infty} R_n(B)$. We name, alternatively, the elements of R as *Rhythm Vectors*. Note that for each given Beat Set we have an associated Rhythm Space. We introduce similarity measure on R as follows. Given an arbitrary rhythm vector $\mathbf{v} = (v_1, v_2, \dots, v_n)$, we define a k -level subsequence $\mathbf{v}^{(k)}$ of \mathbf{v} as any subsequence with k elements extracted from \mathbf{v} , preserving the original order of \mathbf{v} . For example, if $\mathbf{v} = (2, 0, 0, 1, -1, 0, 1, 1)$ we can extract five ordered four-levels sequences, namely, $\{(2, 0, 0, 1), (0, 0, 1, -1), (0, 1, -1, 0), (1, -1, 0, 1), (-1, 0, 1, 1)\}$. It is easy to see that a vector with n elements has $n - k + 1$ k -level subsequences. Now, given two rhythm vectors $\mathbf{v} = (v_1, v_2, \dots, v_n)$ and $\mathbf{w} = (w_1, w_2, \dots, w_m)$ in R consider all k -level sequences of both vectors, that is, the sets $S_{\mathbf{v}}^{(k)} = \{\mathbf{v}_i^{(k)}, i = 1, 2, \dots, n - k + 1\}$ and $S_{\mathbf{w}}^{(k)} = \{\mathbf{w}_j^{(k)}, j = 1, 2, \dots, m - k + 1\}$. If, for example, $m \leq n$ we only can consider sequences with length smaller than m , that is, we must take $1 \leq k \leq \min(m, n)$.

Formally, for each k -level, we define the (i, j) -elements of the k -level Distance Matrix $\mathbf{D}^{(k)}$ of two vectors \mathbf{v} and \mathbf{w} as:

$$[\mathbf{D}^{(k)}(\mathbf{v}, \mathbf{w})](i, j) = d_p(\mathbf{v}_i^{(k)}, \mathbf{w}_j^{(k)}) \quad (2)$$

where $i = 1, 2, \dots, n - k + 1$ and $j = 1, 2, \dots, m - k + 1$. Below we show a visualization

of a general k -level Distance Matrix.

$$\mathbf{D}^{(k)} = \begin{bmatrix} d_p(\mathbf{v}_1^{(k)}, \mathbf{w}_1^{(k)}) & d_p(\mathbf{v}_1^{(k)}, \mathbf{w}_2^{(k)}) & \dots & d_p(\mathbf{v}_1^{(k)}, \mathbf{w}_{(m-k+1)}^{(k)}) \\ d_p(\mathbf{v}_2^{(k)}, \mathbf{w}_1^{(k)}) & d_p(\mathbf{v}_2^{(k)}, \mathbf{w}_2^{(k)}) & \dots & d_p(\mathbf{v}_2^{(k)}, \mathbf{w}_{(m-k+1)}^{(k)}) \\ \vdots & \vdots & \vdots & \vdots \\ d_p(\mathbf{v}_{(n-k+1)}^{(k)}, \mathbf{w}_1^{(k)}) & d_p(\mathbf{v}_{(n-k+1)}^{(k)}, \mathbf{w}_2^{(k)}) & \dots & d_p(\mathbf{v}_{(n-k+1)}^{(k)}, \mathbf{w}_{(m-k+1)}^{(k)}) \end{bmatrix}$$

Although there exist many different measures we restrict our analysis, as mentioned above, to the block distance ($p = 1$) and the Beat Set to $\mathbf{B} = \{0, 1\}$ (Fig. 1).

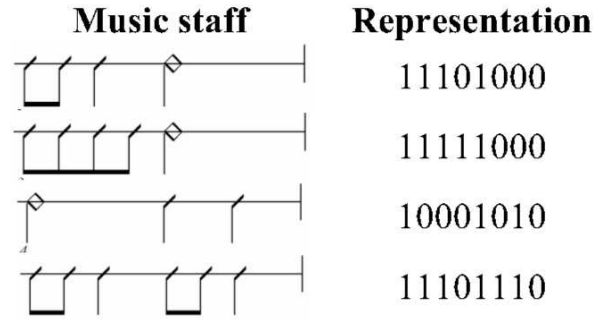


Figure 1: Musical notation and correspondent coding

We get, then, a k -level Distance Matrix whose elements are non negative integers. Now, we define the k -level *Similarity Coefficient* as the

$$c^{(k)}(\mathbf{v}, \mathbf{w}) = \frac{z^{(k)}}{(n - k + 1)(m - k + 1)} \quad (3)$$

where $z^{(k)}$ is the number of zeros in the matrix $\mathbf{D}^{(k)}$. Roughly speaking the similarity coefficient measures the *sparsity* of the matrix $\mathbf{D}^{(k)}$. Greater the coefficient $c^{(k)}$, greater is the similarity between the subsequences of level k . In the extreme case a matrix with all coefficients equal to 1, it means that one of the sequences has a perfect copy of it contained in other one.

Now we can collect all the k -levels coefficients in a vector we name *Similarity Coefficient Vector* (SCV). It reads like

$$\mathbf{C} = [c^{(1)}, c^{(2)}, \dots, c^{(min(m,n))}] \quad (4)$$

In Fig. 2 we show an example of the 3-level Distance Matrix and its respective SCV. Bellow we provide an example of this approach.

Example:

Take the Beat Set as $B = \{0, 1\}$. Let $\mathbf{v} = (1, 0, 1, 1)$ and $\mathbf{w} = (1, 0, 1, 1, 0, 1)$ be two rhythm sequences in R . The possible 3-level sequences for \mathbf{v} and \mathbf{w} are:

- $S_{\mathbf{v}}^{(3)} = \{(1, 0, 1), (0, 1, 1)\}$
- $S_{\mathbf{w}}^{(3)} = \{(1, 0, 1), (0, 1, 1), (1, 1, 0), (1, 0, 1)\}$

Let us take, for the sake of simplicity, $p = 1$ on the Rhythm Space. Since we must take the distance between all elements of each level up to sixth level, one can guess that a large number of evaluations is needed. We show below only the distance between the

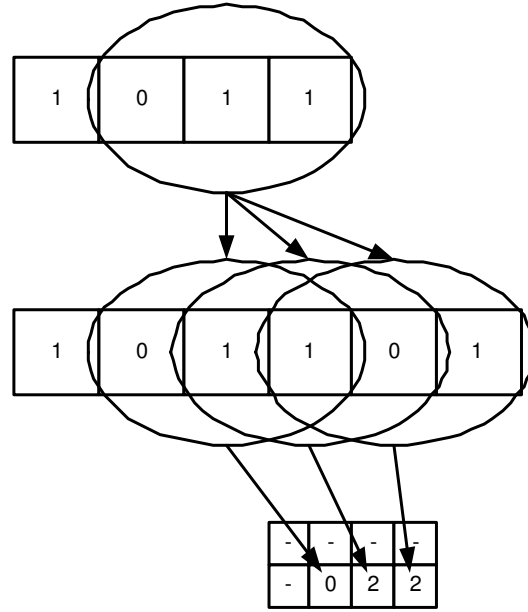


Figure 2: Building the Distances Matrix for the 3-level

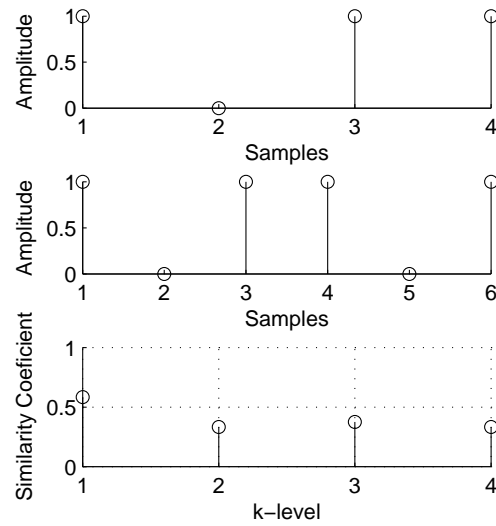


Figure 3: Example of two vectors and their Similarity Coefficients Vector (SCV)

combinations in the 3-level. According to the procedure describe above and shown in Fig. 2, we obtain the (2×4) 3-level Distance Matrix:

$$D^{(3)}(\mathbf{v}, \mathbf{w}) = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 2 & 2 \end{bmatrix} \quad (5)$$

The 3-level coefficient can be read easily from this matrix and it is $c^{(3)} = 3/8 = 0.375$.

The complete SCV (Fig. 3) for the above example is given by

$$\mathbf{C} = [0.5833, 0.3333, 0.3750, 0.3333] \quad (6)$$

In the next section we show the algorithmic implementation and make some additional comments on our model.

3. Algorithm Implementation

We have implemented an algorithm in MATLAB which is able to construct, manipulate, and play the rhythm sequences defined by our model. We restrict our analysis below to rhythms without accentuation and articulation and also without pauses. This is a crude approximation to real rhythms and, in our model, it is accomplished by taking the simplest Beat Set, that is, $B = \{0, 1\}$. These aspects will be added in a further implementation of our formal modal described above.

We also devised a function to play back the input sequences, to do a subjective evaluation of the result of the measure. The events correspond to sinusoidal functions with exponential decay and we introduced a short tone at the starting point as a reference for the beginning of the sequence.

3.1. Similarity Coefficient Vector

The algorithm picks two sequences of elements extracted from the Beat Set and computes, for each k -level, the matrix $\mathbf{D}^{(k)}$. The meaning of a zero in a matrix element, corresponds to a perfect match between sub-sequences of the two input vectors. At this point we have as many matrices \mathbf{D} as the length of the shortest input vector.

The sparsity of the matrix, which means the number of zeros in each $\mathbf{D}^{(k)}$ matrix, will give information on how similar are the subsequences of that particular k -level. The algorithm computes the ratio between the number of zeros and the product of the dimensions of $\mathbf{D}^{(k)}$ for each k -level and stores those values in the SCV.

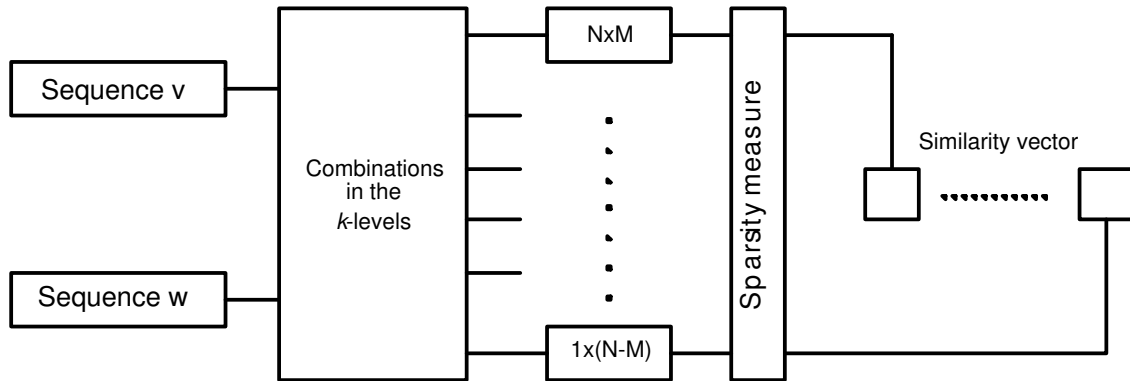


Figure 4: Model for the creation of the SCV between the two vectors

3.2. Analysis by Single Values

In addition to the SCV, we thought that it would also be interesting to reduce the quest for measuring similarity to a single value, enabling an easier comparison between the sequences. We offer three different solutions for this problem.

It is clear to us that the content of the first element provides a low level of information, as it only tells us whether the distribution of events (1s) and no-events (0s) in both of the vectors is even, or it is polarized towards having more events or no-events. On the other hand, the content of the last element of the SCV gives us the highest information. Finding a non-zero value in this position implies that the shortest input sequence exists at least once in the longest input sequence. In most of the comparisons this element will be zero. So the last non-zero element will tell us that what is the size of the longest sub-sequence that is common to both input sequences.

Another parameter that may be useful is the sum of the elements of SCV, which will take into account the coefficients from all the k -levels. For example, by considering

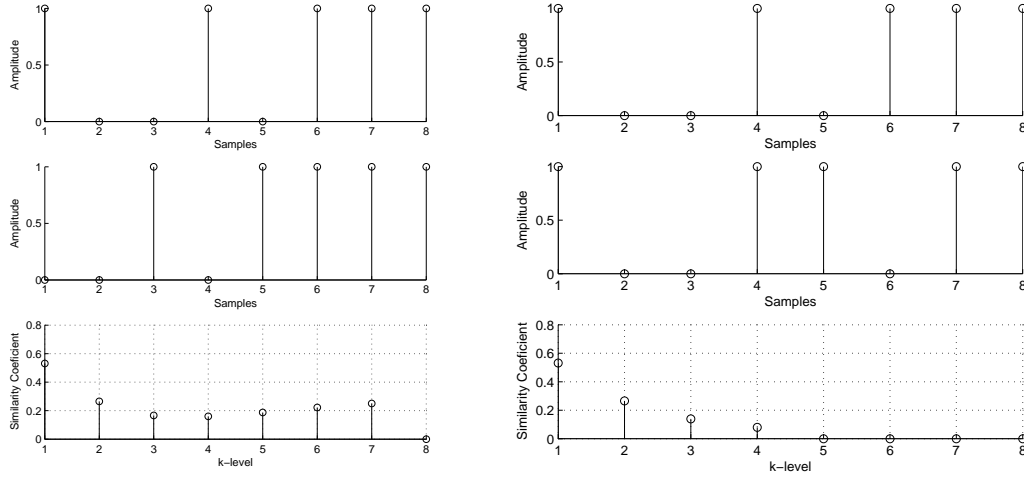


Figure 5: Vector with high values of the Similarity Coefficients values

the input sequence $\mathbf{v} = (1, 0, 0, 1, 0, 1, 1, 1)$ we ran the distance for all possible \mathbf{w} vectors of length 8. In Fig. 5 we present two vectors that show high similarity with the presented sequence. By maximizing the sum of the Similarity Coefficients vector and removing the input vector from the competition, we arrive to the value $\sum C_i = 1.7829$ with the most similar vector being $\mathbf{w} = (0, 0, 1, 0, 1, 1, 1, 1)$ as can be seen in Fig. 5 (left). If instead we use the same vector and minimize the sum of the SCV elements, we get the least similar vector. For the example above the resultant value is $\sum C_i = 0.5179$, and the resultant \mathbf{w} vector will be the no-event vector.

However, the sum above, does not consider, that there is greater importance in the rightmost elements of the vector. This can be achieved by taking a weighted sum of the elements, with an increasing profile of the weights.

4. Computer Applications

The formal model presented in this work is flexible enough to be used in several applications from rhythm analysis to creation of new rhythms, etc. Below we describe just two applications, namely, *Net-Rhythms* related to Neural Networks, developed by one of the authors(JM), and *RGemede* developed by another author (MG) in which AI agents learn rhythmic sequences from one another.

4.1. Neural Networks and Rhythms

Net-rhythms is a tool developed to classify and store rhythmic representations in a neural network. The framework used by this tool is constituted by a Neural Network called the SARDNET [James and Miikkulainen, 1995], an extended Kohonen self-organising feature map [Kohonen, 1985]. This network was developed to study the study of sequences and organization of phonemes in the context of language. We decided to explore its potential in the representation of rhythmic sequences, and new problems arose particularly related to the measurement of the distance between two vectors. The diagram on Fig. 6 explains how the network works.

The rhythms are coded according to the representation depicted in Fig. 1. Whenever a small rhythmic sequence \mathbf{v}_t in time step t reaches the input, the distance from that sequence to all weight vectors w_j is computed. The neuron corresponding to weight more similar to the input, according to the defined distance, is activated and removed from further testing. As time progresses all activations are decayed, implying that after some time

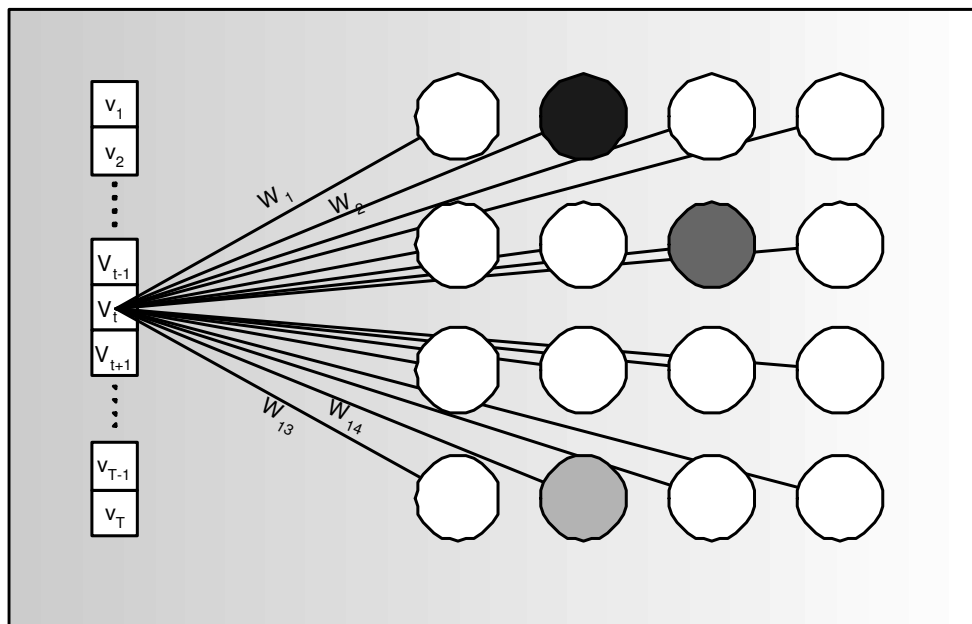


Figure 6: Diagram of the Sardnet with three activated neurons

steps there will be a ladder of activations in the network. In Fig. 6 there are three activated neurons represented by grey tonalities, corresponding the w_{14} weight to the first activated neuron, and w_2 to the last one. Finally, after a complete sequence on time T , the weights from the activated neurons are slightly adapted in order to decrease their distances to input vectors.

This network can represent in a bidimensional space the rhythms that arrive sequentially to the input, and self-organize simulating a learning procedure.

As stated before, the choice of the winning neuron implies the measurement of the distance from the input to each of the neurons from the Sardnet. The distance proposed by the original creators of the network was the Euclidean distance, however there are some problems as this measure. The Euclidean distance does not allow sequences with different lengths and does not capture the similarity between equal sub-sequences that have their position shifted in time. The use of the SCV, and the other measures presented in Sec. 3.2, help solving the problems presented above.

4.2. Agents and Memes: A Rhythm Imitation Society

RGeme is an artificial intelligence system for the composition of rhythmic passages inspired by Richard Dawkin's theory of memes that is being presently developed in the Future Music Lab at the University of Plymouth. According to Dawkins [Dawkins, 1989, Dawkins, 1991], memes are basic units of cultural transmission in the same way that genes, in biology, are units of genetic information. Other researchers have already studied some applications of this concept in music [Cox, 2001, Gabora, 1997]

In *RGeme* a rhythmic composition is understood as the process of interconnecting sequences of basic elements (or "rhythmic memes") that have varied roles in the stream. Intelligent agents learn these roles from examples of musical pieces in order to evolve a "musical worldview" which consists of a "style matrix" of basic rhythms. During the learning stage, agents parse examples of pieces of music in search for rhythmic memes. These (candidates) memes are then compared with the agent's database of memes which we named Style Matrix (see Table 1), and are stored or transformed accordingly. For example, if the candidate meme is not already present in the agent's Style Matrix, it is

copied and it's weight is set to 1. Now the model of distance of rhythm patterns is used in this application in order to upgrade the weight of each meme in Style Matrix. In this way the style of the memes evolves in time. In *RGeme* it was used the block distance. So all memes in the agent's style matrix have their weight upgraded according to their distance to the candidate meme.

Meme	dFL	dLL	nL	W
01011101	1	1	6	1.0385
11011101	1	1	31	1.0424
10001000	1	1	1	1.0181
10010101	1	1	1	1.0171
11011010	1	1	1	1.0159
10011010	1	1	4	1.0090
10011001	1	1	4	1.0075
11111111	1	1	1	1.0040
10000000	1	1	1	1.0000

Table 1: Extract from an Agent Style Matrix

where

- dFL: date the meme was first listened to
- dLL: date the meme was last listened to
- nL: number of times the meme was listened to
- W: upgraded weight

In the second stage, the system creates new rhythmic sequences (Production Phase) according to the musical structures and rules that were previously extracted from the styles of the pieces that were used in the learning stage. At this stage, agents are able to learn from each other's "compositions" and capable of evolving new rhythmic styles by adapting to each other's rhythms. Clearly, new distances and similarity measures as shown above can be implemented in *RGeme*, which, of course could result in a different evolution of the memes society. This is presently under investigation.

5. Conclusion and Perspectives

We presented a model for measuring similarity in a general Rhythm Space, which include all the possible rhythm sequences. The key issue and innovative contribution of this work is the hierarchical subdivision of rhythm sequences in several levels and the construction of a Distance Matrix for each one of them. The information is coded in a Similarity Coefficient Vector (SCV), whose entries estimate the similarity between rhythm sequences in different k -levels. These coefficients are related to the sparsity the k -levels Distance Matrices. We also provided a easier to read single value measure for similarity. In addition we presented two applications for our formal model. Clearly, it can be applied in many other areas of music analysis and composition. It can be also applied on musical learning devices, such as self evaluation systems to relate played sequences to previously defined ones.

There is plenty of room to extend this work. For example, is yet to be done the comparison between this distance with other well established similarity measurements, such as the Levenshtein and Hausdorff distances. In addition to the block distance, we could use new basic p -distances and check which of them is better to the applications the user has in mind.

A further and also important problem is to link the formal similarity in this work to the rhythmic perception of human beings.

In this paper we have only shown the potential of our methods to construct similarity measures. The problem of perception of rhythm sequences deserves a deeper study by itself. This, as well comparisons with other methods, will be done in a future work.

6. Acknowledgments

This work was supported by the Leverhulme Trust (JM), the São Paulo State Research Foundation (FAPESP) (AMJ), and CAPES (Brazil) (MG, AMJ). A.M.J is grateful to the Computer Music Research group from the University of Plymouth, for the hospitality during the making of this work.

References

- Cooper G., Meyer, L. (1963). *The Rhythmic Structure of Music*. University Of Chicago Press.
- Cox, A. (2001). The mimetic hypothesis and embodied musical meaning. *MusicaScientia*, 2:195–212.
- Dawkins, R. (1989). *The Selfish Gene*. Oxford University Press, London.
- Dawkins, R. (1991). *The Blind Watchmaker*. Penguin Books, London.
- Gabora, L. M. (1997). The origin and evolution of culture and creativity. *Journal of Memetics -Evolutionary Models of Information Transmission*.
- Gabrielsson, A. (1973). Similarity ratings and dimension analyses of auditory rhythm patterns i. *Scand. J. Psychol.*, 14:138–160.
- Handel, S. (1989). *Listening: An Introduction to the Perception of Auditory Events*. The MIT Press.
- Hewlett, W. B. and Selfridge-Field, E., editors (2005). *Music Query: Methods, Models, and User Studies*. MIT Press.
- James, D. L. and Miikkulainen, R. (1995). SARDNET: a self-organizing feature map for sequences. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems 7*, pages 577–84, Cambridge, MA, USA. MIT Press.
- Kohonen, T. (1985). The self-organizing map. In *Proceedings of IEEE*, volume 78, pages 1464–1480.
- Minsky, M. (1988). *Society of Mind*. Simon & Schuster.
- Miranda, E. R. (2004). At the crossroads of evolutionary computation and music: Self-programming synthesizers, swarm orchestras and the origins of melody. *Evolutionary Computation*, 12(2):137–158.
- Mongeau, M. and Sankoff, D. (1990). Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175.
- Orpen, K. S. and Huron, D. (1992). Measurement of similarity in music: A quantitative approach for non-parametric representations. *Computers in Music Research*, 4:1–44.
- Povel, D. J. and Essens, P. (1985). Perception of temporal patterns. *Music Perception*, 2(4):411–441.
- Takeda, H., Nishimoto, T., and Sagayama, S. (2003). Automatic rhythm transcription from multiphonic midi signals. In *ISMIR 2003*.

Um Sistema Automático de Transcrição Melódica *

Adriano Mitre e Marcelo Queiroz

Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP)
Rua do Matão, 1010 – 05508-090 – São Paulo, SP – Brasil

amitre@linux.ime.usp.br, mqz@ime.usp.br

Abstract. *A system for the automatic transcription of melodies which implements state-of-the-art techniques of sinusoids estimation is presented. A novel technique, which benefits from robust partial estimates, is proposed for the estimation of fundamental frequency. Furthermore, system's flexibility allows it to serve also as a signal analysis tool, aiding in the production of plots such as sonograms and F0-grams.*

Resumo. *Um sistema de transcrição automática de melodias que implementa diversas técnicas de estimação robusta de senóides é apresentado. Partindo de estimativas confiáveis de parciais, uma nova heurística para determinação de frequência fundamental é proposta. Tendo sido concebido de maneira modular, o sistema pode ainda ser usado como ferramenta de análise de sinais, servindo à produção de gráficos como sonogramas e F0-gramas.*

1. Introdução

O problema da transcrição automática de melodias, embora estudado há vários anos (vide [Roads, 1996, cap.12] para um estudo taxonômico), dificilmente pode ser considerado um problema fechado. Uma variedade de aspectos técnicos e cognitivos fazem corresponder a cada solução proposta um novo conjunto de situações e exemplos (sintéticos ou naturais) em que a solução não fornece a resposta correta ou não fornece a resposta a tempo. Estes dois aspectos - robustez e eficiência - são fundamentais, por exemplo, durante uma performance interativa.

Dentre as dificuldades técnicas destaca-se o problema da resolução temporal versus resolução de frequência, que será discutido em detalhes na seção 6. Do lado cognitivo pode-se mencionar que a determinação da frequência fundamental em um fragmento de áudio é um problema que em geral não admite solução única/objetiva, mesmo de posse do espectro contínuo (com precisão infinita) do fragmento em questão. Todo método de solução é heurístico por natureza e tem maior ou menor grau de acerto de acordo com a afinidade do fragmento analisado com as hipóteses implícitas no método.

Isto posto, pode-se compreender o sistema aqui descrito como uma proposta robusta de solução do problema, fornecendo ao usuário diversas opções de técnicas de refinamento espectral e uma heurística nova, baseada em estimativas confiáveis de parciais, para a determinação de frequência fundamental.

O sistema tem como entrada um sinal de áudio monofônico (por exemplo uma gravação acústica) e produz uma transcrição de seu conteúdo melódico. No contexto deste artigo, transcrição significará a obtenção de uma representação simbólica, mais formalmente um conjunto de quádruplas da forma $(t_0, n, \Delta t, v)$, onde n denota a altura da

*O projeto teve apoio da FAPESP, projeto temático 02/02678-0.

nota, v sua intensidade, t_0 seu início e Δt sua duração. No restante do texto iremos nos referir a essas quádruplas como notas.

Nosso sistema possui três modos de impressão de notas: descritivo, *piano roll* e “MIDIcável”. O primeiro tem como saída um texto de fácil leitura, em que as notas são descritas com letras (notação latina) e oitavas em relação à central, por exemplo $E_b(-1)$. Já o segundo modo produz um arquivo que ao ser carregado no *gnuplot*¹ produz uma visualização das notas em estilo *piano roll*.

O terceiro modo, por sua vez, tem como saída um arquivo em formato CSV² que, ao ser processado pelo utilitário *MIDICSV*³, dará origem a um arquivo MIDI. Neste caso, o andamento, a forma de compasso e a pausa inicial serão fornecidos pelo usuário. É possível ainda, a partir do arquivo MIDI, obter partituras das transcrições com o uso de sistemas como o *Lilypond*⁴.

2. Arquitetura do Sistema

Uma parcela significativa dos sistemas de transcrição encontrados na literatura mesclam tratamento minucioso de alguns aspectos do problema e hipersimplificação de outros. O cuidado devotado a um certo aspecto é tipicamente determinado por sua proximidade com o foco da pesquisa. Como a praxe é apresentar resultados apenas do sistema como um todo, não é possível dimensionar o impacto individual de cada decisão de projeto.

Em vista disso, nosso sistema foi projetado como um conjunto de módulos independentes e desacoplados, ilustrados na Figura 1. Além de oferecer maior liberdade de experimentação, a modularidade de nosso sistema torna mais fácil identificar a causa de uma transcrição mal-sucedida. Por fim, amplia sua funcionalidade, permitindo fácil integração com outros sistemas por meio de *pipes*. Com efeito, a comunicação entre os diferentes módulos de nosso sistema pode ser realizada com *pipes*.

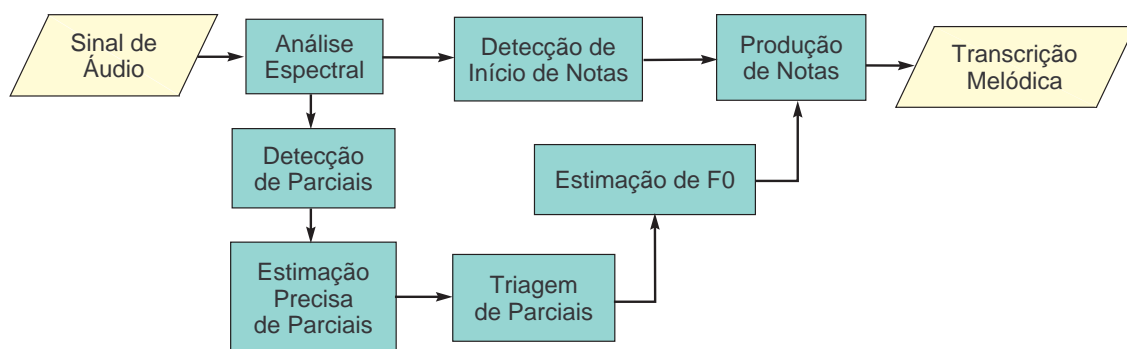


Figura 1: Módulos do sistema e seu fluxo de dados.

O sistema foi codificado na linguagem C padrão ANSI, escolhida em função de sua portabilidade e eficiência. Apesar de não ser uma linguagem orientada a objetos, conceitos deste paradigma foram freqüentemente considerados durante o desenvolvimento do sistema, além das boas práticas de programação [Kernighan e Pike, 2000].

¹disponível gratuitamente em <http://www.gnuplot.info>

²acrônimo, em inglês, para *valores separados por vírgula*.

³idem, <http://www.fourmilab.ch/webtools/midicsv>

⁴idem, <http://lilypond.org/web>

3. Modelo do Sinal

O sistema espera como entrada um sinal constituído por uma sucessão de notas e pausas (i.e., períodos de silêncio). A evolução dinâmica de uma nota musical pode ser descrita, na maior parte das entradas naturais, pelo envelope ADSR, que consiste em uma sucessão de quatro estágios: ataque, amortecimento, sustentação e dissipação. A inspeção de certos aspectos destes estágios é muito importante no projeto de um sistema transcritor. A Tabela 1 apresenta um resumo dessas características.

Tabela 1: Modelo evolutivo de uma nota típica.

Estágio	Energia	Caráter
Ataque	Cresce rapidamente	Transiente
Amortecimento	Decresce	Em estabilização
Sustentação	Permanece constante ou decresce lentamente	Periódico (som harmônico)
Dissipação	Decresce rapidamente	Periódico (som harmônico)

O modelo de sinal adotado no artigo baseia-se em [Serra e Smith III, 1990], mais especificamente na adaptação feita por [Desainte-Catherine e Marchand, 2000]. Essencialmente o sinal de áudio é decomposto como

$$s(t) = d(t) + e(t) \quad (1)$$

onde

- $d(t)$ é a parte determinística (componente de banda-estreita)
- $e(t)$ é a parte estocástica (componente de banda-larga)

A parte determinística é constituída por uma soma de osciladores senoidais com frequências e amplitudes que variam lentamente. Um oscilador de variação lenta é geralmente chamado de parcial, senóide ou componente senoidal. Portanto, a parte determinística pode ser formalizada como

$$d(t) = \sum_{p=1}^P \text{osc}(f_p(t), a_p(t)) \quad (2)$$

onde P é o número de parciais, dados por

$$\text{osc}(f_p(t), a_p(t)) = a_p(t) \cdot \cos(\phi_p(t)) \quad (3)$$

com

$$\frac{d\phi_p}{dt}(t) = 2\pi f_p(t) \quad \text{i.e.} \quad \phi_p(t) = \phi_p(0) + 2\pi \int_0^t f_p(u) \cdot du \quad (4)$$

Como esperamos sons harmônicos, sabemos que as frequências dos parciais relacionam-se de acordo com

$$f_p(t) = p \cdot f_1(t) \quad (5)$$

Deste modo, temos a garantia de que, para todo instante t , os parciais estarão distantes, no domínio da frequência, de pelo menos $f_1(t)$. Formalmente:

$$\min_{1 \leq i < j \leq P} \{|f_i(t) - f_j(t)|\} \geq f_1(t) \quad (6)$$

A garantia é expressa como desigualdade, e não igualdade, por duas razões. A primeira é que alguns instrumentos produzem séries harmônicas incompletas, como um

sintetizador de onda quadrada, que produz apenas harmônicos ímpares. A segunda razão é que em algumas famílias de instrumentos as frequências dos parciais desviam de maneira sistemática dos múltiplos da fundamental.

Nos instrumentos de corda, por exemplo, a frequência dos parciais é dada pela fórmula $f_n = n \cdot f_1 \sqrt{1 + B \cdot (n - 1)^2}$, onde B é o coeficiente de inarmonicidade. Este fenômeno decorre do fato de a velocidade do som em uma corda não ser constante, mas variar em função de sua espessura, comprimento e tensão (logo, B é determinado por estes parâmetros). O efeito psicoacústico da inarmonicidade em instrumentos de corda foi estudado em [Järveläinen et al., 2000].

Os instrumentos melódicos de percussão em membranas, como a marimba, o xilofone ou o vibrafone, divergem ainda mais acentuadamente do modelo harmônico, tanto pela frequência dos parciais quanto pela densidade das séries harmônicas. Apesar disso, experimentos indicam que nosso sistema é capaz de transcrever melodias executadas com estes instrumentos. Em primeiro lugar porque essas divergências não invalidam o espaçamento mínimo entre parciais no âmbito da frequência, necessário à precisa estimação dos mesmos. Em segundo, porque princípios psicoacústicos e de seletividade harmônica guiaram a construção de nosso estimador de F_0 (frequência fundamental), tornando-o mais condizente com a percepção de altura.

A parte estocástica é o que resta do sinal quando se lhe subtrai a parte determinística. Este sinal residual é plenamente descrito por sua densidade espectral de potência, que fornece a potência esperada em cada banda crítica. A justificação dessa afirmação foge ao escopo deste texto e pode ser encontrada em [Serra e Smith III, 1990].

Como se observa na Tabela 1, durante o ataque de uma nota, o sinal tem caráter transiente. A Transformação de Fourier, por sua vez, é apropriada para sinais periódicos. No entanto, o que poderia constituir um problema foi revertido em favor do sistema, que se utiliza de métricas de aperiodicidade do sinal para determinar, com maior exatidão, o início das notas (*note onsets*). Apesar de o sistema implementar diversas técnicas de detecção de início de nota (por exemplo [Duxbury et al., 2003]), foi necessário suprimir sua apresentação por conta do limite de páginas desta publicação.

4. Análise Espectral

O sistema obtém estimativas espectrais por meio da Transformada Discreta de Fourier de Curto Tempo (abreviada em inglês por STDFT). O espectro associado a um dado instante t é obtido da seguinte forma. Seja F_s^{-1} o período interamostral do sinal de entrada, medido em segundos. A duração de cada quadro (*frame*) a ser analisado será determinada pelo usuário, respeitando $\tau = N \cdot F_s^{-1}$ para algum inteiro positivo N . Então a sequência de N amostras contidas no intervalo de tempo $[t - \tau/2, t + \tau/2]$ será multiplicada ponto a ponto por uma função de suavização (também chamada de janela) e o resultado deste produto será por fim transformado pela STDFT. Se o valor de N for apropriado, o sistema realizará o cálculo por meio do algoritmo FFT (*Fast Fourier Transform*), que apresenta desempenho assintótico $O(N \lg N)$, contra $O(N^2)$ da implementação trivial da DFT.

Para um sinal de entrada de duração T segundos, o sistema calculará o espectro associado a todos os instantes $t \in [0, T]$ que satisfaçam, para algum inteiro m , a condição $t = t_0 + m \cdot \tau \cdot (1 - \alpha)$, onde $\alpha \in [0, 1)$ denota o fator de sobreposição escolhido pelo usuário. A parcela t_0 compensa o viés de quadros com número par de amostras, associando-os não ao instante da $N/2$ -ésima amostra, mas ao ponto médio entre as amostras $N/2$ e $N/2 + 1$. Portanto, se N for par, $t_0 = F_s^{-1}/2$. Caso contrário, $t_0 = 0$. Para os casos em que $t < \tau/2$ ou $t > T - \tau/2$ são requeridas amostras fora da extensão do

sinal de entrada. O sistema lida com a questão interpretando o sinal como precedido e sucedido por silêncio (amostras de valor nulo).

Os módulos descritos nas seções 5 a 8 operam em regime *por-quadro*. Por conta disso, informações referentes ao tempo serão omitidas nessas seções.

5. Detecção de Parciais

O módulo anterior fornece, para cada quadro, seu espectro complexo, cujo k -ésimo escaninho (*bin*) será denotado S_k . Ao escaninho de índice k corresponde a frequência $2\pi k/N$ radianos ou, equivalentemente, $2\pi F_s k/N$ Hertz. Para a estimação de F_0 , contudo, interessam apenas informações referentes aos parciais, e não todo o espectro.

Em condições razoáveis, espera-se que cada parcial no sinal de entrada produza um máximo local no espectro de magnitude. A recíproca, no entanto, não é verdadeira. Por conta disso, propuseram-se diversas heurísticas para discriminar entre máximos locais decorrentes de ruído daqueles produzidos por parciais. Uma estratégia adotada com frequência em sistemas de análise/resíntese é o acompanhamento evolutivo de parciais (*partial tracking*), como em [Lagrange et al., 2003], que não pôde ser incorporado ao sistema proposto por não satisfazer a exigência de operação em regime *por-quadro*. É fácil, contudo, acrescentar ao sistema um módulo de filtragem *offline* baseado nessa estratégia.

A heurística utilizada no sistema exige que cada máximo local no espectro de magnitude seja superior aos mínimos locais que lhe são adjacentes em pelo menos δ_{\min} decibéis. Formalmente, dado um $k \in \{2, \dots, N/2 - 1\}$, S_k será classificado como parcial somente se verificar

$$20 \cdot \log_{10} (|S_k|/|S_{k^+}|) \geq \delta_{\min} \quad \text{onde } k^+ = \max \{j > k : |S_k| \geq |S_{k+1}| \geq \dots \geq |S_j|\}$$

e também

$$20 \cdot \log_{10} (|S_k|/|S_{k^-}|) \geq \delta_{\min} \quad \text{onde } k^- = \min \{j < k : |S_j| \leq |S_{j+1}| \leq \dots \leq |S_k|\}.$$

6. Estimação Precisa de Parciais

Para a correta determinação de notas a partir das frequências fundamentais é necessária, na escala de doze notas com temperamento igual (i.e., o sistema de afinação padrão no ocidente), uma exatidão de frequência de pelo menos $F_{0\min} \cdot (\sqrt[12]{2} - 1)$ Hz, onde $F_{0\min}$ denota a frequência fundamental da nota mais grave da sequência melódica de entrada. Para se extrair nuances expressivas como vibratos e glissandos, entretanto, a precisão necessária é muito maior.

A resolução (i.e., capacidade de discriminação) de frequência da STDFT é de $L_w \cdot \tau^{-1}$ Hz, onde L_w denota a largura do lobo central da resposta de frequência da função de suavização, medida em escaninhos. A exatidão de frequência da STDFT é, por sua vez, dada por τ^{-1} e, portanto, difere da resolução apenas por um fator pequeno (tipicamente vale que $L_w \leq 4$). Deste modo, para que seja possível distinguir as notas de um violão de 6 cordas, cuja corda mais grave é convencionalmente afinada em 82,4 Hz, é necessário um quadro de pelo menos $(82,4 \cdot (\sqrt[12]{2} - 1))^{-1} \simeq 204$ ms.

Sinais musicais raramente possuem característica estacionária por tanto tempo. Assim, a utilização de quadros muito longos prejudica a resolução temporal, de modo que o conteúdo harmônico de duas ou mais notas consecutivas pode se misturar no espectro de um quadro. Além disso, uma precisão temporal de 20 ms exigiria um fator de

sobreposição de quadros de 90% e, portanto, uma carga computacional dez vezes maior. Em suma, quadros longos não são a abordagem correta.

Como o sinal de entrada é suposto monofônico, o espaçamento mínimo de frequência entre dois parciais quaisquer será de pelo menos $F0_{\min}$ Hz. Existem, para situações como essa, técnicas para obter estimativas mais precisas a partir de quadros com duração $L_w \cdot (F0_{\min})^{-1}$ s. Isto representa uma grande redução na exigência de tamanho do quadro. No exemplo do violão, um quadro de $2 \cdot (82,4)^{-1} \simeq 24$ ms passa a ser suficiente, considerando-se a janela de Hamming.

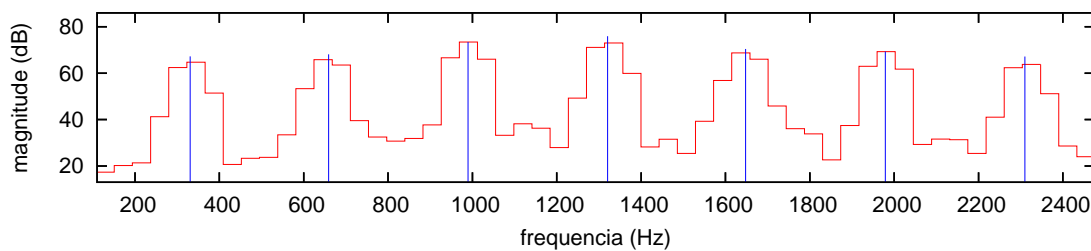


Figura 2: Espectro de magnitude e estimativas precisas dos parciais.

A superamostragem espectral, popularmente referida como *zero-padding*, é uma dessas técnicas. Como o nome sugere, este método consiste no aumento da taxa de amostragem do espectro, o que pode ser alcançado por meio do cálculo da STDFT do sinal suavizado (i.e., já multiplicado pela função de suavização) concatenado a uma sequência de zeros. A desvantagem dessa técnica é seu custo computacional: para aumentar a precisão em m vezes é necessário calcular uma STDFT com tamanho mN .

Outros métodos obtêm estimativas mais precisas dos parciais por meio de interpolação com coeficientes espectrais adjacentes. A interpolação parabólica pertence a essa categoria. Essa técnica beneficia-se do fato de que o lobo central da resposta de frequência da maioria das funções de suavização assemelha-se, em escala logarítmica, a uma parábola. Com os coeficientes de magnitude do pico e seus dois vizinhos, determina-se um polinômio do segundo grau, cujo ponto máximo dará origem à nova estimativa do parcial. Essa técnica é frequentemente reforçada pelo uso concomitante de superamostragem. É possível obter um desempenho ainda melhor com a aplicação de funções de suavização específicas, calculadas pela transformada inversa de uma parábola.

Especificamente para a janela de Hann⁵, Grandke desenvolveu um algoritmo de interpolação que utiliza o coeficiente vizinho de maior magnitude para obter estimativas mais consistentes do que as obtidas pela interpolação parabólica [Grandke, 1983].

Há técnicas mais sofisticadas de estimação de parciais, as quais consideram, além da magnitude, a informação de fase presente no espectro. Dentre essas, destacam-se o Método da Derivada [Desainte-Catherine e Marchand, 2000], que se baseia na relação entre os espectros do sinal e de sua derivada (aproximada com a aplicação de um filtro), e a Reatribuição Espectral [Kodera et al., 1978, Auger e Flandrin, 1995], que redistribui os pontos do reticulado tempo-frequência de acordo com os centros de energia de cada célula. Essas técnicas apresentam desempenho superior, mas possuem a desvantagem de exigirem o cômputo de STDFT adicionais. A Figura 2 exemplifica os resultados obtidos com a Reatribuição Espectral. A Tabela 2 lista as principais técnicas de estimação de parciais aplicáveis a sinais musicais e algumas de suas características.

⁵não obstante seja um tributo ao meteorologista austríaco Julius von Hann, alguns autores referem-se à janela como sendo de Hanning

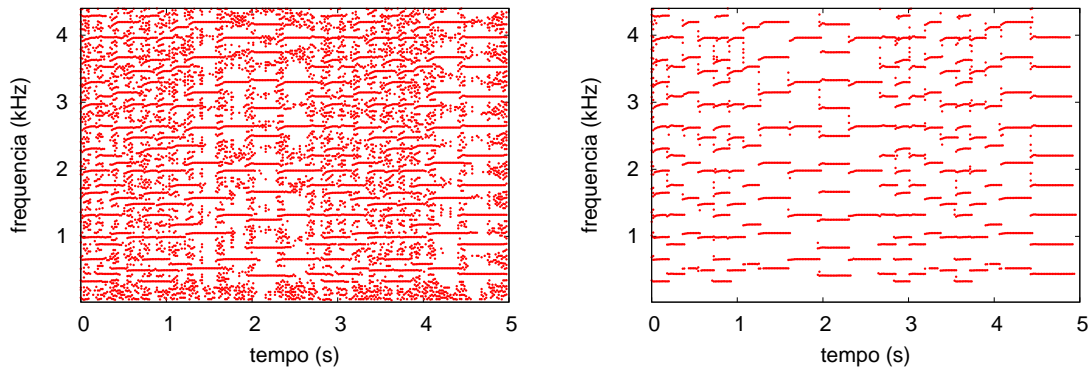
Tabela 2: Estimadores de parciais implementados e suas características.

Técnica	Custo mínimo por quadro	Custo adicional por pico	Função de suavização
Superamostragem de fator m	$O(mN \lg mN)$	$O(m)$	Qualquer
Reatribuição	$O(N \lg N)$	$O(1)$	Qualquer
Derivada	$O(N \lg N)$	$O(1)$	Qualquer
Interpolação de Grandke	$O(1)$	$O(1)$	Hann
Interpolação Parabólica	$O(1)$	$O(1)$	Melhor se específica

Estudos comparativos que investigam o erro médio, a variância e o viés dos principais estimadores de parciais podem ser encontrados em [Keiler e Marchand, 2002] e [Hainsworth e Macleod, 2003].

7. Triagem de Parciais

A quantidade de parciais produzidos pelo módulo anterior é tipicamente elevada, não raro da ordem de centenas. No entanto, nem todos precisam ou devem ser considerados para a estimação de F0. Por conta disso criou-se este módulo, cuja finalidade é filtrar parciais irrelevantes e picos espúrios, i.e., produzidos por ruído. O efeito deste processo pode ser apreciado no exemplo da Figura 3.

**Figura 3: Exemplo do efeito da triagem de parciais.**

Denotaremos por (f_i, a_i) o i -ésimo parcial, com frequência f_i Hertz e magnitude a_i decibéis. O conjunto de todos parciais será representado por Q . Por fim, $P \subseteq Q$ denotará o subconjunto dos parciais principais, obtidos após a triagem.

A triagem se dará em dois passos. O primeiro critério exige que o parcial tenha magnitude igual ou superior a α_{\min} e não inferior à do maior pico do quadro em mais do que Δ_{\max} . Além disso, exige-se que tenha frequência entre ϕ_{\min} e ϕ_{\max} . Os valores para α_{\min} , Δ_{\max} , ϕ_{\min} e ϕ_{\max} são fornecidos pelo usuário. Segue a formalização desse critério.

$$\text{Se } (f_i, a_i) \in P, \text{ então } \begin{cases} f_i \in [\phi_{\min}, \phi_{\max}] \\ a_i \geq \alpha_{\min} \\ a_{\kappa} - a_i \leq \Delta_{\max} \end{cases} \quad \text{onde } \kappa = \min \{i : a_i \geq a_j, \forall (f_j, a_j) \in Q\}$$

O segundo critério de filtragem foi baseado na observação de que a maior parte dos instrumentos produz séries harmônicas com envoltórias aproximadamente contínuas,

ou seja, séries em que a diferença de magnitude entre harmônicos consecutivos é pequena. Denotando por σ_{\max} a variação máxima permitida entre harmônicos sucessivos, segue a formalização deste critério.

$$\text{Se } (f_i, a_i) \in P, \text{ então } \begin{cases} i > \kappa \Rightarrow a_i + \sigma_{\max} \geq a_{i-} \\ i < \kappa \Rightarrow a_i + \sigma_{\max} \geq a_{i+} \end{cases}$$

$$\text{onde } i^- = \max \{j < i : (f_j, a_j) \in P\} \text{ e } i^+ = \min \{j > i : (f_j, a_j) \in P\}$$

8. Estimação de F0

O método que será proposto parte da suposição de que o parcial de máxima magnitude pertence à série harmônica principal, ou seja, é um múltiplo da frequência fundamental. Denotando por f_κ a frequência do parcial de máxima magnitude, nosso conjunto de candidatos à frequência fundamental é, então, dado por

$$C = \left\{ c_n \stackrel{\text{def}}{=} \frac{f_\kappa}{n} : 1 \leq n \leq \left\lfloor \frac{f_\kappa}{F0_{\min}} \right\rfloor \right\} \quad (7)$$

É preciso, neste ponto, estabelecer formas de comparar essas dezenas, eventualmente centenas de candidatos. De acordo com o princípio da seletividade harmônica, o primeiro passo neste sentido é coletar as séries harmônicas correspondentes a cada candidato. Denotando com uma seta para a esquerda a operação de atribuição (por exemplo, $x \leftarrow 7$ significa a variável x recebe o valor 7), apresentamos um algoritmo que cumpre a tarefa.

```

SÉRIE-HARMÔNICA( $f_{cand}, P$ )
1  for each  $(f, a) \in P$ 
2    do  $i \leftarrow \lfloor f/f_{cand} + 0.5 \rfloor$ 
3       $d \leftarrow \text{DESVIO-RELATIVO}(f_{cand} \cdot i, H[i]_{freq})$ 
4       $\hat{d} \leftarrow \text{DESVIO-RELATIVO}(f_{cand} \cdot i, f)$ 
5      if  $\hat{d} \leq \min \{d, \sqrt[24]{2}\}$ 
6        then if  $\hat{d} < d$  or  $a > H[i]_{mag}$ 
7          then  $H[i]_{freq} \leftarrow f$ 
8               $H[i]_{mag} \leftarrow a$ 
9  return  $H$ 

```

Se o i -ésimo harmônico do n -ésimo candidato à F0 estiver presente no espectro, sua magnitude e frequência estarão, ao fim da execução do algoritmo, em $H[n][i]_{mag}$ e $H[n][i]_{freq}$. Caso contrário valerá que $H[n][i]_{mag} = H[n][i]_{freq} = 0$.

Faz-se necessário, então, um modo de quantificar a proeminência de cada candidato em função de sua série harmônica. Isto será feito levando em conta princípios psicoacústicos, em particular o conceito de banda crítica [Roederer, 2002, seção 3.4].

As funções Φ e Ψ , definidas a seguir, são adaptações do modelo de soma harmônica apresentado em [Klapuri, 2004, seção 6.3.3]. A motivação psicoacústica das fórmulas pode ser encontrada na referência e não será reproduzida aqui.

Formalmente, a proeminência do n -ésimo candidato será dada por

$$\Phi(n) = \sum_{i=1}^{I(n)} H[n][i]_{mag} \cdot \Psi(i) \quad (8)$$

$$\text{onde } I(n) = \max \{j \in \{1, \dots, |H|\} : H[n][j]_{mag} > 0\} \quad (9)$$

e $\Psi(i)$ denota a fração de banda crítica que corresponde a cada harmônico em função de seu índice, calculada como

$$\Psi(i) = \begin{cases} 1, & \text{se } i = 1 \\ \min \left\{ \log_{2^{1/3}} \left(i \cdot \sqrt{\frac{i+1}{i}} \right) - \log_{2^{1/3}} \left((i-1) \cdot \sqrt{\frac{i}{i-1}} \right), 1 \right\}, & \text{c.c.} \end{cases} \quad (10)$$

De posse da proeminência dos candidatos, a determinação de F0 será feita em dois estágios. Primeiramente, os candidatos com proeminência relativa à máxima de pelo menos $\beta \in [0, 1]$, pré-fixado, são selecionados. Dentre estes, aquele que possuir a maior média ponderada χ , definida a seguir, é eleito F0. Caso haja empate, vence o candidato de menor frequência. Formalmente, será considerado F0 o candidato de índice φ dado pelas seguinte equações

$$\varphi = \min \{n : c_n \in C^\Phi \text{ e } \chi(n) \geq \chi(m), \forall c_m \in C^\Phi\} \quad (11)$$

$$\chi(n) = \frac{\sum_{i=1}^{I(n)} H[n][i]_{mag} \cdot \Psi(i)}{\sum_{i=1}^{I(n)} \Psi(i)} \quad (12)$$

$$C^\Phi = \left\{ c_n \in C : \Phi(n) \geq \beta \cdot \max_{m|c_m \in C} \{\Phi(m)\} \right\} \quad (13)$$

Com isso, já se tem uma estimativa da frequência fundamental. No entanto, seu valor exato baseou-se na estimativa de frequência de um único parcial, o de máxima magnitude. Ainda que este tenha a estimativa isoladamente mais confiável, um estimador de F0 ainda mais robusto pode ser obtido combinando-se as estimativas dos diversos harmônicos.

Como os estimadores de parciais são essencialmente não-viesados, espera-se que os erros individuais se anulem ao serem acumulados. Sabe-se que a qualidade da estimativa de cada parcial depende, dentre outros fatores, de sua razão sinal/ruído (abreviada em inglês por SNR) e de quão estável encontra-se sua frequência absoluta. Por conta disso, devem ser privilegiados os harmônicos de maior magnitude, que possuem maior SNR, e de menor índice, uma vez que variações na frequência fundamental são multiplicadas pelo índice do harmônico. Considerando estes princípios, desenvolvemos uma fórmula para re-estimação da frequência fundamental:

$$F0' = \frac{\sum_{i=1}^{I(n)} H[i]_{freq}/i \cdot H[i]_{mag} \cdot \Psi(i)}{\sum_{i=1}^{I(n)} H[i]_{mag} \cdot \Psi(i)} \quad (14)$$

onde $H[i]$ denota o i -ésimo parcial da série harmônica de c_φ , ou seja, $H[i] \stackrel{def}{=} H[\varphi][i]$.

9. Produção de Notas

Informações referentes ao tempo foram omitidas nas seções 5 a 8, em que se descreviam operações realizáveis individualmente para cada quadro. A produção de notas, no entanto,

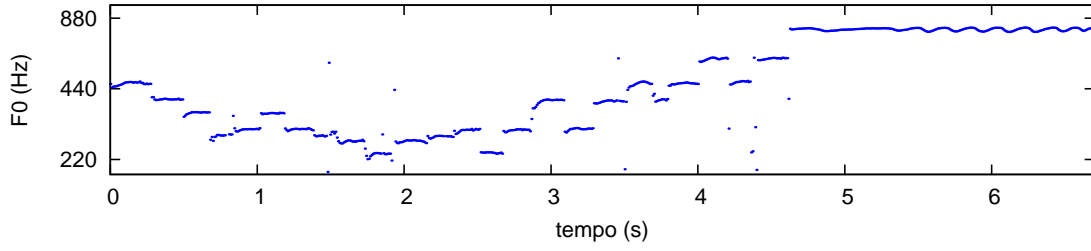


Figura 4: Exemplo de saída do módulo de estimação de F0.

não faz sentido em regime *por-quadro*. Conseqüentemente reaparecerão, nesta seção, informações temporais.

O funcionamento do módulo é descrito pelo algoritmo *Produz-Notas*, em que $freq[Fund[i]]$ denotará a i -ésima estimativa de F0, associada a um quadro centrado no instante $inst[Fund[i]]$. Além disso, $intens[Fund[i]]$ denotará a proeminência da i -ésima estimativa, calculada pela Equação 8. Além de uma seqüência de estimativas de F0 (Figura 4), o módulo de produção de notas recebe o conjunto T dos instantes de início de nota.

O atraso máximo na detecção de F0, i.e., o tempo máximo decorrido entre um início de nota e a detecção da F0 que lhe corresponde, será denotado por σ_{max} . O maior intervalo aceitável sem constatações de F0 no “interior” de uma nota será representado por γ_{max} . Finalmente, Δ_{min} denotará a duração da nota mais curta permitida e ν_{min} , a intensidade mínima aceitável.

O módulo de produção de notas tem essencialmente três finalidades. A primeira é evitar que irregularidades na estimação de F0 ocasionem notas espúrias. A segunda é eliminar o atraso decorrente do fato de as notas muitas vezes não possuírem, em seus estágios iniciais, altura (e, portanto, F0) bem definida. A terceira é impedir que uma seqüência de notas repetidas seja interpretada como uma única nota.

PRODUZ-NOTAS($Fund, T$)

```

1  for  $i \leftarrow 1$  to  $tamanho[Fund]$ 
2      do  $n \leftarrow \lfloor \log_{12\sqrt{2}} (freq[Fund[i]] / 440) + 0.5 \rfloor + 69$   $\triangleright$  número de nota MIDI
3      if  $ultimaConstatacao[Nota[n]] \neq INDEFINIDA$ 
4          then if  $inst[Fund[i]] - ultimaConstatacao[Nota[n]] > \gamma_{max}$ 
5              then if  $intens[Nota[n]] > \nu_{min}$  and  $duracao[Nota] > \Delta_{min}$ 
6                  then IMPRIME( $Nota[n]$ )
7                      REINICIA( $Nota[n]$ )
8      if  $inicio[Nota[n]] = INDEFINIDO$ 
9          then  $t \leftarrow \max \{T[i] \leq inst[Fund[i]]\}$ 
10         if  $inst[Fund[i]] - t \leq \sigma_{max}$ 
11             then  $inicio[Nota[n]] \leftarrow t$ 
12             else  $inicio[Nota[n]] \leftarrow inst[Fund[i]]$ 
13          $ultimaConstatacao[Nota[n]] \leftarrow inst[Fund[i]]$ 
14          $intens[Nota[n]] \leftarrow \max \{intensidade[Fund[i]], intens[Nota[n]]\}$ 
15          $duracao[Nota[n]] \leftarrow ultimaConstatacao[Nota] - inicio[Nota]$ 
16     for  $m \leftarrow 1$  to  $tamanho[Nota]$ 
17         do if  $inst[Fund[i]] - inicio[Nota[m]] > \gamma_{max}$  or  $i = tamanho[Fund]$ 
18             then if  $intens[Nota[m]] > \nu_{min}$  and  $duracao[Nota[m]] > \Delta_{min}$ 
19                 then IMPRIME( $Nota[m]$ )
20                 REINICIA( $Nota[m]$ )

```

10. Resultados e Discussão

Como o sistema teve sua implementação concluída recentemente, os experimentos realizados até o momento não foram muito numerosos e visaram a uma análise essencialmente qualitativa. Nesses experimentos constatamos que os parâmetros fornecidos pelo usuário têm influência decisiva sobre os resultados, de modo que a realização de testes estatisticamente relevantes pressupõe sua otimização.

Não obstante, os experimentos apresentaram resultados animadores. Testes realizados com sinais sintéticos foram extremamente bem sucedidos, muitos dos quais resultaram em transcrições perfeitas. As melodias foram sintetizadas com diversos timbres, indicando que as hipóteses adotadas pelo sistema não estão vinculadas a peculiaridades de determinada família de instrumentos.

Previsivelmente, os testes com gravações acústicas reais foram mais desafiadores. Recursos expressivos, por exemplo inflexões, não raro levam a cenários cuja desambiguação depende de conhecimentos musicológicos. Considerando que o sistema baseou-se apenas em técnicas de processamento de sinais, seu desempenho em tais circunstâncias mostrou-se bastante satisfatório.

Constatamos ainda que a presença de efeitos, naturais ou artificiais, degrada o resultado das transcrições. Em gravações com forte reverberação ou eco, há situações de polifonia artificial, em que uma nota soa simultaneamente com rastros das que a antecederam.

O sistema foi compilado e executado com igual sucesso em ambientes Linux e Windows. Além disso, foi capaz de produzir transcrições em tempo inferior à duração das gravações, demonstrando potencial para aplicações de tempo-real.

Infelizmente, não há uma base de dados universalmente aceita para testar sistemas de transcrição automáticos, o que dificulta a comparação com outras soluções encontradas na literatura. Apesar disso, pretendemos realizar testes sistemáticos em um corpo representativo de gravações acústicas e sinais sintetizados, divulgando os resultados tão logo estejam disponíveis.

11. Conclusão

Neste trabalho apresentamos um sistema de transcrição automática de melodias com desempenho bastante promissor. Por ser modular, o sistema se presta tanto a aplicações típicas de sistemas de transcrição, por exemplo *query-by-humming*, quanto à integração em sistemas maiores, por exemplo de análise-manipulação-resíntese.

Adicionalmente, o sistema pode fornecer resultados de análise espectral, estimativas de parciais e de frequência fundamental, entre outros, em formato textual próprio para ferramentas de impressão de gráficos ponto-a-ponto, tal como o *gnuplot*. Com isso, torna-se útil na produção de, por exemplo, sonogramas e F0-gramas. Em particular, os gráficos desta publicação foram produzidos desta forma.

O sistema está disponível em www.mitre.com.br/asymut/, onde se encontram também exemplos em que se pode comparar a gravação original com a resintetizada a partir da transcrição.

Graças à modularidade do sistema, um desenvolvedor interessado terá facilidade em acrescentar novas técnicas de refinamento espectral, heurísticas para determinação de fundamental, interface gráfica, etc.

Referências

- Auger, F. e Flandrin, P. (1995). Improving the readability of time-frequency and time-scale representations by the reassignment method. *IEEE Transactions on Signal Processing*, 43(5):1068–1089.
- Desainte-Catherine, M. e Marchand, S. (2000). High Precision Fourier Analysis of Sounds Using Signal Derivatives. *Journal of the Audio Engineering Society*, 48(7/8):654–667.
- Duxbury, C., Bello, J. P., Davies, M., e Sandler, M. (2003). Complex Domain Onset Detection for Musical Signals. In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, Londres, Reino Unido.
- Grandke, T. (1983). Interpolation algorithms for discrete Fourier transforms of weighted signals. *IEEE Trans. Instr. and Meas.*, 32(2):350–355. 1983.
- Hainsworth, S. e Macleod, M. (2003). On Sinusoidal Parameter Estimation. In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, Londres, Reino Unido.
- Järveläinen, H., Verma, T., e Välimäki, V. (2000). The effect of inharmonicity on pitch in string instrument sounds. In *Proc. International Computer Music Conference*, Berlin, Germany.
- Keiler, F. e Marchand, S. (2002). Survey On Extraction of Sinusoids in Stationary Sounds. In *Proc. of the 5th Int. Conference on Digital Audio Effects (DAFx-02)*, Hamburgo, Alemanha.
- Kernighan, B. W. e Pike, R. (2000). *A Prática da Programação*. Editora Campus, primeira edição.
- Klapuri, A. (2004). *Signal Processing Methods for the Automatic Transcription of Music*. tese de PhD, Tampere University of Technology.
- Kodera, K., Gendrin, R., e de Villedary, C. (1978). Analysis of time-varying signals with small BT values. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):64–76.
- Lagrange, M., Marchand, S., Raspaud, M., e Rault, J.-B. (2003). Enhanced Partial Tracking Using Linear Prediction. In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, Londres, Reino Unido.
- Roads, C. (1996). *The Computer Music Tutorial*. The MIT Press.
- Roederer, J. G. (2002). *Introdução à Física e Psicofísica da Música*. EdUSP, São Paulo, primeira edição. Tradução Alberto Luis da Cunha.
- Serra, X. e Smith III, J. O. (1990). Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition. *Computer Music Journal*, 14(4):12–24.

A computational model of harmonic chord recognition

Riana Walsh¹, Dr Donncha O' Maidin¹

¹Department of Computer Science & Information Systems, University of Limerick,
Ireland

{riana.walsh,donncha.omaidin}@ul.ie

Abstract. *This computational model of harmonic chord recognition investigates the perception of harmonic chords by peripheral auditory processes and auditory grouping. The frequency selectivity of the auditory system is modeled using a bank of overlapping band-pass filters and a model of inner hair cell dynamics. The periodicity in each frequency channel is determined by autocorrelation. These periodicities are grouped according to pitch class. By computing the intervals between the high priority pitch classes, the model achieves considerable success in recognizing major, minor, dominant seventh, diminished and augmented chords. Its performance is evaluated on chords from instrument samples as well as from an ensemble recording.*

1. Introduction

In the auditory system, harmonics belonging to the same fundamental frequency tend to fuse and be heard as a single sound with a pitch corresponding to that fundamental frequency.

Two harmonic notes form the interval of an octave when their fundamental frequencies are in the ratio 2:1. Western music theory assumes octave equivalence: notes one or more octaves apart have the same harmonic function [Parncutt 1989]. These notes belong to the same pitch class. In Western music theory the harmonic triad is built out of three pitch classes. Both major and minor triads contain the perfect fifth interval. The diminished triad contains the diminished fifth interval (two minor 3rd intervals) and the augmented triad contains the augmented fifth (two major 3rd intervals). Along with perceiving each individual pitch in the triad we perceive the quality of the triad as a whole. Bregman [Bregman 1990] mentions that music may be thought of in terms of a horizontal and a vertical dimension. The horizontal dimension is time and the vertical dimension is pitch relations or harmonies. "The vertical organisation gives us not only the experience of chords but also other emergent qualities of simultaneous sounds, e.g. timbre, consonance and dissonance [Bregman 1990]." A chord can be thought of as a global entity that has its own emergent properties; the chord as a whole becomes the acoustic object we hear.

2. The auditory filters

The spectral analysis carried out by the basilar membrane is modeled using a parallel bank of 160 overlapping gammatone filters, each tuned to a different frequency [Patterson et al 1992]. A Matlab implementation is supplied by Malcolm Slaney [Slaney 1993]. The time domain impulse response of the gammatone filter is:

$$gt(t) = a t^{(n-1)} \exp(-2\pi bt) \cos(2\pi ft + \theta); (t > 0)$$

The bandwidth of the filter is determined by b , and n is the order of the filter and so controls the slope of the skirts. The bandwidth of each filter is described by the Equivalent Rectangular Bandwidth (ERB). This is defined as [Glasberg and Moore 1990]:

$$ERB = 24.7(4.37f/1000 + 1)$$

With $n = 4$ then the filter bandwidth, $b = 1.019 \cdot ERB$ in the current implementation [Patterson et al 1992, Slaney 1993]. The ERB increases with increasing channel centre frequency.

3. The hair cell model

Meddis [Meddis 1985, 1987] described a computational model of mechanical to neural transduction at the hair cell-auditory-nerve fibre synapse. The output excitation function in response to an acoustic stimulus is a stream of spike events precisely located in time. The model describes the production, movement and dissipation of transmitter substance in the region of the hair cell-auditory-nerve fibre synapse. Meddis [Meddis et al 1990] published a computer program of the hair cell model. This program has been implemented in Matlab by Malcolm Slaney [Slaney 1993]. The output of each filter is passed to the inner hair cell model. The model implements one hair cell per frequency channel. The model assumes that the probability of a spike occurrence is linearly related to the amount of transmitter substance in the synaptic cleft between the inner hair cell and its corresponding auditory nerve.

The following equations are computed at each time interval dt and define the operation of the model. The hair cell contains a quantity of free transmitter, $q(t)$. A fraction of this transmitter is released between time t and time $t+dt$ across the membrane into the synaptic cleft.

$$\text{Fraction of transmitter released} = k(t)q(t)dt$$

The permeability of the membrane, $k(t)$, fluctuates as a function of the instantaneous amplitude of the acoustic stimulus, $s(t)$.

$$k(t) = g[s(t) + A]/(s(t) + A + B), \text{ with } [s(t) + A] > 0$$

$$k(t) = 0, \text{ for } [s(t) + A] \leq 0, \text{ A and B are constants } B > A.$$

When $s(t) = 0$ (no acoustic stimulation), spontaneous activity in the auditory nerve is allowed for by $gA/(A+B)$, the transmitter leak rate. In this instance $k(t)$ varies between 0 and g . B is the rate at which $k(t)$ approaches its maximum value as a function of $s(t)$ and $-A$ defines the lowest instantaneous amplitude at which the membrane remains permeable. The synaptic cleft contains a fluctuating amount of transmitter substance $c(t)$. The amount of transmitter lost is:

$$lc(t)dt$$

The amount of transmitter substance returned to the hair cell is:

$$rc(t)dt$$

Following the reuptake, $rc(t)dt$, the transmitter is subject to reprocessing delays and forms a reprocessing store, w .

$$dw/dt = rc(t) - xw(t)$$

A fraction $xw(t)dt$ of the transmitter in this store is continuously released into the free transmitter pool. Free transmitter is replenished at a rate of $y[M-q(t)]$. M is the maximum amount of transmitter in the transmitter pool. The cell transmitter level $q(t)$ is replenished through manufacture and return from the cleft but is depleted by loss into the cleft.

$$dq/dt = y[M-q(t)] + xw(t) - k(t)q(t)$$

The cleft transmitter level $c(t)$ is replenished by release from the hair cell and depleted by loss and return to the hair cell.

$$dc/dt = k(t)q(t) - lc(t) - rc(t).$$

Spike occurrence in the auditory nerve is assumed to be linearly and probabilistically related to the residue of transmitter substance left in the cleft after the loss and reuptake. The probability of a spike occurring is:

$$\text{Prob (event)} = hc(t)dt, h \text{ is a constant.}$$

The parameters of this model were optimised by Meddis [Meddis 1987] in order to simulate as closely as possible known responses at the hair cell-auditory-nerve synapse. The parameters of the model are set to simulate a high spontaneous-rate fibre. A feature of the model is its ability to simulate the decrease in phase locking for high frequency stimuli. The rate at which the transmitter is cleared from the cleft, loss and reuptake, is linked to the models ability to reflect the fine structure of the stimulus. It was noted that although this movement of transmitter causes a decrease in phase locking in the model it is not necessarily a true explanation of phase locking. Phase locking is less evident when this rate is slow relative to stimulus frequency.

4. Periodicity detection

The next stage of the model uses autocorrelation to detect periodicity in each frequency channel. The autocorrelation of a sequence $x[n]$ is:

$$a[m] = 1/Q \sum_{n=0}^{n=Q-m-1} x[n]x[n+m]$$

Q is the length of the data sequence and m is the autocorrelation lag. For $m=0$, $a[m]$ is a calculation of the total energy in the signal. The autocorrelation of a sequence defines how similar a signal is to a time-delayed version of itself. This similarity is greatest when the time delay is zero (the zero lag point). If the signal is periodic then the autocorrelation value at some non-zero value of the lag is close in value to the zero lag. The autocorrelation pattern of a periodic signal is also periodic. The periodicity value of the signal is indicated by the position of the first maximum after the zero lag maximum.

The frequency selectivity of the auditory system refers to its ability to resolve the frequency components of a complex sound. The ear has limited frequency resolution. The discrimination by the auditory system of individual frequency components in the sound depends in part on the frequency resolution of the basilar membrane. Resolution on the basilar membrane is generally described in terms of the critical band. If the action of the basilar membrane is thought of in terms of overlapping

band-pass filters, then the ERB provides a practical measure of the critical band. The waveform of a frequency component that is resolved by a filter is a sine wave of that frequency. The frequency channel is resolved. When more than one harmonic, or frequency component, passes through a filter the output pulses at a rate equal to the frequency difference between the adjacent harmonics. There is a modulation present in the amplitude envelope at the output of the filter. This is an unresolved frequency channel. For a single harmonic sound the periodicity in the amplitude modulation is equal to the fundamental frequency. Autocorrelation is used to determine the main channel periodicity and the periodicity in the amplitude modulation.

In the next stage the envelope of the autocorrelation pattern for each frequency channel is extracted. The envelope of the autocorrelation pattern is the contour formed by the maxima in the pattern. For a resolved frequency channel the envelope pattern forms a straight line. If the frequency channel is unresolved there is a modulation present in the envelope pattern. The least squares method is used to detect envelope modulation in the autocorrelation pattern. A straight line is fit to the envelope of the autocorrelation maxima in each frequency channel. For a channel that is resolved, the straight line will fit quite well to the envelope maxima. For a channel that is unresolved the contour of the envelope maxima will deviate from a straight line. The correlation coefficient is used as a measure of how far the envelope contour of the maxima deviates from the straight line; it indicates the amount of envelope modulation in each frequency channel. A correlation coefficient value of 0.997 was chosen as a threshold value to indicate the presence or absence of envelope modulation. A value greater than this threshold value indicates that no envelope modulation is present. The closer the envelope of the maxima is to a straight line the less modulation there is in the channel. Frequency channels are divided into those with and without envelope modulation.

If the envelope modulation is periodic, its periodicity value is calculated. The criterion for periodicity used is success in a heuristic search for the first maximum which has corresponding maxima at integer multiples of its lag value in the autocorrelation pattern.

5. Interval and chord identification

This algorithm identifies the major, minor, diminished, augmented triads and the dominant 7th chord. A harmonic triad is made up of three pitch classes the root, third and fifth of the triad. More than three notes in the triad chord indicates an octave doubling. For major and minor triads the interval between the root and the fifth is a perfect fifth. For the major triad the interval between the root and the third is a major third and for the minor triad the corresponding interval is a minor third. The diminished triad can be considered as consisting of a minor third interval and a diminished 5th interval from the root, or two adjacent minor third intervals. The augmented triad is made up of a major third interval and an augmented 5th interval from the root or two adjacent major 3rd intervals. The dominant 7th has four distinct pitch classes and consists of a major triad with an added minor 7th interval above the root.

In general if the ratio between the two fundamentals of a musical interval is $m:n$, with m and n integers, every n th harmonic of the first fundamental will correspond in frequency with every m th harmonic of the second. When the ratio is small integers, such as 2:1 in the octave interval, there will be many exact correspondences of

frequencies. However in equal tempered tuning, which was formed out a requirement for equally spaced intervals (in terms of frequency ratio) regardless of tonality, all intervals (apart from the unison and octave) match approximately to the corresponding interval found in the harmonic series. The difference between the nearly coinciding harmonics is small, within 5% of the critical bandwidth [Howard and Angus 2001].

The frequency channels are grouped according to pitch class. The envelope modulation periodicity and the main channel periodicities are collated. Each pitch class is assigned a priority based on the number of channels with that periodicity. The highest priority pitch class is the one that occupies the maximum number of channels. For a single harmonic sound, or an octave interval, the highest priority pitch class is the pitch class to which the fundamental frequency belongs. This is because the fundamental frequency appears as a beat periodicity in addition to possibly being present as a resolved frequency component.

The periodicities corresponding to the root, third and fifth generally appear as high priority pitch classes. The interval size in semitones is computed between all high priority pitch classes, searching for the fifth, and then for the major or minor third.

6. Results

This model was tested on chords created from instrument samples taken from the McGill University master sample (MUMS) CD set and on a recording of the second movement of Bach's Brandenburg Concerto No. 2 in F major. The instruments in the recording are recorder, oboe, violin, cello and harpsichord. From both sets of results it can be seen that the model performs well in identifying chords created from different harmonic instruments.

Table 1 holds the results of the instrument sample chords. The individual instrument samples were not balanced for equal loudness when creating the chords but kept at the original sample level. The durations were measured from the start of the instrument sample. Table 2 shows results for chords taken from the second movement of J.S. Bach's Brandenburg Concerto. The analysis was run on single time frames with starting time points corresponding as close as possible to one fifth, two fifths, three fifths and four fifths of the duration of the sound. The model performed well on major, minor and augmented chords. The performance of the model was less reliable on the dominant 7th and diminished chords. It may be noted that the diminished chord forms part of a dominant 7th chord (i.e. C, E, G, Bb is a dominant 7th and E, G, Bb is a diminished chord).

In table 1 four out of the seven chords were recognized correctly in all the time frames tested. Every chord was identified correctly in the first time frame. In the second and third time frame four out of the seven were correct. In the final time frame six out of the seven were correctly recognized. The G major dominant 7th chord was incorrectly identified in the second and third time frame. In the second time frame it was identified as a minor triad with root B. The F# is the third and sixth harmonic of the oboe B and is strong in this time frame. This same error occurred with the diminished triad in the second and third time frames. For the A major dominant 7th chord the identification as a C# minor triad (C#, E, G#) arises because of the presence of a strong G#, a third and sixth harmonic of the C# on the oboe, and the E is already present to complete the triad.

In the second table three out of the eleven chords were correctly recognised in all time frames tested. In the first and second time frames five and six out of eleven were correctly recognised. In the third and fourth time frame ten out of eleven chords were correctly identified. The second D minor chord in table 2 was identified as an F major chord in the first two time frames. This is due to the C, which is present as the third and sixth harmonic of F and occurs in more channels than D, for these two time frames. The third D minor chord in table 2 was identified as a major chord with root note Bb. The Bb arises as a difference periodicity in the unresolved channels. Both D and F are strong periodicities in this time frame. A and Bb occur with equal strength. In the G major dominant 7th chord analysis the F# is in a significant number of channels. The F# is a third and sixth harmonic of B. Some harmonics present may be stronger than the fundamental frequency.

Table 1. Results of chords built from recorded instrument samples

Chord type and duration (s) of chord	Instruments	Starting point (of duration) for analysis			
		1/5	2/5	3/5	4/5
F major (0.6s)	cello C3, violin F5, oboe A5	Major, root F	Major, root F	Major, root F	Major, root F
G minor (0.5s)	cello G3, violin Bb5, flute D6	Minor, root G	Minor, root G	Minor, root G	Minor, root G
G major dominant 7 th (0.5s)	cello G2, violin F5, oboe B4, flute D6	Major, root G dominant 7 th	*minor, root B	*undefined	Major, root G dominant 7 th
A major dominant 7 th (0.9s)	Cello A3, oboe C#4, piano G4, flute E6	Major, root A dominant 7 th	Major, root A	Major, root A	*Minor, root C#
Diminished B, D, F (0.9s)	oboe B4, flute D6, violin F5	Diminished B, D, F	*Minor triad, root B	*Minor triad, root B	Diminished B, D, F
Augmented D, F#, Bb (0.9s)	oboe D4, piano F#3, violin Bb5	Augmented D, F#, Bb	Augmented D, F#, Bb	Augmented D, F#, Bb	Augmented D, F#, Bb
Augmented C#, A, F (0.9s)	oboe C#4, piano F4, oboe A5	Augmented C#, F, A	Augmented C#, F, A	Augmented C#, F, A	Augmented C#, F, A

Table 2. Results of chords from the 2nd movement of Bach's Brandenburg Concerto No. 2 in F major

Chord type and duration (s) of chord	Bar number	Starting point (of duration) for analysis			
		1/5	2/5	3/5	4/5
D minor 0.7s (cello D3, violin A4, oboe F5)	29 (3 rd beat)	Minor, root D	Minor, root D	Minor, root D	Minor, root D
D minor 0.5s (cello D3, oboe F4, recorder A5)	7 (3 rd beat)	*Major, root F	*Major, root F	Minor, root D	Minor, root D
F major 0.6s (cello C3, violin F5, oboe A5)	35 (1 st beat)	*Minor, root C	Major, root F	Major, root F	Major, root F
G minor 0.4s (cello D3, oboe G5, recorder Bb5)	37 (1 st beat)	Minor, root G	Minor, root G	Minor, root G	Minor, root G
G minor 0.5s (cello G3, violin Bb5, recorder D6)	37 (3 rd beat)	*Major, root C	*Undefined	Minor, root G	Minor, root G
C major dominant 7 th 0.3s (cello C3, violin E5, oboe Bb4, recorder G5)	18 (3 rd beat)	*Undefined	Major, root C, dominant 7 th	Major, root C, dominant 7 th	Major, root C, dominant 7 th
G major dominant 7 th 0.5s (cello G2, violin F5, oboe B4, recorder D6)	22 (3 rd beat)	*Minor, root B	*diminished B, D, F	*Minor, root B	Major, root G
A major dominant 7 th 0.3s (cello A3, violin C#6, oboe G4, recorder E6)	28 (3 rd beat)	Major, root A, dominant 7 th	Major, root A, dominant 7 th	Major, root A, dominant 7 th	Major, root A, dominant 7 th
D minor 0.6s (cello D3, violin A4, recorder F5)	57 (3 rd beat)	Minor, root D	*Major, root Bb	Minor, root D	Minor, root D
Diminished G#, F, B 0.6s (cello G#2, oboe F4, recorder B5)	8 (1 st beat)	Diminished G#, F, B	*undefined	Diminished G#, F, B	Diminished G#, F, B
Augmented A, 0.2s, C#, F (cello A2, violin C#6, oboe A4, recorder F6)	28 (3 rd beat)	Major, root A	Augmented A, C#, F	Augmented A, C#, F	Augmented A, C#, F

7. Conclusions

The performance of a computational model of harmonic chord recognition has been demonstrated. The model was evaluated on a set of chords created from instrument samples and on chords from a recording of the second movement of J.S. Bach's Brandenburg Concerto No. 2. The results demonstrate that the model works well on harmonic chords created from different instruments. Results are good for the major, minor and augmented chords but the performance is less reliable in the case of the dominant 7th and diminished chord. The results for the chords from the Bach recording tend to improve towards the middle and end of the sound (the last two time frames). Errors at the start of the chord may be due to transients or some reverberation of the previous harmony in the recording. In the case of chords created from the instrument samples errors did not occur at the start of the chords.

References

- Bregman A. S. (1990) 'Auditory Scene analysis: The perceptual organisation of sound.' Cambridge, MA: MIT Press.
- Hartmann W. M. (1998) 'Signals, Sound, and Sensation.' Copyright © 1998 Springer-Verlag New York, Inc.
- Howard D.M and Angus J. (2001) 'Acoustics and Psychoacoustics.' Second Edition, Music Technology Series, Focal Press.
- Meddis R. (1986) 'Simulation of mechanical to neural transduction in the auditory receptor.' J. Acoust. Soc. Am. 79 (3), March 1986
- Meddis R. (1988) 'Simulation of auditory-neural transduction: Further studies.' J. Acoust. Soc. Am. 83 (3), March 1988.
- Meddis R., Hewitt, M. J. and Shackleton T. M. (1990) 'Implementation details of a computation model of the inner hair-cell/auditory-nerve synapse.' J. Acoust. Soc. Am. 87 (4), April 1990.
- Meddis R. and Hewitt M. J. (1991) 'Virtual pitch and phase sensitivity of a computer model of the auditory periphery. 1: Pitch identification.' J. Acoust. Soc. Am. 89(6), June 1991.
- Moore B.C.J. (2003) 'An introduction to the Psychology of Hearing.' Fifth Edition, Academic Press Copyright © 2003, Elsevier Science (USA).
- Parncutt R. (1989) 'Harmony: A Psychoacoustical Approach.' Copyright © Springer-Verlag 1989
- Patterson R. D., Robinson, K., Holdsworth, J., McKeown, D., Zhang, C. and Allerhand, M. (1992) 'Complex Sounds and Auditory Images.' Auditory Physiology and Perception, editors Y. Cazals, L. Demany, K. Horner. 1992
- Roederer J. G. (1995) 'The Physics and Psychophysics of Music, An Introduction.' Third Edition, © 1995 by Springer-Verlag New York, Inc.
- Slaney, M. (1993) 'An Efficient Implementation of the Patterson-Holdsworth Auditory Filter Bank.' Apple Computer Technical Report #35. Perception Group-Advanced Technology Group © 1993, Apple Computer, Inc.

Slaney, M. (1993) 'Auditory Toolbox: A Matlab toolbox for auditory modeling work version 2' Technical Report #1998-010. © Apple Computer, Inc. 1993-1994, © Interval Research Corporation 1994-1998.

Modeling Issues for Fine Musical Timbre Characterization

Mauricio A. Loureiro¹, Hugo B. de Paula², Hani C. Yehia³

CEFALA – Center for Research on Speech, Acoustics Language and Music

¹Escola de Música - Universidade Federal de Minas Gerais (UFMG)

³Departamento de Eletrônica – Universidade Federal de Minas Gerais (UFMG)

²Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais (PUCMINAS)

mauricioloureiro@ufmg.br, hugo@pucminas.br, hani@ufmg.br

Abstract. *In order to represent the variety of sounds a musical instrument may produce, it is necessary to find a model that can cope with sound features independent from its scale. In this work, several models of timbre characterization were applied to sample notes in several intensity levels across the whole extension of a clarinet. These models were based on amplitude and frequency time-varying curves of partials, which were measured by Discrete Fourier Transform. Principal Component Analysis techniques and Genetic Algorithms were used to define spectral sub-spaces capable of representing all tested sounds and of grouping them. The K-means clustering algorithm was used to infer timbre classes. Self-Organizing Maps lead to results similar to those obtained by PCA representation and K-means algorithms.*

Resumo. *Para se representar a variedade de sons que um instrumento musical é capaz de produzir, é necessário se utilizar modelos que podem lidar com um conjunto de parâmetros, independentemente de sua dimensão ou escala. Neste trabalho, vários modelos para caracterização do timbre foram aplicados a amostras de notas executadas em várias intensidades, cobrindo toda a extensão do clarinete. Estes modelos se basearam nas curvas de amplitude e frequência dos parciais, obtidos através da transformada discreta de Fourier. Análise por Componentes Principais e Algoritmos Genéticos foram utilizados para se definir um subespaço espectral capaz de representar e agrupar estes sons. O algoritmo de agrupamento das K-médias foi utilizado para se determinar classes de timbre neste subespaço. Mapas Auto organizáveis produziram resultados similares aos produzidos pela PCA e K-médias.*

1. Introduction

Representation of a musical instrument involves the estimation of the physical parameters that contribute to the perception of pitches, intensity levels and timbres of all sounds the instrument is capable of producing. Of these attributes, timbre poses the greatest challenges to the measurement and specification of the parameters involved in its perception, due to its inherently multidimensional nature. Timbre is perceived by means of the interaction of a variety of static and dynamic properties of sound grouped into a complex set of auditory attributes. Due to the multidimensionality of this

attribute, the identification of the contribution of each one of these competitive factors has been the main subject of psychoacoustics research on timbre perception.

In one of the most classic studies on musical timbre, [Grey, 1975] measured subjective judgment of similarity between pairs of timbres from 16 different musical instruments, submitted them to multidimensional scaling (MDS) and built a three-dimensional *timbre space*, in which multidimensional "timbre values" of different instruments were positioned according to their similarity/dissimilarity. Other than mapping geometrically the concept of acoustic similarity, that study also showed the capability of the method for providing a psychological quantification of a relatively complex structure upon quite simple data – similarity/dissimilarity responses between pairs of distinct timbres.

More recent studies were able to relate measurable physical parameters with the dimensions shared by the timbre represented in these spaces, combining quantitative models of perceptive relationships with psychophysical explanations of the identified parameters [Hajda, Kendall *et al.*, 1997; Misdariis, Smith *et al.*, 1998]. The possibility of establishing correlations between purely perceptive factors related to timbre and acoustic measurements extracted directly from sound, directed research on musical timbre towards more quantitative approaches. A historical review of the development of research on musical timbre is found in [McAdams, Winsberg *et al.*, 1995].

A technique commonly used in research on musical timbre nowadays is Principal Component Analysis (PCA). Recent works applying PCA to time-varying amplitude and frequency curves of harmonic components have produced similar results with similar sets of sounds [Cosi, De Poli *et al.*, 1994; Sandell e Martens, 1995; Charbonneau, Hourdin *et al.*, 1997; De Poli e Prandoni, 1997; Rochebois e Charbonneau, 1997; Beauchamp e Horner, 1998]. Multiple Wavetable synthesis (MWS) and Genetic Algorithms have also been commonly used to build orthogonal bases similar to the PCA. However, MWS have the advantage to allow a better understanding of timbre dynamics and its relation to the acoustics of the instrument [Horner, Beauchamp *et al.*, 1993; Horner, 1995a; Horner, 1995b; Horner e Ngai-Man, 1996].

The above mentioned studies on timbre analysis have approached comparisons among different musical instruments outside any musical context. This study investigates methods for representing the variety of sonorities produced by one single musical instrument, searching for models that are able to describe fine timbre attributes that are found in a single instrument timbre class.

1.1. Timbre modeling procedure

There are several ways of producing sound features for timbre modeling. [Tzanetakis, 2005] suggests a Music Information Retrieval (MIR) pipeline composed by the following steps: data acquisition, parameter extraction, feature estimation and information processing (classification, synthesis, and so on). The following sections will describe methods for the construction of a MIR pipeline for fine timbre characterization.

2. Timbre Set Specification

For the data acquisition step, it is important to have a data set properly defined. Since the purpose of this study is to show ways of representing the timbre of a musical instrument upon spectral parameters extracted from samples of sounds performed on that instrument, an adequate set of sounds for such a representation should include as many as possible different timbres, performed along the instrument entire pitch range. The timbre set used in this study was limited to the sound palette commonly produced on musical instruments in traditional classical western music performance, excluding sonorities produced on the instrument on the context of other musical traditions, as well as those regularly used in contemporary music known as “extended techniques”. Only the sustained part of relatively long sounds was considered, excluding attack, decay and transitions between consecutive notes.

Although timbre may vary independently from intensity and duration, its dependence on intensity is evident. This high level of correlation facilitates the sampling of different timbre “values” of the same note upon specification of intensity levels. Thus, different timbres were sampled for each note in the following intensity levels: *pianissimo* (*pp*), *piano* (*p*), *mezzo-piano* (*mp*), *mezzo-forte* (*mf*), *forte* (*f*) and *fortissimo* (*ff*).

3. Spectral Bases for Timbre Characterization

3.1. Spectral Parameters Estimation

The amplitude curves of the harmonic components were used as the initial parameters of the model, and were extracted using the short-time Fourier transform, according to McAulay and Quatieri’s method [McAulay e Quatieri, 1986; Serra, 1997]. In order to reduce the complexity of the data amplitude curves were smoothed by a low pass filter with cut-off frequency of 10 Hz.

3.2. Features measurement by Principal Component Analysis

The parameter space defined by the spectral analysis has as many dimensions as the number of partials extracted. The high correlation of these spectral parameters, presented in both the frequency and time domains, which is a common characteristic of spectral distribution of sounds of musical instruments, allowed an efficient data reduction using Principal Component Analysis (PCA) [Johnson e Wichern, 1998]. Applied to a set of multidimensional variables, PCA calculates an orthogonal basis determined by the directions of maximum variance of the analyzed data. The projections of the original data on this basis, denominated principal components (PCs), follow trajectories that accumulate the maximum variance of the data in a decreasing order. This allows an approximate representation of the data, using only a reduced number of dimensions.

3.3. Features measurement by Multiple Wavetable Synthesis and Genetic Algorithms

Musical sounds can be efficiently synthesized using an automatic genetic algorithm (GA) that decomposes the sounds into a group of wavetables (usually 3-5). The decomposition process consists of finding the optimal group of tables that reconstructs a

signal with minimum distortion. An approximation of the sound can be then constructed by Multiple Wavetable Synthesis (MWS) as a linear combination of these tables [Horner, Beauchamp *et al.*, 1993]. In this work, GA was used for the feature estimation procedure using wavetable components that were sampled from spectral components of the input data itself. This was possible because it was assumed that the recorded database comprises the timbre universe of the clarinet. In this novel approach, the population of the GA is constructed from real measured spectra, instead of being randomly generated, which makes it specially useful for analysis purposes. See [de Paula, Loureiro *et al.*, 2004] for a detailed description of the procedure.

Figure 1 shows the results obtained by the use of PCA and GA for feature estimation of the Timbre space. These bases were calculated using a concatenated group of four notes (16 sounds): B3, C4, C#4 and D4, each one played in four intensity levels: *pianissimo*, *piano*, *forte* and *fortissimo*.

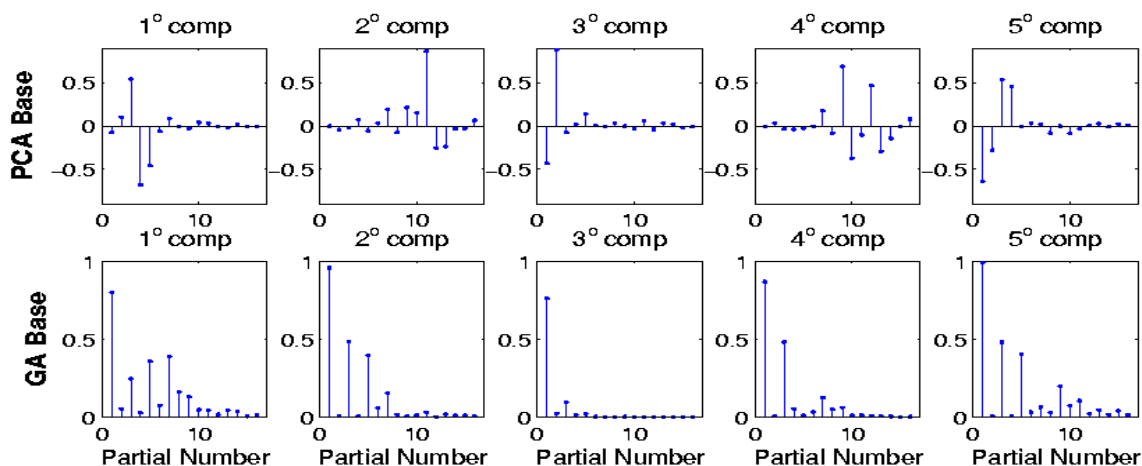


Figure 1: Comparison between the 5-dimensional PCA and GA base. The bases were calculated for the concatenated group of 4 notes (16 sounds): B3 (220 Hz), C4 (233 Hz), C#4 (247 Hz) and D4 (262 Hz).

4 The feature space of the clarinet timbre

The reduction in dimensionality resulted from PCA and GA made it possible the representation of the spectral distribution on low dimensional spaces. Figure 2 shows three-dimensional trajectories of the four notes of Figure 1. The first dimension of the PCA has a high correlation with the intensity level of the sounds and the PC value increases as the sound intensity increases. Although there is no visible correlation for the first component of the GA space, there is a clear intensity grouping and organization along the second and third dimensions of the GA. Since the spectral matching is an optimization process based on randomly generated populations, there is no real meaning in the order of the GA dimensions, and the amount of data each one explains is not ordered in the same way as the PCA. It can also be observed that the range of variation decreases as the dimension number increases in the PCA space, showing a clear hierarchy of dimensions. The same does not happen in the GA space, where all dimensions keep similar ranges of variation.

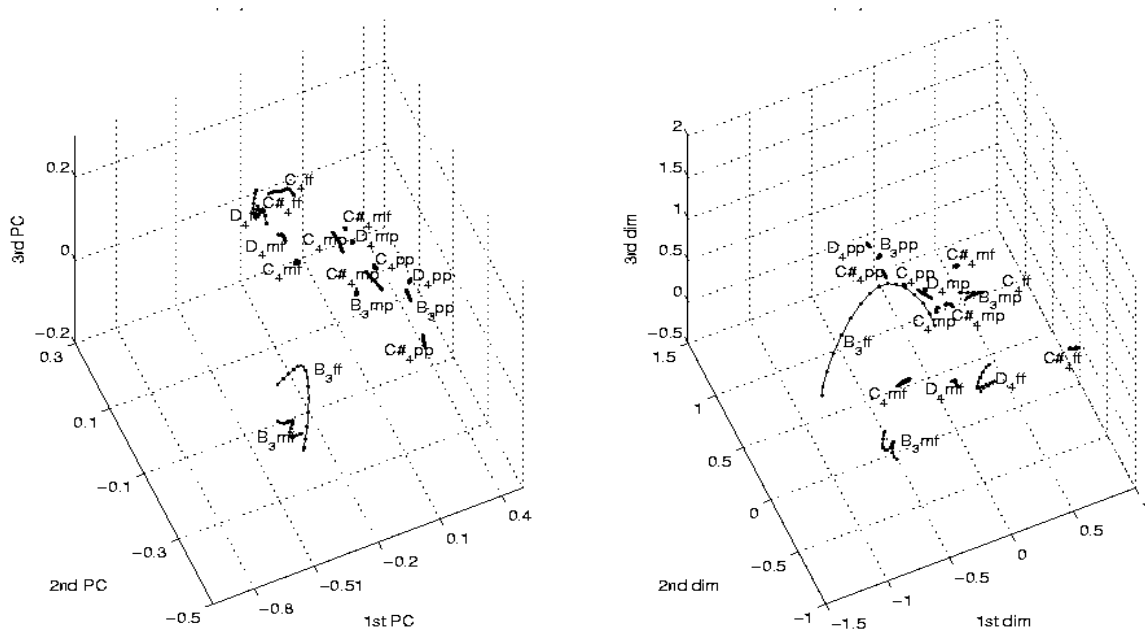


Figure 2: Three-dimensional trajectories of the four notes B3 (220 Hz), C4 (233 Hz), C#4 (247 Hz) and D4 (262 Hz) on PCA (left) and GA (right) spaces.

5. Timbre Classification

5.1. K-means Cluster Analysis

An attempt to investigate the timbre distribution along the entire instrument was made with Cluster Analysis, using the K-means algorithm [Kaufman e Rousseeuw, 1989]. In the present analysis, the variance of a cluster was calculated using the Squared Euclidian Distance (similar results were obtained with different types of distance). To avoid local minima, the K-means was run 40 times and the best solution was chosen. Comparison of timbre parameters among notes of different pitch becomes more complex, as timbre may vary significantly as a function of the note played, depending on the instrument. Clarinet sounds, as used in this study, present irregular variation of timbre from note to note, which can be very accentuated, depending on the region of the instrument, like the abrupt timbre change between the low and mid registers, a well known characteristic of the clarinet. At first, a cluster analysis was performed using the 19 notes (76 sounds) from the low register of the clarinet, from D3 (147 Hz) through Ab4 (415 Hz). Nine clusters provided the best correlation between auditory tests and the classification obtained for this set of sounds [Loureiro, de Paula *et al.*, 2004]. Very few of these sounds had their principal component coordinates split into different clusters and, when this happened, no more than 2 clusters were involved and the cluster assigned to the central part of the sound was always the cluster where the majority of points lied.

Figure 3 shows the 11 lower notes of the clarinet (from D3 through C4), represented by the location of its central frame on the low register *timbre space*. The figure shows a large group of sounds clustered together close to the origin of the space (left), which includes the *pp* or *mp* version (or both) of every note of this set (D3 to C4), except for the Eb3. Sounds *mf* and *ff* are more spread along all three dimensions, showing that intensity level differentiation spread the sounds more strongly than pitch

differentiation. This can be also observed in Figure 4, which orders all 76 sounds of the low register of the clarinet by pitch and shows the cluster to which each one was assigned. Each sound is represented by the location of its central frame on the *low register timbre space*. This figure highlights the correlation of the cluster to intensity level and shows that intensity level variation spreads the sounds more than pitch variation. Auditory tests showed strong coupling of perceived brightness to clusters assignment. Due to the known relationship of spectral centroid to the perception of brightness, cluster labels were ordered according to the mean of the spectral centroid of the group of sounds assigned to it. Note that the first 3 clusters group almost every *pp* and *mp* sounds of the whole set. Moreover, notes of higher pitch in *mf* and *ff* were also assigned to these clusters. While higher pitched notes were grouped more tightly into these clusters, the four last clusters contain only *mf* and *ff* notes of the lower octave, It can be seen that notes of lower pitch tend to have a wider variation in timbre and that timbre becomes more stable and concentrated in lower clusters as the pitch increases.

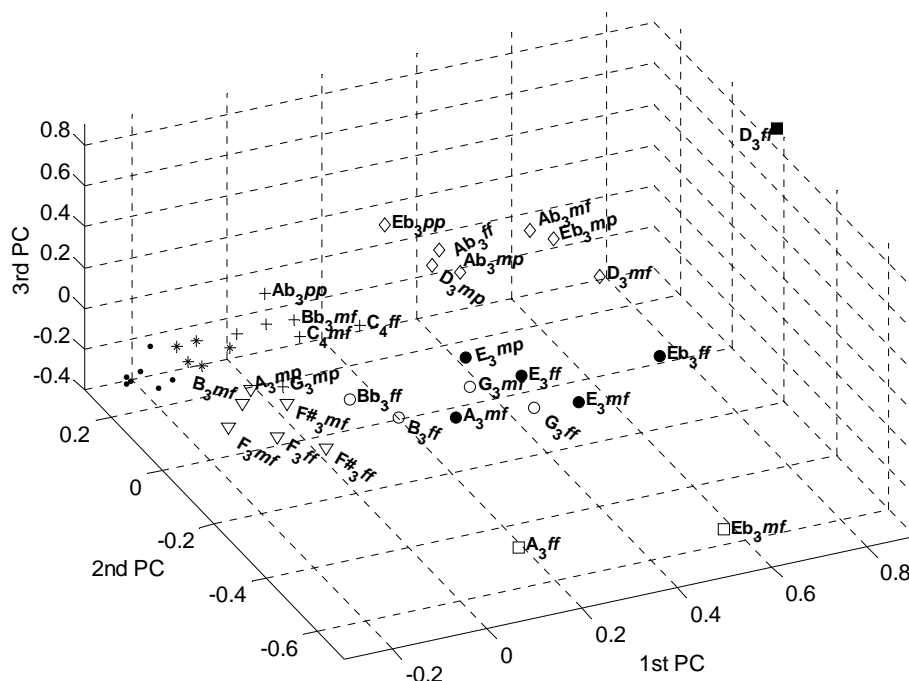


Figure 3: Three-dimensional trajectories of the four sounds of each of the 11 lowest notes of the clarinet, from D3 through C4, in the spectral space defined by the entire low register (points not labeled on the left of the figure correspond to notes: D3 *pp*, E3 *pp*, F3 *pp*, F#3 *pp*, G3 *pp*, Ab3 *pp*, A3 *pp*, Bb3 *pp*, Bb3 *mp*, Bb3 *mf*, B3 *pp*, B3 *mp*, C4 *pp* and C4 *mp*).

A new clustering analysis was then performed using the 33 notes (132 sounds) from the first and second register of the clarinet. Twelve clusters were found to be more adequate for this sound set. Figure 5 shows a similar plot to Figure 4, including also the 14 notes of the second register, A4 through Bb5. With the exception of one *mf* and four *ff* sounds (Bb *mf*, Bb4 *ff*, C# *ff*, B4 *ff* and Bb5 *ff*), all sounds of the second register lied in the five first clusters, together with *pp* and *mp* notes from the lowest register. This shows the tendency of higher notes to be clustered tighter together, reinforcing the correlation of the classification to the variation of the intensity level, as well as the diminishing of timbre variation as pitch increases.

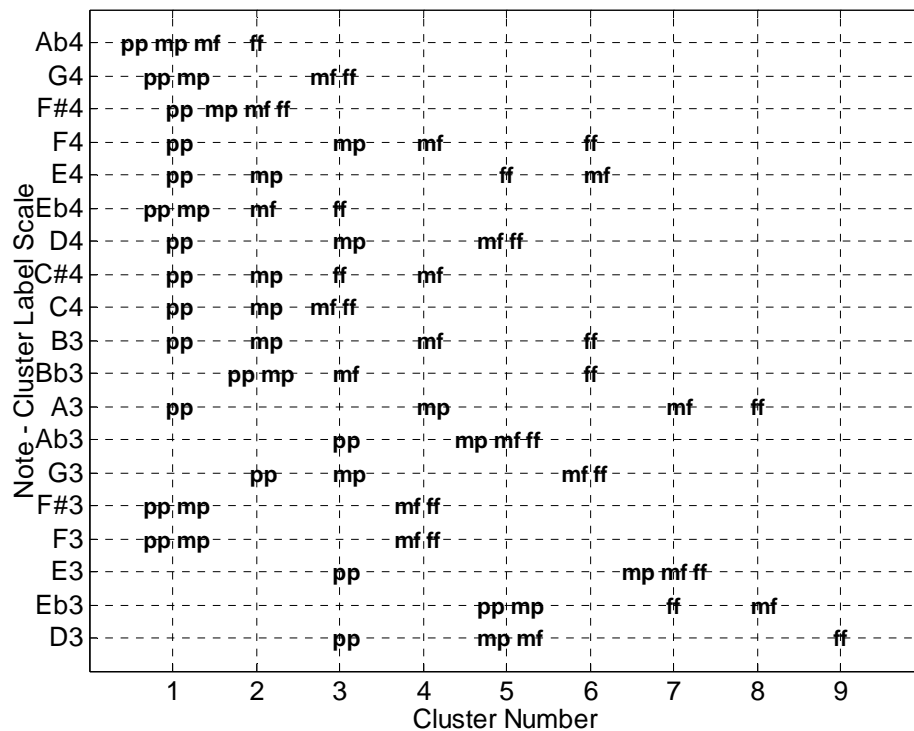


Figure 4: Cluster Label of the 19 notes of the low register of the clarinet. Notes are ordered by pitch and cluster labels by the mean of the spectral centroids.

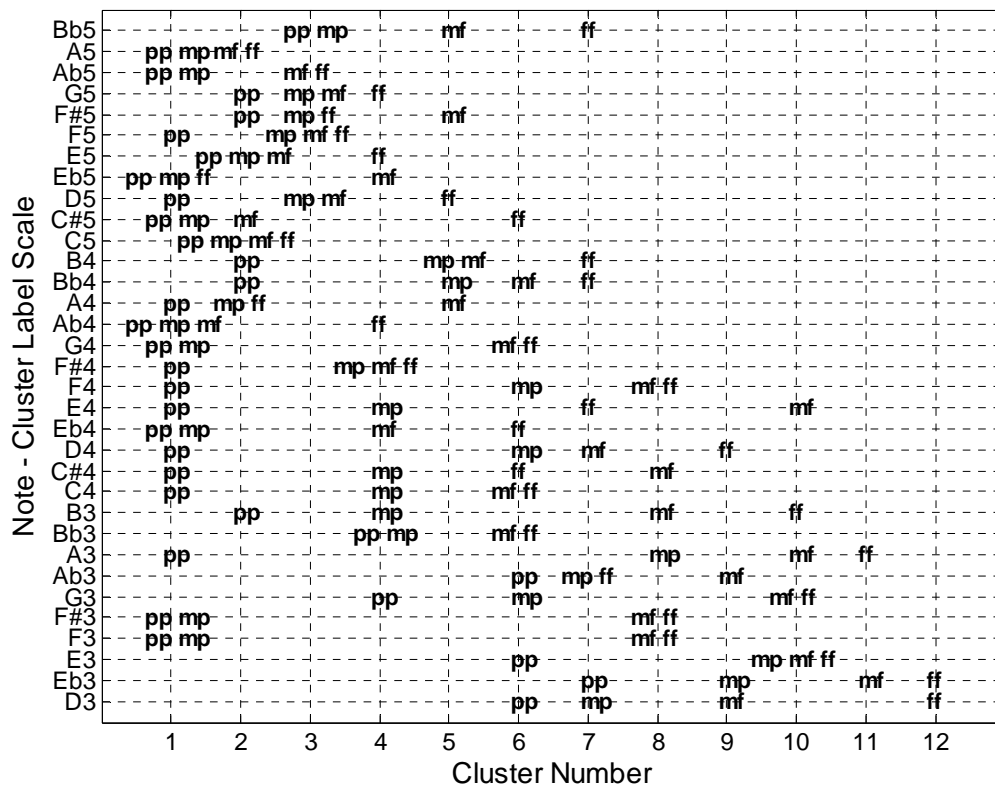


Figure 5: Cluster Label of the 4 four sounds of each of the 33 notes of the first two registers of the clarinet. Notes are ordered by pitch and cluster labels by the mean of the spectral centroids.

5.2. Self-Organizing Maps

Self-Organizing Maps are algorithms formalized by Kohonen for non supervised neural nets, capable of mapping input data of large dimensions into lower dimensional spaces, preserving the essential topological relationships of the original data [Kohonen, 1995]. Because it is not based on *a priori* suppositions about the characteristics of the analyzed data, SOM is a powerful tool to analyze complex and non-linear data, such as musical sounds. Leman [Leman, 1994] proposed a comparison between timbre mapping obtained by SOM and those spaces built by MDS starting from psychological measurements, as a basic reference for cognitive research in music. Toiviainen [Toiviainen, Kaipainen *et al.*, 1995] compared the efficiency of musical timbre representations in spaces built by topological distances calculated by SOM to subjective measurements of similarity. The results of that work proved a high correlation degree between the two domains, suggesting an adaptation of the model of Kohonen to project multidimensional perceptive complexes in this kind of representation. De Poli [De Poli e Prandoni, 1997], Cosi and colleagues [Cosi, De Poli *et al.*, 1994] developed studies on classification of musical timbre using SOM. Faiten [Faiten e Gunzel, 1994] obtained timbre specialization by processing pre-processed spectral parameters with SOM. This paper used a Matlab Toolbox from Versanto and colleagues [Versanto, Himberg *et al.*, 2000].

An hexagonal SOM of size 16-by-9 was used to map the 76 sounds (19 notes) from the low register of the clarinet. Figure 6 (left side) shows the relation of this mapping to sound intensity levels. As in the classification with k-means, *ff* and *mf* sounds tend to be grouped together. Moreover, *pp* and *mp* sounds were also tighter clustered than *mf* and *ff* sounds, as already observed on the spectral *timbre space* of Figure 3. This can be verified by the distance metrics distribution of the SOM shown on the graph on the right side of Figure 6, in which distances between hexagons represent distances between map cells.

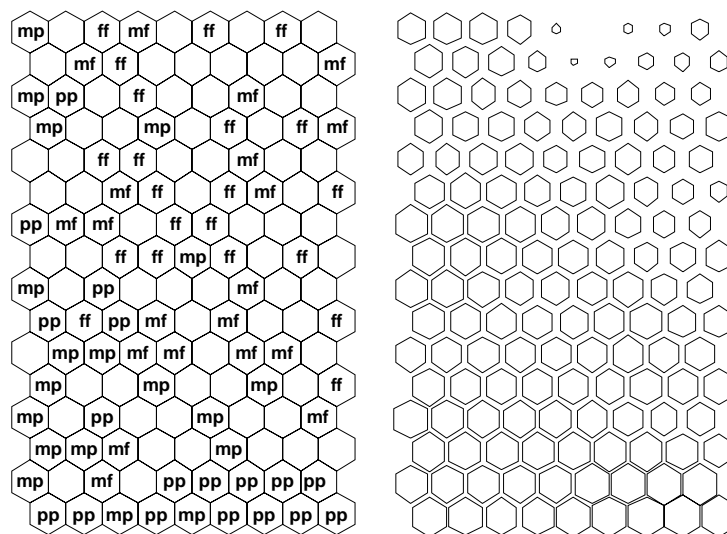


Figure 6: SOM mapping of intensity levels (left) of the 76 sounds (19 notes) of the low register of the clarinet and the distribution of the distance between map cells (right). Closer (larger) cells mean closer matching units.

Figure 7 (c) shows the trajectories of six notes of the lower octave of the instrument, including the notes plotted in figures 7(a) and 7(b): D3, Eb3, F3, F#3 and Ab3. Despite being closer in pitch, they were mapped onto two distinct groups on opposite sides, D3, Eb3 and Ab3 on the upper left corner and F3, F#3 and A3 on the lower right corner. Comparing Figures 4 and 7(c) we observe that notes of the upper left corner were assigned to the same clusters by the K-means as were also the notes mapped on the lower right corner. They were also spread into opposite sides on the spectral *timbre space* (Figure 3). Moreover, in both classifications Ab3 sounds were positioned tightly together, while A3 sounds were widely spread over clusters and cells, corroborating the high correlation between the K-means and the Kohonen map.

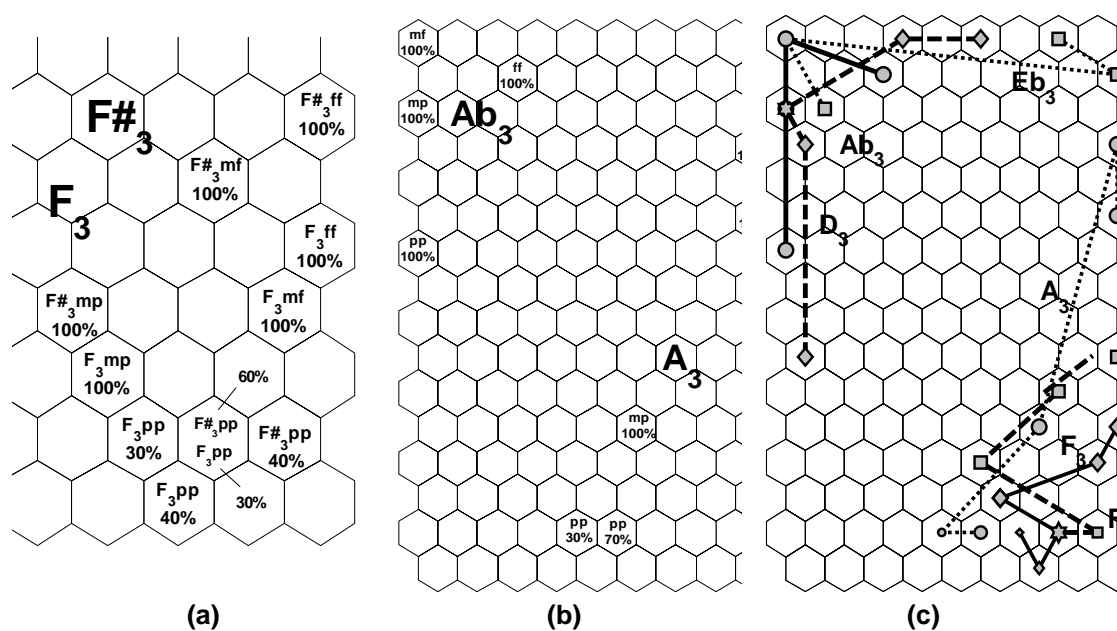


Figure 6: SOM mapping of : (a) notes F3 and F#3; (b) notes A3 and Ab3; (c) trajectories of notes D3, Eb3, F3, F#3 and Ab3.

6. Conclusion

PCA and GA were presented as adequate models for a lower dimension timbre space definition. Auditory tests of discrimination with resynthesized sounds with normalized pitch showed the effectiveness of the representation model presented in this paper, showing a clear relation between the perceived timbre and the cluster label to which the notes were assigned. The construction of spectral sub-spaces involving all possible sounds produced by the instrument made it possible a compact representation of the whole timbre palette of the instrument. Both K-means and Self-Organized Maps provided a descriptive comparison of the dynamic variation of timbre. These representations and clustering techniques showed a strong matching, as they are mapping data from the same spectral *timbre space*. Summarizing, it could be clearly verified across all the results presented in this study that: (i) timbre classes tend to be divided as a function of spectral brightness, which is known to be correlated to intensity level in wind instruments; (ii) the lowest octave of the clarinet exhibit in general much more richness of timbre differentiation than higher pitched notes; (iii) the highest octave of the mid register (from C5 up), exhibit less spectral brightness and less timbre differentiation. The results of this study applied to wider dynamic timbre variation will facilitate the investigation of the use of intentional timbre differentiation by the performer to convey musical expressiveness. Other perspectives for this project is to extend the investigation to shorter sounds, like staccati and pizzicati, as well as attack, decay and transition between notes, for which auditory models seems to be an adequate analysis tool.

7. Acknowledgments

This work was supported in part by CAPES (Brazilian Higher Education funding agency), and by CNPq (National Council for Scientific and Technological Development), Brazil.

8. References

- Beauchamp, J. W. e A. Horner. Spectral Modelling and Timbre Hybridisation Programs for Computer Music. *Organised Sound*, v.2, n.3, p.253-258, 1998.
- Charbonneau, G., C. Hourdin e T. Moussa. A Multidimensional Scaling Analysis of Musical Instrument's Time-Varying Spectra. *Computer Music Journal*, v.21, n.2, p.40-55, 1997.
- Cosi, P., G. De Poli e G. Lauzzana. Auditory Modelling and Self-Organizing Neural Networks for Timbre Classification. *Journal of New Music Research*, v.23, p.71-98, 1994.
- de Paula, H. B., M. A. Loureiro, H. C. Yehia e J. A. Vasconcelos. Representing Timbre Dynamics of a Musical Instrument: comparison between GA and PCA. *IEEE International Workshop on Machine Learning for Signal Processing*, São Luis, Brazil. p. 381-390, 2004.
- De Poli, G. e P. Prandoni. Sonological Models for Timbre Characterization. *Journal of New Music Research*, v.26, p.170-197, 1997.

- Faiten, B. e S. Gunzel. Automatic Indexing of a Sound Database Using Self-Organizing Neural Nets. *Computer Music Journal*, v.18, n.3, p.53-65, 1994.
- Grey, J. M. *An exploration of musical timbre*. CCRMA, Dept. of Music Stanford University. Stanford, Calif., (STAN-M-2), 1975.
- Hajda, J. M., R. A. Kendall, E. C. Carterette e M. L. Harshberger. Methodological Issues in Timbre Research. In: I. Deliège e J. A. Sloboda (Ed.). *Perception and Cognition of Music*. Hove: Psychology Press, p.253-306, 1997.
- Horner, A. Wavetable Interpolation Synthesis Based on Time-Variant Spectral Analysis of Musical Sounds. *Journal of The Audio Engineering Society*, 1995a.
- _____. Wavetable Matching of Dynamic Instruments with Genetic Algorithms. *Journal of The Audio Engineering Society*, v.43, n.11, p.916-931, 1995b.
- Horner, A., J. W. Beauchamp e L. Haken. Methods for Multiple Wavetable Synthesis of Musical Instrument Tones. *Journal of The Audio Engineering Society*, v.41, n.5, p.336-356, 1993.
- Horner, A. e Ngai-Man. Group Synthesis with Genetic algorithms. *Journal of The Audio Engineering Society*, v.44, n.3, p.130-147, 1996.
- Johnson, R. e D. W. Wichern. *Applied Multivariate Statistical Analysis*. Upper Sadlle, New Jersey, 1998.
- Kaufman, L. e P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons, 1989.
- Kohonen, T. *Self-Organing Maps - Springer Series in Information Sciences*. Berlim: Springer Verlag. v.30, 1995.
- Leman, M. Schema-Based Tone Center Recognition of Musical Signals. *Journal of New Music Research*, v.23, n.2, p.169-204, 1994.
- Loureiro, M. A., H. B. de Paula e H. C. Yehia. Timbre Classification Of A Single Musical Instrument. *5th International Seminar on Music Information Retrieval*, Barcelona, Spain. p. 546-549, 2004.
- McAdams, S., S. Winsberg, S. Donnadieu, G. De Soete e J. Krimphoff. Perceptual Scaling of Synthesized Musical Timbres: Common Dimensions, Specificities and Latent Subject Classes. *Psychological Research*, v.58, p.177-192, 1995.
- McAulay, R. J. e T. F. Quatieri. Speech Analysis/Synthesis Based on a Sinusoidal Representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v.34, n.4, p.744-754, 1986.
- Misdariis, N. R., B. K. Smith, D. Pressnitzer, P. Susini e S. McAdams. Validation of a Multidimensional Distance Model for Perceptual Dissimilarities Among Musical Timbres. *Proceedings of the 16th International Congress on Acoustics*, Woodbury, New York: ASA - The Acoustical Society of America. p., 1998.
- Rochebois, T. e G. Charbonneau. Cross-Synthesis Using Interverted Principal Harmonic Sub-Spaces. In: M. Leman (Ed.). *Music, Gestalt and Computing*:

- Studies in Cognitive and Systematic Musicology*. Berlin-Heidelberg: Springer Verlag, p.375-385, 1997.
- Sandell, G. J. e W. Martens. Perceptual Evaluation of Principal-Component-Based Synthesis of Musical Timbres. *Journal of The Audio Engineering Society*, v.43, n.12, p.1013-1028, 1995.
- Serra, X. Musical Sound Modeling with Sinusoids plus Noise. In: A. Piccialli, C. Roads, *et al* (Ed.). *Musical Signal Processing*: Swets & Zeitlinger Publishers, 1997.
- Toiviainen, P., M. Kaipainen e J. Louhivuori. Musical Timbre: Similarity Ratings Correlate with Computational Feature Space Distances. *Journal of New Music Research*, v.24, p.282-298, 1995.
- Tzanetakis, G. A personal history of Music Information Retrieval. *Multimedia and Mathematics*, Banff, Canada. p., 2005.
- Versanto, J., J. Himberg, E. Alhoniemi e J. Parhankangas. Self-Organizing Map in Matlab: the SOM Toolbox. Helsinki, Finland,: Helsinki University of Technology 2000.

“Enquanto eles riem” para Clarinete e computador rodando MAX/MSP – Escrita e Estratégias interativas

Cristiano Figueiró, Anselmo Guerra de Almeida

Laboratório de Pesquisa Sonora Emac/UFG – Campus II - Samambaia - Caixa

Postal 131 Goiânia - Goiás - Brasil - Cep 74001-970

figocris@yahoo.com.br, anselmo@musica.ufg.br

Resumo

Esse artigo expõe o processo de composição da peça “Enquanto eles riem” para Clarinete e Computador rodando MAX/Msp, tanto no âmbito da técnica de programação empregada, quanto da escrita instrumental e alguns aspectos poéticos de construção da peça como por exemplo a intervenção do performer no processo de composição e a interatividade do performer com o computador. O primeiro trecho do artigo expõe os elementos de escrita instrumental e de poética composicional. Na segunda parte do artigo é expostas as estratégias de programação em MAX no sentido de vencer os desafios de interatividade na composição.

Abstract

This article exposes the compositional process of the piece “Enquanto eles riem” to Clarinet and Computer with MAX/Msp, in the field of the programming technique, how in the instrumental writing and some poetic aspects of construction of the piece, for example, the performer intervention in the composition process and the performer interactivity with the computer. The first part of the article exposes elements of instrumental writing and poetic composition. The second part of the article exposes some MAX programming techniques in order to win the challenge of interactivity in the composition.

Enquanto eles riem..., para Clarinete e Computador

Esta peça se tornou um desafio ao longo do processo, por abranger áreas distintas como a escrita e performance tradicional e o universo da produção sonora digital com a idiomatismo dialético da música eletroacústica. Procurou-se não transformar este contraste num elemento castrador de uma linguagem ou outra. Mas, ao contrário, um limite bem estreito onde se possa explorar, de uma maneira consciente e, ao mesmo tempo, lúdica, as

possibilidades, particularidades e idiosincrasias de cada elemento. Nesta parte, iremos enfocar o processo de composição da parte do Clarinete.

A peça se trata de um exercício composicional, que foi sendo trabalhado e definido ao longo dos ensaios entre compositor e Clarinetista. Um dos impulsos composicionais foi a busca pela interatividade entre os universos sonoros do compositor e do performer através do processo composicional, materializado na forma de gravação dos materiais da partitura a diversos níveis de elaboração. Gravações de trechos da peça e improvisos foram e continuam contribuindo como materiais e fontes sonoras para o desenvolvimento da peça. O uso do recurso da interatividade entre compositor e performer foi o primeiro fator estrutural pré-estabelecido. A possibilidade da interferência de informações do performer no processo composicional foi desejada, ainda que esta tenha se concretizado apenas no âmbito da escolha de algumas notas e dinâmicas e algumas frases idiossincráticas de improviso do performer que foram adicionadas à composição com objetivo de tornar a música mais orgânica.

A Seção A tem quatorze compassos e primeiramente é exposta a série, desenvolvida apenas no âmbito das alturas através de serialização livre, de onde saiu o material da peça, já com algumas alterações da série e que se tornaram estruturais no desenvolvimento. Como é o caso da segunda vez em que entra a série, a nota ré é substituída pela nota mi, o que se torna recorrente na peça.



Fig1: série original da peça;



Fig.2: compasso 9 (trecho da peça com a série alterada-nota mi)

A maior seção da peça é a seção B que tem cinquenta compassos e quatro sub-seções, distintas. A primeira parte desta seção começa com a mudança de compasso de 2/4 para 6/8. É estabelecido o acorde de mi maior com quinta aumentada como entidade de base

desta seção, que se contrapõe as notas da série que são desenvolvidas, já com algumas mudanças que estavam presentes na seção A.



Fig.3: compasso 16

A entidade de base é o motivo da construção da segunda parte da seção B, que se consiste apenas de transposições deste acorde, ajustadas numa figuração rítmica quase constante (acelerando) e que faz um contraste com a dinâmica empregada. O que resulta num gestual interessante para realização de processamento com o MAX.

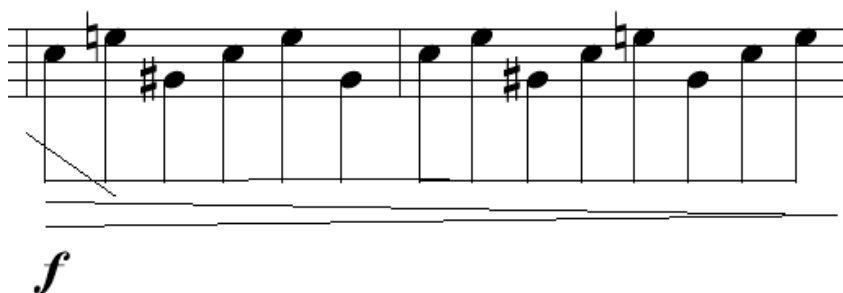


Fig.4: compasso 29-30, figura rítmica de aceleração em contraponto com a dinâmica que vai diminuindo.

A terceira parte da seção B é uma seção mais lenta, em que foi usada a entidade de base transposta como elemento básico do desenvolvimento melódico e harmônico. A partir dessa transposição, chegou-se a escala de lá menor harmônica, que na sua configuração possui três transposições da entidade de base.



Fig.5: compasso 44-46, constr. melódica em lá menor harmônica.

A quarta parte da seção B é um retorno variado à segunda parte, com caráter de cadência virtuosística e serve de ponte para o retorno à seção A que volta, com variações, no compasso 73. Esta nova seção (A'), se caracteriza por um uso mais efetivo do computador e das células rítmico-melódicas, propostas pelo performer, que acabaram sendo incorporadas na composição enquanto células mesmo e não como órgãos estruturais da composição, o que implicaria em um trabalho composicional mais profundo. Como exemplo, foi escolhido um trecho em que, após uma variação melódica da série, foi acrescentada uma célula melódica de "risada", em que o performer simula uma risada com o Clarinete. Estes elementos implicam uma diferença na execução, pois o mais importante não são as notas e sim o caráter específico do trecho.



Fig.6: compasso 91, Clarinete "rindo"

No compasso 97 começa uma seção que é uma ponte para a entrada da seção B'. Nesta ponte podemos notar como foi pensada a condução vocal, pois o Clarinete, apesar de ser um instrumento melódico, com sua sustentação de notas permite uma escrita pseudo-polifônica.

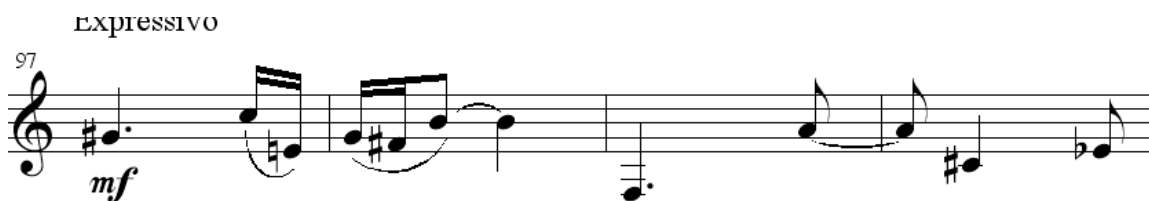


Fig.7: compasso 98-101, escrita a duas vozes.

A seção B' começa no compasso 116 e combina o uso da entidade de base interpolada com a série inteira adaptada ao desenho melódico característico da parte lenta da seção B.



Fig.8: compasso 118-119, interpolação da série com a entidade de base e o desenho melódico

O final da peça consiste de uma pequena coda de cinco compassos em que o último compasso usa um artifício técnico da Clarineta, em que o performer fica assoprando o instrumento sem emitir notas definidas, apenas o ruído do sopro. Isso cria uma suspensão cênica enquanto o computador tece seus últimos comentários. Em relação a interatividade do performer com o computador, podemos dizer que é uma interatividade calcada na condução da partitura, ou seja, o desafio de interatividade é conseguir fazer o computador acompanhar a performance do músico, guiado pela partitura.

Max/Msp – implementações

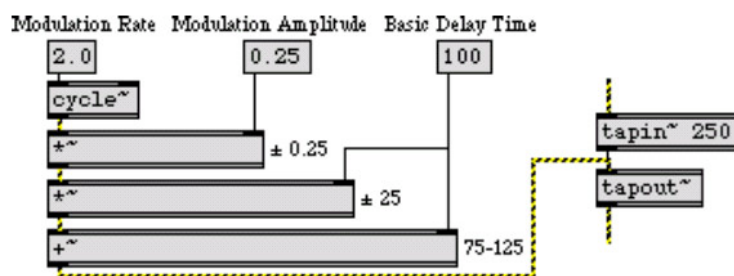
Nesta parte vamos analisar os trabalhos desenvolvidos no ambiente MAX/MSP criados a partir das premissas de interatividade musical que se fizeram necessárias durante a construção dessa peça.

Tendo em vista o pensamento composicional que guiou o processo, e o tipo de interação desejada, procurou-se desenvolver um sistema que possibilitasse que a performance da peça não dependesse de outro performer humano além do clarinetista. Aqui vamos expor dois patches de MAX desenvolvidos nesse sentido.

Interface

Este trabalho se trata de uma interface de performance baseada no tutorial MSP número 30 do MAX. Este *patch* possibilita a captura de áudio e processamento em tempo real, no qual a idéia básica do efeito é acrescentar um oscilador como variação de tempo de *delay*. O que ocasiona um *delay* limpo quando você posiciona a frequência do oscilador em zero e um suave efeito *flanging* com a frequência abaixo de 0.5Hz, variando a densidade do efeito conforme o índice de modulação.

A principal estrutura para o entendimento deste tutorial é a operação de modulação do oscilador(*cycle~*) controlando os parâmetros de *delay* (*tapout~*)(fig. 1). Os objetos *tapin~* e *tapout~* atuam em conjunto criando linhas de *delay*. O sinal original entra no *inlet* esquerdo de *tapin~* que tem como argumento o tamanho da linha de *delay* descrito em milisegundos e que vai ser lido por *tapout~* que tem como argumento o tempo em milisegundos que leva para acionar a primeira linha de *tapin~*. Neste caso, além da saída de *tapin~*, é somada a entrada de *tapout~* os dados de um oscilador (*cycle~*) que realiza a modulação do sinal de áudio. A saída de *tapout~* toma dois rumos: o primeiro vai direto para a saída de áudio e o segundo é escalado por um multiplicador variável (de 0 a 1) e reconectado a entrada de *tapin~* o que causa um efeito que em pedais de guitarra se conhece por *feedback* e se trata de uma repetição contínua das linhas de *delay*.



Modulating the delay time with a low-frequency oscillator

Fig.9 : núcleo básico do efeito flanger.

O trabalho de programação em si, além da análise do tutorial, foi a criação de uma interface de uso em performance. Uma interface é um dispositivo físico ou lógico que faz a adaptação entre dois sistemas, que no caso foram de um lado o *patch* do tutorial do MSP e de outro um performer com a partitura da composição e a possibilidade de improvisar otimizando os recursos do tutorial, ou outro *patch* controlador do sistema que ao acompanhar a performance, liga e desliga essa interface. A construção de uma interface de uso do programa faz parte do processo de aprendizado de uma linguagem. No caso do MAX, escrever interfaces, torna-se básico na aprendizagem, pois toda a estrutura da linguagem é gráfica e voltada para a performance.

Nesse caso foram acrescentados objetos gráficos de controle como botões e sliders verticais, horizontais e em forma de caixa de desenho (**pictslider** - onde o controle é feito com o mouse e valor controlado pelas quatro arestas da caixa). As ligações da interface com o *patch* do tutorial foram feitas através do objeto **patcher** que realiza a conexão e o encapsulamento de um *patch* com outro. Alguns artifícios foram adicionados, no sentido de ampliar as possibilidades interativas do performer com o computador e de facilitar a manipulação do programa durante a performance. Foram acrescentadas envoltórias de segmentos lineares (**line**) ao controle de volume, causando assim um efeito de *fade in* e *fade out*. Os controles de tempo de *delay* e espacialização do sinal processado respectivamente são controlados pelo objeto **drunk**, que é um gerador numérico randômico e que necessita de um comando *bang* para se acionar, que, no caso, é fornecido por **metro**, um metrônomo gerador de *bangs*. O fato de ter alguns parâmetros controlados por gerador independente, que produz parâmetros randômicos durante a performance, empurra o performer a interagir de maneira mais intensa com a resposta do computador.

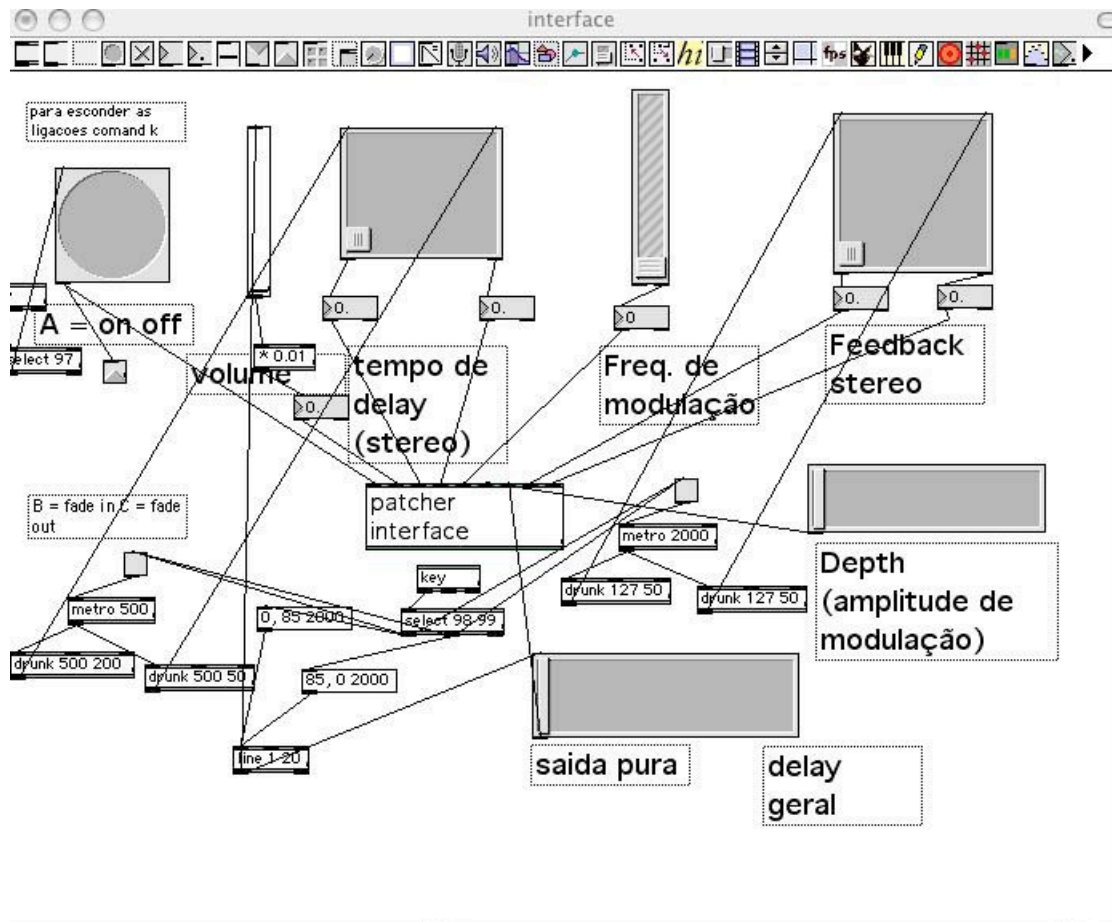


Fig.10: interface em modo de edição

Score-Follower

Essa peça exige dois elementos de interatividade do computador com o performer, que são: acionamento de samples pré-gravados e processamento de áudio em tempo-real. Como a parte do Clarinete é concentrada em uma notação tradicional e desde o princípio a parte do computador não pretendia ser em suporte fixo, fez-se necessário o desenvolvimento de um Score-Follower, que pudesse conduzir a performance do computador em sincronia com o clarinetista.

O caminho escolhido foi mapear a performance a partir de um microfone, transformar estes dados (frequências) em dados MIDI e deixar na memória do programa uma sequência MIDI que sirva de "espelho" para o acompanhamento. A busca pela interatividade com o performer foi a tônica do trabalho. O *patch* supre a necessidade de

uma peça que inclui improvisos do performer e alguns pontos sincronizados com o acionamento de *samples* pelo computador. Ou seja, o computador espera pelo performer os pontos certos de sincronia, que são uma sequência de notas. Os espaços entre os pontos de sincronia são usados para improviso do performer que deve interagir com o universo sonoro que ele mesmo acionou.

O processo do *patch* é muito simples. O interesse vem da forma prática e econômica da organização das ações e a portabilidade, podendo fazer parte de um complexo de outros elementos ou simplesmente mudarmos a função do mapeamento dos dados que, como vimos, podem ser muitas. Os dados de performance podem ser mapeados para o acionamento de *samples* que estejam na memória do programa, controle de parâmetros de síntese ou processamento, ou nível composicional.

A organização deste programa pode ser dividida em quatro partes: reconhecimento de alturas, transferência dos dados da performance para dados MIDI, acompanhamento e acionamento de *samples*. A parte do reconhecimento de alturas foi resolvida com o uso do objeto **fiddle~**, escrito por Miller Puckete (1999). Este objeto é um algoritmo que fornece alguns dados sobre o reconhecimento da performance que está na sua entrada de áudio, como altura e amplitude. No caso, foi usado apenas o reconhecimento de altura, que é endereçado para o objeto **midiformat**, que transforma um valor de altura em dado MIDI.

Uma vez que o programa faz o reconhecimento de altura da performance e transforma isso em dado MIDI, podemos enviar este dado para o objeto **follow**, que é um objeto em que podemos sincronizar performances MIDI. Ou seja, o objeto tem uma sequência MIDI na sua memória e consegue acompanhar uma outra sequência que fica sendo mandada a ele com base na parte gravada. A cada nota que é reconhecida, o objeto manda uma mensagem que aciona um processo, que no caso é o *playback* de *samples*, e que poderia ser uma série de ações possibilitando diferentes níveis de interatividade, como, por exemplo, o acionamento de estruturas sonoras baseadas no histórico do comportamento da performance, sequências randômicas com parâmetros variáveis, processamentos variados do sinal de entrada, etc....

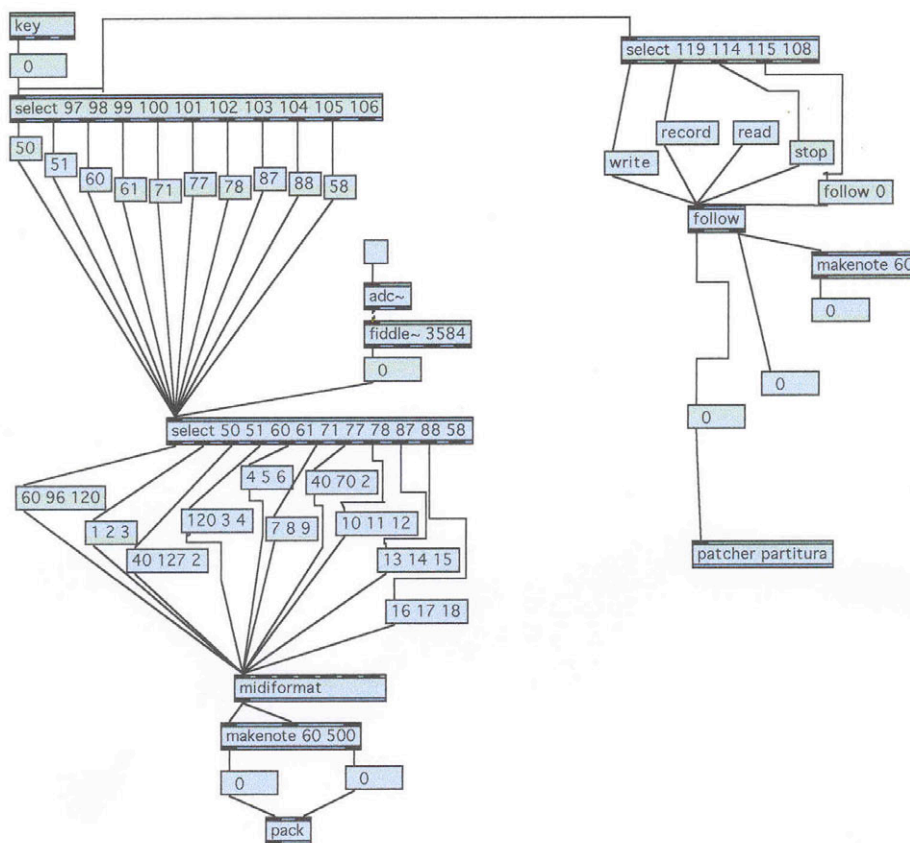


Fig.11: *patch* para MAX que realiza acompanhamento interativo (*Score – Follower*).

Conclusão

O *score-follower* desenvolvido ainda é uma versão instável, ou seja, ainda não é uma ferramenta com a qual podemos contar sem o ônus de erros. Mas consiste de uma estrutura básica de um *score-follower*, pronto para desenvolvimento e implementação em composições. Os principais desenvolvimentos são o trabalho, direto no algoritmo do objeto **fiddle~**, no sentido de estabelecer diferenças entre os timbres dos instrumentos, pois um dos maiores problemas de leitura deste objeto é o fato dele reconhecer muitos harmônicos de cada timbre, enquanto que o interessante para um *score-follower* seria o reconhecimento apenas do harmônico fundamental de cada nota, pois para cada timbre ou espectro harmônico diferente ele se comporta de maneira diferente. Outro passo é o de distribuir o

processamento, pois, neste caso, o mesmo programa reconhece as alturas e realiza o acompanhamento acionando os *samples*. Se conseguirmos deslocar o reconhecimento para um processador e o acionamento do acompanhamento para outro processador, podemos chegar mais perto de uma situação estável. Poderemos também ampliar a lógica deste *patch*, pois poderá ser incluído, por exemplo, a análise de comportamento da performance de maneira que o *patch* responda de maneiras diferentes de acordo com cada performance. A saída para a instabilidade desse *patch* é a performance auxiliada por um performer ao computador, acionando os *samples* e a interface de processamento manualmente.

Podemos fazer uma síntese dos processos composicionais de escrita e de programação que mais foram utilizados: serialização livre de alturas, idiomatismo instrumental como ponto estrutural nas composições, contraste entre narrativas diversas, colagens, granulações, interação com intérprete no nível da performance e da composição, estruturação eletroacústica provinda da escrita instrumental.

Os *patches* de MAX/MSP, pela própria característica de programação do ambiente MAX, são automaticamente células que podem ser incorporadas a novos projetos, ou ter detalhes alterados para servir a novas funções. Por vezes, o mesmo *patch*, com suas possibilidades bem exploradas, pode nos fornecer inúmeras respostas, muitas que nem tínhamos pensado originalmente.

Para finalizar, podemos concluir que o processo composicional de pesquisa serviu para o aprofundamento da busca por uma identidade artística, através da ampliação de técnicas de programação e da reflexão ao redor do fazer musical contemporâneo e suas relações com as novas tecnologias. Identidade artística essa que entendemos como uma constante busca dinâmica da expressão pessoal através do universo que engloba nossa individualidade. Enquanto etapa de formação artística para um compositor, pode-se dizer que esta pesquisa serviu para a ampliação do repertório gestual e idiomático através de possibilidades de ambientes computacionais voltados para a composição, escrita instrumental e combinação de linguagens.

Referências Bibliográficas

- ALMEIDA, Anselmo Guerra de. **A arte como modelo de interatividade**. In: Anais da Anppom, 1999.
- CAESAR, Rodolfo. **Novas interfaces e a produção eletroacústica**. In: Anais do SBCM, Brasília, 1997.
- CORREA, James. **Jardim dos caminhos que se bifurcam**: Processos composicionais. Dissertação de Mestrado, Programa de pós graduação em música, UFRGS, 2003.
- ECO, Umberto. **Obra Aberta**. Perspectiva, São Paulo, 1992.
- FERRAZ, Silvio. **Criação musical com suporte tecnológico**. In: Anais da Anppom, 1999.
- Composição e pesquisa: A categoria compositor-pesquisador ou o compositor que se perdeu num tubo de ensaio**. In: Anais da Anppom, 2000.
- NYMAN, Michael. **Experimental music – Cage and beyond**. Schirmer Books, New York, 1974.
- OPCODE Systems. **MAX Reference Manual**. Pablo alto, 1995.
- ROADS, Curtis. **Computer music tutorial**. MIT Press, Massachussets, 1996.
- TOOP, Richard. **György Ligeti**. Londres, Phaidon Press, 1999.
- WINKLER, Todd. **Composing Interactive Music – Techniques and ideas using Max**. MIT Press, Massachussets, 1993.
- XENAKIS, Iannis. **Determinacy and indeterminacy**. In: Organised Sound, Vol. 1, Nº 3, December 1996. Cambridge University Press, Cambridge, 1996.
- ZAMPRONHA, Edson. **Notação, Representação e Composição – Um novo paradigma da escritura musical**. São Paulo, Annablume/Fapesp, 2000.

Implementação da síntese FM em uma linha de atraso variável e suas possíveis aplicações no processamento de áudio em tempo real

Sérgio Freire

Escola de Música da UFMG - Campus Pampulha - Av. Antônio Carlos, 6627
31270-010 – Belo Horizonte – MG – Brasil

sfreire@musica.ufmg.br

Resumo: *O artigo apresenta uma proposta de transposição das técnicas e do repertório de timbres da síntese FM para o processamento digital de sons quaisquer, baseada no uso de linhas de atraso variáveis. Os cálculos necessários para uma correta adaptação dos parâmetros básicos da FM a essas linhas de atraso são acompanhados da discussão das possibilidades e de alguns cuidados em sua aplicação a sons instrumentais.*

Abstract: *The paper proposes a transposition of the techniques and the timbral repertoire developed by the FM synthesis to the digital processing of any sounds, which is based on variable delay lines. The calculation of the parametrical equivalence between both techniques is followed by a discussion of the possibilities – and some cautions – to be considered in its use with instrumental sounds.*

Introdução

O presente trabalho trata das possibilidades de aplicação da grande experiência adquirida com a síntese digital de som por meio da modulação de frequência (FM) no processamento de sinais de áudio em tempo real. Sua fundamentação se baseia no fato de que o efeito doppler pode ser visto como um caso particular de modulação em frequência (FM). O efeito doppler é facilmente obtido com uma das mais antigas técnicas de processamento sonoro – o atraso (*delay*) –, cuja versão digital oferece considerável flexibilidade na construção de linhas de atraso variável. Propõe-se aqui uma implementação da síntese FM baseada em uma linha de atraso variável equivalente à clássica implementação descrita por John Chowning em 1973. Em seguida são discutidas algumas particularidades e cuidados que devem ser tomados tanto em seu uso com um sinal de áudio em tempo real quanto na aplicação de formas de onda moduladoras diferentes da senoide.

Síntese FM e efeito doppler

A modulação em frequência tem por base a modificação da frequência de um sinal senoidal (chamada de frequência portadora) por um outro sinal senoidal (frequência moduladora). Ao se elevar a frequência da moduladora para uma ordem de grandeza auditiva (acima de 16 Hz), acontecem interações capazes de gerar uma variada gama de timbres. Os parâmetros de um sinal modulado em frequência são os seguintes: frequência portadora, frequência moduladora e desvio máximo da frequência portadora. A amplitude instantânea de uma onda modulada em frequência é dada pela fórmula:

$e = A \sin(2\pi f_p t + I \sin 2\pi f_m t)$, onde A é a amplitude máxima da modulação resultante, f_p é a frequência (em Hz) da onda portadora, f_m é frequência da moduladora e I é o índice de modulação, definido pela razão entre o desvio máximo (da portadora) e a frequência da onda moduladora¹. A variação temporal do índice de modulação permite a construção relativamente simples de timbres que apresentam uma evolução espectral característica. Outro fator de grande relevância sobre o resultado sonoro é a razão existente entre as frequências moduladora e portadora, responsável pelo grau de harmonicidade do timbre resultante.

O efeito doppler é percebido, no caso da audição, como uma alteração da frequência do som escutado, sendo causado pela variação da distância existente entre a fonte sonora e o ouvinte. Ou seja, o emissor e o receptor sonoros não mantêm entre si uma distância constante, que pode ser expressa por uma velocidade fixa ou variável no tempo. O cálculo da frequência resultante é expresso pela seguinte fórmula: $f = f_0 * \left(1 + \frac{v}{v_s}\right)$, onde f_0 é frequência emitida pela fonte sonora, v a velocidade existente entre o ouvinte e a fonte, e v_s a velocidade do som no ar.

Cálculo dos parâmetros da linha de atraso

Pode-se interpretar o atraso provocado em um sinal digital de áudio como a criação de uma distância entre este sinal e o ouvinte. Cada amostra representa uma duração de $(1/f_a)$, onde f_a é a frequência de amostragem. Em uma situação física, uma atraso de N amostras equivale a uma distância de $((N/f_a) * v_s)$ da fonte em relação ao ouvinte. Variar esse atraso significa imprimir uma velocidade relativa entre a fonte e o ouvinte, causando, por consequência, uma variação de frequência.

A implementação aqui proposta se baseia nos mesmos parâmetros da síntese FM, com a diferença de que o desvio máximo deve ser agora medido em amostras. Para se encontrar a relação existente entre os parâmetros de cada caso é necessário primeiramente transformar a variação do número de amostras (respectivamente da distância) em uma variação de velocidade, o que é facilmente obtido pela derivação da equação que representa o desvio instantâneo. Assim, se

$\Delta n(t) = -\frac{N_{\max}}{f_a} * v_s * \cos 2\pi f_m t$ representa a variação instantânea da distância entre a fonte e o ouvinte²,

$\Delta v(t) = \frac{\partial \Delta n(t)}{\partial t} = \frac{N_{\max}}{f_a} * v_s * 2\pi f_m * \sin 2\pi f_m t$ representa a velocidade instantânea entre eles.

Com isso, torna-se possível uma comparação direta entre as variações de frequência em cada um dos casos. A equação seguinte tem como seu membro esquerdo

¹ Ver Chowning (1973), p. 527. Ver também Tempelaars (1996), p. 248-253, para uma descrição mais detalhada das equações envolvidas na FM.

² A opção pela função cosseno com sinal negativo representa uma escolha da fase inicial da variação do atraso. Essa escolha coloca a variação de frequência na linha de atraso em fase com a variação presente na FM, já que a derivada de $-\cos(x)$ equivale a $\sin(x)$.

a variação de frequência devida ao efeito doppler, e no direito a variação de frequência em uma síntese FM:

$$f_p \left(1 + \frac{\frac{N_{\max}}{f_a} * v_s * 2\pi f_m * \sin 2\pi f_m t}{v_s} \right) = f_p + D_{\max} * \sin 2\pi f_m$$

Desenvolvendo ambos os termos e realizando as simplificações possíveis, obtém-se:

$\frac{N_{\max}}{f_a} = \frac{D_{\max}}{2\pi f_p f_m}$. Como D_{\max} é definido pelo produto de I por f_m , chega-se a:

$$\frac{N_{\max}}{f_a} = \frac{I}{2\pi f_p}. \quad (1)$$

O primeiro termo dessa equação é o desvio máximo medido em segundos, que depende – de modo diretamente proporcional - do índice de modulação e – de modo inversamente proporcional - da frequência portadora.

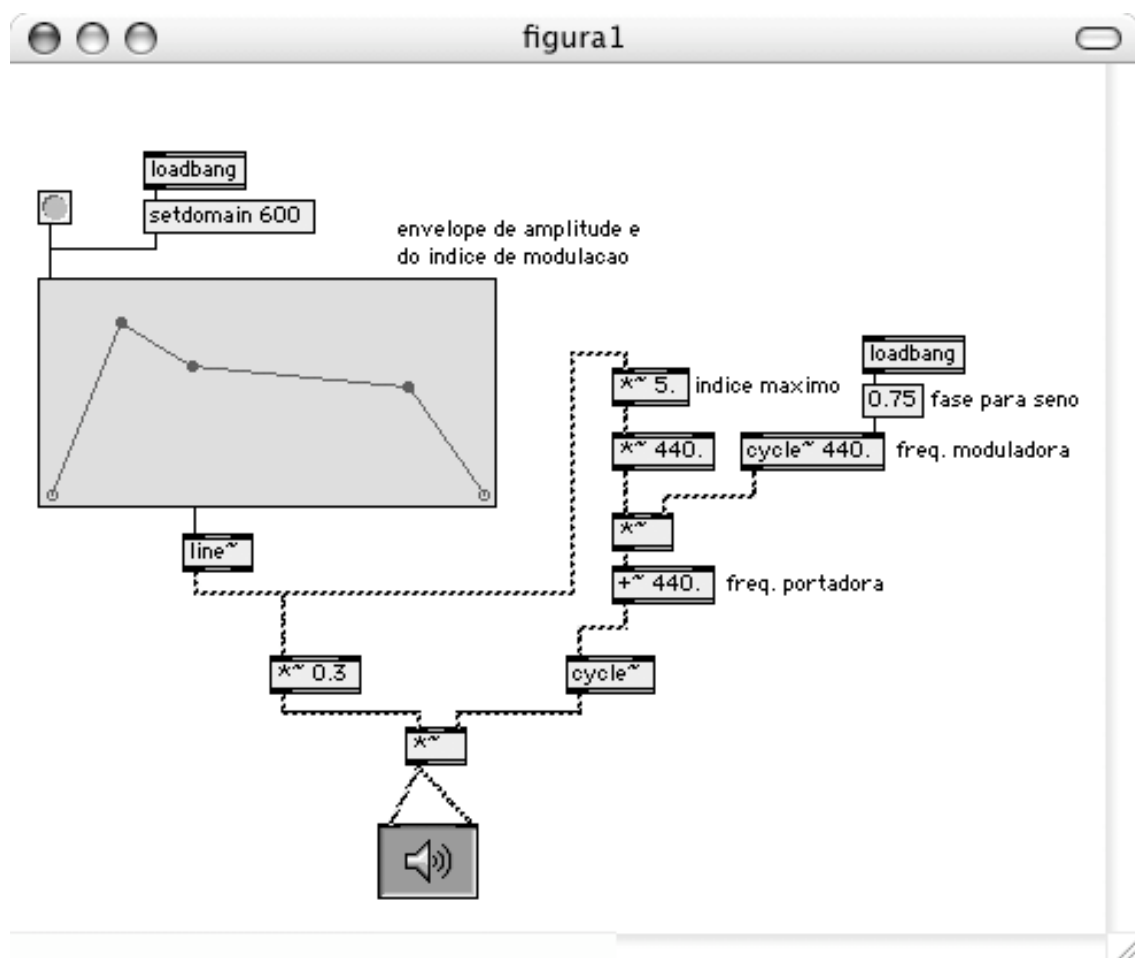


Figura 1: Implementação em max-msp da modulação em frequência de *brass-like tones* – descrita por Chowning (1973), p. 532.

de ca. 199 ms (cálculo advindo da equação 1), o que requer, sem o mecanismo de *wrapping*, um *buffer* mínimo de 398 ms.³

Modulação em frequência de sinais de áudio: precauções e possibilidades

Com o exposto acima, está aberto o caminho para a modulação em frequência de sinais de áudio quaisquer: basta conectá-los à entrada da linha de atraso. Mas, como utilizar esse recurso de processamento, mantendo parâmetros mínimos de comparação e alguma previsibilidade nessa experimentação? Minha primeira opção é a de limitar seu uso a aplicações em tempo real e a sons gerados por um único instrumento: aliam-se aqui uma fácil implementação e reais possibilidades de extensão tímbrica instrumental.

Nesse campo de exploração, os seguintes pontos devem ser considerados, levando-se em conta ainda que diferentes graus de independência ou hierarquia podem aparecer entre eles, dependendo de contextos específicos:

a) os sons a serem modulados em frequência apresentam, obviamente, um espectro harmônico mais rico do que o da onda portadora original (senóide). Esses harmônicos também sofrem a modulação em frequência, e respondem ao índice de modulação de modo proporcional ao seu índice na série harmônica. Ou seja, o segundo harmônico (cuja frequência é o triplo da frequência fundamental) sofrerá uma modulação efetiva três vezes mais forte do que o som fundamental⁴. Sons percussivos com espectro inarmônico podem ganhar uma nova configuração energética entre seus parciais sem perder totalmente sua identidade. Deve-se contar ainda com a possibilidade de surgimento de *aliasing*; para evitá-lo, basta filtrar o sinal de entrada, sempre que necessário.

b) sons instrumentais, são, via de regra, variáveis no tempo, o que gera automaticamente um espectro com características dinâmicas. Porém, variações muito bruscas na forma de onda de entrada (principalmente em frequência e timbre) podem criar modulações inesperadas ou não desejadas.

c) um controle mais refinado do índice de harmonicidade requer o conhecimento preciso da frequência do sinal de entrada. Um algoritmo de detecção de alturas pode ser usado⁵, ou mesmo uma partitura com clara definição das alturas a serem tocadas.

d) é importante definir como será feito o controle do envelope dinâmico do processamento. Na síntese FM, o disparo de um novo som inicia automaticamente os envelopes relativos à amplitude e ao índice de modulação. Aqui, diferentes situações podem ser imaginadas, que vão do simples disparo de envelopes pré-fixados ao controle exercido pelo próprio envelope do som de entrada. Outras possibilidades são abertas pelo uso de controladores contínuos (um pedal de volume, p. ex) por parte do músico.

Pode-se também imaginar que um som instrumental de curta duração seja capaz de gerar um evento mais longo: nesse caso, deve-se implementar algum tipo de

³ A implementação de um oscilador do tipo *look-up table* baseia-se fundamentalmente nessa técnica de *wrapping*, já que pressupõe que um ciclo da onda a represente em uma duração infinita. Moore (1992), p. 159-173, apresenta uma discussão detalhada da implementação de um oscilador desse tipo.

⁴ O mesmo se dá em relação a uma onda complexa que serve a um oscilador do tipo *look-up table*. Um ciclo completo do oscilador gera um ciclo para a frequência fundamental, ao mesmo tempo em que gera três ciclos para o segundo harmônico.

⁵ Tais como *fiddle~*, de Miller Puckette, ou *pitch~*, de Tristan Jehan.

realimentação da linha de atraso que, como no caso do *wrapping* descrito mais acima, deve manter uma relação de números inteiros com o período do som de entrada (onda portadora).

e) os envelopes de amplitude, do índice de modulação e da harmonicidade podem ser controlados de forma totalmente independente; o contexto de sua aplicação é que deve determinar o grau de interdependência.

f) formas de onda alternativas à senóide podem ser usadas para a modulação. Nessa situação, a experiência adquirida com a síntese FM pode ser de grande utilidade para uma certa previsibilidade dos resultados. De maneira bem geral, pode-se afirmar que a influência de uma certa forma de onda sobre a modulação do atraso é equivalente ao efeito de sua derivada aplicada à forma de onda da moduladora em uma síntese FM. Assim, uma onda triangular aplicada à modulação do atraso corresponde a uma onda quadrada aplicada à modulação da frequência de uma onda portadora; o mesmo se dá entre uma onda quadrada e um trem de pulsos.

Dois exemplos bastante simples podem ajudar a ilustrar algumas possibilidades dessa modalidade de processamento:

(1) uma nota sustentada de clarineta é modulada em frequência, com índice de harmonicidade igual a 2, e índice de modulação variando de 0 a 5, e novamente até 0. A figura representa o espectrograma do som original e do som modulado.

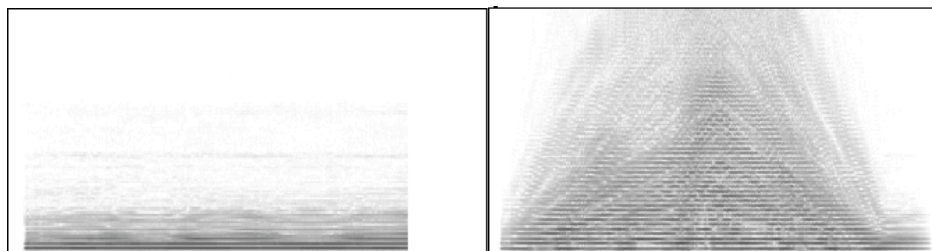


Figura 3: sonogramas de uma nota longa de clarineta (à esquerda), e desta mesma nota modulada em frequência (à direita).

(2) um som de tamborim repetido algumas vezes é modulado com diferentes graus de harmonicidade, com um índice de modulação igual a 2.5 (é utilizada uma frequência fundamental de 481.7 Hz). O novo timbre apresenta, ao lado da variação em alturas, uma acentuada característica metálica.

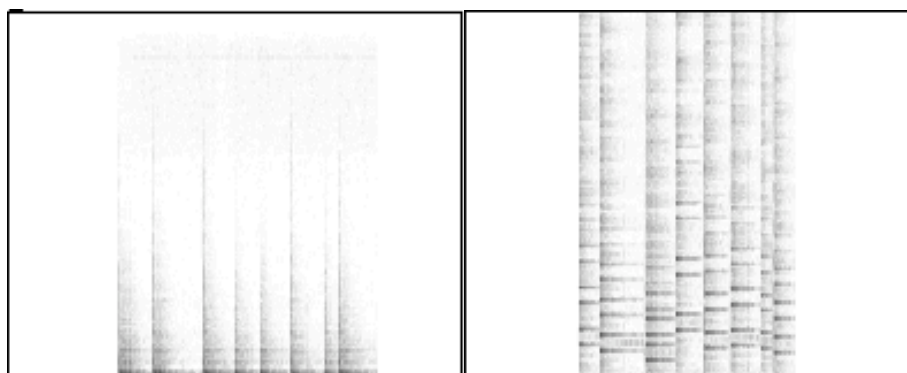


Figura 4: sonogramas de uma sequência rítmica de um tamborim (à esquerda), e desta mesma sequência modulada em frequência (à direita).

Conclusão

A implementação proposta, embora extremamente simples, apresenta um grau considerável de flexibilidade para aplicações em tempo real, principalmente na extensão dos timbres de instrumentos tradicionais. Boa parte dessa flexibilidade vem da impossibilidade de sua aplicação imediata, já que são necessárias diversas definições adicionais: detecção da frequência portadora, tipo de controle dos diferentes envelopes, espectros harmônicos esperados etc. Com isso, espera-se que boa parte da riqueza de timbres, que deu à síntese FM uma relativa hegemonia na geração sonora nos anos 1980 e 90, possa ser adaptada ao processamento de sons de instrumentos acústicos.

Bibliografia

Chowning, John M. (1973), "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation", *Journal of the Audio Engineering Society* 21(7): 526-534.

Moore, Richard F., *Elements of Computer Music*, Englewood Cliffs: Prentice-Hall, 1992.

Puckette, Miller, Apel T. (1998), "Real-time audio analysis tools for PD and MSP", *Proceedings of the ICMC 1998*. San Francisco, Internacional Computer Music Association, p. 109-112.

Schottstaedt, B. (1977), "The Simulation of Natural Instrument Tones Using Frequency Modulation with a Complex Modulating Wave", *Computer Music Journal* 1(4):46-50.

Tempelaars, Stan, *Signal Processing, Speech and Music*, Lisse: Swets & Zeitlinger, 1996.

Bodyweave: ambiente computacional para um encontro poético entre corpo e sonoridades

Andrea C. B. Krotoszynski^{1,2}, Jarbas de Moraes Neto³, Jônatas Manzolli^{2,4}

Comunicação e Artes do Corpo – Pontifícia Universidade Católica de São Paulo – PUC¹

Núcleo Interdisciplinar de Comunicação Sonora – NICS/Unicamp²

Departamento de Matemática – IMECC/Unicamp³

Departamento de Música, Instituto de Artes – DMIA/Unicamp⁴

lalik@ajato.com.br, {jarbas, jonatas}@nics.unicamp.br

Abstract. *This paper describes concepts, creative aspects and the sound design in Bodyweave: computational environment for relating poetically sounds and body. Bodyweave, an interactive interface works as medium and partner with the user in the composition of sound combinations and images of body in movement mini-clips.*

Resumo. *Este trabalho trata da descrição dos conceitos, dos aspectos criativos e do projeto de sonorização de Bodyweave: ambiente computacional para um encontro entre corpo e sonoridades. Bodyweave é uma interface interativa que atua como meio e colaborador do usuário na composição de pequenas animações contendo combinações sonoras e de imagens de corpo em movimento.*

1. Introdução

A proposta de Bodyweave pressupõe a criação de um sistema dinâmico (interface computacional) gerador de composições sonoro-imagéticas em parceria com o usuário, e é constituído pelos elementos: sonoridades, imagens que fazem referência ao corpo humano em movimento, recursos e suporte eletrônicos. O nome Bodyweave (cuja tradução seria “tecer, entrelaçar corpos”) faz menção à idéia de trama, ou tecido, e incorpora o conceito-chave da proposta: transformar elementos isolados em composições animadas geradoras de sentido para o interator, que então se torna implicado no processo de criação dadas as inúmeras possibilidades de relação entre imagens e sons que a proposta interativa oferece.

O processo de elaboração partiu de modelos culturais e técnicos como o haikai, o I Ching, o quebra-cabeça, os princípios do cinema e aplicativos de edição sonora e de vídeo que compartilham características com o processo de composição coreográfica. O Bodyweave encontra-se no âmbito das Artes Interativas Mediadas (AIM), isto é, toda arte interativa que se vale de interfaces eletrônicas para servir de intermediária indiretamente entre o artista e seu público. Também foram tecidas relações entre

Bodyweave e a idéia de *auto-organização* por sua capacidade generativa, ou de emergência. Quanto aos processos de composição musical que a interface proporciona, eles se relacionam ao minimalismo e a outros processos de composição, como a síntese granular e a incorporação do acaso no paradigma da composição.

2. Conceitos e motivação

A investigação conceitual do processo de implementação de Bodyweave procurou localizar coordenadas referenciais no âmbito da arte contemporânea, em especial na Arte Interativa Mediada e Generativa, ao mesmo tempo que buscou, nas relações desenvolvidas no âmbito da idéia de auto-organização, agregar consistência teórica ao processo puramente artístico de criação. Tais referências vem sendo exploradas por (Lali) Andrea Krotoszynski desde sua monografia de conclusão do curso Comunicação e Artes do Corpo: Performance e Auto-organização (2004), o que a levou a procurar a orientação de Jônatas Manzolli, que viria a contribuir com seu conhecimento de auto-organização e atuação no campo de música computacional.

A equipe de trabalho se formou a partir da proposta de uma reflexão intelectual que acompanhasse o processo de criação e desenvolvimento técnico do projeto de Bodyweave; tendo se formalizado com o pedido de bolsa de Iniciação Científica para a Fapesp sob a orientação de Jônatas Manzolli. Jarbas de Moraes Neto, seu aluno do Departamento de Matemática, veio a integrar a equipe como responsável pela implementação da proposta no ambiente computacional. O relatório final intitulado: A Auto-Organização em Obras de Arte Contemporânea Envolvendo Corpo, Sonoridades, Interatividade e Novas Mídias, elaborado pela equipe, foi recebido e aprovado pela Fapesp em 2005.

2.1. Auto-organização e a Arte Interativa Mediada

A idéia de auto-organização (Debrun, 1996) nasce da Teoria Geral dos Sistemas (Bertalanffy, 1968) e da Cibernética (Ashby, 1970), que se propõem a abranger uma perspectiva interdisciplinar entre diferentes ciências, enfatizando a organização intrínseca e a interdependência do mundo em todas as suas manifestações.

A Arte Interativa Mediada (AIM) opera com conteúdos os mais diversos, constituindo, em cada experiência de interação, um conjunto único de idéias, concepções estéticas e redes de significados. Por sua vez, este conjunto singular engendra uma coesão própria. A capacidade de imbricar os elementos disjuntos através de um processo de organização engendra o fenômeno que relacionamos com o conceito de auto-organização.

Michel Debrun (1921-1997), filósofo francês que foi um dos fundadores do Centro de Lógica, Epistemologia e História da Ciência (CLE) da Unicamp em 1977, define a auto-organização como processo de evolução em um sistema, qualquer que seja sua natureza, em direção a um estado de (des)equilíbrio organizado, sem a intervenção de fatores externos ao sistema ou imposição de um ou mais elementos do sistema. A auto-organização ocorre como uma característica operativa que emerge das relações intrínsecas ao sistema. Segundo ele: “*A auto-organização só existe como tal enquanto imperfeita. Ou melhor, enquanto não houver uma ‘maximização’ do sujeito auto-organizador*” (Debrun, 1996, p.29).

Assim, ocorre auto-organização em um corpo, um grupo de pessoas, uma obra artística, se estes forem considerados como sistemas complexos que funcionam através da colaboração entre os elementos que os constituem.

A partir destes vetores, pode-se projetar uma relação imbricada entre processos de organização, complexidade e arte contemporânea, principalmente, no que se refere à AIM. Podemos sintetizar esta relação da seguinte forma: O sistema operativo da auto-organização gerencia o mecanismo de emergência do sistema enquanto a AIM utiliza-se desta capacidade organizacional como mecanismo de construção estrutural que dá a ela sua forma e identidade. Assim, a mediação é o contexto no qual a obra se insere, e a articulação de seu conteúdo, a própria obra que compartilha sua autoria com o usuário – e, em alguns casos como o de Bodyweave, com a interface em si.

3. Sonorização

Cada sonoridade e cada imagem constituem unidades em princípio independentes, e são associadas na proporção 1x1 por atributos estéticos/poéticos predefinidos. As medidas de tempo das unidades sonoras foram escolhidas em proporção com a duração média da soma do tempo de cada um dos seis quadros de cada animação.

3.1. Amostras Sonoras

O processo de armazenamento das amostras sonoras necessárias para a constituição do “vocabulário sonoro” de Bodyweave se deu segundo as seguintes etapas:

- a) Músicas que contivessem elementos sonoros cujas características se mostravam evidentes à escuta da autora foram sendo extraídas de suas fontes, ou seja, de CDs de seu próprio acervo, com o uso do *software* Windows Media Player.
- b) A seguir, os trechos em que se localizavam os elementos de evidência (ou atratores, aqui principalmente considerados por sua potencialidade expressiva ou poética de acordo com a deliberação subjetiva da autora como artista) foram extraídos das músicas através do programa Sound Forge e agrupados conforme parâmetros de similaridade subjetiva, tendo elementos da natureza como modelos para grupos.
- c) Neste momento do processo foi criada a “*Tábua de Elementos e Estados*” a partir de características de elementos da natureza apontadas pelas amostras recolhidas. A tábua delimitou o universo de relações a serem operacionalizadas pela interface em uma matriz 6x6, descrita em detalhes na seção seguinte.
- d) O trabalho de audição e seleção de sonoridades passou a ter a “*Tábua de Elementos e Estados*” como referência.
- e) Por último foi realizada a normalização de volume das sonoridades segundo três parâmetros, para que as sonoridades, em sua apresentação final como *loops*, adquirissem *nuances* de volume. Os parâmetros foram condicionados segundo máxima intensidade de pico para depois adquirirem os valores -3, -6 e -9 dB, conforme deliberação da autora.

A pesquisa iniciada em trabalhos anteriores, e, fundamentada na manipulação de quadros (*still frames*) de vídeo em experiências de diferentes combinações, permutações, extrações e adições, em seqüências de no mínimo dois quadros e no

máximo oito, trouxe resultados estéticos e artísticos que motivaram a continuação destes procedimentos, acrescentando-se a eles a experiência de vincular cada quadro a uma unidade sonora, de forma que a cada composição visual produzida pela animação dos quadros correspondesse uma composição sonora. Em Bodyweave experimentamos fornecer um som correspondente a cada imagem nos grupos de unidades a serem operacionalizadas pelo usuário. Tal opção revela relações com conceitualizações da música contemporânea, como por exemplo, os *screens* de Xenakis e com o conceito de Schaeffer de *objeto sonoro*, que, no presente trabalho, constarão apenas como indicações para futuras reflexões sobre o processo.

3.2. Taxonomia e Herança Poética

Para buscar a melhor correspondência entre sonoridade e imagem, de forma que a unidade som-imagem pudesse ser identificada como um elemento dotado de características próprias, foi desenvolvida uma *Tábua de Elementos e Estados (TEE)*.

A idéia de criar a TEE surgiu a partir da observação do sistema do I Ching (*I Ching – O livro das mutações*, livro-oráculo chinês que começou a ser desenvolvido há cerca de 3.000 anos e chegou à forma que conhecemos hoje por volta de 200 a.C.) e de sua estrutura de duas linhas distintas (a linha inteira e a linha interrompida), combinadas em oito trigamas que geram os 64 hexagramas diferentes que compõem o oráculo a ser interpretado pelo consulente.

CHAVE PARA OS HEXAGRAMAS

Trigrama Exterior Trigrama Interior	Ch'ien ☰	Chen ☷	K'an ☵	Ken ☶	K'un ☴	Sun ☳	Li ☲	Tui ☱
Ch'ien ☰	Ch'ien 1	Ta Chuang 34	Hsü 5	Ta Ch'u 26	T'ai 11	Hsiao Ch'u 9	Ta Yu 14	Kuai 43
Chen ☷	Wu Wang 25	Chen 51	Chun 3	I 27	Fu 24	I 42	Shih Ho 21	Sui 17
K'an ☵	Sung 6	Chieh 40	K'an 29	Meng 4	Shih 7	Huan 59	Wei Chi 64	K'un 47
Ken ☶	Tun 33	Hsiao Kuo 62	Chien 39	Ken 52	Ch'ien 15	Chien 53	Lü 56	Hsien 31
K'un ☴	Pi 12	Yü 16	Pi 8	Po 23	K'un 2	Kuan 20	Chin 35	Ts'ui 45
Sun ☳	Kou 44	Heng 32	Ching 48	Ku 18	Sheng 46	Sun 57	Ting 50	Ta Kuo 28
Li ☲	T'ung Jen 13	Feng 55	Chi Chi 63	Pi 22	Ming I 36	Chia Jen 37	Li 30	Ko 49
Tui ☱	Lü 10	Kuei Mei 54	Chien 60	Sun 41	Lin 19	Chung Fu 61	K'uei 38	Tui 58

Figura 1. Reprodução de tabela para consulta do nome do hexagrama sorteado (Legge, 1984)

A estrutura do I Ching é em si interativa, cabendo ao consulente formular sua questão e sortear, por métodos sugeridos no livro, seis linhas que vão constituir dois trigramas, que, por sua vez, irão gerar um hexagrama. O oráculo correspondente ao hexagrama sorteado se encontra na tabela de hexagramas. O oráculo pressupõe um simbolismo que se relaciona com a cultura monárquica chinesa e as tradições confucionista e taoísta. O exercício de buscar congruência entre o oráculo sorteado e a questão formulada, se considerado independentemente da simbologia própria envolvida, constitui uma dinâmica similar à que queremos implementar no decorrer da interação do

usuário com a interface. Isto é, queremos que o usuário estabeleça relações de natureza estética e simbólica durante sua interação com Bodyweave.

Nossa intenção foi a de desenvolver uma interface capaz de gerar narrativas cujo conteúdo é em parte fornecido pelo artista (coleção de imagens e sons) e em parte formulado pelo usuário. Depois de ter colecionado um número razoável de sonoridades numa faixa de duração entre décimos de segundo e 5 segundos, estabelecemos categorialmente um sistema que caracterizasse diferentes famílias de imagens e sons relacionados.

A TEE foi desenvolvida para este fim. Os elementos, ou categorias, surgiram de associações subjetivas sugeridas pelos próprios sons extraídos do acervo musical em CD. Utilizamos também, como referência, a proposta estética extremamente sintética e ampla do haikai tradicional, que se baseia exclusivamente em imagens da natureza. A TEE se configurou como reproduzida a seguir.

	FOGO	ÁGUA	TERRA	METAL	MADEIRA	AR
FOGO	fogueira	ebulição	magma	derretido	carvão	rarefeito
ÁGUA	ebulição	corrente	lama	polarizado	decomposição	vapor
TERRA	magma	lama	chão	oxidado	viva	livre
METAL	faísca	gelo	grumo	bruto	celulose	bolha
MADEIRA	brasa	umidade	fértil	limalha	âmbar	sopro
AR	fumaça	vapor	poeira	pó	farpa	vento

Figura 2. Tabela criada especialmente para Bodyweave

Haikai é um estilo de escrita poética da cultura clássica japonesa que percorreu uma trajetória de mais de 20 séculos e continua sendo praticada atualmente. Determinados conceitos, como a sinceridade por exemplo, vinculam a prática da poesia ao cultivo do caráter e do espírito, fortalecendo valores taoístas e budistas inerentes à tradição. Imagens da natureza, o fluir do tempo, a contemplação e fruição da beleza e a consciência da vida neste mundo transitório são seu conteúdo principal.

Os preceitos do haikai forneceram algumas de suas propriedades como referências para o tipo de interação que desejávamos estabelecer entre Bodyweave e potencial usuário. Particularmente, nos interessávamos em incorporar a estratégia típica do haikai, que busca na expressão em palavras, de maneira direta, aquilo que se passa no coração do autor. No haikai, a observação direta, sem reflexão, por um impulso do espírito, é considerada desejável, ao contrário da elaboração intencional de um bom poema. Esta observação consiste no exercício da visão livre, desvinculada de opiniões pessoais e valores transitórios e deve ser aprendida e aperfeiçoada pelo praticante de haikai.

Em Bodyweave, a intenção foi a de estimular o usuário a interpretar as animações resultantes de sua interação com a interface utilizando este tipo de estratégia de associação espontânea. Ou seja, as imagens e amostras sonoras funcionariam como versos do haikai.

Entre os versos que constituem um haikai não há necessidade de se estabelecer um nexo óbvio. Às vezes, a relação entre as partes é de natureza metafórica, em que uma parte ilumina e dá significado a outra. São utilizados recursos como a justaposição direta de imagens de alguma forma complementares, ou a utilização de uma estrofe final que apresenta um tipo de comentário ou exemplo do que se estabelece nas estrofes anteriores.

A estética do haikai nos forneceu um modelo de estratégia para que combinações de poucos elementos resultasse em peças sintéticas, breves e ricas em imagens e significados. O sistema formado por um número limitado de imagens e sons adquire integridade estético/conceitual por meio desta estratégia, ao mesmo tempo em que permite o surgimento de associações inesperadas entre seus elementos.

4. Visualização

Em Bodyweave, assim como em experiências anteriores, foi empregada a idéia derivada do mecanismo de animação. Imagens bidimensionais mostradas em sequência, na velocidade de 24 quadros por segundo, dão ilusão de movimento devido ao fenômeno óptico da persistência retiniana que é o princípio básico do cinema. Esta propriedade foi utilizada neste trabalho no limite entre a ilusão de movimento e a leitura de situações em sucessão em menos quadros por segundo do que 24, o que resulta em uma ação que não induz à ilusão de movimento contínuo, mas que ainda assim orienta o espectador a estabelecer relações associativas entre um quadro e outro. O conjunto de quadros assim apresentados agencia espontaneamente a capacidade interpretativa do observador. Este recurso encontra-se muito bem fundamentado na obra do cineasta e teórico russo Eisenstein (1898-1948), e enxergamos similaridades de sua utilização no caso de palavras na técnica poética do haikai (como concebida, por exemplo, por Haroldo de Campos em seu livro *A arte no horizonte do provável*) e na idéia de contínuo em estruturas sonoras, como as apresentadas por compositores de música contemporânea na década de 1950, como Stockhausen, Ligeti e Penderecki.

Também foram identificadas outras referências relevantes – e que se encontram hibridizadas na gênese de Bodyweave – que aparecem nos processos de animação e reiteração de quadros. Estes são de caráter minimalista (segundo Steve Reich e Philip Glass, entre outros) e se valem de processos de repetição que levam a graduais transformações no tempo.

A experiência de criação de pequenas animações nos mostrou que mesmo poucos quadros encadeados permitem a construção de uma narrativa por parte do observador, e esta se torna ainda mais significativa quando acompanhada de sonoridades. Estabelecemos a medida de seis quadros para cada animação, de forma a permitir uma interação dinâmica e eloquente ao mesmo tempo.

5. Implementação computacional

Macromedia Flash MX 2004 foi a plataforma considerada mais adequada, pois trabalha com imagens e sons com agilidade, permitindo também posterior implementação na Internet.

5.1. Randomização da disposição das bolinhas

Para criar uma dinâmica interator-interface de caráter lúdico, no qual o interator fosse convidado a realizar opções que resultassem em uma resposta autonôma da interface, desenvolvemos uma animação inspirada no Movimento Browniano (movimento descrito por uma agitação irregular, em todas as direções, de algumas partículas suspensas em um meio fluido). A animação ocorre logo após a realização da primeira tarefa do interator: escolher, entre as 36 bolinhas dispostas no menu, três delas, o que ocorre somente depois que o usuário clicar sobre cada uma das escolhidas.

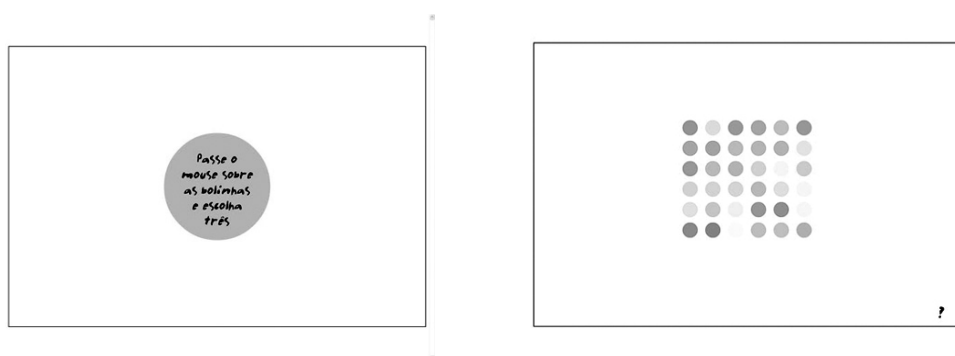


Figura 3. Screen shots: a) primeira ação na tela b) menu apresentado para escolha do usuário

A interface apresenta então a simulação de um *random walk* (ou “movimento aleatório”) como contrapartida da interface a ação do interator. Na simulação, são apresentadas 16 bolinhas que integram a família de cada uma das três bolinhas previamente escolhidas, que se espalham no espaço delimitado na tela com o sabor do acaso; surgindo no próximo ambiente da interface “embaralhadas”.

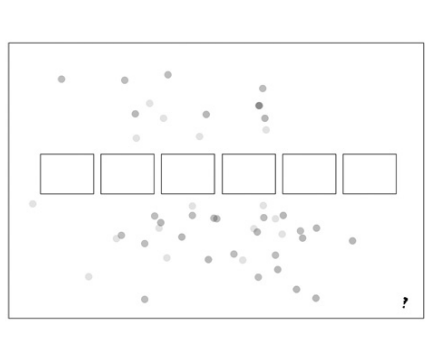


Figura 4. Screen shot de uma configuração formada depois que as 48 bolinhas se espalham randomicamente pelo espaço da tela

5.1.2. Pesos

A segunda intervenção solicitada ao interator é a de escolher seis bolinhas, entre as 48 opções que se estabeleceram randomicamente no espaço delimitado. Tendo sido realizada a escolha, o interator deve clicar sobre cada uma delas e, em seguida, arrastar uma a uma, posicionando-as nos seis campos vazios no centro da tela.

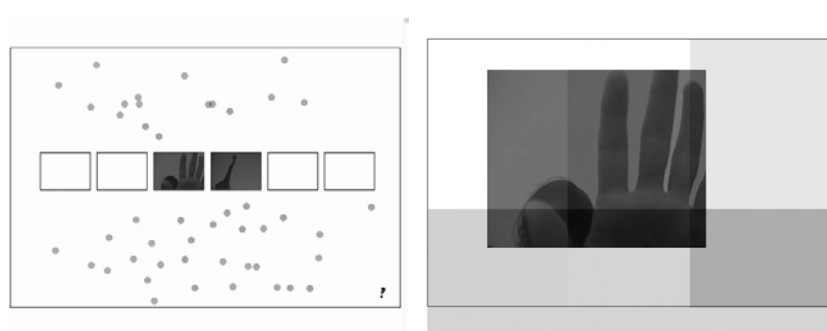


Figura 5. a) *Screen shot* de duas bolhinhas escolhidas e arrastadas para dois dos retângulos disponíveis b) momento de formação da animação: máscaras coloridas translúcidas cobrem o campo na tela para proporcionar uniformidade na tonalidade de cor entre frames de diferentes famílias

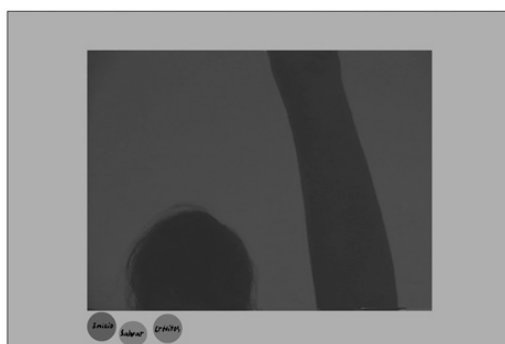


Figura 6. animação em curso

Neste momento é aplicado um algoritmo randômico que confere diferentes pesos a cada um dos seis *frames* que equivalem a cada uma das seis bolinhas escolhidas e que se revelam como tais a partir daí, ou seja, as bolinhas se revelam como *frames*.

Os pesos de valores estipulados entre 1 e 10 fornecem ritmo e associam a duração sonora com a permanência do quadro no tempo, o que confere uma unidade intrínseca à animação sonorizada resultante desta intervenção.

A unidade animada tem a duração total de dois segundos (os *loops* apresentados variam entre 3 e 10 segundos), sendo que cada *frame* adquire um peso que vem a determinar a sua permanência na animação. Assim chegamos às seguintes fórmulas para a permanência relativa das imagens:

$$\text{Soma dos pesos : } S = \sum_{i=1}^6 p_i, \text{ com } p_i \in N \text{ e } 1 \leq p_i \leq 10 \quad (\text{Eq. 0.1})$$

$$\text{Normalização : } d_i = \frac{2p_i}{S}, \text{ onde } d_i \text{ é a duração que a imagem } i \text{ terá nos dois segundos} \\ (\text{Eq.0.2})$$

A partir deste esquema de proporções cada som se relaciona na animação conforme a seguinte configuração:

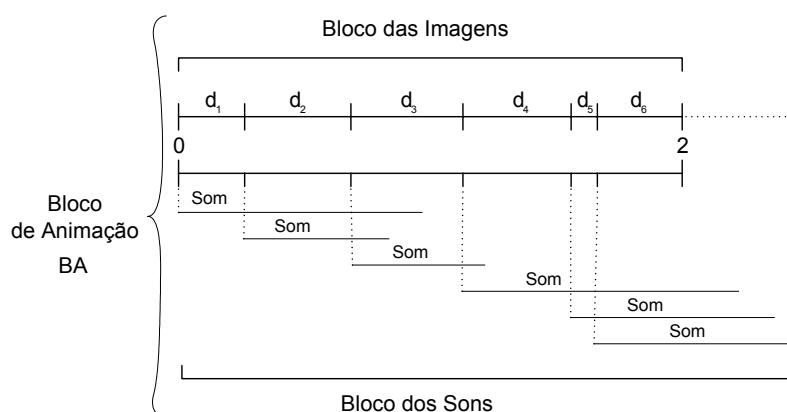


Figura 6. Esquema do processo de distribuição de pesos relativos a cada diferente composição tomando-se a duração d_i (i.e. Eq. 1.0 e 2.0) de cada uma das amostras sonoras como parâmetro para a permanência de cada frame na animação. A soma das durações individuais deu origem a duração total da animação ou loops (utilizamos valores entre 3 e 10 segundos).

Na interação entre som e imagem buscamos relações que estivessem alinhadas com o conceito inicial do trabalho: tecer, entrelaçar corpos. Neste sentido, entendemos que estes “dois corpos” deveriam se entrelaçar de forma orgânica dando origem a um tecido, uma textura sonoro-visual. Assim, à medida em que as imagens sucediam-se na animação, os blocos sonoros, relacionados a cada uma delas, eram superpostos. A duração de um som poderia ser maior que a duração de uma imagem, mas a soma das durações dos sons não seria maior que a duração total da animação. Com isto, obtivemos um jogo textural onde a interface criou condições para que o público explorasse diferentes configurações e pudesse experimentar através da escuta as diferentes matizes sonoras relacionados os elementos da Tabela apresentada na Figura 2.

6. Discussão

Uma reflexão decorrente das investigações prático-conceituais deste processo evidencia o fato de que a gênese na interação público-obra, no caso da AIM, pressupõe um controle relativo por parte do artista como sujeito criador, que compartilha a responsabilidade de conclusão da obra com cada interator como sujeito(s) co-criadore(s).

Esta consideração pode sugerir que a arte contemporânea vem traçando um percurso em que os paradigmas da estética modernista (o domínio do artista sobre a obra, o seu significado e o direcionamento da experiência perceptiva-interpretativa de seu público) vão progressivamente ganhando características de processos de auto-organização na proposta pós-moderna. Isso pode indicar que a vertente da arte interativa e de criação distribuída vem percorrendo um caminho que requer graduações em maior ou menor proporção de processos de auto-organização.

Nesta linha de pensamento existiria um gradual declínio da importância do sujeito-criativo na passagem do processo artístico autoral para o processo de auto-organização. Ambos os processos acontecem a partir da autonomia de um sistema, porém o sistema autoral é auto-referente e a função do sujeito é hegemônica, enquanto o sistema auto-organizado só pode ser produto de uma dinâmica sistêmica entre as partes, deslocando a hegemonia para a própria dinâmica de produção.

Naturalmente, o controle no processo artístico interativo mediado não se localiza exclusivamente no artista ou no processo, ou ainda no público interator, mas se distribui em instâncias dispersas em todo sistema. Em cada proposta, esta distribuição será enfocada pelo artista, ao mesmo tempo que sugerida pelas próprias articulações intrínsecas ao processo como um todo.

Neste sentido, poderíamos encarar o conjunto de mecanismos propostos para AIM como um processo de auto-organização secundário (Debrun, 1996), pois os elementos de imbricamento colocados pelo artista serão alvo de interação co-criativa do público.

Por outro lado, ao propor um processo interativo mediado, o artista incorpora a irreversibilidade dos processos pelos quais a ciência clássica tem sido questionada (Prigogine, 1997).

Portanto, na medida em que estas duas entidades (autor/emissor e público/receptor) passam por um processo de imbricamento e fragmentação, é necessário que se faça uma reflexão sobre a natureza do processo criativo, da autoria e da concepção da *poiesis* (num sentido ampliado), não mais com um foco “auto-poiético”, para que se possa descrever e estudar com propriedade a Arte Interativa Mediada (AIM).

Agradecimentos

Nossos sinceros agradecimentos ao apoio e estrutura das seguintes instituições que viabilizaram a realização desta pesquisa: Fapesp, NICS e Curso Comunicação e Artes do Corpo PUC-SP.

Referências

- Ashby, W. R., “Uma introdução à Cibernética”, Editora Perspectiva, São Paulo, 1970.
- Bertalanffy, L. V., “General System Theory”, Braziller, New York, 1968.
- Bresciani Filho, E.; D’Ottaviano, I. M. L. (1996), “Conceitos básicos de sistêmica” In: Debrun, M.; Gonzales, M. E. Q.; Pessoa JR, O.; D’Ottaviano, I. (Org.) “Auto-

- Organização: estudos interdisciplinares”, Editora da Unicamp, Coleção CLE 18, Campinas.
- Campos, H., “A arte no horizonte do provável” São Paulo: Perspectiva, 1977.
- Dantas, L.; Fracetti, P.; Taeko Doi E., “Hai Kai, antologia e história”, Editora da Unicamp, Campinas, 1990.
- Gonzales, M. E. Q.; D’Ottaviano (2000), “Auto-Organização: organização, estudos interdisciplinares”, Editora da Unicamp, Coleção CLE 30, Campinas.
- Debrun, M., “A idéia de auto-organização” In: Debrun, M.; Gonzales, M. E. Q.; Pessoa JR., O. (Org.) “Auto-Organização: estudos interdisciplinares” Editora da Unicamp, Coleção CLE 18, Campinas, 1996.
- Eisenstein, S., “Film Form”, Harcourt, Brace and Company, New York, 1949.
- Krotoszynski, A., (2004) “Performance e auto-organização”, monografia de conclusão de curso Comunicação e artes do corpo – PUC, São Paulo.
- Krotoszynski, A., Manzolli, J., Moraes Neto, J., (2005) “A Auto-Organização em Obras de Arte Contemporânea Envolvendo Corpo, Sonoridades, Interatividade e Novas Mídias”, Relatório de Iniciação Científica – Fapesp, São Paulo.
- Legge, J., “I Ching, O livro das mutações”, p.50, Hemus Editora, São Paulo, 1984.
- Lévy, P., “A inteligência coletiva. Por uma antropologia do ciberespaço”, Trad. Rouanet, L. P., Edições Loyola, São Paulo, 1998.
- Gardner, H., “The Mind’s New Science: A History of the Cognitive Revolution”, Basic Books, Inc., New York, 1985.
- _____. “Estruturas da mente. A teoria das inteligências múltiplas”, Artes Médicas, São Paulo, 1994.
- Kelso, S., “A auto-organização e os caminhos da consciência”, Jornal da Unicamp, n. 212, maio de 2003, Campinas.
- Manzolli, J., “Auto-organização: um paradigma composicional” In: Debrun, M.; Gonzales, M. E. Q.; Pessoa JR., O. (Org.) “Auto-Organização: estudos interdisciplinares” Editora da Unicamp, Coleção CLE 18, Campinas, 1996.
- Manzolli, J.; Gonzales, M. E. Q.; Vershure, P., “Auto-Organização e criatividade”, (2000) In: D’Ottaviano, I. M. L.; Gonzales, M. E. Q. (Org.), “Auto-organização: estudos interdisciplinares”, Editora da Unicamp, Coleção CLE 30, Campinas.
- Manzolli, J., (2005), “Compondo com o mundo real: paisagem sonora de labirintos entrelaçados” Tese de Livre Docência – Instituto de Artes da Universidade de Campinas.

gemini flux: Música electrónica, improvisación y código abierto

Andrés Cabrera

Universidad Javeriana – Departamento de música

Cr. 7 #40 – 62 – Bogotá - Colombia

andres@geminiflux.com

Sinópsis. *'gemini flux' es un proyecto de música electrónica con una gran dosis de improvisación y tratamiento de audio en tiempo real. Es una exploración en técnicas de improvisación dentro de la música electrónica y en el control inusual de herramientas electrónicas acompañada de elementos visuales para buscar un nuevo estilo de 'teatro musical'. El desarrollo de la obra '01-radioreloj' se hizo usando herramientas muy variadas y la ejecución se hace principalmente con herramientas de código abierto. La creciente complejidad del proyecto durante su fase de composición generó necesidades y problemas adicionales que definieron la interfase de ejecución y condujeron a la adopción de Linux como sistema operativo para el procesamiento de audio.*

Abstract. *The 'gemini flux' project plays electronic music incorporating plenty of improvisation and real-time audio processing. It explores improvisational techniques within electronic music and employs unusual control interfaces for the electronic tools, coupled with visual elements to create a new kind of 'musical theater'. Development of the piece '01-radioreloj' was done using many different tools and its performance employs primarily open source software. The growing complexity and needs of the project during composition, shaped the interface design, and led to the final adoption of Linux as operating system for audio processing.*

1. Introducción

El proyecto 'gemini flux'[4] (en mal latín 'fluir doble') surge como un deseo de explorar técnicas de interpretación en tiempo real usando software avanzado, llevando el resultado hacia un lenguaje musical directo y asequible, simultáneo con un montaje atractivo en vivo. 'Gemini flux' está compuesto por Andrés Cabrera, que cuenta con amplia experiencia en el campo de la post-producción audio-visual y la docencia, además de participar en el desarrollo de plug-ins para *csound* y Gonzalo Sagarmínaga quien es un reconocido intérprete y compositor de música para cine y televisión.

La influencia creativa que ha dictado el rumbo de la concepción musical es la obra 'Black on White' de Heiner Goebbels [5]. Esta obra es una emocionante mezcla de teatro e improvisación musical. Al inicio de la concepción de 'gemini flux', fue claro que se quería buscar un efecto similar al de esta obra, pero a través de herramientas muy diferentes. Mientras que 'Black on White' utiliza virtuosos músicos académicos que son al mismo tiempo músicos y actores, 'gemini flux' mezcla la experiencia de la música electrónica y el rock con herramientas tradicionalmente asociadas a la composición electroacústica, en un montaje en el que los músicos y aún el computador son actores. El resultado de la primera exploración de 'gemini flux' es '01-radio reloj'. Es un montaje marcadamente teatral, que requiere que la tecnología que soporta en gran medida el espectáculo, funcione sin que se evidencie un esfuerzo por parte de los interpretes por hacerla funcionar. Para lograr esto la mayoría del control de la ejecución

(incluyendo cambio de secciones) se hace vía MIDI -sin manipulación directa del computador-, así la tecnología adquiere un carácter propio e independiente de los músicos y se evitan pausas largas entre las secciones dando gran fluidez y una sensación de unidad. Aunque ésta fue desde el principio la idea original, los medios por los cuales se ha llevado a cabo han ido transformándose según las necesidades y a medida que se han tenido nuevos recursos disponibles (actualizaciones de software o hardware).

A pesar de ser una obra principalmente musical, por tener músicos interpretando instrumentos en vivo, '01-radioreloj' tiene una indispensable presencia de video y de teatralidad. Una proyección de video presenta diferentes propuestas visuales de acuerdo con la actividad musical, que tienen profunda relación con la propuesta musical, y en muchos casos aclaran o ayudan a definir esta propuesta. La práctica de que imágenes de video acompañen la música electrónica está usualmente asociada al trabajo de los VJ, y aunque algo de esta estética popular urbana esté presente en el video de 'gemini flux', la forma en que el video se entrelaza tanto temática y conceptualmente como rítmicamente con la música le da un carácter especial. La proyección de video, por incluir procesamiento de imágenes capturadas en vivo, subraya la relación entre músicos, actuación y tecnología. Se habla de teatralidad porque los músicos entran al escenario y se comportan en él como actores, es decir, sus acciones además de producir música, los definen como entidades en el escenario. Aunque estos elementos visuales son inseparables de la obra, este documento se centrará en los aspectos técnicos y estéticos que atañen a la creación sonora y no pretende por esto dar una visión completa de la obra.

1.1. Herramientas para la ejecución

Actualmente, aparte del sonido de los músicos en vivo, la mayoría de '01-radio reloj' proviene del proceso de dos computadores (uno corriendo Linux -Fedora Core 2 Planet CCRMA- [6] que se encarga del audio y otro corriendo Windows XP, encargado de la imagen). La razón de mezcla de sistemas operativos surge de necesidades específicas que sólo se satisfacen en uno de los dos sistemas, y que se discutirán más adelante. El procesamiento y generación de audio está a cargo de *csound* [1] (y en pequeña medida de *pure data*[9]). El video es generado por *pure data* usando su módulo de video *Gem* [14]. Los computadores están interconectados bidireccionalmente usando OSC [12] a través de UDP en cable ethernet.

Además del computador como generador y procesador de sonidos, están presentes diversos instrumentos como guitarras eléctricas, harmónica, bajo eléctrico, un teclado Korg Triton, rototoms, platillos y voces, todos conectados a la consola, y siendo amplificados con su sonido original y en algunos casos específicos con reverberación o flanger de una unidad de efectos externa.

Existe otra categoría de instrumentos, que fueron diseñados específicamente para '01-radio reloj', que involucran una mezcla de interacción acústica con procesamiento y síntesis electrónica. Fueron diseñados para ser efectivos tanto como efecto teatral e histriónico como para lograr timbres y posibilidades interpretativas inusuales. El primero es un sillín de batería que se menciona más abajo en la sección 2.5, el segundo una grabadora de periodista que se enciende y su reproducción es procesada en la sección 2.9, y un tercero que se ha llamado 'hidrófono' se menciona en la sección 2.11.

El control de la ejecución se hace desde tres controladores: Un teclado Radium49 (de 4 octavas y 16 controladores MIDI), un controlador MIDI infrarrojo Eyris (que puede enviar tanto notas como información de control) y un guante P5 (que por medio de USB puede enviar información de control).

El sonido de toda la obra es cuadrafónico, tanto en los procesos en tiempo real como en las secciones que tienen elementos pre-grabados. Los cuatro canales que salen del computador y los sonidos provenientes de los instrumentos entran a un mezclador principal donde son espacializados adecuadamente. En algunas secciones las voces entran directo a la consola y son amplificadas (con algo de reverberación de un procesador externo), mientras que en otras, las voces pasan por procesos electrónicos en el computador.



Figura 1. Guante P5 y controlador Eyris

2. Estructura de '01-radio reloj'

La obra '01-radioreloj' intenta imitar la experiencia de escuchar música en radio presentando una serie de canciones continuas sin relación (ni estilística ni sonora) entre ellas. Cada canción tiene un estilo muy individual, y sus propios motivos histriónicos y experimentos interpretativos, lo que resulta en una serie de unidades totalmente independientes. Se hace un gran barrido por diferentes estilos de música electrónica, y aunque se presentan principalmente estilos populares de los últimos 20 años, también se escuchan sonidos reminiscentes de música electrónica académica -en la introducción y el sueño-. La presentación completa tiene una duración aproximada de una hora.

2.1. Introducción

La introducción, que inicia con el computador solo en escena (programado para arrancar a cierta hora específica), presenta un inicio que busca confundir al espectador sobre lo que va a suceder posteriormente. Presenta una serie de 'campanazos' realizados por medio de síntesis granular a partir de una muestra de audio de la segunda sección. Se inicia con unos sonidos profundos extraños -generados a partir de la granulación de la muestra transponiendo los granos y recorriéndola lentamente- que van transformándose hasta convertirse en uno de los motivos principales de la siguiente sección [10]. Esta transformación se logra acercando los parámetros de la síntesis cada vez más al punto en que el sonido se convierte en la muestra original -es decir sin transposición y usando la velocidad original-.



Figura 2: Leit-motiv

El motivo de la muestra es un leit-motiv en la obra y se compone de tres pares de notas sincopadas en un instrumento electrónico reminiscente a un xilófono. Tanto la música como el video de esta sección son simples y mecánicos, para introducir y dar carácter al 'personaje' del computador, presentándolo como una máquina fría que sin intervención humana, es capaz sólo de crear sonidos estériles. Este inicio contrasta fuertemente con el resto de la obra, que es muy viva e histriónica.

2.2. Primate melancólico

Esta sección surge a partir del motivo generado en la introducción. Es una pieza de elementos minimalistas y totalmente diatónica, que a pesar de ser más emotiva que la anterior conserva su carácter frío. En esta sección el video empieza a mostrar colores, que aún parecen primitivos, acompañando el surgimiento de armonías consonantes en la música. Se escuchan tres líneas instrumentales: el motivo generado en la introducción, que al repetirse constantemente genera una especie de pasacaglia, una línea alta de sonido similar a una flauta y un bajo que inician con motivos muy sencillos que van ganando movilidad melódica y armónica a lo largo de la sección. Los sonidos provienen del sintetizador Absynth [8] de Native Instruments, que por su versátil control de envolventes de parámetros de la síntesis permite crear sonidos muy orgánicos.

2.3. Ionosphere

Cuando empieza a morir el sonido de la sección anterior, se dispara el sonido de un radioreloj que ha sido previamente sincronizado para que inicie en este momento. Una emisora que ha sido seleccionada durante el ensayo empieza a sonar y luego empieza a ser amplificada. En este momento entra 'el técnico' (Andrés Cabrera) en escena. El personaje de 'el técnico' está a cargo de controlar los procesos electrónicos y de manejar la acción relacionada con la tecnología, y será siempre el puente entre el computador y el segundo personaje: 'el músico' (Gonzalo Sagarmínaga). Es un personaje frío que no interactúa con el público, y su relación en el escenario es siempre con el hardware.

En esta sección empieza a hacer un muestreo (usando líneas de retardo) de lo que esté sonando en radio en el momento, que luego se combina con estática y otros ruidos generados por el radioreloj, para comenzar a crear una textura rítmica, que se mezcla luego con una música previamente preparada con un ritmo similar a una champeta o un reggetón. El muestreo del radio se mezcla y se manipula por medio de filtros y otros efectos para dar variedad, y se vuelve prominente en dos secciones en las que se detiene la champeta, e inicia un ruido de ambulancias y luego de guerra. Aquí se hace fuerte uso de la espacialización cuadrafónica.

2.4. Blues

Mientras termina 'Ionosphere', entra al escenario 'el músico', el segundo personaje, que representa la parte emocional en la obra, porque su interacción con lo tecnológico es mínima (y cuando se da más adelante en 'electric cowboy' es cómica). Empieza a tocar una improvisación de estilo blues con guitarra y armónica, y luego comienza a ser muestreado igual que el radio para generar texturas rítmicas densas. 'El músico' también es muestreado en video, y se ven varias imágenes de él en bucles de diferente duración en la pantalla. Esta sección es improvisada sobre una estructura predefinida muy libre, y se genera un estilo algo contradictorio por combinarse la improvisación estilo blues con una rítmica y bucles característicos de música techno.

La voz en esta sección improvisa además de la melodía, la letra, en un idioma que suena como inglés pero que no tiene significado, lo que algunas veces se conoce como 'singlés'. Este 'singlés' -improvisación en melodía y texto- estará presente en todas las demás secciones con excepción de 'pussycat'.

2.5. Parsons

Cuando termina el blues, con una serie de ciclos arítmicos, se inicia la siguiente sección, que tiene un carácter totalmente electrónico, con una base rítmica que va aumentando en textura

mientras 'el músico' toca con baquetas de batería el sillín de una batería, que alimenta unos resonadores virtuales configurados desde el controlador MIDI, dando unos sonidos inusuales que son percutidos pero afinados. La percusión electrónica se hace más densa hasta que hay una explosión en la que entra un bajo sintetizado y una línea de sintetizador que parece generada por un arpegiador. En esta sección 'el músico' toca rototoms y platillos que son procesados con flanger. Luego 'el técnico' improvisa en guitarra eléctrica procesada para generar notas muy largas y de ataque lento, mientras 'el músico' improvisa en un teclado. Acercándose al final, 'el músico' grita y susurra al micrófono pasando por un waveshaper para darle distorsión, y enviándolo a un delay cuya presencia y feedback son controlados por MIDI [3].

2.6. ACDC

Cuando finaliza 'Parsons' se da comienzo a ACDC una canción que es mucho menos mecánica, y aunque de sonido muy electrónico, tiene más movilidad. Se exploran sobretodo efectos en la voz, como distorsión por waveshaping, vocoder y delay. El 'músico', además de improvisar la letra y la melodía (sobre un modelo básico), improvisa en el órgano. En esta sección se explora la relación entre video y música generando texturas visuales contrastantes que son paralelas a los procesos contrastantes que sufre la voz (distorsión-vocoder).

2.7. Nobody wan

Esta especie de interludio es una pieza lenta que busca dar un toque de tranquilidad y emotividad. Tiene dos secciones muy claras que aunque relacionadas tímbricamente, pueden categorizarse como oscura y clara. Representa una salida a la claridad y la tranquilidad a partir de un estado de angustia pasiva. Para mantener el carácter contemplativo, esta sección no tiene ningún elemento histriónico como las anteriores, sino que se centra en la emotividad de la música únicamente. Aquí la improvisación se da en guitarra y voz.

2.8. Electric cowboy

Esta sección presentada por el 'músico', sin acompañamiento de video o electrónica, hace un comentario tácito y humorístico sobre la tecnología, porque el intérprete para generar acordes con el teclado, sobre los que improvisará con la armónica, pega con cinta las teclas para que queden sonando. Esta sección 'primitiva' es un fuerte contraste con la siguiente altamente tecnológica.

2.9. Sueño

En 'Sueño' se presenta una experimentación con controladores inusuales y programación de gestos para la generación de notas. Todo el sonido es generado por el 'técnico' a partir de dos controladores (un Eyriss -controlador MIDI infrarrojo- y un guante P5), que dirigen tanto el sonido como el video. La programación de la respuesta del computador a estos controladores se da de dos maneras: la manera tradicional que es mapeando un parámetro del controlador a un parámetro que controla la síntesis, y la otra es a partir de definición de gestos. Los gestos son simples comparaciones de estado del controlador (el guante P5), que cuando se cumplen generan una acción determinada. Por ejemplo, cuando se flexionan dos dedos (el anular y el meñique), se activa la posibilidad de generar notas de un instrumento virtual que dependen de la altura (eje y) del guante. Otro caso es que cuando se cierran los dedos como un puño, se envía información a *csound* para que dispare unas muestras de la pieza granular introductoria y de 'Primate melancólico'. El gesto de flexionar el pulgar, por ejemplo, apaga todas las notas.

La ejecución tiene un orden definido, ya que el mismo gesto cambia de función (luego de varias veces de hacer el mismo gesto, este produce diferente resultado), y aunque se tiene mucha

libertad en el tiempo, el orden se ajusta a un texto (ver más abajo).

El 'técnico' utiliza en una mano el guante P5 y con la otra utiliza el controlador infrarrojo Eyris, lo que resulta en una extraña 'danza' con los brazos. El Eyris se configura para que envíe mensajes de noteon y noteoff, que aunque pertenecientes a la escala de do mayor, se les convierte usando *pure data* en un arpeggio de un acorde disminuido.

En esta sección, el 'músico' enciende una pequeña grabadora de periodista frente a un micrófono, y un texto grabado es luego procesado por varios delays con tiempos y filtros diferentes, creando unas extrañas texturas rítmicas a partir de la voz. Algunos de los sonidos generados vienen de 'primate melancólico' y la introducción, aunque son difícilmente reconocibles en este contexto. Esto acompaña los otros sonidos generados por los controladores (a través de plugins VST [11]) creando una extraña textura surrealista y onírica.

La sección comienza usando una grabación de los sampleos del radio que se hicieron en 'Tonosphere'. Inician claramente reconocibles hasta que se transforman en sonidos extraños, como el momento en el que se entra en el sueño. Toda la actividad es realizada por el 'técnico' que se convierte en una especie de hipnotizador controlando el sueño del 'músico' (esto es evidente en el video en el que salen figuras de su cabeza) y llevándolo a una especie de pesadilla.

2.10. Pussycat

'Pussycat' es la continuación de la pesadilla. Es una pieza en estilo 'electro', construida exclusivamente a partir de muestras procesadas (transposición con envolventes, filtros, etc.) de sonidos de una fábrica de mantenimiento de aviones, y una simple línea de bajo sintetizado. Por esto tiene un sonido agresivo y mecánico. Está cantada por una mujer y es la única que tiene un texto 'coherente'. Hace extenso uso de paneos cuarafónicos para crear la textura percusiva.

2.11. Party

Esta sección inicia con una introducción de estilo cinematográfico, que desemboca en una canción alegre con influencia del tecno pop de los años 80. Es una canción que busca terminar en clímax por ser la última sección en la que los músicos están en el escenario. En esta sección aparece el 'hidrófono', un instrumento diseñado para esta obra, que consiste de una olla de cocina con un pick-up de teléfono que luego es procesada por un agresivo waveshaper diseñado en *csound*, controlando volumen y profundidad del waveshaper desde MIDI. El efecto es a la vez emocionante y cómico, como un gran solo de guitarra eléctrica 'a la Hendrix' con una olla. Estos solos de 'hidrófono' están intercalados con las secciones 'pop', produciendo una estructura inusual. Esta canción está conectada con la siguiente por transición al estilo DJ (por crossfade).

2.12. Primate alegre

Esta sección final es una repetición idéntica de la sección 2.2 'Primate melancólico', pero esta vez, se ha agregado un beat electrónico y otras texturas rítmicas (a partir de elementos percusivos procesados con modulación y delay), que junto con el aumento del tempo, le dan gran claridad rítmica a la pieza (particularmente al motivo que se ha repetido a lo largo de la obra), y mucha alegría. Este gran contraste de atmósfera con la primera aparición de 'Primate', representa el resultado de la inspiración agregada a la tecnología. Es despojar la tecnología de su aspecto frío e inhumano para convertirla en algo vibrante y emocional. El computador cierra la obra solo en el escenario tal como empezó.

3. El proceso de composición

3.1. Inicio

El primer paso hacia '01-radio reloj' fue el diseño de varios algoritmos para *csound* a finales del 2003. Estos algoritmos permanecen en las secciones 'Ionosphere' y 'Blues' y generan las líneas de delay independientes. También en esta primera fase se hizo la composición de la introducción usando síntesis granular -también en *csound*- y 'Primate melancólico' que se compuso en Sonar 3¹ usando principalmente el sintetizador de software Absynth de Native Instruments. También se hicieron los primeros bocetos para el video de estas dos secciones usando las herramientas gráficas de *csoundAV*.

Al unirse Gonzalo al proyecto se inicia la composición del grueso de la obra usando principalmente Sonar. La composición se hizo en un secuenciador comercial tradicional, pero siempre se tenía presente el rol de la improvisación y la interpretación en vivo, dejando siempre espacio para éstos. A lo largo de este proceso se planeaba generar el audio y el video usando un solo computador corriendo *csoundAV* en Windows XP y haciendo la edición del código usando winXsoundPro. La razón por la que se eligió el software de código abierto *csound* fue por su gran versatilidad tanto para el procesamiento y generación de sonido como por la posibilidad de tener gran interactividad. Otra razón por la que no se eligió algún software comercial como Live o Reason fue que se quería que las secciones de la obra fluyeran ininterrumpidamente y sin manipulación del computador, algo imposible con estos programas ya que se tendría que cargar un nuevo archivo para cada sección. Sin embargo, pronto surgieron las siguientes necesidades adicionales:

- Se vio la necesidad de usar en tiempo real plugins VST para tener una paleta versátil de sonidos, y *csoundAV* no tenía (ni tiene aún) la capacidad de correrlos. Como primera solución se escribió un módulo (en C++) que permitía usar plugins VST en *csound5*, pero *csound5* aún no era apto para uso en tiempo real en Windows en esa época, así que se debió recurrir a *pure data* para alojar estos plugins. Esto implicó que se debía tener un computador adicional ya que no es posible en Windows XP que dos programas independientes accedan a una tarjeta de sonido cuando se quieren usar drivers de baja latencia como ASIO. Desde este momento se comenzó a separar el proceso de audio entre *csound* y *pure data* en dos computadores, sincronizándolos a través de MIDI. La sincronización se hizo a través de mensajes note-on en el canal 15, con un valor de nota para encender y otro diferente para marcar el final de sección. La información de controladores MIDI pasaba directamente del computador con *csound* al de *pd* para controlar parámetros de los plugins VST, que en un principio fueron Cyanide2 (distorsión por waveshaping, gratuita) y Orange Vocoder (de Prosoniq).
- Se comenzó a presentar dificultad para hacer ensayos que no iniciaran desde el principio, por la estructura de la sección de score de *csound*. Se intentó armar un sistema usando macros para simplificar esto, pero seguía siendo engorroso. Adicionalmente, el código había alcanzado un tamaño considerable, y no era fácil hacer ajustes con un editor tradicional. La solución llegó con blue [13] -un front-end para *csound* escrito en java- que permite iniciar la ejecución de *csound* desde cualquier punto en el tiempo.² Blue sigue siendo el editor actual de archivos de *csound* en '01-radio reloj'.

1 Sonar es un secuenciador MIDI y grabador multi-track de audio digital.

2 Esta limitación no se ha superado del todo, y siempre toca tocar secciones desde el principio.

- Las herramientas gráficas de *CsoundAV* empezaron a ser insuficientes, y se transportó con considerable esfuerzo el trabajo ya hecho a *pure data+gem* que ofrece una gama más amplia de herramientas de video y sobretodo la posibilidad de procesar el video capturado por una cámara en tiempo real. Esto implicó nuevos retos de sincronización, ya que se vió que no era confiable correr el audio y el video en una sola instancia de *pd*, sino que debían correr en instancias separadas. Esto supuso la necesidad de instalar un cable virtual (MidiyokeNT), y un patchbay MIDI virtual (Midiox) para intercomunicar las dos aplicaciones que estuvieran en un mismo computador.
- Empezó a notarse un comportamiento errático e impredecible de *CsoundAV* al usar muchos *if...then* dentro de un instrumento, que se requerían para seleccionar la pista que debía sonar según la sección, así que se asignó el trabajo de reproducción de pistas a *pd*.

En esta fase se empezó a asentar la primera versión funcional de '01-radio reloj', en la que se usaron dos computadores de la siguiente forma:

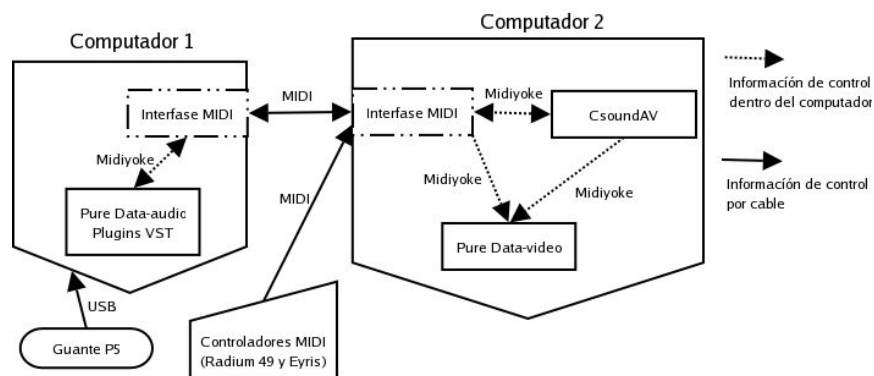


Figura 3. Esquema de la primera versión funcional

Se diseñó un simple protocolo de sincronización generado por *csound*, que no incluía ninguna información de posición sino solo gatillos de inicio, que ha demostrado ser suficiente, ya que no se requiere en ningún momento una sincronización precisa entre las aplicaciones, sino sólo que cada una ejecute la tarea adecuada según el momento. Este protocolo de mensajes de note-on, permanece en la versión actual.

El protocolo cuenta principalmente con dos partes. Una que usa el canal 15 indica a los programas qué sección de la obra está sonando. La sección determina qué algoritmos deben activarse o apagarse, tanto en audio como en video. Los canales 1-8 determinan controles específicos de cada sección, siendo la primera 'Ionosphere' y la octava 'party' (la sección 'Electric cowboy' no tiene ningún control MIDI). Por esto en cada cambio de sección se envía un mensaje desde *csound* que indica que una sección termina -e inicia la siguiente-, y que se debe escuchar en el canal MIDI siguiente, teniendo que configurar al controlador MIDI (Radium49) para que transmita por el canal siguiente. La ventaja de separar las secciones por canal MIDI garantiza que no se pueden activar funciones indeseadas -como activar por error una función de otra sección-.

La organización de la función de las teclas del controlador para ayudar a la memoria se hizo agrupando funcionalidad y separando funciones por octava. El controlador usado es un Radium49, que tiene 49 teclas (4 octavas), 8 sliders y 8 perillas (pots- numeradas 9 a 16). Por ejemplo la sección 'Ionosphere' -que debe transmitir por el canal MIDI 1- se controla así:

Nota	Función
C4 a A4 (teclas blancas)	Estas notas controlan el muestreo del radio, enviando la entrada de audio a una de las 6 líneas de delay, cada una representada por una tecla blanca. Cuando la tecla esta presionada, el audio es enviado a la línea de delay.
F#4, G#4 y A#4	Estas tres teclas negras controlan respectivamente el filtro, flanger y gating que se aplica al sonido. Cuando la tecla está presionada, el efecto está activo.
C5	Inicio del kick drum rítmico. Tiene efecto en el video también.
D5	Inicia la secuencia pregrabada. Debe iniciar sincronizada al kick que se disparó con C5. Produce un cambio en el video.
F5	Re-dispara el kick en el caso de que haya quedado desajustado o impreciso con respecto a los loops (del radio) que ya están sonando.
A5	Graba al disco duro el sonido del loop más largo que luego se usará en la sección 'sueño'.
F#5, G#5, A#5, C#6, D#6, F#6	Controlan diferentes efectos y secciones del video.

Tabla 1. Teclas de control para la sección 'Ionosphere'

Adicionalmente a estas teclas, controladores MIDI tienen efecto en el proceso. En el caso de Ionosphere están distribuidos así:

Slider	Función
1 al 6	Controlan independientemente la retroalimentación de cada una de las líneas de delay.
7 y 8	Controlan la frecuencia y resonancia del filtro resonante pasa-bajos que se activa con F#4.

Tabla 2a. Controles MIDI para la sección 'Ionosphere'

Perilla	Función
9, 10, 11	Nivel de las líneas de delay 1,2 y 3
12	Nivel de las líneas 4, 5 y 6
13	Nivel master de las líneas de delay
16	Nivel de la señal del micrófono para amplificar el sonido del radio

Tabla 2b. Controles MIDI para la sección 'Ionosphere'

La configuración de teclas y controladores para las demás secciones sigue patrones similares organizando funciones en la misma octava en las teclas blancas o las negras. La tecla C7 (96) indica en todos los casos cambio de sección. Esta opción de cambiar de sección por medio de comandos en tiempo real en vez de estar programada es necesaria porque algunas de las secciones no tienen una duración definida, sino que dependen de cómo se desarrolle la improvisación.

La elección de usar el computador 2 para *csound* y el video obedeció a que el trabajo que realizaba el *csound* (líneas de delay, control maestro) era menos exigente para el procesador que el que realizaba pure data (que debía alojar plugins VST, y reproducir archivos de audio largos), y se escogió el computador con el hardware de gráficas más apropiado para este trabajo.

Con esta disposición se terminó la primera versión funcional de '01-radio reloj' que ha sido presentada en el Teatro Colón de Bogotá como concierto de cierre del ciclo anual 'Colón Electrónico' en diciembre de 2004 y luego en la Corporación Colombiana de Teatro en febrero de 2005.

Sin embargo este sistema presenta algunas complicaciones que se han querido corregir:

- El hecho de usar MIDI ha sobrecargado el sistema en secciones donde se envía mucha información, particularmente cuando se envía información acerca del nivel *rms* del sonido. Esta información que genera el *pd* de audio se envía 60 veces por segundo con una precisión de 14-bits usando dos controladores diferentes. Como esta información debe entrar por la interfase MIDI al computador 2, y luego ser repartida a través de dos cables MIDI virtuales -uno conectado a *csound* y el otro a *pd*, el impacto en el desempeño del sistema es de consideración. Experimentos usando OSC para la transmisión de esta información demostraron que es más ligero para el sistema además de ofrecer mayor precisión en el tiempo y en los valores. Sin embargo *csound* hasta hace muy poco tiempo (y solo en versiones beta) soporta OSC.
- Tener el audio separado en dos computadores implicaba tener que usar una consola de más de 16 canales, que tiende a ser más costoso e impráctico. Lo ideal sería que un solo computador produjese el audio, ya que la limitación no es la complejidad del proceso, sino el hecho de que en Windows XP dos programas no pueden acceder simultáneamente al hardware con la baja latencia requerida para trabajo en tiempo real.
- Ocasionalmente ocurría que el sistema se caía cuando recibía una nota MIDI, por causas que parecen apuntar a algún problema en el subsistema o librerías MIDI de Windows cuando se corren varias aplicaciones MIDI y hay una carga alta en el procesador. Se ha visto que este problema está relacionado con el disco duro externo que aloja los archivos de audio.
- Otro problema molesto surgía por un bug en *gem* con el objeto [separator] que aún no ha sido corregido. La ejecución del video debió separarse en dos partes, teniendo que cerrar *pd* entre una y otra. Sin embargo, al cerrar *pd* en estas circunstancias, por alguna razón aún desconocida, la información MIDI dejaba de viajar entre las aplicaciones a través de los cables virtuales. Esto implicó que se debía reiniciar el computador 2 en medio de la presentación, algo que se escondió con la improvisación acústica de armónica y teclado por parte del 'músico' -llamada 'electric cowboy'- mientras el 'técnico' reiniciaba y arrancaba de nuevo el procesamiento de audio y de video, con las correspondientes conexiones MIDI virtuales y la configuración de los plugins VST.

El problema era una paradoja, porque ¿Cómo separar el video del audio si se deben correr dos programas de audio que deben estar en computadores separados sin usar un computador más? En ese momento, me encontraba experimentando con *Linux* hacía algún tiempo, y la posibilidad de usar *csound* y *pd* a través de *Jack* [2] permitiría hacerlo. Sin

embargo el cambio no sería sencillo.

En primer lugar, la versión de *Csound* que se estaba utilizando era *CsoundAV* 0.043, [7] que contenía funciones aún no disponibles en la versión oficial de *Csound* 4 para *Linux*, así que se trabajó para portar los opcodes de *CsoundAV* a *Csound5* (que tiene una arquitectura más moderna, y es más fácil de extender). Afortunadamente, por la misma época (a principios del 2005), la comunidad de desarrolladores llegó una versión estable (aunque aún beta), que corría eficientemente en tiempo real y era capaz de conectarse a *Jack*.

Los plugins VST (Absynth, Orange Vocoder, Cynanide2 y JX220) que se habían usado son programas de Windows, que después de algún trabajo se hicieron correr sobre Linux usando WINE y *jack_fst*. Sólo uno de ellos no funcionó (JX220) pero como no era uno de timbre demasiado importante, se substituyó por *polyiblit* que dió unos timbres similares. Para realizar la mezcla y nivelación de estos plugins se conectan a través de Jack a *csound* y *pd* que se encargan de la nivelación.

Inicialmente se pensó en conectar el guante P5 directamente a *Linux*, pero las versiones más recientes del kernel (se está usando 2.6.10) no soportan aún el guante, así que se debió escribir un pequeño programa en *pd* (en Windows) que convirtiera la información MIDI del guante a OSC. El montaje definitivo está presentado en la figura 4.

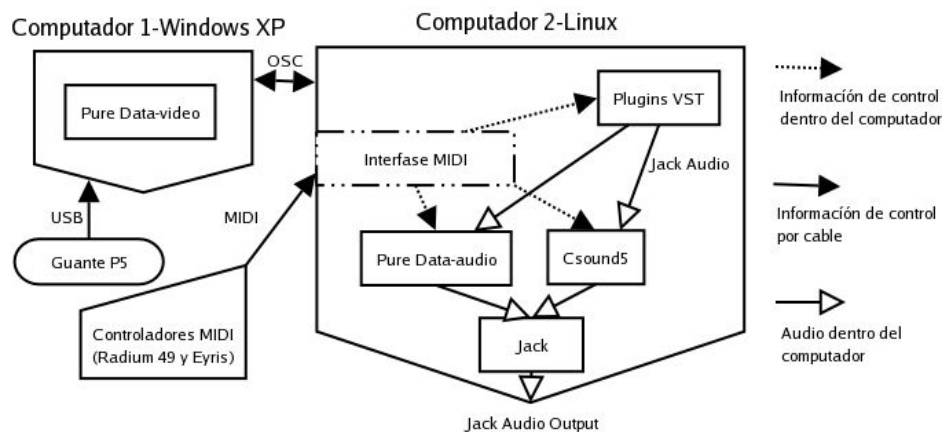


Figura 4. Esquema de la versión final de montaje

4. Conclusiones

El proceso de desarrollo de 'gemini flux' por sus intensos requerimientos técnicos, y por sus ambiciosas intenciones estuvo lleno de obstáculos impredecibles e inesperados. Como la intención y la dirección del proyecto siempre estuvo clara se hicieron todos los esfuerzos para que las limitaciones técnicas nunca frenaran el proceso y la visión creativa. El software de código abierto estuvo presente desde la concepción del proyecto, pero la adopción de *Linux* como sistema operativo de uno de los equipos fue una consecuencia inesperada. Con este documento se quiere demostrar la madurez y versatilidad de estas herramientas gratuitas, en un contexto en el que brillan sus mayores fortalezas.

La experiencia con Linux ha sido inmensamente positiva, y aunque es cierto que es un sistema operativo que puede tomar tiempo para conocer, ha posibilitado la ejecución de 'gemini flux' dando una estabilidad que no se había logrado en Windows. Para futuras producciones de 'gemini flux' se espera usar linux en todo el proceso, incluso para el video. De esta forma se simplificará la interacción entre los diferentes programas, y se facilitará la compatibilidad

futura por no usarse software propietario. Además, el hecho de usar software de código abierto permite extender y modificar la funcionalidad del software según sea necesario.

5. Referencias

- [1] **Boulanger, R.** 2000. ed. *The Csound Book*. MIT Press. Cambridge, Massachusetts.
- [2] **Davis, P, et. al.** JACK- Jack Audio Connection Kit. <http://jackit.sourceforge.net/>
- [3] **Dodge, C. , Jerse, T.** 1997. *Computer Music*. 2nd Edition. Schirmer Thompson Learning.
- [4] **Gemini Flux.** <http://www.geminiflux.com>
- [5] **Goebbels, Heiner.** 1998. *Black on White*. <http://www.heinergoebbels.com>.
- [6] **Lopez-Lezcano, F.** 2002. The Planet CCRMA Software Collection. Proceedings of ICMC 2002. International Computer Music Association. San Francisco.
- [7] **Maldonado, Gabriel.** 2001. *CsoundAV*. <http://www.csounds.com/maldonado/>
- [8] **Native Instruments Inc.** *Absynth*. http://www.native-instruments.com/index.php?id=absynth3_us
- [9] **Puckette, M.** 1996. *Pure Data: another integrated computer music environment*. Proceedings of ICMC 1996, Goeteborg, Sweden. International Computer Music Association.
- [10] **Roads, C.** 2001. *Microsound*. MIT Press. Cambridge, Massachusetts.
- [11] **Steinberg.** *VST Plugin standard*. <http://www.steinberg.de>
- [12] **Wright, M., A. Freed.** *Open Sound Control: A New Protocol for Communicating with Sound Synthesizers*, Proceedings of ICMC 1997. International Computer Music Association. San Francisco.
- [13] **Yi, S.** Blue. <http://www.csounds.com/stevenyi/blue/>
- [14] **Zmoeling, I.** maintainer. *Gem*. <http://gem.iem.at/>

5.1. Hardware de control

Eyris Infrared MIDI controller. <http://www.synesthesiacorp.com/indecks.html>

P5 glove. http://www.zzz.com.ru/index.php?area=pages&action=view_page&page_id=11

Radium49. http://www.midiman.com/products/en_us/Radium49-main.html

Silicon Child

Eloi Fernando Fritsch, Rafael de Oliveira, Rodrigo Avellar Muniagurria

CME – Centro de Música Eletrônica
Instituto de Artes – Universidade Federal do Rio Grande do Sul (UFRGS)
Rua Senhor dos Passos, 248, 6º andar
90020-180 – Porto Alegre – RS – Brasil
`musica.eletronica@ufrgs.br`

Abstract. *This paper describes the procedures which composer E. F. Fritsch employed to create the experimental electro-acoustic piece Silicon Child. This piece uses musical material obtained from the processing of recorded sound and from sound synthesis. It is created through the use of computer programmed automated processes within the Max/MSP environment, which allow the modification of sound. The composer must trigger some processes and interact with the programs, thus giving the work an interactive character.*

Resumo. *O presente artigo aborda os procedimentos utilizados pelo compositor E. F. Fritsch no desenvolvimento da peça eletroacústica experimental Silicon Child que utilizou material musical obtido através de processamento de sons gravados e síntese sonora. A peça é criada pela utilização de processos automáticos programados no computador através do ambiente Max/MSP. Estes processos possibilitam a modificação do som por processamento. O compositor necessita disparar alguns processos e interagir com os programas, atribuindo a obra caráter interativo.*

1. Introdução

O presente artigo apresenta os processos composicionais utilizados para a criação da obra *Silicon Child*. A obra é composta de três seções, cada uma produzida por uma rotina programada em Max/Msp. *Silicon Child* necessita da intervenção do compositor durante a execução para disparar eventos e controlar rotinas. O artigo discorre sobre a pesquisa de materiais musicais, o plano da obra, a difusão sonora e o detalhamento técnico sobre os procedimentos implementados em MaxMsp.

2. A pesquisa de materiais musicais

A pesquisa dos materiais musicais foi dividida em duas etapas:

A síntese sonora realizada no sintetizador Korg MS-2000R, no qual, através das técnicas de síntese subtrativa, foi possível criar um catálogo de sons eletrônicos, que compreende sons ruidosos e complexos através do emprego da filtragem, ressonância, LFO e processamento por “delay”.

A gravação e processamento de gargalhadas e balbuciar de um bebê, na época, com um ano e meio de idade. Todos os sons foram registrados e, posteriormente, processados pelo vocoder do sintetizador Korg MS-2000R e pelo plug-in Sci-fi do Áudio Suíte do Pro tools. As características dos sons processados remetem o ouvinte à “criança silicone”, ou seja, a criança que está cercada pelo aparato tecnológico e que irá influenciar seu comportamento.

3. Plano da obra

Com relação ao plano da obra, as seções da composição foram organizadas em três partes, sendo a *Silicon Child Part I* caracterizada por não possuir uma duração fixa pré-determinada, pois a duração depende do controle da execução do compositor durante a apresentação da peça musical. *Silicon Child Part I* e II possuem um tempo determinado e o compositor apenas dispara outros processos durante a execução da peça. A obra não possui uma duração total definida por causa da *Silicon Child Part I* e das pausas entre as seções.

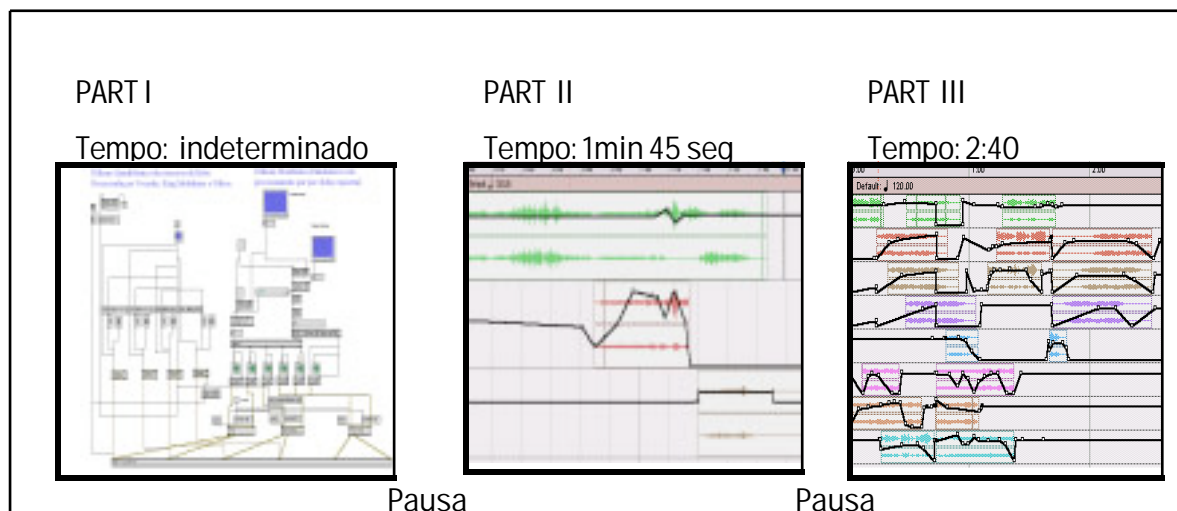


Figura 1. Plano visual das três seções da obra e suas durações.

3. Difusão sonora da peça

A difusão sonora da peça foi projetada para sistema hexafônico de alto-falantes. Trilhas de áudio foram criadas em sistema Pro tools. Os áudios foram disparados em diferentes alto-falantes pelo programa Max/Msp.

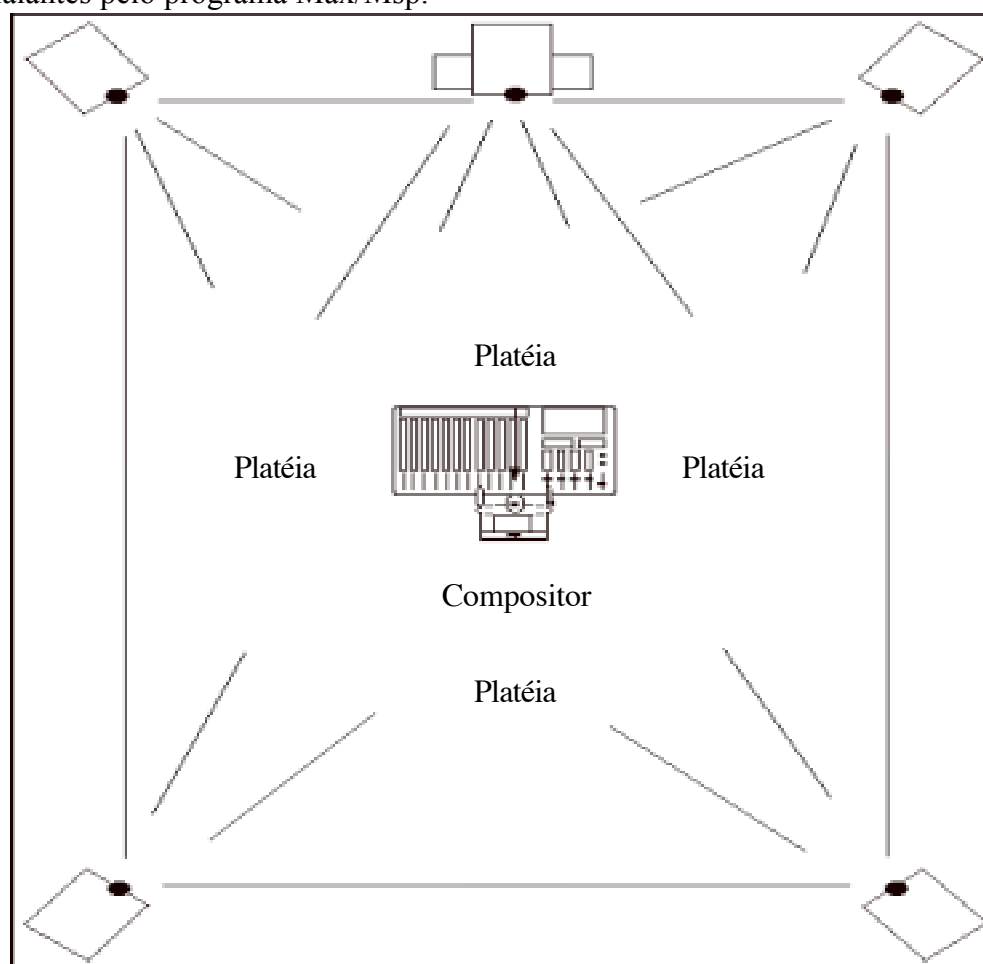


Figura 2. Posicionamento dos elementos envolvidos na difusão da peça eletroacústica

Para a difusão sonora foram utilizados 5 monitores ativos Event, 1 Sub-woffer ativo Tanoy, Software MAX/MSP, Programas desenvolvidos pelo compositor no MAX/MSP, Plug-in Spectral Delay, Computador Pentium 4 Toshiba, Interface M-Audio Firewire 410.

4. A Obra Eletroacústica

4.1 Silicon Child Part I

Na Silicon Child Part I foi utilizado um programa em Max/MSP apresentado na Figura 1. O lado esquerdo do programa permite ativar a execução do arquivo de áudio através da seleção do valor 1 nas caixas de mensagem. A seleção do valor 0 irá desligar o áudio. Cada voz está associada a um alto-falante de acordo com o objeto “dac~”. O lado direito

do programa envia valores de alturas randômicos para um plug-in de delay spectral que cria múltiplas cópias. O efeito de delay gera uma nova percepção para os eventos musicais idealizados pelo autor. Todo o controle da emissão do material randômico fica a cargo do executante que poderá improvisar durante a audição das vozes pré-gravadas.

Difusao Quadrifonica da conversa do Bebe
Processada por Vocoder, Ring Modulator e Filtros

Difusao Hexafonica Randomica com
processamento por delay espectral

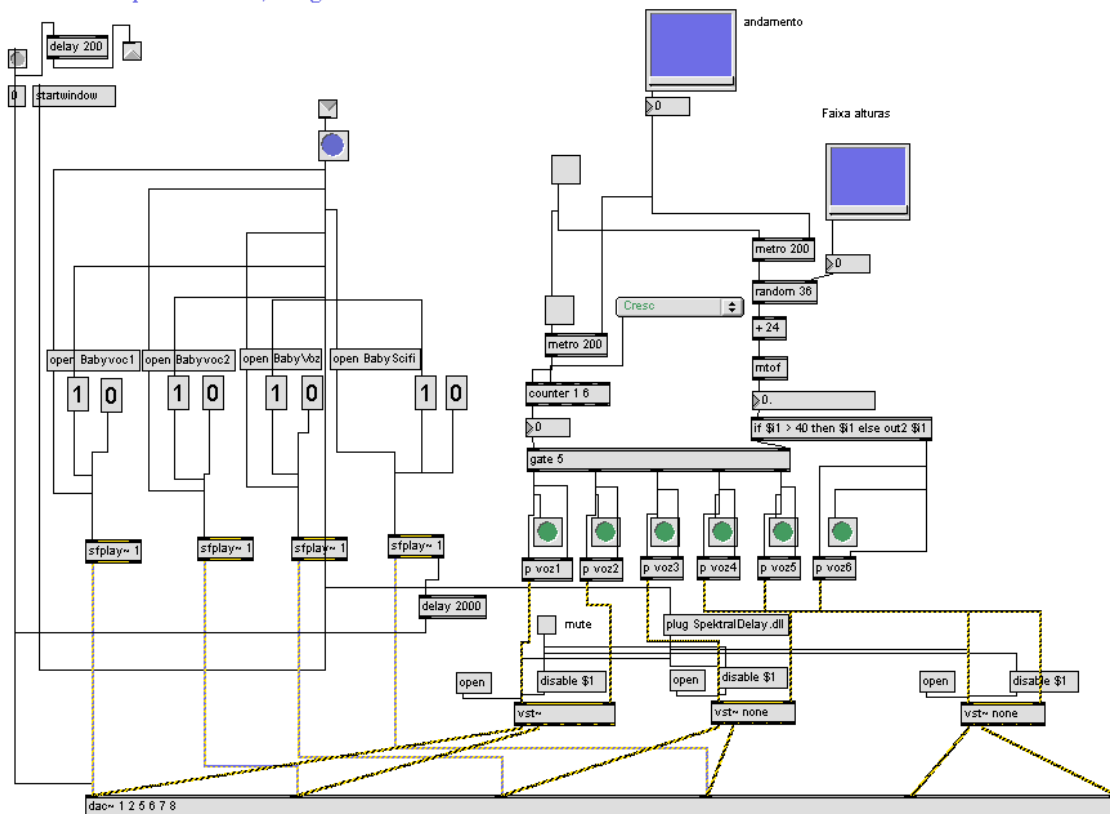


Figure 3. Programa em Max/MSP para a difusão sonora de Silicon Child Part I

O Sub-woffler sempre irá executar os sons graves da composição enquanto cada uma das quatro gravações diferentes da voz do bebê serão projetadas no ambiente. Cada voz irá soar sempre na mesma caixa criando a percepção de quatro vozes distintas bem localizadas no ambiente de projeção sonora. Os sons eletrônicos processados pelo delay são enviados por todas as caixas.

4.2 Silicon Child Part II

Silicon Child Part II é composta pela execução de um trecho de música eletroacústica composto por sons do MS-2000R e o som produzido pelo mouse movimentado pelo compositor no quadro da Figura 4. Além disso, o compositor tem a possibilidade de ativar processos randômicos para movimentar o mouse automaticamente, produzindo alturas aleatórias produzidas por osciladores senoidais. Os quatro quadrados à direita do quadro de movimento do mouse servem para ativar e desativar os processos randômicos. O quadro também possibilita enviar os sons produzidos para quatro alto-falantes. De

Difusao Quadrifonica para composicao Interativa e Estereofonica de Audio Pre-gravado

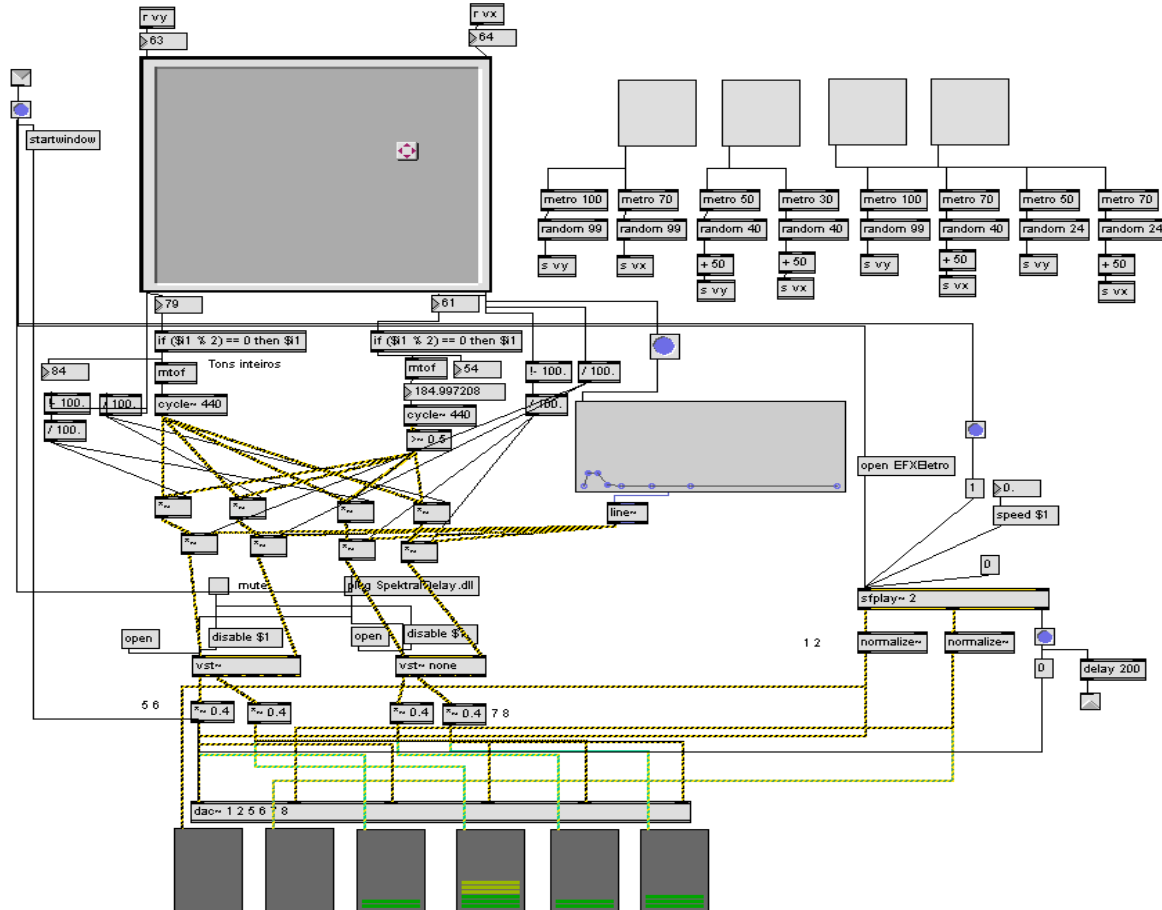


Figure 4. Programa em Max/MSP para a difusão sonora de Silicon Child Part II

Três alto-falantes executam os sons eletrônicos providos do movimento do mouse enquanto dois alto-falantes executam a passagem eletrônica pré-gravada. Os sons eletrônicos providos do Mouse causam uma sensação de movimento na platéia por serem executados em andamento rápido enquanto a passagem eletrônica pré-gravada causa uma sensação de profundidade pela ambiência artificial criada por efeito de reverberação e pouco movimento.

Na Silicon Child Part III é executada uma passagem de música eletroacústica com intervenções das vozes do bebê. Apenas duas vozes são executadas no decorrer da seção. Uma voz granulada e outra robotizada. Enquanto a passagem eletroacústica é reproduzida pelo Max/MSP, o compositor dispara as vozes realizando uma improvisação.

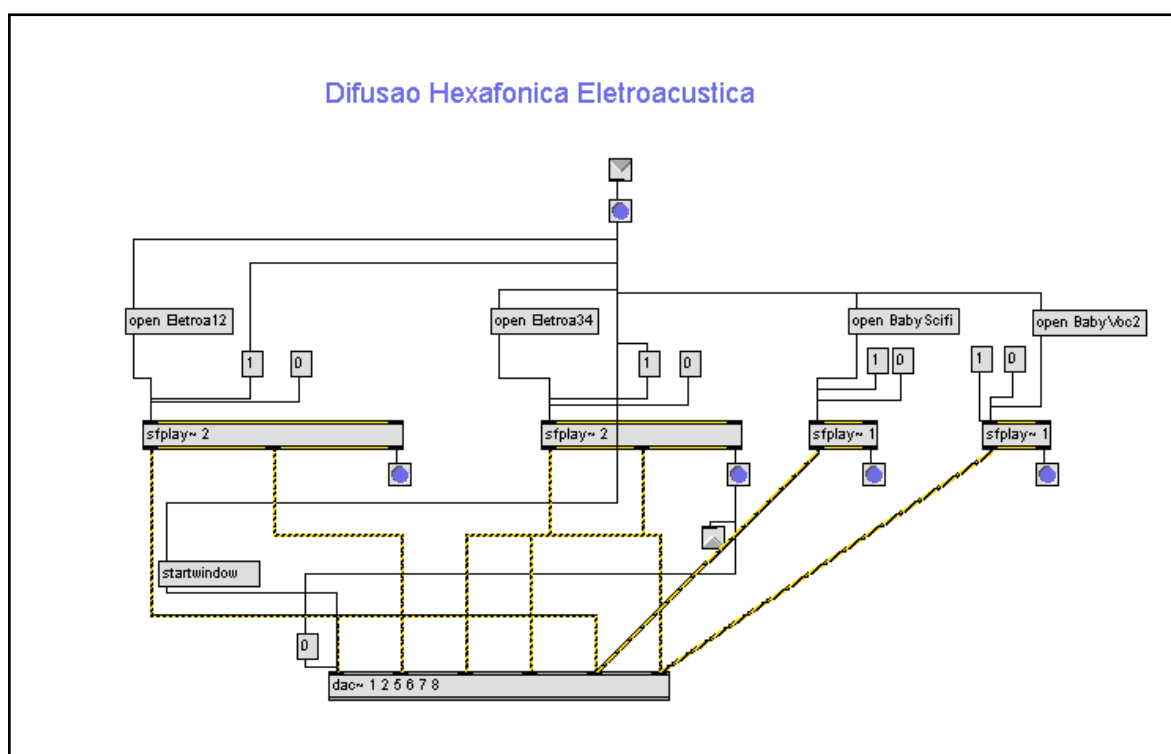


Figure 5. Programa em Max/MSP para a difusão sonora de Silicon Child Part III

Seis alto-falantes executam os sons eletrônicos provindos da passagem musical eletroacústica pré-gravada enquanto dois deles executam, simultaneamente, as vozes do bebê.

5. Conclusão

Neste projeto foram elaborados programas de computador para a execução e espacialização de música eletroacústica que permitiram ao compositor criar e apresentar uma composição auxiliada por computador. Os programas utilizados pelo pesquisador foram eficientes e fundamentais para a realização da obra musical. A relevância do projeto concentra-se principalmente no âmbito musical. Entretanto, a utilização de algoritmos programados no sistema Max/Msp foram indispensáveis na realização sonora das idéias composicionais.

Através dessa iniciativa foi possível convidar outros compositores, pesquisadores e estudantes de música para apresentarem suas composições e experimentações eletroacústicas em um mesmo concerto. A apresentação do concerto só foi possível devido à utilização do computador portátil financiado pela FAPERGS.

Através dessa pesquisa, foi possível estudar, desenvolver e apresentar novas composições incentivando e promovendo a pesquisa em música eletroacústica no Rio Grande do Sul.

A primeira audição pública da obra foi realizada em 30 de Setembro de 2004 para um público de aproximadamente 300 pessoas no Auditório Tasso Correa do Instituto de

Artes da UFRGS durante o primeiro concerto *Projeção Sonora de Música Computacional*. A música eletroacústica apresentada neste concerto contou com uma platéia atenta aos detalhes da composição. Este foi o primeiro concerto do gênero realizado na Universidade Federal do Rio Grande do Sul com peças eletroacústicas de professores e alunos da Instituição. A composição foi projetada através de cinco monitores ativos Event e um subwoofer Tanoy conectados às saídas da interface Firewire M-Audio ligada ao computador.

Antes da apresentação o compositor realizou um relato da pesquisa desenvolvida e dos resultados obtidos. Após explanou sobre o funcionamento do sistema e como este é utilizado para produzir a peça *Silicon Child*.

Durante a execução da peça os programas, controlados em tempo real, foram projetados para que a platéia conseguisse observar o funcionamento deles e a interação do compositor com o sistema criado.

Referências

- Dobrian, Christopher. (1998) *MSP - The Documentation*. Cycling '74, San Francisco, CA.
- Dodge, Charles; Jerse, Thomas A. (1997) *Computer music: synthesis, composition, and performance*. Schirmer Book, New York, NY, USA.
- Fritsch, E. F.; Vicari, R; Cunha, A.C. (2002) *MEPSOM – Método de Ensino de Programação Sônica para Músicos*. (Tese de Doutorado). Porto Alegre: CPGCC da UFRGS.
- Miranda, Eduardo Reck. (1998) *Computer Sound Synthesis for the Electronic Musician*. Focal Press, Oxford.
- Miranda, Eduardo Reck. (2001) *Composing Music with Computers*. Focal Press, Oxford.
- Winkler, Todd. (1998) *Composing Interactive Music – Techniques and Ideas Using MAX*, MIT.
- Winsor, Phil. (1989) *Automated Music Composition*, USA.

Apoio: Fapergs, CNPq, UFRGS.

Interação Tímbrica na Música Eletroacústica Mista

Ignacio de Campos

Departamento de Música, Instituto de Artes - Unicamp

ignaciiodc@uol.com.br

Abstract. *The interaction between electroacoustic and instrumental environments in the mixed electroacoustic music may present problems regarding their differences of compositional approaches and timbric divergences. Through a meticulous technical study of the two environments this project aims to find a poetic solution for these timbric divergences proposing an electroacoustic work in which the sonic material involved is circumscribed by the timbres of the acoustic instruments used in the work itself.*

Resumo. *A interação entre os meios eletroacústico e instrumental na música eletroacústica mista pode ser problemática no que tange à diferença escritural e à divergência tímbrica. No presente projeto propôs-se encontrar uma solução poética à divergência tímbrica dos meios através de um estudo técnico detalhado, que resulta da concepção da obra eletroacústica mista cuja interação dos meios se circunscreve na delimitação do timbre no material eletrônico a partir dos próprios timbres dos instrumentos acústicos envolvidos na obra.*

1. Introdução

O presente artigo visa descrever, em termos gerais, meu projeto de mestrado no qual se trata da questão da divergência tímbrica dos meios instrumental e eletrônico na música eletroacústica mista sob o recorte do tempo diferido, isto é, *non real-time*.

Propôs-se encontrar uma solução poética à divergência tímbrica dos meios através de um estudo técnico detalhado, que resulta da concepção da obra eletroacústica mista cuja interação dos meios se circunscreve na delimitação do timbre no material eletrônico a partir dos próprios timbres dos instrumentos acústicos envolvidos na obra. A proposta seria a de criar para o meio eletroacústico uma paleta tímbrica graduando desde timbres similares ao timbre original do instrumento até timbres perceptivelmente distantes através de variações controladas dos parâmetros sonoros resultantes da própria análise espectral do timbre original. Para isso, o estudo envolve três fases que encontram projeção prática permeadas pelos constructos psicoacústicos: a análise espectral paramétrica de um som instrumental seguida de manipulações e geração modelada dessas variáveis para futura concreção do material eletrônico final através da síntese, que resultará em timbres similares, variados, expandidos do timbre original. Para se encontrar as maneiras pelas quais são obtidas as sensações de similaridade e distanciamento tímbrico que validassem as manipulações de parâmetros propostas, contou-se com estudos de psicoacústica. Assim, as decisões tomadas nas escolhas dos

parâmetros de análise, nas variações propostas desses parâmetros e nos algoritmos de síntese derivam, em parte, de referências às respostas do nosso sistema auditivo. As fases de análise, programação das variações e síntese envolvem o uso de diferentes *softwares* que são abordados no trabalho (AudioSculpt, OpenMusic e Csound). Outros utilitários computacionais que dão suporte paralelamente aos principais processos estudados são o SoundHack e o programa utilitário de análise de *phase vocoder* chamado PVANAL.

2. Seleção de Parâmetros

Os diferentes níveis de representação naturalmente existentes entre um som instrumental ‘pronto’ criado já como um complexo sonoro, via de regra definido através de uma simbologia de alto nível já codificada, e o objeto sonoro eletroacústico podem ser equiparados através da modelização da configuração tímbrica instrumental em parâmetros que podem ser individualmente trabalhados, envolvendo a descrição dos comportamentos freqüenciais, graus de harmonicidade/inarmonicidade, envelope dinâmicos e rugosidade. Foram realizadas pesquisas e estudos de correspondências entre as alterações de cada um dos parâmetros expostos acima e o distanciamento/aproximação sensorial de uma ‘identidade’ tímbrica derivada da configuração instrumental.

Os parâmetros adotados na investigação de distanciamento/aproximação sensorial são:

- relação freqüencial entre os parciais (harmonicidade/inarmonicidade)
- alteração da amplitude média dos parciais (redistribuição da energia do espectro)
- alteração da distribuição das freqüências no espectro
- micro-variações randômicas dos tempos de início de cada parcial
- variações na relação temporal de ADSR dos parciais
- variações na relação de amplitude entre o ataque e a extinção dos parciais
- mudança de fase dos parciais
- *jitter*
- *shimmer*
- quantidade relativa de dados não-determinísticos (transientes e ruídos) acrescentados à parte determinística (ressíntese senoidal de parciais estáveis)
- seleção de dados (filtro psicoacústico)

Desses parâmetros, os sete primeiros são manipulados através de *patches* criados em OpenMusic e os quatro últimos aplicados através de *scripts* de Csound.

3. Preparação do Áudio e Análises de Fourier

A análise escolhida para a leitura dos parâmetros oriundos dos sons gravados foi a FFT (*Fast Fourier Transform*) que, para que gere resultados nos quais haja uma correspondência ‘mais fiel’ ao conteúdo espectral original, deve analisar sons com o mínimo de dados não-determinísticos possível. Dessa forma pode-se evitar o efeito colateral da falsa interpretação de ruídos e transientes como parciais estáveis, esperados nesse tipo de análise. A separação da parte não-determinística do áudio a ser analisado é realizada através do processo de ‘*spectral extractor*’ do *software* SoundHack apenas fornecendo os limites superior e inferior das taxas médias de mudança de freqüência das

componentes analisadas. Obtida a parte do som com taxas médias de variação menores que um determinado *threshold* (em Hertz por *frame* de análise), passa-se ao estágio da análise via FFT no AudioSculpt.

Aqui se inicia uma extensa série de cuidadosas decisões a serem tomadas na escolha dos parâmetros de análise mais adequados para cada tipo de som seguindo intuítos específicos: tamanho da janela de análise, tipo da janela de análise, número de *overlaps*, número de ‘canais’ na análise (também chamado FFT *size*), tamanho mínimo para que as componentes analisadas sejam consideradas parciais, flutuação máxima de frequência a partir da qual passa-se a considerar um novo parcial, distância máxima entre dois parciais abaixo da qual pode-se uní-los e *thresholds* mínimo e máximo de amplitude que estabelecem o âmbito considerado para análise. Pode-se estabelecer ainda relações entre o tamanho real da janela de análise escolhida e um tamanho ajustado para uma potência de dois para que haja ‘*zero-padding*’ em tamanho suficiente para uma análise mais precisa, isto é, uma maior definição espectral.

O processo em AudioSculpt para a obtenção das análises desejadas é o ‘*partial tracking*’ que exportará, em formato texto, dados referentes aos tempos de início e de extinção de cada parcial, suas frequências médias e amplitudes médias em dB.

4. Modelagem e Criação de Dados para Síntese

Através do OpenMusic podemos fazer programações visuais (*patches*) de métodos e processos de manipulação de dados adequando-os para as necessidades musicais apresentadas na elaboração da obra mista. Assim, foram criados *patches* que permitem a exploração sonora a partir da alteração de parâmetros e fatores que irão dirigir as sensações de fusão e segregação sonoras e o distanciamento/aproximação de uma identidade sonora, ou seja, a criação de parâmetros sonoros que criem sons cujo modelo inicial seja o som instrumental. Ao contrário do que se poderia pensar inicialmente, os limites impostos pelo modelo não são estreitos, pois pode-se chegar a configurações totalmente distintas daquelas de origem com a vantagem de se ter uma referência e de se poder controlar o grau de dissimilaridade do modelo, ajustando-o estática ou dinamicamente.

Os *patches* criados foram, segundo as categorias dos parâmetros abordados, foram:

No campo frequencial

- 1) transformação de um espectro harmônico em um espectro inarmônico
- 2) transformação de um espectro inarmônico em um espectro harmônico
- 3) compressão e dilatação progressiva exponencial bipolar frequencial do espectro a partir de um âmbito central dado
- 4) redistribuição progressiva exponencial bipolar de frequências sem alteração do âmbito espectral

No campo temporal

- 5) micro-alterações dos tempos de início de cada parcial
- 6) relação variável de duração entre os segmentos do envelope dinâmico dos parciais (proporção interna do envelope)

No campo das amplitudes

- 7) redistribuição da energia contida no espectro
- 8) criação de uma relação variável de amplitude entre o pico máximo e a amplitude de sustentação do som

No que concerne às fases

- 9) criação de posições de fase individuais para os parciais em cinco configurações
 - a. em fase (a mesma posição de fase para todos os parciais)
 - b. fases opostas entre parciais consecutivos no âmbito frequencial
 - c. fases randômicas com distribuição uniforme
 - d. fases com distribuição gaussiana
 - e. fases com distribuição beta

No que concerne ao espaço

- 10) distribuição de parciais no espaço estereofônico

Os *patches* executarão tarefas de adaptação, alteração e criação de dados segundo os algoritmos elaborados para cada parâmetro e exportarão os dados resultantes formatados para serem lidos como *scripts* de Csound (*scores*). Eles fornecerão listas dos novos parciais com as descrições necessárias para a posterior síntese aditiva executada em Csound - número de parciais, *onsets*, durações, amplitudes, frequências, fases, posições no espaço estereofônico e envelope dinâmico dos parciais.

Segue uma breve descrição de algumas das funções listadas acima.

1) Neste *patch* toma-se como princípio que o som de origem, cujos parâmetros se queira modificar, ou para o qual se queira criar, seja harmônico. A lógica consiste em relacionar cada frequência lida da análise a um harmônico e a partir da frequência virtual desse harmônico, criar desvios de até 22% em frequência acima ou abaixo da frequência original, o que corresponde a aproximadamente três semitons mais um quarto de tom (≈ 344 cents) para cada lado. Com ‘frequência virtual’ refiro-me à frequência obtida a partir da multiplicação da frequência fundamental (dado extraído diretamente do AudioSculpt em um processo paralelo e inserido no *patch*) pelo número do harmônico correspondente (calculado dentro do *patch*). O porquê ela é virtual e não ‘real’ deve-se ao fato de que os parciais harmônicos de um som instrumental nunca correspondem exatamente à frequência calculada a partir da série harmônica.

2) A lista de frequências extraídas da análise é comparada à lista de frequências harmônicas virtuais. O arredondamento é essencial, pois a simples razão entre as frequências fornecerá a razão real entre elas, seja esta harmônica ou inarmônica. A razão arredondada, fornecendo um número inteiro, assegura a harmonicidade da relação. Depois de comparadas as listas, se for verificado que as frequências das análises são pertencentes ao âmbito de um tom abaixo a um tom acima da frequência virtual, a frequência virtual é adotada no lugar da primeira, se não, não haverá alteração, garantindo que parte dos parciais continuem inarmônicos e mantendo, dessa forma, uma riqueza do espectro frequencial maior que se todo o espectro fosse totalmente harmônico.

3) A idéia contida neste *patch* é a compressão ou dilatação do espectro de frequências de maneira progressiva e exponencial com fatores de progressão independentes para o grave e para o agudo excluindo deste processo o âmbito especificado. As alterações acontecerão para além desse âmbito central segundo dois fatores fornecidos: o fator para modificações abaixo do âmbito (nomeado como *fator_inf*) e o fator para modificações acima do âmbito (nomeado como *fator_sup*). Os dois têm funcionamento similar e independente, podendo mesmo não haver alteração em um dos ‘lados’.

4) Neste caso, diferentemente do anterior, o âmbito de frequências do espectro original é preservado. Em alguns casos pode-se querer, por vários motivos, preservar os limites de frequência. Assim, há uma redistribuição das frequências do espectro, mas com um reescalamento das novas frequências a fim de manter o âmbito fixo.

7) O mascaramento espectral (discutido no corpo da dissertação) impede a percepção, em condições muito específicas, de uma componente do som pela proximidade frequencial de outras componentes espectrais com maior intensidade. É sobre este ponto especificamente que se situa o campo de atuação do *patch* de alteração de amplitude. A idéia desenvolvida aqui é trazer à tona uma série de componentes do espectro que não poderiam ser percebidas antes devido ao mascaramento espectral. A esse processo chamei desvelamento tímbrico, pois revela uma parte do timbre antes encoberta (mascarada). Mas, antes de apenas elevar as amplitudes das componentes imperceptíveis (ou menos perceptíveis que outras), aumentando a energia sonora global, foi realizado um deslocamento ou redistribuição de determinadas ‘quantidades de amplitude’ com valores fornecidos pelo usuário. Dessa maneira, pode-se, em parte, manter o contorno de amplitude e a soma de energia do espectro do som original. O que poderá mudar, além da alteração de regiões frequenciais percebidas, é a sensação de intensidade resultante devido a um grande número de fatores psicoacústicos em jogo.

9) Por questões de dificuldade de gerenciamento de grandes quantidades de dados, não foram lidas as fases originais das componentes espectrais, as leituras não seriam possíveis através das análises de parciais via *partial tracking*. Foi proposta então a criação de cinco diferentes configurações de fase para os parciais, cada uma delas com arranjos de valores bem diversificados. As aplicações das distribuições de fases randômica uniforme, gaussiana e beta se deram no eixo das frequências das componentes e não no eixo temporal.

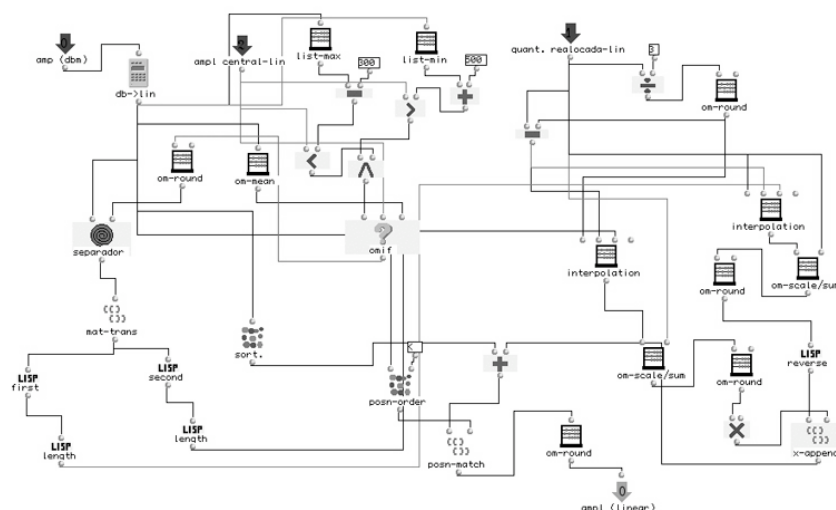


Figura 1. Patch de fornecimento de dados para a realocação da energia do espectro.

5. A Síntese

O modelo de síntese adotado foi o modelo espectral por ser este, entre todos, o mais detalhado e maleável e o que se ajusta perfeitamente à proposta de trabalho. O *software* escolhido foi o Csound na versão MacCsound de Matt Ingalls que possibilita, além da síntese, processamentos e programações de novas funções.

As implementações são de três categorias - implementação de modelos psicoacústicos, manipulação de dados para a síntese e técnicas de preparação e endereçamento de dados – que agrupam as seguintes propostas:

1) Modelos psicoacústicos

- limiar de audibilidade (*Audible Threshold of Hearing*)
- *jitter*
- *shimmer*
- distribuição espacial

2) Manipulação de dados para a síntese

- senóides + ruído
- hibridização

3) Técnicas de preparação e endereçamento de dados

- criação do arquivo de análise de *phase vocoder*
- integração dos processos para a síntese final

A aplicação desses tópicos em Csound se valeu, em parte, pelo uso de funções préprogramadas - os *opcodes*, e em parte pelo uso de expressões matemáticas, como foi o caso da implementação filtro criado a partir do limiar de audibilidade:

```

iampdbm = dbamp(p4)           ;conversão para dBm
iampspl = (iampdb*120)/90.308 ;conversão para dB SPL
iamplin = p4                  ;amplitude linear
ifrq = p5                     ;constante de freq.

ithres = (3.64*(ifrq/1000)^(-0.8)) - (6.5*2.71828^(-0.6*((ifrq/1000)-3.3)^2)) + (10^(-3)*(ifrq/1000)^4)

iampf = (iampspl < ithres + 4.99 ? 0 : iamplin) ;teste/decisão

```

Figura 2. Expressão para aplicação de filtro em Csound baseado no limiar de audibilidade (ithres).

Outros parâmetros empregados provenientes de estudos psicoacústicos foram o *jitter* e o *shimmer* (flutuações nas frequências e amplitudes das componentes de um som, respectivamente), ambos responsáveis pela sensação de riqueza e naturalidade dos sons instrumentais. Dessa forma, dá-se prosseguimento às etapas de modelização do som de origem. O Csound já possui *opcodes* apropriados para implementação de *jitter* e *shimmer*, bastando escolher o tipo de algoritmo dentre os disponíveis e os parâmetros adequados a cada situação, isto é, a cada tipo particular de som.

Por fim, a parte mais trabalhosa foi devido ao acréscimo da parte ruidosa do som, não-determinística, eliminada antes da análise de FFT. Assim, em vez de adotarmos a síntese aditiva como única fonte de geração de som, optou-se por ‘hibridizar’ o som final acrescentando ruídos extraídos do som original à síntese. O processo inclui a importação do áudio original para o Csound, a análise de *phase vocoder* com o utilitário PVANAL, a leitura da frequência e amplitude de cada um dos canais resultantes da análise e suas filtragens através de uma função de filtragem recursiva estabelecida no processo de programação de funções (chamada também *opcode*). Obtido o ruído resultante da subtração de todos os componentes analisados pelo *phase vocoder* do áudio original, mixamos à síntese obtida como encerramento de todo o processo. A mixagem evidentemente pode ser ajustável e pode conferir diferentes caracteres sonoros, de modo a aproximar ou distanciar os sons eletrônicos obtidos daqueles sons instrumentais de origem.

```

opcode FFTFilter, a, akkpp

asig, ktime, kq, idep, icnt  xin

if (icnt >= idep) goto next

asig      FFTFilter asig, ktime, kq, idep, icnt+1
kfrq, kamp pvread ktime, "d.pvc", icnt
kamp = (kamp > 1 ? 1/kamp : 1)
asig      pareq asig, kfrq, kamp, kq

next:
xout asig
endop

```

Figura 3. Programação da filtragem recursiva em Csound para extração da parte não-determinística do som.

6. Conclusão

Há as conclusões técnicas, relativas à eficiência e pertinência dos procedimentos envolvidos de acordo com o que foi proposto e há as conclusões, situadas mais no campo subjetivo, relativas à qualificação das sonoridades resultantes voltadas para o uso

composicional na música eletroacústica mista. O grande âmbito de associações percebidas, entre a verossimilhança com o timbre original e a ausência completa de qualquer referência perceptiva, nos conduz a possibilidades inumeráveis, assim como suas classificações. Reduzir esse âmbito incomensurável a meia dúzia de passos de interação entre sons instrumentais, ditos ‘acústicos’, e sons eletrônicos seria desconsiderar totalmente as nuances e características de nossa percepção e da estrutura dos sons. Seriam necessários, na verdade, estudos exaustivos sobre as considerações a serem feitas na escolha dos processos, mesmo assim sem a intenção de esgotar as possibilidades de combinação de alteração conjunta de diversos parâmetros simultaneamente. O que se pôde perceber no decurso deste estudo é que variações contínuas e regulares de um ou outro parâmetro sonoro para a síntese não mostrou uniformidade perceptiva: pequenas alterações iniciais dos parâmetros de análise originais, principalmente no que tange ao campo das alturas e sua estrutura, mostraram distanciamentos perceptivos abruptos e vice-versa. Alterações de outros parâmetros, como fase, criaram situações sonoras divergentes quando a estrutura dos valores de fase também divergiam, mas pouquíssimas variações percebidas quando os valores mudavam mantendo-se as suas estruturas. Assim é porque cada parâmetro diz respeito a uma estrutura perceptiva diferente e suas mudanças só poderão ser entendidas se estudadas dentro de sistemas restritos e controlados. Como recurso de composição, vale mais a livre exploração sonora/musical das combinações fornecida pelos abertura e abrangência dos procedimentos criados aqui com esse intuito.

Referências

BATTIER, Marc. De la Phonographie à la Lutherie Numérique. L’Épiphanie du Son Artificiel. In: Journée d’Informatique Musicale – 9^a édition, 2002, Marseille.

Proceedings. Disponível em:

<http://perso.wanadoo.fr/gmem/evenements/jim2002/articles/L01_Battier.pdf>. Acesso em: 10 fev. 2004.

BOULANGER, Richard (Ed.). *The CSound Book – Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*. Cambridge: The MIT Press, 2000, 740 p.

BREGMAN, Albert S. *Auditory Scene Analysis – The Perceptual Organization of Sound*. Cambridge, Massachusetts: MIT Press, 1990, 773 p.

CAMPOS, Ignacio de. *Interação Tímbrica na Música Eletroacústica Mista*, 2005. Dissertação, 205 f. (mestrado) - Pós-graduação do Instituto de Artes, Unicamp.

JENSEN, Kristoffer; MARENTAKIS, Georgious. *Hybrid Perception*. 1st Seminar on Auditory Models. Lyngby, Denmark, 2001.

MALT, Mikhail. *Les Mathématiques et la Composition Assistée par Ordinateur - Concepts, Outils et Modèles*, 2000. Tese, 874 f. (doutorado) - Ecole des Hautes Etudes en Sciences Sociales. Recebido do próprio autor em 9 ago. 2003.

***Cidade* de Gilberto Mendes: o toca-discos e o gravador como instrumentos musicais?**

Denise Garcia, Clayton Rosa Mamedes

Departamento de Música – Instituto de Artes –
Universidade Estadual de Campinas (Unicamp) – Campinas – SP – Brazil

d_garcia@iar.unicamp.br, mamedes@iar.unicamp.br

Abstract. *The present paper is the first result of the project "Faces of Brazilian electro-acoustic music: the Música Nova Group and the pioneers in the use of technological resources" and "Recovery, production and restoration of the electro-acoustic parts of Gilberto Mendes music", the second integrated to the first and both with the support of FAPESP. The objectives of these projects are the study, and the recovery of the works of the composers from São Paulo who signed at the sixties the Manifesto Música Nova, which has used at first the resources of sound technology. In this paper we present the result of the study and a new technological version of the music **Cidade** (1964) by Gilberto Mendes.*

Resumo. *O presente artigo trata de um primeiro resultado parcial dos projetos "Faces da música eletroacústica brasileira: o Grupo Música Nova e seu pioneirismo na utilização de recursos tecnológicos" e "Recuperação, produção e restauro das partes eletroacústicas das obras mistas de Gilberto Mendes", o segundo integrado ao primeiro e ambos apoiados pela FAPESP. Tem como objetivo o estudo e recuperação das obras dos compositores paulistas signatários do Manifesto Música Nova que utilizam pioneiramente equipamentos tecnológicos. O presente artigo traz o resultado do estudo e versão tecnológica atualizada da obra **Cidade** (1964) de Gilberto Mendes.*

Introdução

Reza o Manifesto Música Nova 'um compromisso total com o mundo contemporâneo' e, dentre outros tópicos, o uso de 'processos fono-mecânicos e eletroacústicos em geral' para a música a ser produzida então. Esse manifesto foi concebido em 1963 e quatro dos seus mais importantes signatários são os compositores Damiano Cozzella, Rogério Duprat, Gilberto Mendes e Willy Corrêa de Oliveira (Cozzella, 1963).

Iniciada em março de 2005, nossa pesquisa, apoiada pela FAPESP, quer se aproximar da produção que efetivamente utilizou recursos tecnológicos disponíveis nos anos sessenta e setenta por parte desse grupo de compositores. Essa produção não publicada, pouco documentada e parcialmente perdida por deterioração dos meios (fitas analógicas) foi objeto de estudo apenas em uma dissertação de mestrado (Maués, 1989) cujo tema era a música eletroacústica brasileira em geral do período de 1956 a 1981. Pretende-se neste projeto a restauração e reconstrução possível das obras em meios digitais, a disponibilização dessas obras para divulgação, estudos e pesquisas teóricas que situem essa obra pioneira no devido lugar no panorama histórico da música eletroacústica brasileira.

O que aqui apresentamos é um primeiríssimo resultado parcial (dado o início recente da pesquisa), mas que dá conta de demonstrar bem a dimensão do nosso trabalho. Trata-se do estudo e trabalho com a obra *Cidade* (1964) de Gilberto Mendes, primeiro compositor focalizado em nossa pesquisa. Essa obra, elaborada sobre poema homônimo de Augusto de Campos, utiliza vários meios, vozes, instrumentos musicais (piano, percussão e contrabaixo), três toca-discos, um gravador e diversos aparelhos eletrodomésticos (televisão, máquina de escrever, calculadoras, outros aparelhos a escolha dos intérpretes). Trata-se de um de seus Teatros Musicais nos quais as partes cênicas - embora não detalhadas em partitura - têm importância fundamental na obra.

Em *Cidade* o ambiente cênico é o de uma loja-escritório, no qual há a presença de personagens 'secretárias', 'vendedoras' que manipulam os eletrodomésticos, máquinas de escrever e outros, tendo o regente um papel cênico ambíguo de 'regerente' (expressão cunhada por Haroldo de Campos para a obra de Mendes). A sobreposição dos mais diferentes trechos musicais em colagens (interpretados ao vivo pelos músicos e difundidos por toca-discos e gravador), alternadas com a interpretação do poema por três narradores, constrói metaforicamente o som caótico do ambiente urbano das grandes cidades nos anos sessenta.

Igor Linz Maués em sua dissertação classifica a obra *Cidade* como 'música eletroacústica em eventos para meios diversos (multimídia)'. Afirma que ela utiliza dentre os instrumentos descritos acima, o uso de duas fitas magnéticas monofônicas com sons urbanos numa realização com elementos teatrais. Na verdade a parte de tape, a ser montada pelo intérprete, pede montagens de trechos musicais diversos retirados de discos, gravação de locução, do poema e de partes para piano.

A primeira característica então comum à época dentre os compositores brasileiros que se envolveram com a utilização de meios eletrônicos é a concepção da obra em partitura textual e indicações para que o intérprete produza a fita magnética. O compositor concebe a montagem do tape, mas a sua produção fica ao encargo do intérprete. Maués refere-se a essa conduta do compositor nos anos 60 da seguinte maneira:

Muitos compositores sentem o caráter de permanência da fita magnética como um obstáculo a mais agindo contra uma possível abertura na interpretação da obra. Uma parte, uma vez realizada na fita magnética, não pode ser modificada com facilidade, sendo além disso forçosamente limitada as condições técnicas do local e da época. Naturalmente esse fato ocorre também com a notação, que pode no entanto assumir diferentes graus de determinismo. No caso da música eletroacústica, uma possibilidade de contornar esse problema é a de deixar para o intérprete a tarefa de realizar ele próprio a fita magnética. Essa atitude é frequentemente integrada à poética de compositores brasileiros refletindo na forma de notação. (Maués, 1989, p.43)

No entanto, essas obras, por outro lado, correm o risco de serem apresentadas apenas pelos poucos intérpretes que se dispõe a produzir a parte de tape. Quando esta é opcional, muitas vezes, ela é deixada de lado pelo intérprete. No caso de *Cidade*, Gilberto Mendes cita apenas três performances em solo brasileiro: no Teatro Municipal de São Paulo, sob a direção de Conrado Silva (data não citada), no Museu da Imagem e do Som e na Pinacoteca de São Paulo, dirigida por Carlos Kater (nos anos 70), e no evento 'A Trama do Gosto' (exposição no Pavilhão da Bienal de São Paulo), dirigida por Mara Campos (Mendes, 1994, p133).

Além da parte de tape, a obra requer também a utilização de três toca-discos e 15 trechos de gravações musicais de época (Lps) a serem levantadas em mais de uma cópia

para sua reprodução simultânea nos três aparelhos. A utilização de equipamentos de áudio e eletrodomésticos é também observada como traço característico da música desse período:

Fontes eletrônicas frequentes, além das fornecidas por sistemas de música eletrônica propriamente ditos, foram instrumentos amplificados eletronicamente..., rádios portáteis..., toca-discos... e outros aparelhos eletrodomésticos... O uso de rádios e aparelhos eletrodomésticos está, além da atitude insólita de cunho neo-dadaísta, também relacionado a chamada "estética da pobreza", como desenvolvida pelos artistas latino-americanos. (Maués, 1989, p.42)

A primeira pergunta que nos ocorreu a respeito da razão da utilização de três toca-discos e um gravador na obra estudada duas respostas possíveis e ainda não confirmadas com o compositor. A primeira é que a obra foi concebida como colagem musical: não apenas as partes executadas nos toca-discos e no gravador, mas também uma grande parte dos trechos executados ao vivo pelos músicos são curtos recortes do repertório, seja da música coral renascentista, seja de música de cabaré, ou melodia operística (os trechos em partitura não tem assinaladas as suas origens). Gilberto Mendes assinala que a inspiração para esta obra foi um quadro de Rauschenberg exposto na Bienal de São Paulo nos anos 60, declarando a influência da pop art norte-americana (Mendes, 1994, p.134).

O fato de o compositor requerer o uso de toca-discos para essas citações musicais e não requerer esses trechos tocados ao vivo pelos intérpretes têm a ver com uma recente conceituação bastante atual, cuja teorização ainda se esboça e que nos foi trazida ao Brasil pelo musicólogo francês François Delalande: o que o compositor quis colocar em sua música não foi apenas as citações musicais, mas sobretudo a citação dos 'sons' que são signos de uma época. Sem nos aprofundarmos neste aspecto particular, pois isso iria muito além dos objetivos deste artigo, vemos que o fato de Gilberto Mendes indicar as gravações de músicas correntes no universo social brasileiro dos anos 60 (Beatles, Roberto Carlos, Elis Regina, trechos orquestrais de Tchaikovsky, Offenbach e outros, indicações de gravações específicas, algumas com o registro do LP) têm o desejo de recriar o 'som' dessa época.

Não seria muito difícil rascunhar uma pequena semiologia do "som", tomados nesse nível grosseiro de grandes categorias musicais, que colocam em relação conjuntos de traços morfológicos precisos ataques, volumes, regularidade, etc.) com valores ou escolhas de vida que caracterizam os grupos sociais. Todo mundo sabe que as músicas são correntemente utilizadas pelas comunidades humanas para se reconhecerem. Mas o que é suposto aqui é mais preciso: é que o "som" é o primeiro índice (ao lado de outros...), ao ponto que é suficiente escutar um trecho de apenas um segundo para saber se tal música é das "nossas" ou não. É o som-assinatura. (Delalande, 2001, p.21)

É a voz daquele cantor, o som daquele instrumentista, o som resultante das técnicas de gravação de um período que transformam as sonoridades resultantes em uma marca, um índice. A técnica tem, segundo Delalande, fortes conotações estéticas. É uma coleção desses índices que o compositor deseja trazer à sua obra.

A segunda razão que nos ocorre para a utilização desses equipamentos específicos é que esta seria a solução técnica mais barata e fácil, já que na época era fácil coletar três toca-discos, mais fácil do que entrar em um estúdio e fazer as montagens em vários canais com todas citações musicais. Além disso a performance dos três tocadores de discos e o tocador de tape se casa perfeitamente com a concepção cênica do teatro musical com ares de 'happening', uma outra vertente artística tão ao gosto dos anos 60. O acréscimo

do gravador aos toca-discos tem como única função a de já trazer algumas montagens dos trechos musicais realizadas caseiramente em um único canal.

As indicações das entradas dos toca-discos, trechos a serem tocados e as montagens prévias dos trechos em *tape* são relativamente estritas, por vezes em segundos ou com delimitações dos compassos das músicas. Outras vezes, o compositor assinala relativamente o trecho (o início por exemplo) sem maiores exatidões. Isso faz com que haja uma relativa fixidez a cada apresentação da obra mas não uma total fixação do resultado, sendo condizente com a afirmação de Maués assinalada acima.

Desta forma, sons trazidos nesses equipamentos não cumprem o papel de objetos sonoros abstratos, materiais retrabalhados cuja origem foi escondida pelo compositor, mas são sons-índices que traçam um quadro da cultura musical de uma época. Como nos lembra Maués e podemos confortavelmente relacionar a sua classificação a essa obra de Mendes, as colagens desses trechos têm forte conotação metalinguística:

Desempenha um papel importante em obras brasileiras, o uso do som gravado e reproduzido de maneira realista. Como técnica, esse recurso é extremamente simples. Porém no contexto da obra pode assumir aspectos extremamente complexos. O meio eletrônico nesse caso desempenha diferentes funções. A mais imediata é a deslocação espacial ou temporal sem modificação de elementos não presentes. O som gravado é então ouvido em primeiro plano ou de fundo e a fonte original do som é reconhecida, às vezes tomada como presente. Quando executada em sincronia com partes ao vivo, atua muitas vezes como extensão desses elementos.... pode ainda, ao repetir eventos já ouvidos atuar como memória sonora... ou ao remeter a outra gravação, como citação metalinguística.(Maués, 1989, p.41)

Antes de adentrar o estudo que realmente nos traz a este artigo, colocamos aqui como sinal uma pergunta sem a devida resposta, resposta esta que esperamos poder encontrar quando o percurso desta nossa pesquisa estiver mais adiantado: obras como essa de Gilberto Mendes, nas quais o compositor concebe o uso de *tape*, sem contudo entrar ele mesmo no estúdio e esculpir ou compor concretamente o seu som ou sua música, quando os sons reproduzidos por esses equipamentos têm o caráter estrito de citação metalinguística, podem essas obras serem classificadas como músicas eletroacústicas, tal qual denominou Maués?

Entretanto, podendo ou não serem chamadas de músicas eletroacústicas essas obras introduziram um universo sonoro nas músicas eruditas brasileiras dos anos 60 e 70, gravadores, toca-discos e amplificadores passaram a fazer parte do instrumental musical desse repertório. E o grande problema que aflora disso, e que está ocupando músicos envolvidos com tecnologia no mundo todo é: as tecnologias e esses instrumentos se tornam obsoletos com uma rapidez assustadora, de formas que passadas umas dezenas de anos as obras não se conservam, os meios se deterioram, as tecnologias e equipamentos que as produziram desaparecem do mercado. Por isso um dos principais objetivos do nosso trabalho é o de restauro de fitas analógicas e de buscar uma versão atualizada para as tecnologias não mais em uso. No caso de *Cidade*, uma substituição para os três toca-discos, o gravador analógico, os LPs e a fita analógica. Se fôssemos buscar soluções caseiras, bastaria substituir os toca-discos e gravador por toca Cds e LPs e fita por Cds. Mas pensamos em uma solução mais condizente com a tecnologia disponível ao músico que trabalha com tecnologia, mais especificamente, uma solução que desse conta das realizações simultâneas em tempo real: a programação de um *patch* no ambiente Max MSP.

Primeiramente fizemos o levantamento das gravações das obras requeridas pelo compositor, os LPs dos anos 60. Para melhor compreensão da partitura-texto, fizemos uma transcrição desta em novo formato, de maneira a ficar mais clara a forma musical e as partes nas quais atuam os toca-discos e o gravador. Como dissemos acima, a razão da utilização do gravador foi provavelmente a de já trazer montado pelo menos em um instrumento as colagens dos mesmos trechos musicais apresentados parcialmente também nos toca-discos. Por esta razão, os quatro equipamentos ficaram como correlatos no desenho do *patch*.

A escolha do computador se deu pela facilidade de uso final, pela maleabilidade que ele oferece para uma performance e por o considerarmos o melhor meio para simular uma realidade de outra época, além de se adaptar perfeitamente à temática da obra.

O processo de programação partiu dos seguintes princípios: criar uma interface de uso fácil e intuitivo, porém que remetesse diretamente às imagens originais da obra musical, e que compilasse todo o processo de performance dos materiais gravados em um único computador. Para o caso específico dessa obra, partimos da idéia de simular quatro *players* em uma tela, sendo três toca-discos e um gravador, aparelhos originalmente integrantes da performance, com instruções específicas e por vezes simultâneas. Esses quatro *players* foram montados de maneira a funcionar independentemente, para que o intérprete mantenha o pensamento próximo à idéia inicial presente na obra, buscando assim uma interferência menor sobre o resultado da performance desejado pelo compositor, e que também não implicasse na reescrita da partitura, devido ao ajuntamento de quatro *performers* em um único, assim como uma mudança na linguagem adotada na partitura.

Tomando como referência nesse trabalho a simulação dos eletrodomésticos citados, realizamos a construção de réplicas. Para isso disponibilizamos ao o usuário acesso apenas àqueles controles presentes nesse aparelhos que são necessários para a execução da obra.

O *patch* é apoiado por uma interface gráfica que ilustra as imagens originais da obra, que se quer simular, e objetiva seu uso intuitivo por parte de um usuário que não esteja envolvido e acostumado com o processo e a linguagem de programação em Max/MSP, pensando sempre em facilitar a execução da obra. Para isso, no processo de programação procuramos eliminar funções de controle dos dados específicas desse software, como o envio de *mensagens* (*Message Box*) ou o controle de valores por *numboxes*, substituindo por equivalentes gráficos como *sliders*, *menus* ou botões gráficos, que atualmente estão presentes e são o padrão em qualquer tocador de áudio feito para o usuário comum, além de remeter diretamente aos suportes originais. Com isso, o intérprete não inicia o funcionamento do *patch* com uma mensagem de start para o *object dac*, mas sim com botões On/Off (Liga/Desliga).

Dessa forma, todo o processo de programação foi ocultado, obtendo uma tela limpa e clara para o intérprete, com apenas os controles necessários para a execução das respectivas partes presentes.

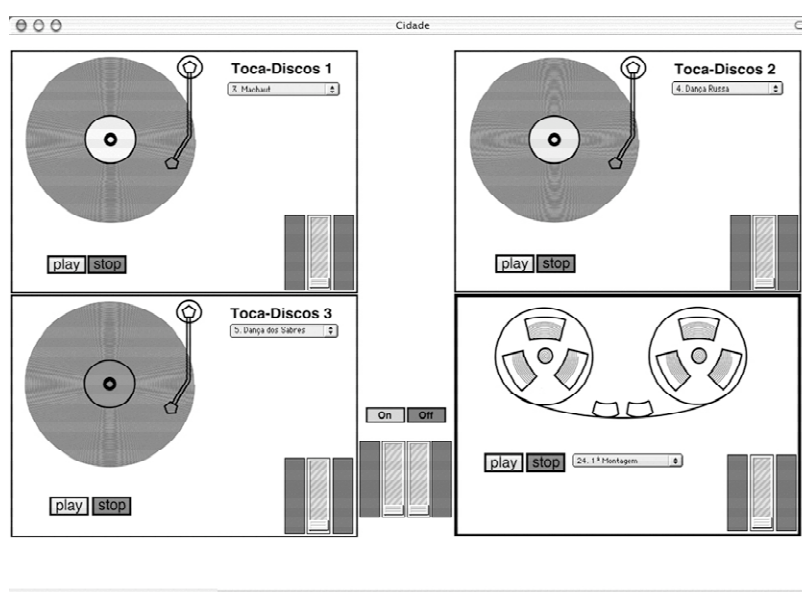


fig.01 imagem do patch fechado

Procuramos também automatizar ao máximo o processo de performance: os áudios são selecionados automaticamente por um menu, em ordem linear de acontecimento (assim que é finalizada a instrução anterior, a próxima é preparada). Esse menu também é passível de ser controlado pelo intérprete, sem alterar a performance do programa. Ao intérprete cabe disparar a ação, que é orientada na performance pelo regente. Controles de volume independentes para cada *player* e um geral, estéreo, foram disponibilizados para o intérprete para uma maior controle da do sinal enviado. Eles são ajustados a um *preset* automaticamente quando o usuário aciona o sistema Liga/Desliga do *patch*. Quando a execução de um trecho de áudio é interrompida, seja pelo desligamento do *patch* ou pela mensagem *stop* dos *td*'s/gravador, a leitura volta para o início do trecho interrompido.

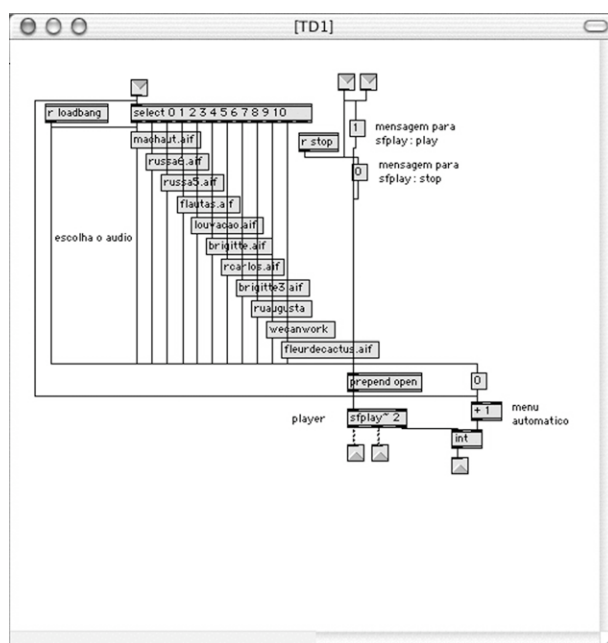


fig.02 - janela de programação do Toca-Discos 1

Embora não seja muito complexo, todo o sistema foi pensado para evitar problemas na sua utilização, sempre pensando em um intérprete leigo em Max. Para isso, evitamos a instalação de recursos desnecessários, que complicariam sua utilização e exigiriam hardwares com maior desempenho. O patch foi pensado para rodar em qualquer computador Macintosh com o sistema operacional MacOS X.2 ou superior.

A escolha da programação em Max se deve à estabilidade do software, e ao recurso disponível que permite compilar todo o patch em um programa externo, que trabalha independente de instalação desse programa, pago. Com isso, o resultado final será um aplicativo, gratuito, como já dito para usuários de MacOS X.2 ou superior, que será disponibilizado para os intérpretes por intermédio do CDMC-Unicamp.

Conclusão

Esta programação em substituição aos equipamentos já obsoletos será brevemente apresentada ao compositor, que dará o aval final sobre sua manutenção. Uma prioridade no nosso trabalho é não macular o projeto poético do compositor, apenas facilitar o acesso e a possibilidade de o intérprete produzir a obra e desta forma restaurar a existência da obra no repertório dos intérpretes e do público.

Referências bibliográficas

- Brasil. Ministério das Relações Exteriores. Departamento de Cooperação Cultural, Científica e Tecnológica. *Gilberto Mendes: catálogo de obras*. Brasília, 1976.
- Cozzella, Damiano et al. *Música Nova. Invenção*. São Paulo, número 3, ano 2, p.5-6, junho 1963.
- Delalande, François. *Le Son des musiques*. Paris: Buchet/Chastel-INA/GRM, 2001.
- Linz Maués, Igor. *Música eletroacústica no Brasil: composição utilizando o meio eletrônico (1956–1981)*. Dissertação de Mestrado. São Paulo: Escola de Comunicações e Artes da Universidade de São Paulo, 1989.
- Chion, Michel; Reibel, Guy; *Les musiques éleetroacoustiques*. Paris: Ed. INA GRM / Edisud; ISBN: 2-85744-015-4 (1976)
- Gendre, Claude; *Les Magnétophones*; Ed. Fréquences, distribuído por Ed. Radio; (1981).
- Mannis, José Augusto; *L'Electroacoustique dans la Musique d'Aujourd'hui*; Memoire en maitrise, Université de Paris VIII; 1987
- Mendes, Gilberto. *Uma odisséia musical: dos mares do Sul à elegância pop-art déco*. São Paulo: Edusp, 1994.

New Feature For Automatic Speech/Music Discrimination

Jayme Garcia Arnal Barbedo¹, Amauri Lopes¹

¹Department of Communications – FEEC – UNICAMP
C.P. 6101 – CEP 13.081-970 – Campinas – SP – Brazil
{jgab,amauri}@decom.fee.unicamp.br

Abstract. *This paper presents a new mechanism for automatic speech/music discrimination (SMD). Such feature is based on the concept of multiple fundamental frequencies. The performance of the feature in terms of correct classifications is evaluated for a wide variety of audio signals, and factors such as computational complexity and robustness are also investigated. The results are compared to those ones reached by previous techniques.*

1. Introduction

In the last decade, the demand for techniques able to automatically discriminate between music and speech signals has risen dramatically. There are several technologies that can benefit from the advances achieved in this area, as Automatic Speech Recognizers and Automatic Music Transcriptors, which must be fed with the appropriate signals. Other applications for speech/music discrimination techniques are the hearing devices and the automatic selection of FM radio stations.

The first researches in SMD have reached about 95% of accuracy in their experiments [Saunders 1996], [Scheirer and Slaney 1997]. Several works have followed those early proposals [Carey et al. 1999], [Cho et al. 2003], [El-Maleh et al. 2000], [Jarina et al. 2002], [Lu et al. 2002], most of them presenting accuracy between 92% and 98%.

There are two characteristics that are common to all speech/music discriminators: 1) the great number of features extracted from the signals and 2) the use of techniques such as Gaussian Mixture Models (GMM), Hidden Markov Models (HMM) and k-Nearest Neighbors (KNN) to combine such features. These approaches have a number of drawbacks associated: high computational and programming complexity and a large number of degrees of freedom, reducing the robustness of the approach. The technique presented here overcomes most of such limitations, since it is extremely simple to implement, requires little computational resources and is composed of only one feature, meaning that it is very robust to a wide range of situations.

2. Feature Extraction

Before the feature extraction itself, the signal must be properly formatted to fit the process requirements. The first step is to identify if the signal is monophonic or has more than one channel. In the first case, no action is taken; otherwise, the channels must be combined using a simple arithmetic average. In this work, the signals are sampled at 48 kHz and divided into frames of 1,024 samples, corresponding to time intervals of 21.3 ms. The frames are 50% superposed and are weighted by a Hanning window.

The strategy presented here is based on the signal main fundamental frequencies

(f_0) detection. Since the signals analyzed here have several sound sources, some kind of processing is necessary to ease the detection of the f_0 of each sound source. Most of the techniques described in the following were inspired in the multipitch analysis model presented in [Tolonen and Karjalainen 2000]. The strategy is illustrated in Figure 1.

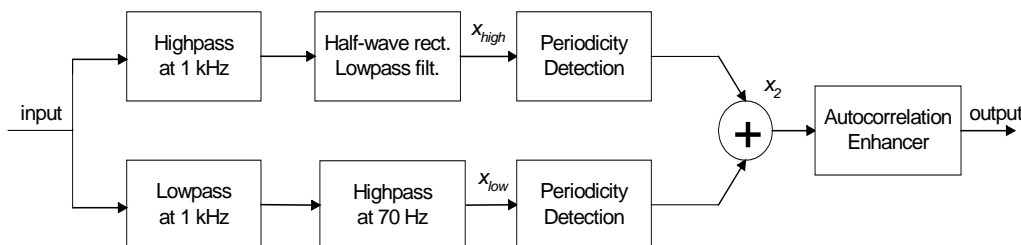


Figure 1. Strategy to estimate multiple fundamental frequencies.

In Figure 1, the input consists of the signal frames, and is divided into two bands (low and high frequencies) by a filtering process with cut-off at 1 kHz. The low frequency portion is also submitted to an extra filtering to block frequencies below 70 Hz. The high frequency portion is then submitted to a half-wave rectification. After that, it is lowpass filtered with a filter similar to that used to determine the low frequency portion.

The periodicity detection, which results in x_2 in Figure 1, is based on the concept of “generalized autocorrelation”, and is given by

$$x_2(n) = \text{IDFT} \left[\left| \text{DFT}(x_{low}(n)) \right| + \left| \text{DFT}(x_{high}(n)) \right| \right], \quad (1)$$

where DFT and IDFT represent the Discrete Fourier Transform and its inverse, respectively, and n is the time index.

The peaks of the autocorrelation given by $x_2(n)$ are good indicators of potential fundamental frequencies. However, since the signals have multiple sound sources, $x_2(n)$ can show lots of spurious information that can potentially lead to wrong estimations. To reduce the amount of unwanted information, a peak pruning technique is applied. Firstly, a half-wave rectification is applied to clip negative values of $x_2(n)$. The resulting function is expanded in time by a factor-two oversampling and subtracted from the clipped autocorrelation function. This procedure eliminates all peaks with twice the time lag of a higher amplitude reference peak. The technique also removes near-zero values of the autocorrelation function. In the present work, the procedure was applied to eliminate peaks with twice and three times the time lag of the reference peaks.

The next step is to identify the three main peaks of the enhanced autocorrelation function for each frame. Those three peaks are taken as the f_0 of the three main sound sources of the frame. If less than 3 sources are present, only one or two peaks will be identified. The estimated frequencies are then converted to the MIDI scale, according to the procedure described in [Tzanetakis and Cook 2002] and given by

$$m = 12 \log_2(f/440) + 69, \quad (2)$$

where f is the frequency in Hz and m is the MIDI number. All frequencies with same MIDI number are counted over all frames, generating a histogram whose bins are the MIDI notes.

It was observed that most of the analyzed speech signals have frequencies whose corresponding MIDI numbers are equal or greater than 100, while music signals rarely present such high frequencies. This probably occurs because of short speech segments with low energy and high frequency whose period is defined enough to be detected by the detection procedure. Next section will describe how this information is explored in order to provide a reliable differentiation between speech and music signals.

3. Tests and Results

The database used in the tests is composed by 2,587 wav-format audio files sampled at 48 kHz and quantized with 16 bits, and it is divided into speech and music files.

As commented before, it was observed that speech signals often presents higher f_0 than music signals. Therefore, the first task is determining the proportion of high frequencies that leads to the best discrimination between speech and music signals. It was observed that the following rule led to the best results: given a histogram, if the proportion of MIDI values equal or higher than 100 is over 0.1%, the signal is considered speech; otherwise, the signal is considered music. Table 1 summarizes the results obtained.

Table 1. Results

Group	Right Classification Percentage
Speech (all files)	94.01%
Speech (only files without environmental noise)	96.05%
Music (all files)	93.63%
Music (without rap files)	94.87%

As can be seen, the percentage of right classifications lies between 93 and 96%. In the case of speech signals, the performance is very good even when strong environmental noise (street, office, nature sounds) is present, indicating that the strategy is very robust to extreme conditions. In the case of music, it was observed that for some musical genres, like classical and rock, the percentage of right classification is near 100%, while for musical genres that have several elements of actual speech, like rap, the correctness can drop to values below 80%.

Comparing the proposed procedure with some methods in the literature, one can conclude that there are previous techniques presenting slightly better discrimination accuracy. The best results were achieved by [Lu et al. 2002], which used a very complex strategy to reach a precision of about 98%. However, when the comparison is done taking into account not only the discrimination but also the robustness of the discriminator to unexpected situations and the computational effort demanded by the method, the conclusion is that the strategy here presented is clearly superior. As commented before, since this proposal depends only on one feature, it presents a great robustness to unexpected situations, as can be observed in Table 2. Additionally, it demands low computational effort, indicating that the procedure can be used in real time applications, even when the available computational resources are scarce.

4. Conclusions

This paper presented a new strategy to discriminate between speech and music signals.

The technique consists of the extraction of a single feature based on the concept of multiple fundamental frequencies.

The performance of the strategy in terms of correct estimates is competitive with previous works. Additionally, it presents a clearly advantage in terms of robustness and computational complexity. The characteristics of this technique make it appropriate to be used in applications where potentially problematic conditions, like degradations and environmental noise, are expected. Finally, it can be used in real-time applications.

There are several possible directions for future research. A possible enhancement can be achieved with the improvement of the process used to estimate the multiple fundamental frequencies. Another interesting line of research is trying to combine the strategy here presented with another successful techniques. At last, some new features based on the histograms generated in the fundamental frequency estimation can be created and combined, in such a way the results are improved without adding significant computational effort. Two novel features that have already been implemented and that have shown good results are the ratio between the amplitude of the histogram peak and the histogram sum, and a measure to the variation between the bin amplitudes of consecutive MIDI notes.

Acknowledgements

The authors would like to thank Fapesp for supporting this research under grant n. 04/08281-0.

References

- Carey, M.J.; Parris E.S.; Lloyd-Thomas, H. (1999) "A comparison of features for speech, music discrimination", *Proc. of ICASSP99*, pp. 149-152.
- Cho, Y.-C.; Choi, S.; Bang, S.-Y. (2003) "Non-negative component parts of sound for classification", *Proc. IEEE Int. Symp. Signal Processing and Information Technology*, Darmstadt, Germany.
- El-Maleh, K.; Samouclian, A.; Kabal, P (1999). "Frame-Level Noise Classification in Mobile Environments", *Proc. IEEE Conf. Acoustics, Speech, Signal Proc.*, Phoenix, AZ, USA.
- Jarina, R.; O'Connor, N.; Marlow, S. (2002) "Rhythm Detection for Speech-Music Discrimination in MPEG Compressed Domain", *Proc. of the IEEE 14th International Conference on Digital Signal Processing 2002*, Santorini, Greece, pp. 129-132.
- Lu, L.; Zhang, H.-J.; Jiang, H. (2002) "Content Analysis for Audio Classification and Segmentation", *IEEE Trans. on Speech and Audio Proc.*, vol. 10, no. 7, pp. 504-516.
- Saunders, J. (1996) "Real-Time Discrimination of Broadcast Speech/Music", *Proc. of the IEEE Intern. Conf. on Acoustics, Speech, and Signal Processing*, vol. 2, pp 993-996.
- Scheirer, E.; Slaney, M. (1997) "Construction and Evaluation of a Robust Multifeature Speech-Music Discriminator", *Proc. of ICASSP97*, pp. 1331-1334, Munich, Germany.
- Tolonen, T.; Karjalainen, M. (2000) "A Computationally Efficient Multipitch Analysis Model", *IEEE Trans. on Speech and Audio Proc.*, vol. 8, no. 6, pp. 708-716.

Sistema Especialista para Detecção do Timbre da Clarineta

Leonardo Carneiro de Araújo

¹CEFALA – Centro de Estudos da Fala, Acústica, Linguagem e Música
UFMG – Universidade Federal de Minas Gerais

leoca@cefala.org

Abstract. *It is presented bellow a method for automatic identification of the clarinet. The schema described bellow is based on the analysis of the decomposed acoustic signal using wavelets functions. For each sample of an instrument it is automatically identified the attack, sustain and release regions. In each of the regions a decomposition process is carried out and finally it is got as a result the mean and variance of the wavelets coefficients. Those are the input data of the fuzzy or neural network used in the identification of the clarinet. The experiments show it is possible to acquire a low level of error rate.*

Resumo. *É apresentado um método para identificação automática da clarineta. O esquema proposto é baseado na análise da decomposição do sinal acústico através de funções wavelets. Para cada amostra de um instrumento são identificados automaticamente a região de 'attack', 'sustain' e 'release'. Para cada uma dessas regiões é feita a decomposição e é obtida a média e a variâncias dos coeficientes de wavelet. Esses serão os dados de entrada da rede fuzzy ou neural para identificação da clarineta. Os experimentos mostram que é possível obter uma boa taxa de acertos.*

1. Introdução

Timbre é uma característica sonora multidimensional e, por conseguinte, de difícil definição, em contraposição à altura e a intensidade percebida que podem ser facilmente definidas por serem características unidimensionais. O timbre é comumente definido como tudo o que caracteriza um som que não é a sua intensidade e sua altura. Esta inclusive é a própria definição adotada pela ANSI (American National Standards Institute) "... that attribute of auditory sensation in terms of which a listener can judge that two sounds, similarly presented and having the same loudness and pitch, are different". Sabemos também que o timbre é uma característica que não pode ser tomada isoladamente das demais. A altura, a duração e a intensidade influenciam o timbre sonoro percebido.

A tarefa de extrair características timbrísticas do sinal acústico não é trivial. Muitos estudos de psicoacústica mostram que a parte estacionária do sinal acústico não é, muitas vezes, suficiente para prover toda a informação necessária ao reconhecimento de um instrumento. Existe então características importantes nos transientes do sinal (ataque e decaimento). Como não sabemos, precisamente, definir o que é timbre, é de se esperar que a tarefa de classificar ou segregar sons com relação as suas informações timbrísticas seja, por conseguinte, imprecisa. No entanto, isso não nos impede de formular estratégias e extrair características, que, embora possam não se assemelhar ao processo realizado pela mente humana para fazer esse tipo de segregação, podem ser tão eficientes quanto.

Sabemos que existem alguns fatores que são determinantes na constituição da informação timbrística. O conteúdo espectral e, também, a forma como este varia no

tempo é extremamente importante na determinação do timbre. Algumas possíveis abordagens seriam: determinar, através de uma transformada de Fourier com janela, a forma como o espectro do sinal evolui. Uma segunda possibilidade seria obter para uma janela deslizante sobre o sinal os coeficientes mel cepstrais que representam o sinal sob a janela. A forma de evolução desses coeficientes poderia, de alguma forma, conter a informação timbrística e assim ser utilizada na separação e identificação de padrões.

Utilizaremos aqui uma terceira opção a transformada wavelets. Veremos adiante algumas características dessa transformada (sessão 2.1). A forma como os coeficientes da decomposição em wavelet variam no tempo pode nos dar um bom parâmetro que capte a informação timbrística de um som musical.

Inicialmente utilizaremos uma forma bem simples de averiguar como variam os coeficientes da transformada wavelet do sinal. Tomaremos apenas as médias e as variâncias dos coeficientes wavelets nas primeiras escalas de decomposição. Escolhidos os parâmetros que utilizaremos, resta-nos decidir de que maneira faremos a classificação com base nesses parâmetros. As redes neurais artificiais podem conseguir bons resultados na classificação dos vetores de características timbrísticas, uma vez que são capazes de aprender tudo o que são capazes de representar.

1.1. Dados utilizados na pesquisa

O conjunto de dados utilizados nesta pesquisa consiste em amostras de notas dos seguintes instrumentos: canto, clarineta, flauta transversal, trombone, trompete e viola. Todas as amostras foram gravadas no estúdio da Escola de Música da UFMG. Como dispomos de uma quantidade muito superior de amostras de clarineta do que dos demais instrumentos, optamos por fazer um sistema especialista que distingue a clarineta dos demais instrumentos.

2. Abordagem do Problema

Uma das vantagens em se utilizar wavelets é que sua teoria já está bem consolidada, principalmente com relação a construção de bases e algoritmos eficientes para transformação, graças a trabalhos de Mallat, Daubechies, dentre outros. Uma outra característica peculiar das wavelets é a existência de diversas bases, sendo possível escolher a base mais adequada para o trabalho em questão, embora tal escolha seja feita com base apenas experimental. Além dessas características citadas, a complexidade computacional das wavelets é linear com o número (N) de coeficientes computados ($O(N)$) enquanto outras transformações obtêm complexidade da ordem de $N \times \log_2(N)$.

2.1. Decomposição wavelet do sinal de áudio

As funções wavelets formam uma base para um subespaço das funções quadrado integráveis e possuem ainda uma característica peculiar que será aqui importante. As wavelets fornecem uma análise de múltiplas resoluções através de um seccionamento do subespaço inicial em diversos subespaços encaixantes. Esses subespaços contêm as diferenças de informações entre duas versões de subespaços em resoluções diferentes. Se os subespaços de baixa resolução são obtidos a partir de filtragens passa-baixas, os subespaços gerados pelas funções wavelets em cada nível são obtidos a partir de filtragens passa-banda. Além disso as funções wavelets possuem suporte compacto, dessa

forma cada função da base terá uma contribuição local para o sinal a ser decomposto. Ou seja, a contribuição das funções wavelets é local tanto em tempo, quanto em frequência.

Existe então um algoritmo rápido que utiliza os filtros espelhados em quadratura para realizar a mudança de escala. Tipicamente um filtro passa-baixas e um filtro passa-banda (H e G respectivamente) são utilizados. A convolução com o filtro passa-baixas resulta em uma representação de baixa resolução do sinal A e a convolução com o filtro passa-banda resulta na versão de detalhes do sinal D .

A transformação de escalas é feita através do seguinte algoritmo rápido: $a_{j,n} = \sum_k \overline{h_k} a_{j-1,2n+k}$ e $d_{j,n} = \sum_k \overline{g_k} a_{j-1,2n+k}$ onde $a_{j,n}$ e $d_{j,n}$ são os coeficientes correspondentes a análise de multiresolução, representando o sinal na escala j , obtidos da filtragem da versão do sinal na escala $j - 1$.

Optamos por fazer uma análise do sinal, efetuando apenas duas etapas de decomposição. Teremos assim, para cada sinal três vetores que o representam: o vetor da representação de baixa resolução, o vetor dos detalhes de nível 2 e o vetor dos detalhes de nível 1.

Cada amostra de áudio foi segmentada automaticamente para separar o sinal em regiões transientes (ataque e decaimento) e de estado estacionário ("sustain"). Cada uma dessas regiões foi tratada como uma sinal independente para o qual foi efetuada a análise de multiresolução, obtendo assim os três vetores acima citados.

Teremos assim um total de 9 vetores por amostra. Mas cada vetor possui muitos coeficientes o que tornaria impraticável utilizá-los como entrada de uma rede fuzzy. Optamos então por apresentar para a rede apenas as médias e as variâncias de nossos coeficientes. Teremos então 18 entradas na rede, sendo 9 médias e 9 variâncias, 3 dessas para cada um dos segmentos da nota.

3. Resultados experimentais

As amostras amostras originais foram normalizadas para que todas tivessem a mesma amplitude eficaz. Após serem normalizadas elas foram automaticamente segmentadas. Após obtidos os segmentos das amostras de áudio, para cada um deles foi obtido a média e a variância dos coeficientes de wavelets na primeira e segunda escalas.

O conjunto de dados foi separada em conjunto de treinamento, conjunto de validação e conjunto de teste. Este conjunto foi montado de forma a ter o mesmo número de padrões para a clarineta e os demais instrumentos.

Com a rede fuzzy utilizada (que possui 18 entradas e 18 funções de pertinência), obtivemos alguns resultados muitos bons que são ilustrados nas figuras 3. Nesses resultados a taxa de acerto chegou a atingir 100%. Como limiar de separação adotamos o valor 0.5, de forma que qualquer resultado obtido abaixo de 0.5 é considerado zero (acerto, para o caso ilustrado na figura) e qualquer resultado acima de 0.5 é considerado um (erro, para o exemplo abaixo).

Uma rede MLP com 3 camadas intermediárias foi treinada utilizando a técnica 'early stoping'. Para a melhor rede obtida, cujo resultado de treinamento consta na figura 4, obtivemos 100% de acerto nos testes utilizando o conjunto de teste. Mas a grande

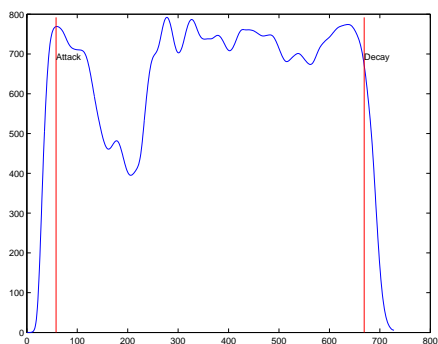


Figura 1

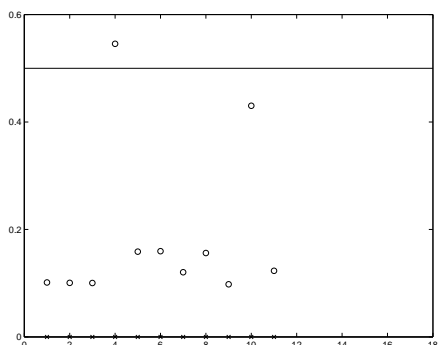


Figura 3

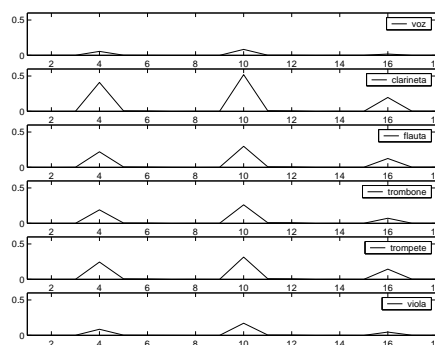


Figura 2

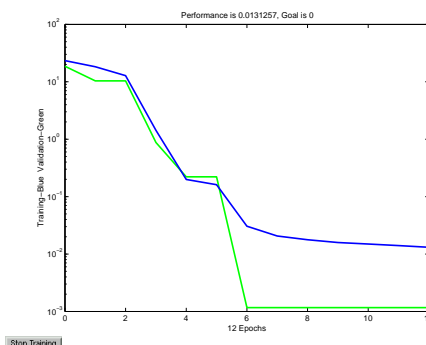


Figura 4

maioria das outras redes obtidas possuíam percentagem de acerta de aproximadamente 90%.

4. Conclusão

Os parâmetros extraídos das amostras de som (notas) aparentemente mostraram-se eficientes para representar a característica timbrística de instrumentos, uma vez que com base nesses parâmetros apenas foram obtidas altas taxas de acertos utilizando quer uma rede neural MLP ou uma rede fuzzy. Um próximo trabalho será coletar mais amostras dos demais instrumentos e tentar fazer um classificador capaz de reconhecer os diferentes instrumentos.

Referências

- Garcia, C., Zikos, G., and Tziritas, G. (2000). Wavelet packet analysis for face recognition. *Image Vision Computing*, 18:289–297.
- Haykin, S. (1999). *Neural Networks*. Prentice Hall, 2nd edition.
- Jang, J.-S. R., Sun, C.-T., and Mizutani, E. (1996). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Pearson Education, 1st edition.
- Mallat, S. (1998). *A Wavelet Tour of Signal Processing*. Academic Press, 2nd edition.

SOAL: Ferramenta para análise musical no ambiente Open Music

André Lira Rolim^{1*}, Bruno Jefferson^{1*}, Didier Guigue^{2†}

1-Departamento de Informática – Universidade Federal da Paraíba - Campus 1 -
João Pessoa – Paraíba – Brasil

Andrefloyd10@yahoo.com.br, bruno@lavid.ufpb.br

2- Departamento de Música – Universidade Federal da Paraíba - Campus 1 -
João Pessoa – Paraíba - Brasil

dguigue@cchla.ufpb.br

Resumo. Este artigo apresenta uma biblioteca de funções que implementamos com a linguagem LISP e CLOS, e que se encontra integrada ao ambiente OpenMusic. A SOAL (Sonic Object Analysis Library) é destinada à análise de arquivos MIDI de música do Séc. XX, dentro de certas premissas analíticas que iremos também brevemente explicitar.

Abstract. This paper shortly presents a library of functions, implemented in LISP and CLOS, we have developped as part of the OpenMusic environment. The goal of SOAL - SonicObjectAnalysisLibrary - is to analyze MIDI files of 20th Century music, according to some primeses we will also briefly explain.

1.Introdução

O suporte teórico deste projeto de biblioteca de funções é uma abordagem da música não-tonal do Séc. XX a partir do conceito de *objeto sonoro*. As premissas e a metodologia foram extensivamente descritas e exemplificadas em publicações anteriores [Guigue 1997, Guigue 2005a, Guigue 2005b]. De forma resumida, podemos sintetizar que, no domínio da música acústica, o que chamamos de *objeto sonoro* pode ser definido como a combinação e interação de componentes musicais primários (*classes de notas*) com secundários. Estes são componentes de ordem estatístico e/ou relativo, tal como intensidades, densidades, e, geralmente falando, componentes que quantificam ou qualificam as modalidades de preenchimento do “espaço” e do “tempo” musicais. É uma estrutura de nível intermediário, entre o nível elementar e atômico das classes de notas e o nível superior da macro-estrutura. A forma como esses objetos são interligados pode vir a ser um importante vetor da estruturação da música do século 20, quando o timbre se tornou tão ou mais funcional quanto a organização das alturas [Guigue 2005b]. SOAL é distribuída aos membros do Forum *OpenMusic*¹.

*Bolsista CNPq do programa PIBIC da UFPB, processo 109368/2003-7.

† Projeto de pesquisa financiada pelo CNPq.

¹ Mais informações: <http://forumnet.ircam.fr/> (menu *OpenMusic*).

Também é disponibilizada gratuitamente no site do GMT na UFPB, onde se desenvolve este trabalho [<http://www.cchla.ufpb.br/gmt/> (menu “software”)].

2. Estrutura da biblioteca

SOAL é dividida em duas pastas principais que agrupam as funções essenciais de análise dos arquivos MIDI: a pasta *Achronic Analysis*, que apenas considera as informações de notas (*pitches*) e intensidades (*velocity*) dos arquivos analisados, e a pasta *Diachronic Analysis*, que processa todas as informações relativas as modalidades de distribuição no tempo dos eventos (*notas*).

Ambas as pastas são divididas em duas sub-pastas para organizar as funções de acordo com seu objetivo principal, que é, seja de ordem quantitativo (primeira sub-pasta de cada pasta, medindo o âmbito, espacial ou temporal, do objeto analisado), seja de ordem qualitativa (segunda sub-pasta), qualificando as modalidades de distribuição dos eventos de acordo com diversos paradigmas.

SOAL disponibiliza também outras pastas: uma, chamada *piano specific*, contém uma seleção de funções otimizadas para música de piano; outra, *stats & utils*, oferece uma coleção de funções mais genéricas para propósitos de análise estatística e matemática, incluindo entre outras, vários modelos de cálculo do desvio padrão otimizado para dados musicais. Finalmente, fica a função de leitura dos arquivos MIDI, e cujas informações em saída são no formato que todas as demais funções de *SOAL* esperam receber em entrada.

3. Utilizando a SOAL

A função que extrai dados dos arquivos MIDI para serem utilizados nas demais funções da biblioteca é chamada de *multi-midi-reader*, e desde a versão 1.2 agora distribuída, suporta a leitura recursiva de um número teoricamente infinito de arquivos MIDI como entrada². O *multi-midi-reader* extrai quatro tipos de dados, a saber: *notes* (notas), *onsets* (disparos), *durations* (durações) e *velocities* (velocidades).

De posse dessas informações, o usuário pode então utilizar as várias funções da biblioteca, podendo até agregar funções da linguagem LISP e demais funções definidas no *OpenMusic*, aproveitando as facilidades oferecidas pela interface deste ambiente. A figura 1 mostra um *patch* típico utilizando *SOAL*³. Nela o *multi-midi-reader* extrai informações de dois arquivos MIDI, onde as suas saídas estão ligadas a outras duas funções da *SOAL*, *file-duration* e *smaller-impulse*, que calculam o tempo e o menor impulso de cada arquivo, respectivamente. O exemplo mostra também que algumas funções de *SOAL* chamam outras funções, a exemplo da *events-density* aqui representada, que calcula a densidade relativa de eventos em relação a duração completa dos arquivos MIDI (cf. a documentação on-line na mesma figura). Vemos que *events-density* recebe os resultados das funções *file-duration* e *smaller-impulse* como entrada. Como todas as operações padrão em *OpenMusic*, o resultado das avaliações de *SOAL* é mostrado na janela LISP chamada “Listener”.

² A limitação do número de arquivos depende da limitação de hardware disponível.

³ A biblioteca é distribuída com alguns patches demonstrativos.

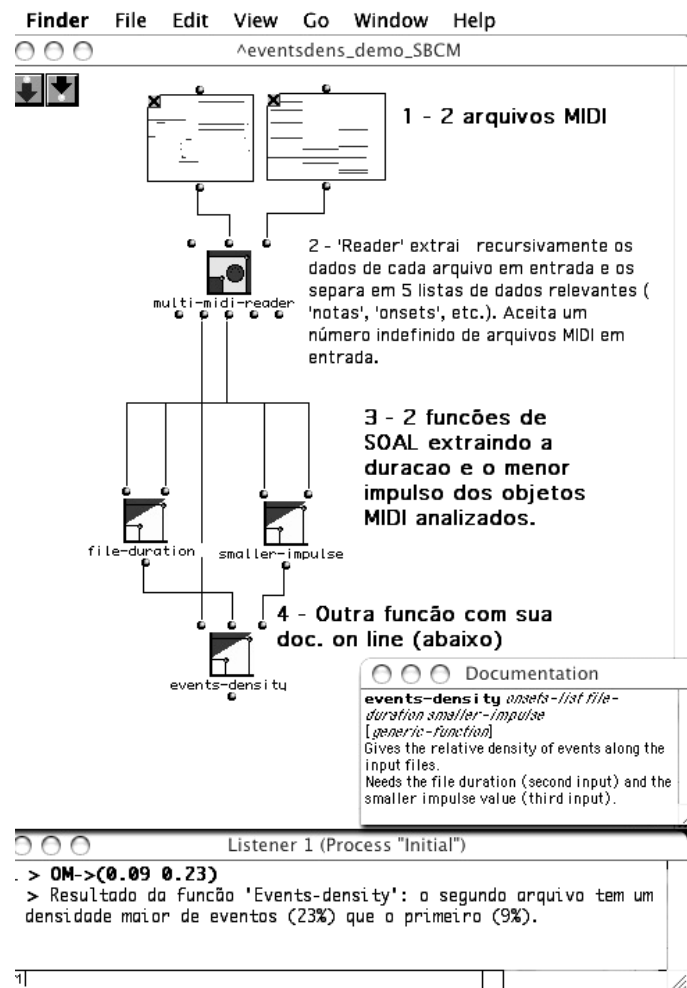


Figura 1: Um exemplo de aplicação da SOAL: o cálculo da densidade de eventos de dois arquivos MIDI utilizando 4 funções da biblioteca.

4.Implementando a SOAL

O uso de *LISP* na implementação da *SOAL* proporcionou grande suporte na implementação da *SOAL*. Visto que a necessidade de constantes cálculos matemáticos e ou estatísticos sob um ou mais conjuntos de dados de mesma natureza, nos obrigam a manipular inúmeras listas de dados, a linguagem *LISP* oferece grandes facilidades nas operações das mesmas. [Touretzky 1990]

O código fonte da biblioteca *SOAL* é organizado em oito arquivos: *Soal.lisp*, *OM-Methods.lisp*, *space.lisp*, *stats.lisp*, *time.lisp*, *piano-specific.lisp*, *constants.lisp* e *aux.lisp*, de modo que, como todos esses arquivos são empacotados no ambiente *Open Music*, qualquer função em um arquivo pode referenciar outra função contida em outro. Além dessas funções declaradas nos arquivos, a *SOAL* pode utilizar funções que *OpenMusic* implementa, fazendo referencia direta a elas, como no exemplo de código fonte mostrado na figura 2, onde temos uma função usada no *multi-midi-reader*, a *OperaMidiFiles*, que utiliza a função do Open Music *mf-info* para obter informações do primeiro elemento do parâmetro *midiFiles*.

```
(defun OperaMidiFiles(midiFiles)
  (first (mf-info (first midiFiles))))
```

Figura 2: declaração da função *OperaMidiFiles*, que retorna a primeira informação obtida pela função do Open Music, *mf-info*, referente ao primeiro arquivo conectado ao leitor *multi-midi-reader*.

5. Conclusões e trabalhos futuros

Neste curto texto, foi apresentada uma ferramenta computacional construída para fins de análise musical no ambiente *Open Music*. A construção da biblioteca *SOAL* se justifica pela inexistência de ferramentas automatizadas de avaliação (principalmente estatística) da estrutura de objetos musicais dentro deste ambiente especializado para a composição erudita. Ele interessa não somente ao analista, ao estudioso ou ao musicólogo, como também ao próprio compositor, que pode assim avaliar imediatamente as qualidades estruturais dos objetos gerados a partir de determinado processo composicional em fase de execução ou de experimentação.

SOAL se encontra em contínuo desenvolvimento. Em suas próximas versões serão trabalhadas a extração e a implementação de funções em cima dos assim chamados “controles contínuos” [MIDI 1996] assim como o desenvolvimento de outras funções de análise acrônica e/ou diacrônica dos eventos.

Referencias

Guigue, D. (1997). “Une Étude ‘pour les Sonorités Opposées’. Pour une analyse ‘orientée objets’ de l’œuvre pour piano de Debussy et de la musique du XX siècle”. Villeneuve d’Ascq : Presses Universitaires du Septentrion, 560 p. Acessível em forma condensada no servidor do GMT [<http://www.cchla.ufpb.br/gmt/> (menu “papers”)].

Guigue, D. (2005a). Sonic Object Analysis Library — OpenMusic Tools for analyzing musical objects structure. Electronic documentation of SOAL. Paris: IRCAM, 27 p. Também disponível no servidor do GMT [<http://www.cchla.ufpb.br/gmt/> (menu “software”)].

Guigue, D., Onofre, M. F., Rolim, A. (2005b). SOAL for music analysis: a study case with Berio’s *Sequenza IV*, 12^a Journées d’Informatique Musicale – JIM 2005. Paris: Actes des JIM 2005, p. 13-18. [<http://jim2005.mshparisnord.net/articles.htm>].

MIDI Manufacturers Association (1996). Complete MIDI 1.0 Detailed Specification, Version 96.1.

Touretzky, D. S. (1990). COMMON LISP: A Gentle Introduction to Symbolic Computation. Redwood City: The Benjamin/Cummings Pub. Co.

Waveforms Synthesis by Evolutionary Processes

José Fornari¹, Jônatas Manzolli^{1,2}, Adolfo Maia Jr.^{1,3}

¹ Núcleo Interdisciplinar de Comunicação Sonora (NICS), UNICAMP,
13.081-970, Campinas, SP, Brazil.

²Instituto de Artes, UNICAMP,
13.081-970, Campinas, SP, Brazil.

³Departamento de Matemática Aplicada, IMECC, UNICAMP,
13.081-970, Campinas, SP, Brazil.

fornari, jonatas, adolfo@nics.unicamp.br

Abstract. *We present a model for sound synthesis based on Genetic Algorithms (GAs) and the correspondent algorithm named ESSynth. It creates an evolutionary sequence of waveforms which converges to a Target Sound Set and can be used to several purposes such as real time sound environment, or even collect the resultant sounds for an off-time composition, electroacoustic music, etc.*

1. Introduction and Motivation

GAs have been, in the past few years, frequently applied to generate and manipulate evolutionary musical material. Biles [Biles, 1990] presented a genetic algorithm-based program that mimics a student learning to improvise jazz solos under the guidance of a human mentor. In Horowitz's [Horowitz, 1994] development, an interactive system uses GAs to develop a criteria to distinguish rhythmic patterns producing a large number of variations. One of the present authors has also studied applications of GAs to interactive composition [Manzolli et al., 1999]. Similar to the approaches described above, our previous research used MIDI data to control music events in real time. Yet, using a different heuristic, we created a system named Vox Populi [Moroni et al., 2000], a hybrid system formed of an instrument and a compositional environment. Evolutionary Algorithms were used by Johnson [Johnson, 1999] to develop a computer system for sound design. Roads [Roads, 1994] used genetic algorithms in granular synthesis to facilitate the regulation of its parameters.

Since it is, in general, difficult to combine quantitative and qualitative descriptions of a given sound, we apply the concept of Evolutionary Systems to develop a methodology for sound synthesis or, generally speaking, for Timbre Design. This method was named as ESSynth and is the principal point of this paper. ESSynth uses a set of *Target Waveforms* (in EA jargon, target population) to describe a timbral tendency generated out from an initial set of waveforms (initial population), new variants (generations) *similar* to those ones in the target. This similarity is measured by evaluations of a *Fitness Function*. ESSynth can be seen as a ruled-based algorithm that uses an implicit set of rules for generating waveform variations. In this way we want to formulate mathematical and computational models, in which we can define suitable genetic operations, such as crossover and mutation, operating as waveform transformations as well as defining measures for waveform similarities. By controlling the Target and Initial Populations, it is possible to create organized sound patterns or, at a higher level, to compose a piece of music. For the latter, an external device linked to the ESSynth such as a Wavetable engine is necessary in order to playback the musical sequences in real time.

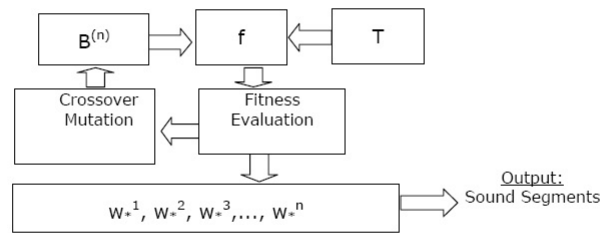


Figure 1: Basic ESSynth Diagram

In ESSynth the sound is dynamically generated by evolution, rather than being a static outcome. So GA are used as to generate sonic process instead of searching for an optimal sound. Some ESSynth characteristics are:

1. ESSynth has an intuitive controllability and easily generates "rich" sounds. For example, it merges the best of linear and non-linear synthesis methods.
2. ESSynth is a non-monitored learning machine that automatically pursues the best sound in a population (i.e. Wavetable) and improves each waveform according to the user's intuitive parameters, by handling the waveforms in the Target population.
3. The meaningful outcome is not only the sound segment but also the evolutionary trail of waveforms as time goes by. It comes along with the dynamic changing nature of sounds

2. ESSynth Description

There are three basic structures which comprises the kernel of ESSynth which we list below.

1. The population of the n -th generation is given by a set of m waveforms $\mathbf{P}^{(n)} = \{\mathbf{v}^{(n,1)}, \mathbf{v}^{(n,2)}, \dots, \mathbf{v}^{(n,m)}\}$. The number m of waveforms could change from population to population, but in this work, for the sake of simplicity, we keep it fixed. The initial population is denoted by $\mathbf{P}^{(0)} = \{\mathbf{v}^{(0,1)}, \mathbf{v}^{(0,2)}, \dots, \mathbf{v}^{(0,m)}\}$.
2. $\mathbf{T} = \{\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, \dots, \mathbf{t}^{(m^*)}\}$, denotes the Target population. Observe that the target can have a different number of elements of the populations in each generation, that is $m \neq m^*$.
3. A function f named *fitness function* used to evaluate the n -th generation's best individual denoted by a vector $\mathbf{b}^{(n)}$.

A simplified diagram of ESSynth is shown in Fig. 1. Waveform variants are produced by applying the genetic operations crossover and mutation on the population of individuals, for each generation $\mathbf{P}^{(n)}$. So they are external interventions producing a modified population in order to be selected from it a new best fitted individual. An interesting ESSynth's feature is to make waveform patterns dynamical sequences in real time. Biologic evolution produces species diversity. As a GA based application ESSynth creates and manipulates complex generations of sound material. In short, the operator crossover increases the waveform co-variance and mutation produces random population variations. These two genetic operations are defined in Φ , the collection of all finite sets (populations) of waveforms, as follows.

We show the algorithm that controls the selection of individuals from a generation to the next one presumably leading to the populations with more evolved individuals which here means those ones with higher fitness values. The algorithm is as follows.

1. Generate an initial population $\mathbf{P}^{(0)}$ as well the Target population \mathbf{T} of waveforms. This could be done importing them from a wavetable or from another application as synthesizers software.
2. Define the Fitness Function.
3. Find the best element.
4. Apply the Genetic Operators crossover and mutation shape on the actual generation and obtaining the next population.
5. Evaluate the distance to the Target Population using the Hausdorff Distance.
6. Repeat the steps (3), (4) and (5) for each generation.
7. Halt the process when the distance is less or equal a prefixed number ϵ

3. Experimental results and analysis

In this section we present a set of experiments we produced to evaluate the potential of the method. These experiments can be found at the link <http://www.nics.unicamp.br/fornari/essynth1/>.

We developed a methodology to evaluate and understand how the ESSynth works. This analytical process was based on the following criteria:

1. generate a family of sounds using populations in which the target and population have a very clear spectral discrepancy.
2. choose an individual in the population and plot his spectral evolution using a sonogram.
3. generate a sonogram representing the best individual spectral evolution.
4. compare (b) and (c) to evaluate similarities between the evolution of an individual and the overall evolution of the best one.

We run some experiments to evaluate the spectral evolution of the method using sine waves and white noise. The population of sine waves was generated constructing a scale using a logarithmic spread of frequencies within the interval $[f_{min}, f_{max}]$, where $f_{min} = 80Hz$, $f_{max} = 4000Hz$. Eq. (1) describes how we generated the initial population and Figure 2 shows the sonogram of a population generated by this formula.

$$P(i, j) = \sin \left[2\pi \left(\frac{N}{2M} \right) \left(\frac{f_{max}}{f_{min}} \right)^{\left(\frac{1}{M} \right)} \left(\frac{j}{f_s} \right) \right] \quad (1)$$

where $i = 1, \dots, M$, where is the number of individuals in the population as presented in Table 2.0, $j = 1, \dots, N$, where is the number of samples in the waveform as presented in Table 2.0.

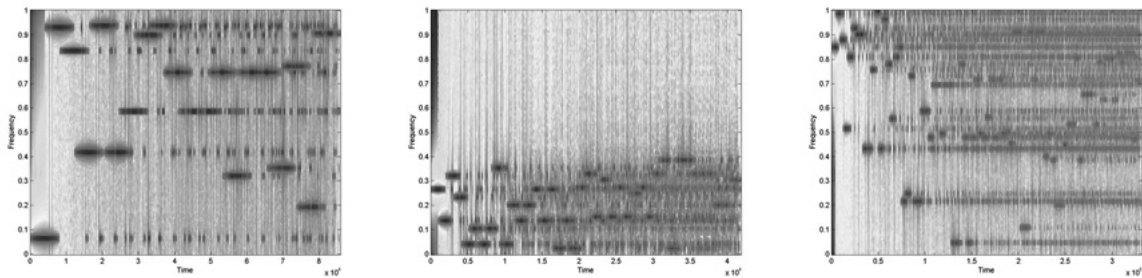


Figure 2: The spectral evolution sonogram of the best individual for three experiments.

4. Conclusion and Further Research

As far as our experimental results had shown, ESSynth is able to generate interesting sound results that dynamically change in time. The evolutionary characteristic of ESSynth methodology allows to explore time domain in a new fashion of sound synthesis, given by the continuous change on the sound segment. So its output can be used to real time composition as well as off time electroacoustic music. These perceptual sound changes are guided by the sound characteristics expressed by the individuals of the target set. If the target set remains unchanged the sonic evolution tends to converge, to a unique sound segment. Nevertheless, ESSynth allows the user to change the Target in real time driving the evolutionary sound to explore other sound regions. Also it is important to stress that we have used Euclidian Metric for its simplicity. Actually there is an infinite number of distinct metrics which could be used. The user/musician must choose which one is best for his/her purpose.

A possibility for further research is to have a population set that is not fixed in size. This implies in having a floating size population, what is closer to the biological reality. In addition we can think of a ripping period for individuals in the population, before they are able to reproduce. This may be also named as *childhood*. During this period the individual would be able to acquire information (learning) from other individuals within the population. Our experimental results shown here point that ESSynth could be a very interesting method for sound synthesis mainly for real time applications. Finally, it worth to mention that aesthetic issues related to sound outputs of ESSynth, although important to a concrete use of it, deserves a deeper investigation which is out of scope of this work.

5. Acknowledgments

This work was supported by São Paulo State Research Foundation (FAPESP) and CAPES (Brazil). A.M.J is grateful to the Computer Music Team of Plymouth University for the hospitality during the making of this work.

References

- Biles, J. (1990). A genetic algorithm for generating jazz solos. In *The Language of Electroacoustic Music*, pages 61–93. Ed. Emmerson.
- Horowitz, D. (1994). Generating rhythms with genetic algorithms. In *Proceedings of the 1994 International Computer Music Conference*, pages 142–143.
- Johnson, C. G. (1999). Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In Wiggins, G. A., editor, *Proceedings of the AISB Workshop on Artificial Intelligence and Musical Creativity*, Edinburgh.
- Manzolli, J., Moroni, A., Von Zuben, F., and Gudwin, R. (1999). An evolutionary approach applied to algorithmic composition. In *Proceedings of VI Brazilian Symposium on Computer Music*, pages 201–210, Rio de Janeiro.
- Moroni, A., Manzolli, J., Von Zuben, F., and Gudwin, R. (2000). Vox populi: An interactive evolutionary system for algorithmic music composition. *Leonardo Music Journal*, 10:49–55.
- Roads, C. (1994). Genetic algorithms as a method for granular synthesis regulation. In *Proceedings of the 1993 International Computer Music Conference*, Edinburgh.

Funchal: a System for Automatic Functional Harmonic Analysis

Ricardo Scholz, Vítor Dantas, Geber Ramalho

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7.851 – 50.732-970 – Recife – PE – Brazil

{reps,vcd2,glr}@cin.ufpe.br

Abstract. *Functional harmonic analysis is an important task in music composition, accompaniment, arrangement and others. However, the solutions are still not satisfactory. The proposed process is divided into two levels: the first one extends one of previous works in the domain to carry out a richer analysis of chord grids and is where the very analysis is performed, and the second one is devoted to correct some conceptual inconsistencies concerning enharmonic spelling of chords. Both levels use an engine to make inferences on some rule bases, which can be easily improved by addition of new rules.*

1. Introduction

Functional harmonic analysis consists of attributing functions to each chord of a given chord grid, considering the context in which the chords are inserted. The main targets criteria for evaluating a system that performs functional harmonic analysis are correctness, extensibility and ability to recognize most of the patterns used.

The previous works studied did not reach some of these criteria. Our proposed process is divided into two levels: analysis and chord names correction. The first extends the work of Pachet [Pachet 1991], reused by Ramalho [Ramalho 1997]. We extended the approach to deal with more complicated structures, such as modal borrowing and secondary dominants and added a level for correcting chord spelling mistakes.

Section 2 will present the state of the art on this subject. Next, section 3 discusses in depth the approach that we used to do the automatic functional harmonic analysis. Section 4 shows the results obtained and conclusions of this work.

2. State of the Art

The perception of harmony in music is a task that is done very naturally by humans, but has been proven to be a complex task to describe algorithmically. The modeling of this task was the target of many psychologists [Krumhansl, 1990] and computer science researchers. For instance, Bharucha have done experiments with connectionist networks to simulate harmonic perceptions [Bharucha, 1991].

Some works assumed basic harmonic structures as part of the input, in form of rule-bases, such as Lerdahl and Jackendoff's generative theory of tonal music [Lerdahl and Jackendoff, 1983] and Narmour's implication/realization model [Narmour, 1990]. Temperley did recently extensions to these works [Temperley, 2001], implemented his

solutions and did extensive tests. Other authors used case-based reasoning to avoid the flaws of assuming wrong predefined structures [Sabater, Arcos, and Mántaras, 1998].

Besides, a number of studies deal with what is known as roman numeral harmonic analysis, that is the way it is taught in music schools. Some authors proposed context-free grammars to realize the analysis iteratively, generating complex harmonic structures [Ulrich, 1977] [Steedman, 1984]. Pachet's approach is different and consists in dividing the analysis in three steps: search for recurrent patterns, overlapping patterns removal and finally unclassified chords are given a function. [Pachet, 1991].

In our work, we used Pachet's division in levels, and we have made a number of extensions to recognize more complex harmonic structures. We also added new a step to correct the names of chords that can be misspelled in the input. Our analysis level deals only with pitch information, without considering the spelling, such as Temperley's analysis did.

3. Automatic Functional Harmonic Analysis

The approach proposed works like growing islands, so it deals with modulation automatically, since neighbor islands are independent and can have different tonalities. In addition to this, the rules proposed deals with complex harmonic structures, as modal borrowing, secondary dominant and composite functions. Some of them are not covered by any of the cited works. The rules that will be shown were identified based on interviews with senior musicians with a good knowledge on functional harmony.

3.1. Chord Pattern Matching

This step consists in searching for recurrent and unambiguous chord sequences. We consider simple patterns sequences of chords with no complex harmonic structure. Unambiguous patterns are chord sequences that cannot have two possible harmonic analysis. Examples of such patterns are II-V, IV-V-I, etc.

We use a rule base coupled with a first order inference engine, called JEOPS [Figueira 2000] that works on forward chaining. Most of the rules in this base were adapted from Ramalho [Ramalho 1997]. The others were created based on the result of a manual analysis of many songs with Jazz and Bossa Nova harmony.

All the rules apply to consecutive chords, considering only the number of steps between these roots. Once a rule is fired on a set of chords, a new pattern is created with the chords' functions. Then, the pattern is inserted in the patterns list. To any set of chords, if more than one rule apply, all will fire. We treat the ambiguous cases in the third step of analysis.

3.2. Overlapping Patterns Removal

After applying the first rule base to match the predefined patterns, the pattern list may contain overlapping patterns. The goal of this analysis step is to eliminate all the overlaps. To reach this goal, the rules of this base eliminate from the knowledge base the patterns considered to have low priority or to be redundant.

The rules were defined from the possible overlaps generated by the first analysis step. In a deeper analysis, we can notice that there are two types of overlaps: total, if a

pattern is contained by another, or partial with a common function, if the last function of the first pattern matches the first function of the second. Only one rule fires on each set of overlapping patterns, according to the rule priority.

3.3. Gaps Classification

Finally, the third rule base has the goal of identifying the functions of the chords that didn't match any pattern. Once a rule has been fired to a chord, no other rule can fire to the same chord. This step is responsible for recognizing the most complex harmonic structures, as modal borrowing, secondary dominants and composed functions.

First, we try to classify the chords according to the harmonic fields of previous and following patterns. If the chord belongs to one of these harmonic fields, it is inserted in the pattern with its respective function in that harmonic field. Otherwise, we try to classify it using a composite function by considering the previous and following chords as roots of an harmonic field and trying to find a function that belongs to one of these harmonic fields and can be applied to the gap. If the rule fires, the gap is inserted in the pattern to which its neighbor chord belongs.

The next attempt is to classify the gap as a modal borrowing of its neighbor patterns. Then, a specific rule was created to classify substitute dominants, since this function is not identified in any other rule. Analogously, when it is not possible to classify the gap as substitute dominant of its neighbor patterns, we try to classify it as a substitute dominant of its neighbor chords.

Pachet [Pachet 1991] and Ramalho [Ramalho 1997] do not consider some of these cases, as modal borrowing and composite functions.

3.4. Chords Names Correction

There is a big amount of song lyrics mixed with chord notation available on the web. The approach was designed to make use of these inputs. However, web songs often have enharmonic spelling mistakes, as in the sequence $IIm^7 V^7 I^{7M}$ that can be found as $Ebm^7 G\sharp^7 Db^{7M}$, while the correct option should be $Ebm^7 Ab^7 Db^{7M}$.

It is a good feature for the analysis level not to distinguish enharmonic notes, such as $G\sharp^7$ and Ab^7 , since this way the functions may be identified even if the chord grid has mistakes. Since the system knows what the correct chord names are, based in the analysis made, it corrects the pitch name of the misspelled chords. All other works cited do not regard to this question.

The rules of this base works as follows: first, we try to use tonalities that had been used before within the same song. Then, if the first attempt is not possible, we use a tonality transition table. Finally, each pattern tonality is corrected locally, based in its last chord.

The use of the tonalities transition table has the goal of making it easier to read and execute the music, avoiding the use of complex tonalities or rough tonality changes. The table specifies what tonality, between all enharmonic options, the following pattern may use, according to the current pattern's tonality. The transition table is based on the fifths cycle and avoids double sharpened and double flattened tonalities.

4. Conclusions

Our system was built aiming to analyze music which harmony is functional. The tests were realized with Jazz and Bossa Nova songs, because of their rich harmony. We used a set of twelve harmonic analysis hand-made by senior musicians for comparison with the automatic analysis. The twelve songs of the test set comprise a total of 810 chords. In our experiments, while the specialist analysis left 3,52% of the chords unclassified, the automatic analysis left 1,76%. However, considering that the specialist analysis is the ideal analysis, 9,16% of the chords were classified by the system with alternative functions. Considering this context, the framework has reached good results with this small set of songs.

We have taken advantage of previous works, mainly Pachet [Pachet, 1991] and Ramalho [Ramalho, 1997], and we have made some significant extensions. More cases are now covered by the pattern matching level, for instance composite functions and modal borrowing. We also added a step for correcting the chord names after the analysis, which did not exist in previous works.

References

- Bharucha, J. J. (1991) "Pitch, harmony, and neural nets: A psychological perspective", Music and Connectionism, The MIT Press.
- Figueira, C. S. F. (2000) "JEOPS: Integração entre Objetos e Regras de Produção em Java", Centro de Informática, UFPE, Brazil.
- Krumhansl, C. L. (1990) "Cognitive foundations of musical pitch". New York, USA, Oxford University Press.
- Lerdahl, F. and Jackendoff, R. (1983) "A Generative Theory of Tonal Music", Cambridge, MA, USA, The MIT Press.
- Narmour, E. (1990) "The Analysis and Cognition of Basic Musical Structures: The Implication-Realization Model", Chicago, USA.
- Pachet, F. (1991) "A Meta-Level Architecture Applied to the Analysis of Jazz Chord Sequences", Institute Blaise Pascal – Laforia, Université Paris VI, France.
- Ramalho, G. L. (1997) "Construction D'un Agent Rationnel Jouant Du Jazz", PhD Thesis. Université Paris VI, France.
- Sabater, J. and Arcos, J. L. and Mántaras, R. L. (1998) "Using Rules to support Case-Based Reasoning for harmonizing melodies", Multimodal Reasoning, AAAI Press.
- Steedman, M. J. (1984) "A Generative Grammar for Jazz Chord Sequences", University of Wariwik and University of Edinburgh, Scotland.
- Temperley, D. (2001) "The Cognition of Basic Musical Structures", Cambridge, MA, USA, The MIT Press.
- Ulrich, J. W. (1977) "The Analysis and Synthesis of Jazz by Computer", Computing and Information Science Department. University of New Mexico, Albuquerque, New Mexico, USA.

Transcrição Automática de Sinais de Áudio Monofônicos

Narciso Trevilatto Junior¹, Jayme Garcia Arnal Barbedo¹, Amauri Lopes¹

¹Departamento de Comunicações - FEEC - UNICAMP
Caixa Postal 6101, CEP: 13.083-970, Campinas - SP - Brasil
{narciso,jgab,amauri}@decom.fee.unicamp.br

Abstract. *This paper describes a method to capture pitches and durations of musical notes. It is applied on monophonic digital signals generated by only one sound source. The method is based on the estimation of the fundamental frequency over time by calculating the signal autocorrelation. Processing techniques are applied to adjust the method to different temporal and spectral features of the signals. The validation of the method is made by using a data bank specially developed to include the widest range of real possibilities the method can face. Finally, advantages and weaknesses are pointed out and suggestions are presented to further improvement of this work.*

Resumo. *Este artigo propõe um método para extração das alturas e durações de notas musicais. A aplicação do método é voltada a sinais gerados por uma única fonte sonora. O método baseia-se na estimação da frequência fundamental ao longo do tempo através do cálculo da autocorrelação do sinal. Técnicas de processamento são aplicadas a fim de sintonizar o método às diferentes características temporais e espectrais apresentadas pelos sinais. A validação do método é realizada usando um banco de dados especialmente desenvolvido para incluir a mais ampla gama de condições reais com as quais o método poderá se deparar. Por fim, pontos fortes e fracos são destacados e sugestões para continuidade do trabalho são apresentadas.*

1. Introdução

Transcrição automática de música é o ato de se extrair uma linguagem simbólica de um sinal de áudio, contendo todas as notas com suas alturas, durações, intensidades, timbres, etc. Apesar desse processo poder ser realizado sem ajuda computacional, uma ferramenta eficaz para esse propósito seria de grande utilidade no meio musical.

Estudos em transcrição automática começaram na década de 70 [Moorer 1975], mas resultados satisfatórios só têm sido alcançados por métodos atuando em sinais de áudio sujeitos a restrições. A proposta deste trabalho é a transcrição automática de sinais monofônicos (uma única fonte sonora). Apesar de relativamente mais simples, uma vez que só há necessidade de se encontrar uma nota por vez, a solução deste problema é essencial para o desenvolvimento de métodos mais sofisticados.

2. Estimativa da Frequência Fundamental

Na transcrição de sinais de áudio monofônicos, é imprescindível a obtenção da altura de cada nota. A essas alturas estão associadas frequências, que normalmente coincidem com a frequência fundamental (f_0) do sinal. O método proposto se baseia no cálculo de f_0 para estimar as alturas das notas.

Existem muitas possibilidades para se detectar a frequência fundamental. Um estudo mais específico é apresentado em [Gerhard 2003], [Gerhard 2002] e [Middleton 2003]. O método utilizado neste trabalho é o da autocorrelação, pois foi considerado a melhor opção para maximizar a relação entre eficiência e complexidade. Esse método é estudado em [Bello, Monti e Sandler 2000] e [Hamidi-Toosi e Laska 2004].

2.1. Método da Autocorrelação

O método da autocorrelação (AC) é uma medida do grau de semelhança entre o sinal e ele mesmo deslocado no tempo. Se o sinal tem uma periodicidade, essa semelhança será maior quando o deslocamento equivaler ao período da onda. A frequência fundamental do sinal é obtida através de

$$f_0 = F_s / n_d , \quad (1)$$

onde F_s é a frequência de amostragem e n_d é o deslocamento do primeiro pico da AC. Para a obtenção das notas musicais, é necessário rastrear f_0 ao longo do tempo. Adota-se o seguinte procedimento: uma janela de 50 ms é deslocada com passos de 25 ms por toda a duração do sinal. Para cada janela aplica-se a AC e calcula-se a f_0 correspondente. Esse tamanho de janela permite a detecção de ondas com período máximo de 25 ms (40 Hz). Existe então um compromisso entre o limite inferior da frequência e a menor duração de uma nota que pode ser detectada.

2.2. Extração do Pico Correspondente à Frequência Fundamental

O comportamento da AC pode apresentar grande variação, dependendo da fonte sonora considerada. A análise da AC pode apresentar problemas devidos, por exemplo, à presença de harmônicas cuja energia relativa é grande se comparada a f_0 . Nesse caso podem surgir outros picos na autocorrelação antes daquele que corresponde a f_0 .

Para detecção do pico correto, a função autocorrelação é expandida no tempo através de sobre-amostragem e subtraída da própria função de autocorrelação original. O efeito deste procedimento, tomando-se como referência um pico com índice de tempo n , será o de eliminar ou atenuar o possível pico com índice de tempo $2n$. O procedimento elimina também o pico localizado no índice de tempo zero.

O máximo dessa função de autocorrelação modificada geralmente corresponde ao primeiro pico (deslocamento de um período). Porém, podem existir picos maiores que o primeiro. O procedimento então é procurar por um novo pico entre o máximo e a origem. Se a amplitude desse novo pico estiver acima de um certo limiar relativo ao máximo, este é assumido como o atual candidato. O procedimento é realizado recursivamente até que não se satisfaça a condição do novo pico ultrapassar o limiar.

3. Estimativa das Alturas (Frequências) e Durações das Notas

Nesta seção são descritos os processamentos necessários para que o programa forneça em sua saída estimativas confiáveis para a frequência e duração de cada nota. Devem ser definidas todas as notas e pausas presentes no sinal, assim como suas alturas e instantes inicial e final.

O resultado do método da autocorrelação é um vetor contendo os valores de f_0 encontrados para todas as janelas. São necessárias algumas manipulações para se extrair desse vetor os parâmetros desejados. Essas manipulações são detalhadas a seguir.

Normalmente, os instrumentos musicais são afinados usando alguma nota como referência. Podem ocorrer situações em que determinado instrumento é afinado usando uma frequência ligeiramente diferente da esperada. Quando isso ocorre, todas as notas do instrumento são deslocadas em frequência e um ajuste deve ser feito para a obtenção das notas corretas. Primeiramente, o vetor das frequências é recalculado de acordo com:

$$m = 12 \log_2 (f_0 / 440) + 69, \quad (2)$$

onde k_n é a nota MIDI correspondente a cada frequência. A seguir, o algoritmo busca por segmentos de 125 ms (quatro janelas) em que as quatro notas k_n associadas têm uma variância menor que um certo limiar. Satisfeita essa condição, a média das quatro notas é calculada. Essas médias são subtraídas dos valores inteiros mais próximos resultando um conjunto de valores entre -0,5 e 0,5. Por fim o vetor de notas é subtraído da média desses valores. Com isso, as notas devem assumir posições mais próximas a valores inteiros, reduzindo a chance de se arredondar valores para notas erradas.

Uma última manipulação é realizada sobre o vetor com o propósito de explicitar os instantes de início e término de cada nota. Isso é alcançado definindo-se uma duração mínima de 125 ms (quatro janelas) para as notas e pausas. A sequência final será então composta por subsequências com quatro ou mais valores iguais. Segmentos com valor “zero” indicam ausência de periodicidade (nenhuma nota executada).

O método para detecção do início e final de cada nota segue os seguintes passos: 1) procura-se uma sequência de quatro pontos consecutivos com o mesmo valor, o primeiro dos quais definindo o início de uma nota; 2) o final da nota é dado pelo último ponto após o qual surgem quatro pontos consecutivos com valores diferentes do seu; 3) se o algoritmo determinar que a próxima nota se inicia até quatro pontos após o final da anterior, os pontos de transição são considerados pertencentes a essa nova nota, caso contrário o valor nulo é associado a esses pontos, caracterizando a ausência de nota.

4. Resultados

O banco de dados usado é composto por 100 arquivos de 15 segundos, amostrados a 44,1 kHz e quantizados com 16 bits. Foram avaliados sinais provenientes de cordas, sopros, piano e voz masculina e feminina. Os resultados obtidos são apresentados na Tabela 1.

Tabela 1. Resultado das simulações.

Fonte Sonora	Total de Notas	Notas Detectadas com Sucesso	Notas Falsas Detectadas	Índice I
Cordas	507	484	45	0,87
Sopros	1805	1712	93	0,90
Voz	492	463	69	0,80
Total	2804	2659	224	0,87

Obs.: $I = (NotasCorretas - NotasFalsas) / TotalNotas$

Na tabela, quanto mais próximo de 1 o valor de I , mais preciso o método. Como se pode observar, a menor eficiência ocorreu na análise de sinais de voz. Dentre os motivos encontrados, pode-se citar: a) em sinais de voz a transição entre notas é frequentemente realizada de um modo suave (*legato*), tornando mais frequente a ocorrência de erros nessas regiões; b) este tipo de sinal possui sons fricativos mais longos do que nos instrumentos, tornando possível a detecção de uma nota falsa nesses

intervalos de tempo; c) estes sinais possuem uma maior quantidade de desafinações, associadas à liberdade de interpretação do cantor.

A saída do programa apresenta notas discretizadas. Por esse motivo, efeitos relativos a variações contínuas de frequência (*glissando*, *vibrato*) não são apontados. A detecção do pico correspondente a f_0 é outro ponto que deverá ser aperfeiçoado, especialmente na análise de sinais com harmônicas de grande intensidade. Apesar disso, o método apresentou um desempenho próximo daqueles encontrados na literatura. Além disso, é importante frisar que as técnicas aqui apresentadas são apenas a primeira etapa na implementação de um método de transcrição de sinais de música confiável e robusto. Novas técnicas de rejeição de harmônicas vêm sendo testadas com sucesso. Estas técnicas, em conjunto com a incorporação de lógicas baseadas em teoria musical, deverão melhorar significativamente os resultados alcançados.

5. Conclusão

Neste artigo foi proposto um método para transcrição automática de sinais de áudio monofônicos. O algoritmo faz uso do método da autocorrelação para detecção de f_0 . Uma quantização da f_0 de cada nota é realizada através de sua transformação para a escala MIDI. Este procedimento aumenta a precisão das estimativas, embora faça com que efeitos como *vibrato* ou *glissando* sejam ignorados. As durações das notas foram estimadas tendo como base os instantes em que o sinal se torna ou deixa de ser periódico.

O método obteve o desempenho esperado para esta primeira etapa de desenvolvimento do transcritor automático de música. Os resultados alcançados mostram boa confiabilidade do sistema, principalmente nos casos em que os instrumentos geram notas com valores de frequência discretos. Porém, existem pontos que precisam ser mais bem explorados. Dentre as medidas que vêm sendo adotadas para melhorar a precisão do método, pode-se citar o aperfeiçoamento do algoritmo de detecção do pico da autocorrelação de modo a evitar a detecção de picos referentes a harmônicas da frequência fundamental. Adicionalmente, versões futuras do algoritmo deverão incorporar a visualização de vibratos e glissandos, fazendo com que o conjunto de informações fornecidas se torne mais completo. Por fim, as notas deverão ter suas probabilidades de ocorrência alteradas através da implementação de uma lógica baseada em teoria musical.

6. Referências Bibliográficas

- Bello, J. P., Monti, G. and Sandler, M. (2000) "Techniques for Automatic Music Transcription", Int. Symp. on Music Inf. Retrieval (ISMIR), Plymouth, Mass., USA
- Gerhard, D. (2003) "Pitch Extraction and Fundamental Frequency: History and Current Techniques", University of Regina Technical Report TR-CS 2003-06
- Gerhard, D. (2002) "Computer Music Analysis", Simon Fraser University, School of Computing Science, Technical Report CMPT TR 97-13, Burnaby, BC, July.
- Hamidi-Toosi, F. e Laska, J. (2004) "Pitch Detection for the Next Millenium and Beyond" www.ews.uiuc.edu/~laska/ece320/files/Report.pdf
- Middleton, G. (2003) "Pitch Detection Algorithms", <http://cnx.rice.edu/content/m11714>
- Moorer, J. (1975) "On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer". Ph.D. thesis. Department of Music, Stanford University.

Validating the Lattice Boltzmann Method for the Characterization of Impedances in Pipes

Andrey R. da Silva*, Gary Scavone

¹Computational Acoustic Modeling Laboratory – Music Technology Area - McGill University
555 Sherbrooke Street West – H3A 1E3 Montreal, QC

andrey.dasilva@mail.mcgill.ca, gary@music.mcgill.ca

Abstract. *This paper proposes the application of the Lattice-Boltzmann method (LBM) for the characterization of the acoustical impedance and sound radiation properties of wind instruments. The validation of the method is achieved by simulating the radiation of an open unflanged pipe with LBM and comparing the impedance related data and radiation results with those obtained through the exact solution derived by Levine and Schwinger for the same model.*

1. Introduction

One important issue in sound synthesis by physical modeling is to create simplified representations of the dynamic behavior of a musical instrument, trying to take into account only the parameters that play a significant role on the auditory perception so that the model can be efficiently reproduced in real time.

In the case of the wind instruments, these parameters can be well described in terms of acoustic impedances. The knowledge of the instrument impedances allows solutions to be found for the sound fields inside the instrument bore and the sound power transmitted into the surrounding air. In other words, it is essential to describe the interaction between the instrument and other systems (lips, vocal tract, environment, etc.) and also to represent the way the instrument radiates sound as a function of frequency and space. For the purpose of sound synthesis, the impedance information can subsequently be represented by digital filters as described by [Scavone, 1999]. The determination of the impedance is not a straightforward task, however. The existing analytical expressions are restricted to systems with simple geometries. In the other hand, experimental procedures are usually frequency limited and have other geometrical issues. Furthermore, it relies on the existence of a real prototype. Alternatively, the acoustic impedance of more complex systems can be derived numerically.

This paper presents the validation of Lattice Boltzmann Method (LBM) for the numerical determination of the acoustic impedance of a simple pipe. The validation is done by comparing the numerical results obtained from a Lattice Boltzmann model of an open pipe with the exact theoretical results for the radiation and impedance of an unflanged open pipe derived by [Levine and Schwinger, 1948].

2. A Brief description of the Lattice Boltzmann Method

The fundamental theory of the LBM has been thoroughly described in many publications, e.g., [Succi, 2001] and [Gladrow, 2000]. Although relatively new, LBM has

*Supported by CAPES.

been extensively studied in many disciplines. However, its use in acoustics has been explored by only a limited number of authors. [Buick et al., 1998] has studied the propagation of waves considering dissipative effects for different boundary conditions and [Buick et al., 2000] have applied it on the simulation of shock wave fronts. [Skordos, 1995] simulated the mechanism of sound generation of an organ pipe by using a 3D LB model. [Lallemand and Luo, 2003] studied the propagation of waves by using different lattice models.

The main idea of the method is to create a discretized version the Boltzmann equation, which describes the fluid at the microdynamic level by a distribution function. Once discretized, the equation can be approximated numerically. The discretization is implemented by the use of a lattice grid where each lattice represents a limited number of velocity states that can be taken by a particle in the fluid. At each discrete time interval all the particles of a domain are found at lattice sites where they collide and exchange momentum according to their previous values. The LBM code used in our simulation was implemented in Matlab and uses a two dimensional squared lattice model known as D2Q9 where each particle in a fluid can assume 9 different velocity directions.

3. Method Validation

The validation of the method was based on the comparison of results obtained by the LB simulation and the theoretical results obtained by [Levine and Schwinger, 1948] for an open-end unflanged pipe, which had also been experimentally validated by [Ando, 1968].

3.1. Model Setup

The model was built with a grid of 500 X 500 lattices, assuming a discretization $N = 1000$ lattices per meter. The space step between neighbor grid points, δx , was, therefore, equal to 1 mm. The criteria proposed by [Wilde, 2004], which suggests that the highest wavelength being analyzed should be resolved with at least 12 lattices was adopted, so that a numerical error for the phase velocity of less than to 1 per cent could be guaranteed for a frequency resolution up to 28 kHz.

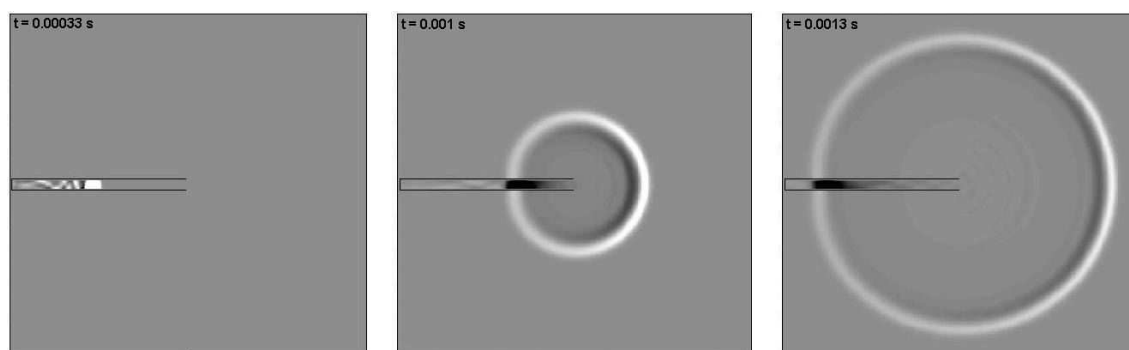


Figure 1: Sound radiation from the pipe model at different time steps

An axisymmetric model of the pipe was simulated by implementing a bounce-back scheme in which the walls of the pipe were considered to be dry lattices. This means that, during the evolution process, the particles propagating in the direction of the walls would be reflected back in the opposite direction conserving the same momentum. The pipe was modeled to be 25 cm long with an inner diameter of 2.5 cm. The lattices at the outer boundaries of the grid were set to have an anechoic behavior so that all the acoustic energy reaching them would be totally absorbed instead of being reflected back into the

grid domain. The pipe was excited with an impulse at the closed end. The impulse was generated by applying a density disturbance equal to 0.1 per cent higher than the fluid density at $t = 0s$. It was expected that the density disturbance would propagate as a wavefront along the pipe until it reached the pipe open end where it should be partially reflected back into the pipe and partially radiated to the outside. Figure 1 depicts the axisymmetric model of the pipe within the lattice grid with anechoic boundaries.

Two probes were placed at the open end of the pipe model to acquire pressure, $p(t)$, and velocity, $v(t)$, at every time step in order to determine the output impedance of the pipe and, consequently, the reflectance. Moreover, 38 probes were displaced in a semi-circle centered at the pipe's open end with a radius of 12.5 cm to acquire the radiated pressure.

3.2. Impedance Results

The complex values of the particle velocity, $V(\omega)$, and pressure, $P(\omega)$, in the frequency domain were obtained by a fast Fourier transform of the time-domain signals acquired at the pipe's open end. These results were used to derive the complex impedance, $Z(\omega)$, and the reflectance magnitude, $R(\omega)$. Figures 2 depicts the comparison between the results for the normalized impedance and reflectance acquired numerically and the theoretical results of Levine and Schwinger plotted as a function of ka , where a is the radius of the pipe, $k = \frac{\omega}{c}$ is the wavenumber, where ω is the angular frequency, and c the speed of sound in the fluid.

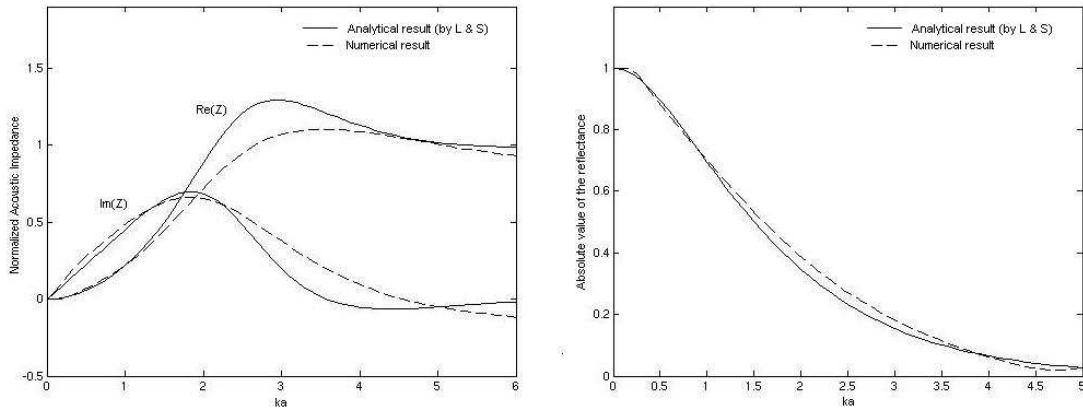


Figure 2: Normalized impedance and reflectance as functions of ka

The results for the impedance (Figure 2 on the left) agree well with a discrepancy of less than 5 per cent for values of $ka < 2$, which corresponds to a linear frequency equal to 8.7 kHz for a pipe of 1.25 cm radius at an air temperature of 20 °C and pressure equal to 1 atmosphere. The plots of the absolute value of the reflectance (Figure 2 on the right) show a good agreement for the whole range up to $ka = 5$. The maximum discrepancy can be found around $ka = 2.3$, which corresponds to a value of 4 per cent higher than the theoretical result.

3.3. Radiation Results

The results for the sound radiation were obtained from the data acquired by the probes placed at the semi-circumference around the pipe's open end and compared with the theoretical results. Figure 3 depicts the polar plots of the sound radiation in the form of power gain for three different values of ka . A good agreement can be seen with the results predicted theoretically, especially when $ka > 2$. The discrepancy for lower values of ka may

be attributed to the existence of an acoustic near field between the pipe's open end and the acquiring probes.

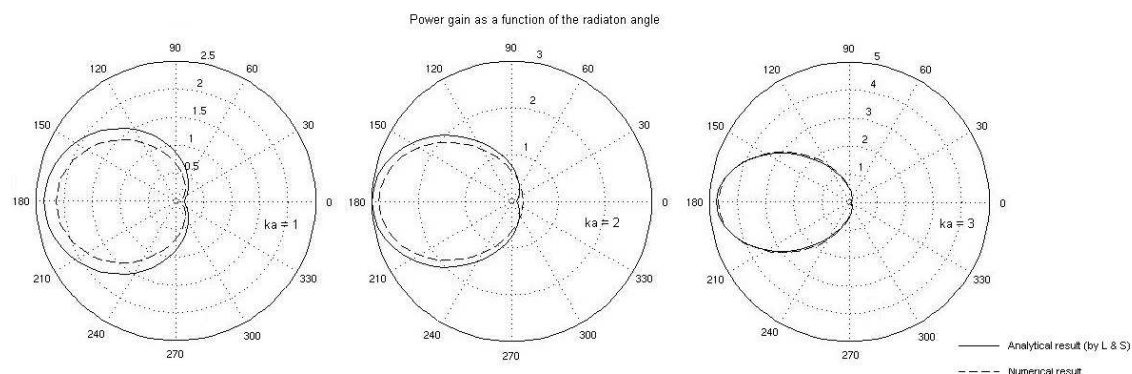


Figure 3: Radiation power gain as a function of angle for different values of ka

4. Conclusions

The motivation of this study was to validate LBM as a tool for determining impedance and radiation properties of wind instruments. The results acquired by the LBM have shown that it is a potential method to be applied on the determination of acoustic impedance and radiation properties of complex geometries. The most significant advantage over the classical continuum methods (Finite Elements, Finite Volumes, etc.) consists on its simplicity of implementation and facility of simulating wave propagation within complicated boundary conditions in the time-domain.

References

- Ando, Y. (1968). Experimental Study of the Pressure Directivity and the Acoustic Centre of the Circular Pipe Horn Loud Speaker. *Acustica*, 20(1):366–369.
- Buick, J. M., Buckley, C. L., Greated, C. A., and Gilbert, J. (2000). Lattice Boltzman BGK Simulation of Nonlinear Sound Waves. *J. Phys. A*, 33:3917–3928.
- Buick, J. M., Greated, C. A., and Campbell, D. M. (1998). Lattice BGK Simulation of Sound Waves. *Europhysics Letters*, 43:235–240.
- Gladrow, D. W. (2000). *Lattice-gas Cellular Automata and Lattice-Boltzmann Models*. Springer-Verlag Berlin, 2nd edition.
- Lallemand, P. and Luo, L. S. (2003). Theory of the Lattice-Boltzmann Method: Acoustic and Thermal Properties. *Europhysics Letters*, 68(3):150–175.
- Levine, H. and Schwinger, J. (1948). On the Radiation of Sound from an Unflanged Circular Pipe. *Physical Review*, 73(4):383–406.
- Scavone, G. P. (1999). Modeling Wind Instrument Sound Radiation using Digital Waveguides. *Proceedings of ICMC 1999*, pages 355–358.
- Skordos, P. (1995). Modeling Flue Organ Pipes: Subsonic Flow, Lattice-Boltzmann and Parallel Distributed Computers, PhD Thesis, MIT.
- Succi, S. (2001). *The Lattice Boltzmann Equation for Fluidynamics and Beyond*. Oxford University Press, 1st edition.
- Wilde, A. (2004). Application of the Lattice-Boltzmann Method in Flow Acoustics. *SWING Workshop*, Aachen.

Um Aplicativo em Max/MSP para Geração de Material Musical Utilizando Técnicas de Síntese Granular

Rafael de Oliveira, Luciano Vargas Flores, Eloi Fernando Fritsch

CME – Centro de Música Eletrônica
Instituto de Artes – Universidade Federal do Rio Grande do Sul (UFRGS)
Rua Senhor dos Passos, 248, 6º andar
90.020-180 – Porto Alegre – RS – Brasil

oliveira_cme@yahoo.com.br, lflores@cpovo.net, 00099197@ufrgs.br

Abstract. *This short paper presents GRANU, an application written in Max/MSP to ease the quick production of musical material for composers who want to make use of the granular synthesis and manipulation of audio samples. GRANU reaches this by offering several options for the control of the various parameters of granular synthesis, in a clear and simple user interface. The paper also includes a report and some impressions of a compositional study conducted as a means to validate this work.*

Resumo. *Este artigo apresenta um aplicativo em Max/MSP, GRANU, destinado a facilitar a geração rápida de material musical para o compositor interessado em empregar síntese e manipulação granulares de amostras de áudio. GRANU atinge este objetivo por oferecer diversas opções para o controle dos vários parâmetros envolvidos na síntese granular, em uma interface simples e clara. O artigo inclui ainda o relato e as impressões resultantes de um estudo composicional realizado para validação do trabalho.*

1. Introdução

Este trabalho faz parte da pesquisa “Laboratório de Música Eletroacústica Experimental”, cujos objetivos incluem elaborar um catálogo de sons processados por computador para utilização na composição de música eletroacústica experimental. Partindo deste objetivo específico propôs-se a criação de um conjunto de aplicativos em Max/MSP [Cycling '74 2005] para processar amostras de áudio e gerar material musical. Foi realizado um estudo inicial de objetos disponíveis publicamente que incrementassem o ferramental do ambiente de programação Max/MSP, dos quais foram selecionadas algumas coleções de objetos para síntese granular, FFT (transformada rápida de Fourier) e modelagem física. A partir destas foram desenvolvidos três programas para estender a paleta de opções e facilitar o trabalho do compositor na elaboração de materiais musicais: GRANU, ApFFT e GMM [Oliveira e Fritsch 2004].

Este artigo descreve e discute em detalhes um destes aplicativos, GRANU, destinado à síntese e à manipulação granulares de amostras de áudio. Nossa principal preocupação ao desenvolvê-lo insere-se na discussão de como facilitar, para o músico, o controle do processo de síntese granular e de seu resultado final. Na síntese granular, as características de cada grão, individualmente, podem ser bastante controláveis, mas prever como será o resultado da combinação de milhares torna-se muito difícil. Muitas

alternativas já foram sugeridas: do controle a partir de autômatos celulares [Miranda 1998] até a descrição gráfica de eventos para controlar a evolução dos grãos e sua quantidade [Wenger e Spiegel 2005]. Nosso trabalho propõe o controle direto, pelo usuário, dos diversos parâmetros envolvidos na síntese granular, através de uma interface gráfica que simplifique o acesso a esses controles e apóie sua compreensão e a segurança em seu uso. Espera-se que, apoiando a segurança do músico no controle total do processo de síntese, ele adquira igualmente, com a prática, a capacidade de controlar e prever o resultado final, chegando mais facilmente ao resultado sonoro que procura.

2. Desenvolvimento e Descrição do Aplicativo

A coleção de objetos selecionada para a criação do GRANU foi a Granular ToolKit 1.02 / 1.03 / 1.10, desenvolvida por Nathan Wolek, da School of Music at Northwestern University [Wolek 2002]. Consiste de seis *externals* (objetos) e alguns *abstractions* (pré-programação) para síntese granular a partir de uma amostra de áudio. Nesta coleção o foco das atenções foram os *externals*, já que eles permitem maior liberdade de emprego e assim facilitam a criação de programas inéditos. Iniciamos o desenvolvimento com o foco em dois destes objetos: *grain.phase~* e *phasor.shift~*.

O *grain.phase~* faz com que sejam disparados grãos seguindo as mudanças de fase de um objeto *phasor~*, um oscilador de fase, e assim a frequência dos grãos é dada pela frequência do *phasor~*. O *grain.phase~* trabalha coletando o grão que irá utilizar de uma amostra sonora previamente carregada em um buffer. Devem ser indicados o ponto de coleta do grão na amostra original e o envelope de amplitude do grão em um segundo buffer. Uma característica interessante deste objeto é o fato de que o tamanho do grão também é determinado pela frequência do *phasor~*: a cada mudança de fase deste, o *grain.phase~* inicia a execução do grão a partir do instante já indicado da amostra original. O grão continua sendo executado até reiniciar na próxima mudança de fase do *phasor~*, o que, portanto, acaba produzindo grãos de duração inversa à frequência deste último. A cada grão é aplicado o envelope de amplitude definido no segundo buffer.

Para controlar a frequência de geração de grãos pelo *phasor~* optamos por utilizar envelopes, que indicam a evolução desta frequência no tempo para cada um dos dez geradores do GRANU. Estes envelopes foram implementados na interface na forma gráfica, facilitando sua definição pelo músico. O valor máximo para o tempo nos envelopes é igual à duração da amostra original. O valor máximo da frequência é de 100Hz e o mínimo é de 5Hz, o que gera grãos entre 10ms e 200ms de duração. O controle dos pontos de retirada dos grãos para cada gerador é feito usando-se dez barras horizontais, dispostas na interface logo abaixo da visualização da amostra original, dando ao usuário uma noção bastante clara desses instantes em relação à amostra.

No *grain.phase~* tem-se ainda controle sobre a “afinação” do grão (o *pitch*, ou altura, no sentido de velocidade de reprodução). Os valores da afinação são multiplicadores do incremento de amostragem dos buffers de grão (“frequência de leitura” do buffer do grão). A afinação no GRANU também é controlada usando-se dez barras, estas verticais, uma para cada gerador de grãos, com os valores 2 para máximo, 0 para mínimo e 1 para o valor central (mesma frequência da amostra original).

O objeto *phasor.shift~* mostrou-se útil na implementação de um controle de frequência único para todos os geradores, proporcionando mais esta funcionalidade para

o usuário. O *phasor.shift~* recebe uma frequência e a distribui entre objetos *phasor~* com pequenas defasagens. No GRANU foi empregado para defasar os dez geradores de grãos, já que todos trabalham na mesma frequência fixa quando esta função está ativada.

Quando estamos trabalhando com vários objetos *phasor~* em paralelo temos o problema de que todos estarão sempre na mesma fase, fazendo com o que os grãos gerados iniciem todos nos mesmos instantes. Isto pode causar intervalos perceptíveis entre grãos quando trabalhando em frequências muito baixas, e provocar a ilusão de que se tem menos geradores de grãos do que na realidade estão sendo usados, desvirtuando a intenção original para o papel de cada gerador de grãos. Com a defasagem dos geradores pelo *phasor.shift~* estes problemas desaparecem.

Uma terceira opção implementada é o controle da geração dos grãos a partir de uma análise da amostra original. A amostra é analisada em sua formação espectral através do objeto *fffb~*, nativo da extensão MSP, que realiza FFT em dez parciais e suas amplitudes. Estes resultados são analisados por um objeto *peakamp~* que determina os valores das amplitudes das parciais (de 0. a 1.). Estes valores são multiplicados por 100 para serem compatíveis com valores de frequências dos geradores de grão. Com este processo conseguimos um controle “automático” da frequência dos geradores, dependente da análise espectral da amostra original, incluindo seu comportamento no decorrer do tempo, o que poderia produzir resultados musicalmente interessantes.

As três formas de controle de produção dos grãos desenvolvidas neste trabalho estão disponíveis no GRANU através da seleção em um menu *drop-down*, cujas alternativas são “Individual”, “Geral” e “Pela amplitude”. Outros controles disponíveis na interface com o usuário do GRANU são o envelope de amplitude da amostra de saída e o volume geral (Figura 1).

3. Validação: Um Estudo Composicional

Para validar a utilização do GRANU no contexto dos objetivos iniciais do projeto realizamos experimentos de manipulação de uma única amostra (uma gravação de fala feminina) visando obter um grande número de sons diversos resultantes. Além disso prosseguimos para a condução de um primeiro estudo composicional utilizando estes sons experimentais, de modo a determinar o valor musical dos materiais gerados.

Iniciou-se o processo com um envelope final de duração igual à da amostra original. Manipulou-se a afinação para criar “referências” de alturas, bem como as frequências dos grãos através dos envelopes, gerando amostras de baixa frequência e amostras com altas frequências para criar contrastes. Em seguida procurou-se manipular o envelope final da amostra para tentar conseguir sons curtos (pulsos) dos quais pudessem derivar possibilidades rítmicas. Experimentou-se ainda com o controle geral de frequência, de forma a criar timbres diversos que depois foram novamente carregados no GRANU e reprocessados. Também utilizou-se o controle automático por análise, mas os resultados não se mostraram tão interessantes quanto o esperado para esta função, talvez porque a intencional falta de controle sobre este efeito acaba tornando o processo “aleatório demais”, podendo dificultar a geração de um bom material.

Foi produzido um catálogo de cem amostras diversas para a composição. Buscou-se criar um estudo de pelo menos 2 min., com estrutura formal simples (A B) para testar os contrastes de materiais de forma clara. A parte A consiste de pulsos e sons

curtos (trabalho mais rítmico) em oposição à parte B, que se utiliza de materiais longos (trabalho de texturas e modificação timbrística). O resultado musical foi satisfatório e os materiais gerados apresentaram uma imensa variedade, mostrando o grande potencial do programa como ferramenta para o compositor gerar seu catálogo de sons. O resultado do estudo pode ser conferido no site www.musicaeletronica.ufrgs.br/cme.

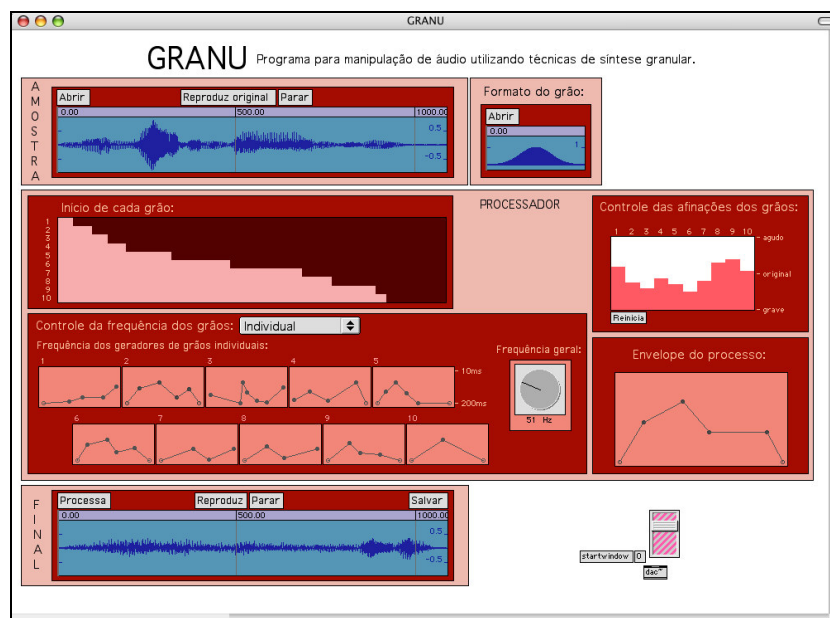


Figura 1. Interface com o usuário do GRANU.

4. Considerações Finais

A partir dos resultados obtidos com a criação do catálogo experimental e com o estudo composicional concluímos que o GRANU atinge os objetivos para os quais foi projetado, tanto em termos musicais quanto como facilitador do processo de geração de materiais musicais. Os programas desenvolvidos nesta pesquisa estarão disponíveis a alunos, pesquisadores e comunidade através do Centro de Música Eletrônica da UFRGS.

Referências

- Cycling '74 (2005) "Max/MSP for Mac and Windows". www.cycling74.com/products/maxmsp.html, jun. 2005.
- Miranda, E. R. (1998) "Computer Sound Synthesis for the Electronic Musician". Oxford: Focal Press.
- Oliveira, R. de; Fritsch E. F. (2004) Três Programas para Laboratório de Música Eletroacústica Experimental: Estudo e Desenvolvimento. In: Feira de Iniciação Científica da UFRGS, 13., 2004, Porto Alegre. "Livro de Resumos...". p. 933.
- Wenger, E.; Spiegel, E. (2005) "MetaSynth 4.0 User Guide & Reference". Redwood City, CA: U&I Software LLC. www.uisoftware.com/MetaSynth, jun. 2005.
- Wolek, N. (2002) "A Granular Toolkit for Cycling74's Max/MSP". In: SEAMUS 2002 National Conference, 2002, Iowa City. www.lowkeydigitalstudio.com/nathanwolek.

Rythmus Environment: um Ambiente para Construção de Ferramentas Educacionais de Instrumentos de Percussão

Suzana Mesquita de Borba Maranhão¹, Érika Pessoa Araújo²,
Matheus Cabral de Araújo Gois¹, Virgínia Barbosa³, Geber Lisboa Ramalho⁴

¹Departamento de Informática (PUC-Rio)
Caixa Postal 38.097 – 22453-900 – Rio de Janeiro – RJ – Brazil

²Programa de Engenharia de Sistemas e Computação (COPPE/UFRJ)
Caixa Postal 68511 – 21941-972 – Rio de Janeiro – RJ – Brazil

³Laboratório de Etnomusicologia – Universidade Federal do Rio de Janeiro (UFRJ)
Rua do Passeio 98 – sala 28 – Centro – 20021-090 – Rio de Janeiro – RJ – Brazil

⁴Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE – Brazil

{suzana,mc}@inf.puc-rio.br, epa@cos.ufrj.br, virgobarbosa@hotmail.com,
glr@cin.ufpe.br

Abstract. *This paper describes a framework that helps the development of new educational tools to percussive instruments. The framework was validated through an instantiation about the alfaia instrument used in Maracatu context. This work is original and it uses Brazilian culture elements.*

Resumo. *Este artigo descreve um framework para o desenvolvimento de novas ferramentas educacionais para instrumentos de percussão. O framework foi validado através de uma instanciação sobre a alfaia no contexto do Maracatu. O trabalho é original e utiliza elementos da cultura brasileira.*

1. Introdução

Embora existam várias aplicações educacionais para o ensino de instrumentos musicais, poucas delas focam em instrumentos de percussão. O *Rythmus Environment* agrupa um conjunto de recursos para facilitar a construção de aplicações com propósito de ensinar instrumentos de percussão tocados com uso de baquetas, como o caixa e o surdo.

Os recursos do *Rythmus Environment* foram utilizados durante a construção da ferramenta *Rythmus Maracatu*. O *Rythmus Maracatu* objetiva ensinar a tocar alfaia no contexto do Maracatu de Baque Virado. O Maracatu¹ foi escolhido porque é um ritmo nacional com popularidade crescente em que alguns autores do artigo possuem experiência musical. Além disso, a transmissão do conhecimento musical do Maracatu muitas vezes ocorre pelo processo de observação e repetição. Barbosa [Barbosa, 2005] realiza uma excelente discussão sobre o processo de aprendizagem de Maracatu. Dentre

¹ Existem outras manifestações de Maracatu, como o Maracatu de Baque Solto. No entanto, este artigo irá sempre se referir ao Maracatu de Baque Virado.

os instrumentos tradicionais de Maracatu, a alfaia foi escolhida por ser tocada com as duas mãos utilizando baquetas, por não atingir, em geral, velocidade de execução das mãos excessivamente alta e por normalmente seguir o ritmo cujo andamento é estabelecido por um outro instrumento, no caso o caixa.

O Maracatu de Baque Virado é uma manifestação popular do carnaval de Pernambuco originado a partir das tradições dos escravos africanos no século XVII. Além de ser um ritmo, essa manifestação também envolve danças, roupas, canções, fatores históricos, religiosos e envolvimento da comunidade de origem do grupo. Existem vários grupos de Maracatu em Pernambuco, como Leão Coroadado e outros no Brasil, como o Rio Maracatu, e no mundo, como o Maracatu Stern der Elbe². O ritmo também está presente no repertório de compositores populares e eruditos, como Gilberto Gil, Lenine e Guerra-Peixe [Guerra-Peixe, 1980].

Nenhum trabalho com propósito de criar um ambiente para facilitar a construção de novas ferramentas de ensino de instrumentos de percussão foi encontrado na literatura. O que existem são APIs para manipular fluxos de áudio, como o Java Sound [Sound, 2005], que foi, inclusive, utilizada no desenvolvimento do *Rythmus Environment*. Existem algumas ferramentas que se propõem a ensinar instrumentos de percussão, principalmente bateria. Como exemplo, pode-se mencionar o *D'Accord Drums Player* e o *Mr. Drumstix Music Studio Lite*. Alguns jogos também podem transmitir algum conhecimento nessa área como o *Donkey Konga*³, que faz uso do instrumento bongô. Ainda assim, o *Rythmus Maracatu* também é original, uma vez que não foi encontrada outra aplicação para ensinar a tocar alfaia e nem nenhum outro instrumento no contexto de Maracatu.

As próximas seções detalham o *Rythmus Environment*, o *Rythmus Maracatu*, além das contribuições obtidas e trabalhos futuros.

2. Rythmus Environment

O *Rythmus Environment* é um ambiente desenvolvido para facilitar a construção de aplicações educativas de instrumentos de percussão. As aplicações construídas utilizando o referido ambiente podem facilmente prover funcionalidades de apresentação de ritmos e reprodução desses pelo usuário, incluindo a possibilidade de realizar avaliação do desempenho do usuário ao efetuar as reproduções.

O *Rythmus Environment* utiliza como entrada dados de áudio e eventos de sincronismo. Os dados de áudio são do instrumento a ser ensinado e outros, que podem ser utilizados como acompanhamento para enriquecer o ambiente musical em que o ensino se desenvolverá. Exemplos de eventos de sincronismo são aqueles que indicam as ações que as mãos devem realizar no instrumento a ser ensinado para gerar um som específico. O ambiente possui funções para manipular o áudio original de modo a prover funcionalidades como *play*, *stop*, mudança de volume e andamento, reprodução de um som específico e gravação de ações do usuário. Para realizar a avaliação do que foi

² Respectivamente disponível em: <http://www.leaocoroadado.org.br/>, <http://www.riomaracatu.com.br/> e <http://www.maracatu.de/>.

³ Respectivamente disponível em: <http://www.daccordmusic.com/eng/>, <http://www.howlingdog.com/> e <http://www.donkeykonga.com/>.

tocado, algumas métricas de desempenho do usuário foram definidas e são calculadas através da comparação entre o áudio original e o executado pelo usuário. Por fim, é possível acompanhar o desenvolvimento de um usuário entre diferentes execuções persistindo suas informações de desempenho.

O *Rythmus Environment* foi desenvolvido utilizando linguagem de programação Java. Suas principais funções foram implementadas em quatro subsistemas: *Leitor*, *Seqüenciador*, *Avaliador* e *Usuário*. O *Leitor* recebe os dados de entrada lendo um arquivo MIDI [Young, 1996] e disponibiliza os dados de cada instrumento e eventos de sincronismo. O *Seqüenciador* provê funcionalidades de manipulação do fluxo de áudio. O *Leitor* e *Seqüenciador* utilizam a API Java Sound. O *Avaliador* calcula o desempenho do usuário e o subsistema *Usuário* persiste seu histórico de execuções. Todos esses subsistemas são encapsulados pelo subsistema *Fachada*, que provê um acesso único a todas as funcionalidades da interface gráfica (GUI). O subsistema *GUI* contém classes para apresentar e sincronizar mídias sonoras e gráficas em um Java Applet. No entanto, parte das classes desse subsistema devem ser definidas pela aplicação a ser instanciada.

3. Rythmus Maracatu

Rythmus Maracatu (ver Figura 1) é um protótipo de uma ferramenta para ensinar a tocar alfaia de Maracatu. A ferramenta permite exibir exemplos de execuções de Maracatu (que chamaremos de toque) através de um áudio e de uma seqüência de imagens e que um usuário possa tentar reproduzir esses toques utilizando o teclado do computador.

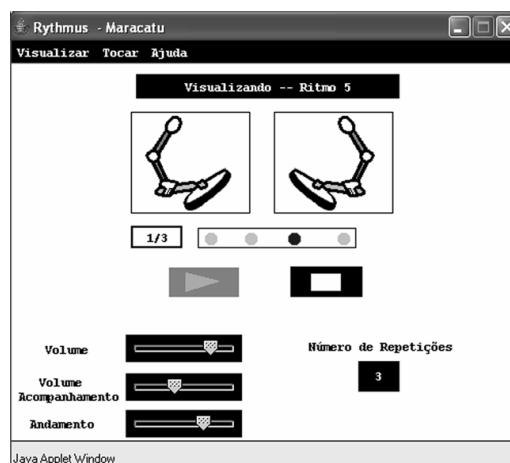


Figura 1. *Rythmus Maracatu*.

Além da alfaia, o protótipo pode reproduzir o som de um instrumento de acompanhamento para auxiliar na marcação do tempo, no caso o caixa, uma vez que, como anteriormente comentado, esse instrumento normalmente dita o andamento do Maracatu. Utilizando a ferramenta, o usuário escolhe se deseja visualizar ou executar um determinado toque, ajusta o volume da alfaia e do caixa, o andamento e o número de repetições do exercício e aperta o botão *play*. Ao total, existem oito toques diferentes, classificados em três níveis de dificuldade. Durante a realização de cada exercício, além da apresentação audiovisual do toque, também existe a apresentação de um metrônomo, para marcar as quatro divisões de cada compasso, e do total de repetições já realizada. Quando o usuário termina de tocar um exercício, uma mensagem é exibida com a avaliação do seu desempenho.

4. Contribuições e Trabalhos Futuros

Este trabalho propõe o ambiente *Rythmus Environment* para facilitar a construção de ferramentas educacionais de instrumentos de percussão. A uso da linguagem de programação Java na sua implementação permite gerar aplicações disponíveis na Internet sem necessidade de instalação. O ambiente foi inicialmente validado através do desenvolvimento do *Rythmus Maracatu*. O protótipo pode ser encontrado no seguinte endereço: <http://www.telemidia.puc-rio.br/~smbm/maracatu/>.

Dois requisitos foram fortemente considerados durante o desenvolvimento desse trabalho. A usabilidade, primeiro deles, foi delegada às instâncias, uma vez que seria muito difícil modelar interfaces de todos os possíveis instrumentos. Uma discussão sobre a usabilidade do *Rythmus Maracatu* foi realizada por Maranhão et al. [Maranhão, 2003]. O segundo requisito foi o realismo, que é fortemente afetado pelas limitações de se ensinar um instrumento pelo computador. É conhecido que a aprendizagem de um instrumento envolve treinamento de vários músculos do corpo [Barbosa, 2005]. No entanto, o computador pode ser utilizado como uma ferramenta auxiliar de ensino. Outros requisitos relacionados ao realismo foram tratados pelo trabalho como a similaridade entre o som produzido e o do instrumento original e o sincronismo entre mídias.

Um teste inicial de usabilidade do protótipo foi realizado. A opinião coletada foi que a interface do programa é completa e fácil de utilizar. Os usuários rapidamente obtiveram melhorias no desempenho da execução dos toques. No entanto, como esperado, eles não aprenderam a manejar o instrumento.

Muitos trabalhos futuros podem ser realizados. Em relação ao *Rythmus Environment*, é necessário validá-lo através da construção de novas instâncias de ferramentas. O ambiente também poderia suportar novos meios de entrada e manipular outros formatos de áudio (como MP3 e AAC) para aumentar a fidelidade do som. Por fim, seria interessante existir diferentes perfis de usuários nas ferramentas instanciadas. No que diz respeito ao protótipo, poder-se-ia apresentar animações mais realistas, outros instrumentos de acompanhamentos e outros toques de Maracatu. Além disso, testes de usabilidade mais abrangentes com diferentes perfis de usuários são de fundamental importância para uma melhor validação do mesmo.

Referências

- Barbosa, Virgínia. (2005) “A Continuidade das mudanças musicais construindo reconhecimento: a experiência do Maracatu Estrela Brilhante (Recife)”. Dissertação de Mestrado em Musicologia, UFRJ, Rio de Janeiro, Brasil.
- Guerra-Peixe, C. (1980) “Maracatus do Recife”, Irmãos Vitale, 2ª edição.
- Maranhão, S.M.B. et al. (2003) “Ensino de Maracatu de Baque Virado através de Software Educativo”. Taller Internacional de Software Educativo (TISE’03), Chile.
- Sound. (2004) “Java Sound” <http://java.sun.com/j2se/1.4/docs/guide/sound>, dezembro.
- Young, R. (1996) “The MIDI Files”, Prentice Hall, 1ª edição.

Desenvolvimento e implementação de uma codificação para definir estruturas musicais

Pedro Kröger¹

¹Escola de Música da UFBA – Parque Universitário Edgard Santos, Canela
40110-150 Salvador, Bahia

kroeger@pedrokröger.net

Abstract. *There are many musical codifications for different purposes, however, to our knowledge, there is not so many research to develop a specific codification to define musical structures in a higher level of abstraction like “return all subjects in the fugues of Bach’s Well-Tempered Clavier” or “return all real answers” (which actually mean “get all answers, transpose a fifth below, compare with the respective subjects and return the ones that are real”). Our goal is to develop a specific codification for music fragments that allows a music description in higher levels beyond note/duration, including formal structures and idiosyncratic elements and to develop a prototype in Lisp to test and help the development of this codification.*

Resumo. *Existem codificações musicais para diversos fins, contudo não é de nosso conhecimento nenhuma pesquisa envolvendo uma codificação específica para definir estruturas musicais em um nível mais alto de abstração como “mostre todos os sujeitos das fugas do Cravo Bem Temperado de Bach”, ou “mostre todas as respostas reais das fugas do Cravo Bem Temperado” (o que na verdade significa “pegue todas as respostas, transponha uma quinta abaixo, compare com os sujeitos e mostre as que são reais”). Nosso objetivo é desenvolver uma codificação específica para fragmentos musicais que permita a descrição musical em níveis mais altos além da nota/duração—incluindo estruturas formais e elementos idiosincráticos—e desenvolver um protótipo em Lisp para testar e auxiliar o processo de desenvolvimento dessa codificação.*

1. Introdução

A codificação musical é a busca da representação de elementos sonoros. Exemplos simples de codificações incluem a representação das notas musicais por letras (C para dó, D para ré, e assim por diante) ou números (0 para dó, 1, para dó#, 2 para ré, e assim por diante). Codificações musicais têm sido usadas desde os primeiros esforços do homem para transcrever sons. Ainda que a codificação musical não esteja limitada à aplicações para computador é nessa área que encontramos o maior número de pesquisas, dadas as possibilidades de processamento oferecidas pelo computador [Selfridge-Field, 1997, p. 3–5]. Existem codificações para performance (e.g. MIDI), síntese sonora (e.g. Csound), jogos (e.g. *Music Macro Language*), notação musical (e.g. *Common Music Notation*, SCORE, GUIDO), braile, bibliografia musical (e.g. Plaine, Easie) e análise de canções folclóricas (e.g. Essen, FRELMUS). Novas codificações como SDML e MusicXML [Good, 2001] em geral lidam apenas com elementos de notação como compasso, notas, durações, etc. Tem se dado uma grande relevância para o desenvolvimento de sistemas para retornar dados a perguntas por “melodias cantadas” [Uitdenboger, 2000] e a criação de ferramentas para procura em bibliotecas digitais, retornando dados como gênero, tempo, executante, compositor,

etc. [Clausen et al., 2000, Dunn, 2000, Bainbridge et al., 2003]. Algumas pesquisas lidam com o conceito de “unidades composicionais” [Berggren, 1997] e reconhecem a necessidade de agrupar a representação em unidades como “frases” [Roland, 2000], contudo não é de nosso conhecimento nenhuma pesquisa envolvendo uma codificação específica para definir estruturas musicais em um nível mais alto de abstração. Um trabalho semelhante [Holger, 2001] usa o formato GUIDO, não tem abstração de alto-nível e seu objetivo principal é o de codificação para banco de dados, não de estruturas musicais.

Acreditamos ser importante ter um sistema que seja inteligente o suficiente para poder lidar com dados musicais em um nível mais alto de abstração como “mostre todos os sujeitos das fugas do Cravo Bem Temperado de Bach”, ou “mostre todas as respostas reais das fugas do Cravo Bem Temperado” (o que na verdade significa “pegue todas as respostas, transponha uma quinta abaixo, compare com os sujeitos e mostre as que são reais”). Outros exemplos incluem acordes para o estudo de orquestração, elementos formais, harmônicos e motivicos. Esse sistema deveria usar uma codificação específica para permitir esses recursos.

2. Objetivos

Nosso objetivo é desenvolver uma codificação específica para fragmentos musicais que permita a descrição musical em níveis mais altos além da nota/duração, incluindo estruturas formais, e elementos idiosincráticos (como a noção de sujeito/resposta em fugas, ou grupos de temas na forma sonata, ou motivos em certos tipos de composições) e desenvolver o protótipo de um programa de computador para testar e auxiliar o processo de desenvolvimento dessa codificação.

3. Justificativa

Essa pesquisa se justifica não só pela originalidade, mas porque vários problemas de representação podem ser investigados, como elementos que se sobrepõem [Roland, 2000], como obter um todo a partir de segmentos, como desenvolver codificações otimizadas para certas tarefas como acordes para o estudo de orquestração, acordes no estudo de harmonia, redução analítica em relação à versão original, redução para piano em relação à versão original.

4. Metodologia

Primeiramente será desenvolvido o núcleo básico do programa que servirá como base para a implementação das codificações. Esse núcleo será baseado em um conjunto de funções em Lisp que já implementamos para lidar com dados musicais. Para garantir que o processo da pesquisa ocorra tranquilamente, durante essa fase vamos oferecer um treinamento rápido aos bolsistas nas tecnologias usadas (Lisp e LilyPond) e elementos de metodologia científica.

Em seguida desenvolveremos as quatro etapas básicas da pesquisa. Em cada uma serão analisados os problemas de codificação de um grupo específico de problemas oriundos de grupos de exemplos musicais. Os grupos escolhidos foram as Fugas do Cravo Bem Temperado de Bach, Corais Harmonizados de Bach, diversos acordes de orquestra (para estudo de orquestração), e Sonatas para Piano de Beethoven. Os recursos para a codificação necessária serão implementados no protótipo; os exemplos serão codificados, e uma série de testes serão conduzidos para confirmar a viabilidade e eficiência da codificação. Para tanto o programa deverá ser escrito de maneira extensível, para permitir que novos recursos possam ser acrescentados facilmente, inclusive por usuários.

Em cada uma dessas quatro etapas serão produzidos resultados parciais que também serão usados nos testes das etapas seguintes, garantindo uma compatibilidade maior.

Para implementar o protótipo usaremos a metodologia *bottom-up* de desenvolvimento e a linguagem de programação Lisp devido a seu poder, flexibilidade e eficiência. Lisp tem se mostrado uma excelente linguagem para a criação de linguagens de domínio específico [Graham, 1993]. Usaremos o programa de tipografia musical LilyPond¹ como motor básico para gerar partituras visuais a partir da nossa codificação.

5. Trabalhos relacionados

O Humdrum foi criado para facilitar a possibilidade de propor perguntas e respostas às questões relacionadas a pesquisa no campo musical [Huron, 1997]. Ele pode extrair trechos usando expressões regulares ou através do comando `extract`, que extrai colunas dentro da representação “tabular” do Humdrum [Huron, 1999]. Ele provê grande extensibilidade, contudo não de uma maneira padrão (existem mais de 40 formatos). O *Themefinder*² é um exemplo de uso do Humdrum. Ele permite a busca por notas, intervalos, grau da escala, e contorno. Essa é uma demonstração da limitação desse tipo de codificação que lida apenas com elementos como compasso, notas, durações, etc, conforme citamos anteriormente. Apesar de novas *spines* (uma coluna com informação) poderem ser criadas é difícil criar representações em um nível mais alto de abstração.

O MusicXML foi criada para ser uma representação para notação intercambiável entre diferentes representações. Ela possui suporte para notação musical, performance, análise, e extração de dados. Contudo ela foi criada para ser “suficiente” e não “otimizada” para essas aplicações [Good, 2001]. O MusicXML não só não possui abstrações em níveis mais alto como explicitamente divide a representação em compassos, tornando difícil a representação de estruturas que englobam partes de compassos.

References

- Bainbridge, D., Cunningham, S. J., and Downie, J. S. (2003). Analysis of queries to a wizard-of-oz mir system: Challenging assumptions about what people really want. In *ISMIR 2003: Proceedings of the Fourth International Conference on Music Information Retrieval*.
- Berggren, U. (1997). Encoding of compositional units. In Selfridge-Field, E., editor, *Beyond MIDI: the handbook of musical codes*, pages 451–458. The MIT Press, Massachusetts.
- Clausen, M., Engelbrecht, R., Meyer, D., and Schmitz, J. (2000). Proms: A web-based tool for searching in polyphonic music. In *ISMIR 2000: Proceedings of the First International Conference on Music Information Retrieval*.
- Dunn, J. (2000). Beyond variations: Creating a digital music library. In *ISMIR 2000: Proceedings of the First International Conference on Music Information Retrieval*.
- Good, M. (2001). Musicxml for notation and analysis. In Hewlett, W. B. and Selfridge-Field, E., editors, *The virtual score: representation, retrieval, restoration*, volume 12 of *Computing in Musicology*, pages 113–124. The MIT Press, Massachusetts.
- Graham, P. (1993). *On Lisp: Advanced Techniques for Common Lisp*. Prentice Hall.

¹Cujo desenvolvimento nós fazemos parte (lilypond.org/web/about/thanks).

²<http://www.themefinder.org>

- Holger, H. (2001). Guido/mir—an experimental musical information retrieval system based on guido music notation. In *ISMIR 2001: Proceedings of the Second International Conference on Music Information Retrieval*.
- Huron, D. (1997). Humdrum and kern: Selective feature encoding. In Selfridge-Field, E., editor, *Beyond MIDI: the handbook of musical codes*, pages 375–401. The MIT Press, Massachusetts.
- Huron, D. (1999). *Music Research Using Humdrum: A User's Guide*. Center for Computer Assisted Research in the Humanities, Stanford, California.
- Roland, P. (2000). Xml4mir: Extensible markup language for music information retrieval. In *ISMIR 2000: Proceedings of the First International Conference on Music Information Retrieval*.
- Selfridge-Field, E., editor (1997). *Beyond MIDI: the handbook of musical codes*. The MIT Press, Massachusetts.
- Uitdenbogerd, A. (2000). Music ir: Past, present, and future. In *ISMIR 2000: Proceedings of the First International Conference on Music Information Retrieval*.

Poucas Linhas de Ana Cristina de Sílvia Ferraz: uma breve análise a partir de conceitos do próprio compositor

Flávio Ferreira da Silva, Maurício Alves Loureiro

Escola de Música - Universidade Federal de Minas Gerais (UFMG)

flavioferreira@ufmg.br, mauricioloureiro@ufmg.br

Abstract: *this article presents a brief analysis of Poucas Linhas de Ana Cristina for clarinet and live electronics of Silvio Ferraz. We believe that the musical analysis is an important tool to understand the musical discourse, then, we search an approach that unite the written part (score) and the electronic part to understand the music as a whole. The analysis presented here followed the own composer's concepts presented on his book Livro das Sonoridades.*

Resumo: *este artigo apresenta uma breve análise da música Poucas Linhas de Ana Cristina para clarineta e eletrônica ao vivo de Silvio Ferraz. Acreditamos que a análise musical é uma importante ferramenta para se compreender o discurso musical, então, procuramos uma abordagem que unisse a parte escrita (partitura) com a parte eletrônica no entendimento do todo musical. A análise aqui apresentada segue os conceitos do próprio compositor apresentados no seu livro Livro das Sonoridades.*

1. Introdução

A utilização musical das novas tecnologias eletrônicas e digitais a partir do final de década de 1940 trouxe muitas novidades e alterações para todos os estágios do fazer musical: composição, performance e audição. Utilizando instrumentos e sonoridades antes considerados “não musicais”, a produção musical afastou-se da recepção, sendo a primeira apresentação para um público só em 5 de outubro de 1948 no *concert de bruit* [Palombini, 1999]. Segundo Boulez, desde o início do século passado nossa cultura tem sido orientada pelo historicismo e conservadorismo, “onde os modelos selecionados para o ensino são tirados de um período extremamente circunscrito na história da música” [Boulez, 1986, p. 6-7], ficando a recepção musical defasada em relação à sua produção.

No território da performance, a música eletroacústica começou tirando o instrumentista do cenário musical, fazendo música totalmente nos novos suportes como o acetato e as fitas de rolo, difundida através dos alto-falantes. A música eletroacústica mista, unindo as tecnologias eletrônicas e digitais aos instrumentos tradicionais, trouxe o intérprete instrumentista de volta ao palco em 1952 numa composição de Bruno Maderna chamada *Musica su Due Dimensioni* [Menezes, 1999, p. 13] e assim, surgiu uma série de novidades/difículdades para a performance musical, entre elas a notação.

O intérprete agora toca o seu instrumento, controla interfaces e interage com alto-falantes e a partitura, incompleta, configura-se apenas como um guia para o instrumentista, não comportando a parte eletrônica.

Assim, foi atingido também o campo de estudo da análise musical, “porque a análise se baseia na tradição da escrita musical e não na música efetivamente sendo tocada ou escutada. O que a análise musical tem observado é a partitura e não o resultado sonoro ou a experiência da escuta” [GUBERNIKOFF, 2003, p. 31].

O trabalho aqui apresentado é parte de uma pesquisa de mestrado em andamento que pretende contribuir para a montagem deste quebra-cabeça, unindo parte escrita (partitura) e parte eletrônica na compreensão do discurso musical. Como objeto de estudo, foi selecionada a peça *Poucas Linhas de Ana Cristina* para clarineta e processamento digital do compositor paulista Sílvio Ferraz, levando em conta dois fatores importantes: (1) é uma composição de autor brasileiro contemporâneo, permitindo contato direto com o compositor; (2) segundo o compositor, a obra pode ser executada em uma versão sem o processamento, despertando grande interesse em saber qual é o seu papel e qual a sua relação com a parte escrita no contexto musical.

Para atingir os objetivos propostos, pretende-se: (1) elaborar uma análise musical do material escrito separadamente; (2) elaborar uma análise musical a partir da obra com o processamento; (3) analisar tecnicamente o processamento digital, focando nos seus aspectos de extensão dos recursos instrumentais; (4) realizar entrevistas com o compositor; (5) propor sistemas para que o intérprete tenha algum controle sobre a eletrônica; (6) realizar a performance da obra levando em consideração os resultados da pesquisa.

Este artigo apresentará alguns pensamentos composicionais de Sílvio Ferraz que possam auxiliar a análise, além de pré-análises da parte escrita (partitura) e do processamento (*patch anacris*) como um primeiro passo para encontrar suas relações.

2. Pensamentos Composicionais de Sílvio Ferraz

Os princípios composicionais de Sílvio Ferraz passam longe da idéia de organização de material, de sons ou de notas. Em seu *Livro das Sonoridades* ele apresenta direções composicionais mais baseadas na sensação, no sentimento, na estética e na expressividade.

“Compor é como fazer uma casa. (...) Fazer um território, fazer uma casa ou nicho é como que deixar claro que ali vive alguém, vive alguma coisa.” [Ferraz, 2005, p. 35].

A partir desta concepção cria conceitos que se tornam essenciais para o processo de sua composição: *ritornelo* - criar um território, sair e voltar a ele; *giro* – derivado de *ritornelo*, “... é mais do que apenas um aspecto da música; é um aspecto de musicalidade que atravessa uma grande área da vida”; *devir*: captar forças não humanas, que não nos são sensíveis e torná-las sonoras; *diferença*: sair do cotidiano e abrir-se a uma nova escuta.

Com conceitos podemos entender que, para Ferraz, o papel do compositor é abrir-se ao desconhecido, ao diferente, reconhecer nele forças não humanas e torná-las sonoras, criando territórios musicais e, assim, “ejetar o ouvinte do território firme, indo fixá-lo em estados totalmente transientes de escuta”. Além de criar territórios, é necessário também dismantela-los, gerando partículas que formam locais de instabilidade, “terrenos que muitas vezes podem ser a interseção de dois ou mais terrenos”. Aqui encontramos um outro conceito importante em sua obra: o futuro. Estas partículas, este terreno de instabilidade, estão mais ligadas ao futuro, pois não estão localizadas nem no presente nem no passado:

“São tais partículas que chamamos de material composicional, e o desmanche da forma e da matéria, o modo deste desmanche, já é também uma força do futuro, ou que conecta com o futuro” [ibidem, p. 39].

Ferraz apresenta ainda uma postura bastante crítica a respeito da relação da música com a fonte sonora, com o material sonoro e com o pensamento musical. Para ele, estando a criação subordinada a quaisquer destes aspectos, estaria presa e não seria um fato em movimento:

“... o seu material, que não se confunde com a matéria (o som) e nem se opõe à forma (abstração da matéria), se revela durante o percurso. No plano de composição não há caminho: o caminho se faz ao caminhar” [ibidem, p. 51].

Compor é construir uma casa para alguém morar. Mas quem vive nela? Um intervalo, uma sonoridade, um gesto. Mas também o ouvinte. Então ele passa a brincar com o ouvinte. Leva-o a passear por um território desconhecido, por um

“processo cujo objeto é a sensação: ouvir não o som, mas aquilo que não está no som, que está no ouvir, aquilo que está no escutar, que está em operar cortes, em operar territórios, em dismantelá-los” [ibidem p. 45].

Para Sílvio Ferraz, o poder expressivo da música está em criar, juntamente com o território, as suas saídas, a desterritorialização: “compor tecendo um território como se tece uma linha de fuga, como quem foge”.

“Se desenho um lugar, e faço com que o ouvinte viva um pouco neste lugar, posso brincar também de fazer com que ele se sinta tranqüilo naquele lugar, ou com que tenha esta tranqüilidade abalada quando, de repente, e isto tem de ser de repente, o faço sentir-se arrastado para fora daquele lugar (...) O efeito surpresa! (...) E para fazer este lugar, o recurso talvez seja este de reiterar elementos, de fazer com que as coisas girem numa pequena roda, uma cantilena, um ritornelo, uma ladainha, um caleidoscópio, uma caixinha de música. E a surpresa é justamente aquele momento em que alguma coisa foge da ladainha (...) De repente uma nota trai a harmonia, desfaz o perfil principal da frase musical, uma sonoridade leva para outro espaço de ressonâncias” [FERRAZ, 2005, p. 37].

3. Poucas Linhas de Ana Cristina

3.1. Parte escrita

Conhecendo alguns traços do pensamento do compositor, e tendo um olhar mais cuidadoso, é possível deduzir o que seria, nesta música, o território, a casa, e quais são os pontos de instabilidade, nele inseridos.

Antes de iniciar a análise propriamente dita, ressaltamos que como a peça foi escrita para clarineta em si bemol, todos os nomes de nota deste artigo se referem às notas escritas, ou seja, uma segunda maior acima do que soa.

No início da peça, constrói-se o território a partir da repetição do intervalo de terça menor nas mesmas duas notas (mi – sol).

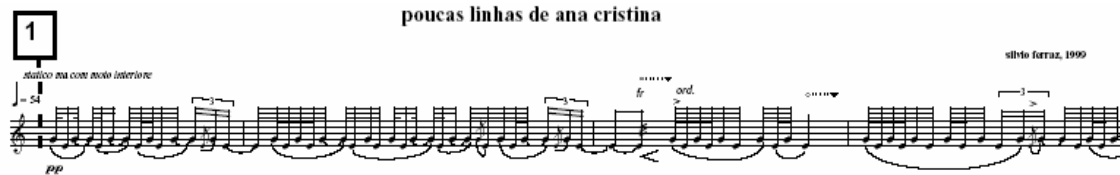


Figura 1: Poucas Linhas de Ana Cristina de Silvio Ferraz. Primeiro pentagrama da página 1.

O efeito surpresa surge com pequenas variações semitoniais que abrem o leque intervalar e o número de notas utilizadas.

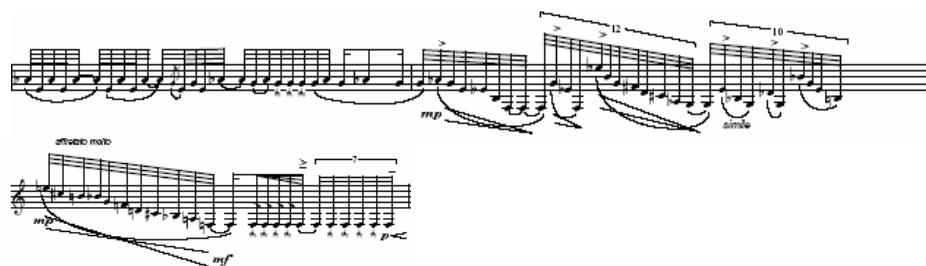


Figura 2: Poucas Linhas de Ana Cristina. Segundo e início do terceiro da página 1. Aqui, aumenta-se o número de intervalos e notas.

Ao escolher os materiais, Ferraz garante que “os mesmos que servem de eixo, trazem sobre si as linhas sobre as quais se sai deles, pois é sobre este eixo e não fora dele que advém a escapada”. Estando o território, o eixo central, construído na terça menor mi – sol, é a partir dela, abrindo suas extremidades com o lab e o mib, que inicia a saída, possibilitando o uso de mais notas e intervalos. Mas, considerando-se que os elementos que causam instabilidade estão ligados ao futuro, as escalas com acentos e quáterteras (fig. 2) encontram sua conexão no final do terceiro pentagrama da página 2.

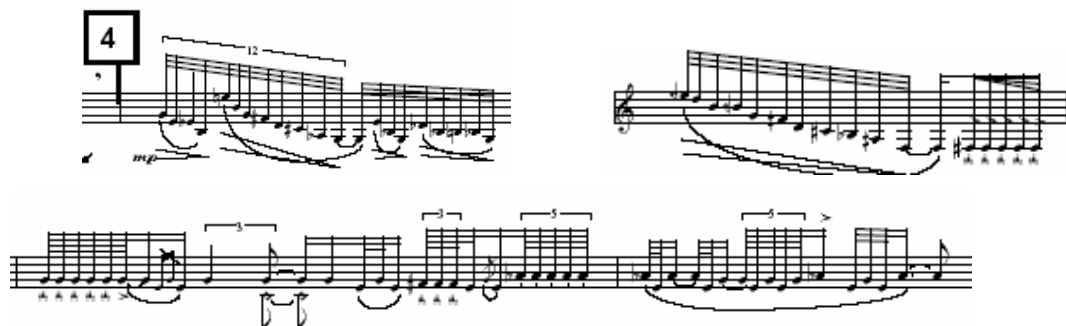


Figura 3: Poucas Linhas de Ana Cristina. Final do terceiro e quarto pentagrama da página 2.

E este encontra o seu elo ainda mais adiante, no final do terceiro pentagrama da página 3.

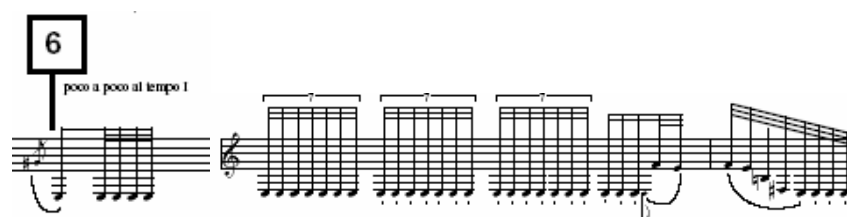


Figura 4: Poucas Linhas de Ana Cristina. Final do terceiro pentagrama da página 3.

Procurou-se até aqui identificar os momentos da música mais semelhantes entre si, excluindo aqueles que apresentam maior diferença. Acredita-se que os momentos semelhantes, unidos, formam o território inicial e que os diferentes sejam transições, momentos de corte, escapadas.

Localizado o território inicial, a casa, quem vive nela? Quais são as notas, intervalos, sonoridades ou ritmos principais na construção desta música?

Se como disse Sílvio Ferraz, o próprio terreno traz as suas linhas de fuga, ou seja, os próprios elementos utilizados na construção musical sugerem a sua elaboração ou variação, observando os momentos de estabilidade e de instabilidade e comparando-os, poderemos encontrar o que eles têm em comum e chegar aos personagens principais, àquilo que gerou a obra.

A primeira linha de fuga (ou transição) está no seguinte trecho.

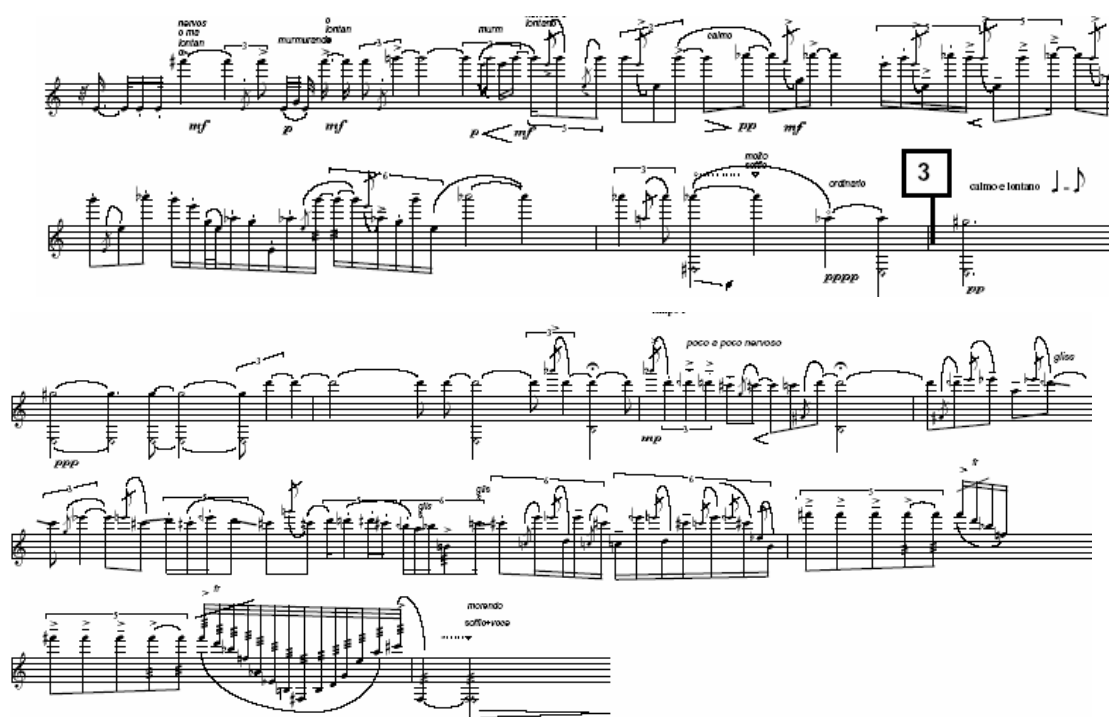


Figura 5: Poucas Linhas de Ana Cristina. Quinto e sexto pentagrama da página 1 e três primeiros pentagramas da página 2.

Nos dois primeiros pentagramas da figura 5, encontramos quatro elementos importantes: (1) grande exploração da tessitura com a utilização de agudos; (2) uma espécie de pedal na nota mi que é a mais grave em quase todo o trecho; (3) um

movimento cromático em volta da nota sol no agudo (fa# - sol - lab) e (4) uma insistência na terça menor mi – sol.

Dentro desta região de transição, encontramos uma inserção que aqui denomino subterritório. (Chamo de subterritório por estar inserido em um momento de transição).

Este subterritório é, na realidade, uma citação da peça *Ninphea* para oboé solo do próprio compositor. Nele surge um elemento totalmente novo: a exploração de espaços microtonais: quarto de tom e glissando. Este elemento será reiterado no transcórre da música, exercendo assim, sua conexão com o futuro.

O segundo terreno de corte, de dismantelamento, está baseado principalmente na terça menor inicial com suas variações semitonais.

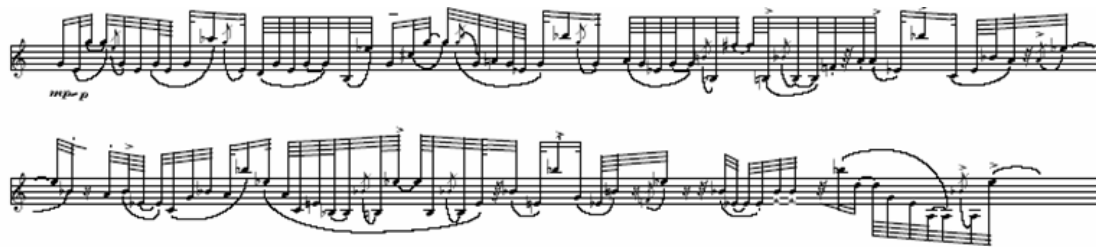


Figura 7: Poucas Linhas de Ana Cristina. Segundo momento de transição: dois pentagramas finais da página 2.

Sua elaboração rítmica está construída a partir da sequência rítmica que podemos chamar de α :



Figura 8: Poucas Linhas de Ana Cristina. Segundo momento de transição: sequência rítmica α (excerto da figura anterior).

e do seu espelhamento, que podemos chamar de β :



Figura 9: Poucas Linhas de Ana Cristina. Segundo momento de transição: sequência rítmica β (excerto da figura 7).

apresentando pequenas alterações como troca de notas por pausas, por exemplo, a cada aparição da sequência.

O próximo e último território criado é mais “nebuloso” do que os anteriores, pois não parece trazer uma relação material com a obra.

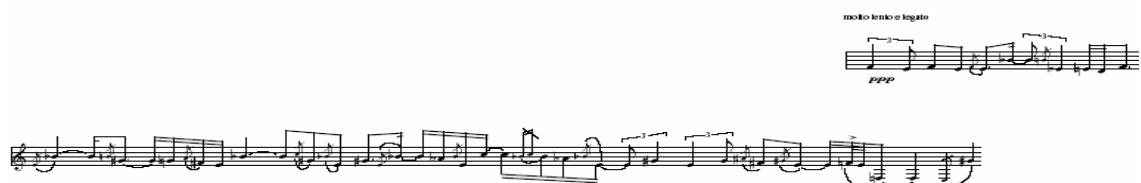


Figura 10: Poucas Linhas de Ana Cristina. Terceiro momento de transição: final do segundo pentagrama e terceiro pentagrama da página 3.

Em duas das três linhas de fuga apontadas (primeira e terceira), o elemento comum com o território inicial é a terça menor com as notas mi – sol. Podemos considerar que estas notas com seu intervalo são os protagonistas, “deixando claro a quem ouve que existe uma sonoridade em círculo (um intervalo musical, algumas notas, um som concreto, um tipo de ressonância, um gesto, uma figuração melódica ou harmônica), que ali mora alguém” [ibidem, p. 36].

O principal objeto deste *patch* é o *harmv2~*. Segundo o *Jimmies for MSP Handbook*, este objeto “desempenha uma transposição em tempo real de um sinal de áudio, alternando constantemente, por fade cruzado, “grãos” transpostos do som de entrada” [IRCAM, 1998, p. 20]. *Grosso modo*, o sinal de entrada é inserido em uma linha de *delay*, onde utiliza um algoritmo de *delay* continuamente variável dando o efeito de transposição do sinal original (IRCAM, MaxMSP 4.5, tapout.help).



Os valores de *window size* (tamanho da janela), *delay time* (tempo de resposta entre o sinal de entrada e o de saída) e *pitch transposition* (transposição de nota) recebidos pelo objeto são controlados pelo usuário. O *patch* executa basicamente duas funções principais: 1) amplia os recursos da clarineta gerando grupos de três notas (geradas por três *harmonizers* utilizados em paralelo) a cada nota tocada pelo

instrumento; 2) cria melodias próprias em glissando a partir de notas tocadas pela clarineta, formando uma sombra em contraponto (principalmente nas notas longas da clarineta). Para cada uma das funções, o *harmv2~* deve ser utilizado de forma diferente. Para os “acordes”, o valor de *pitch transposition* deve ser fixo e especificado nos *sliders* de transposição. Para os glissandos, estes mesmos valores devem ser móveis, deslizando os *sliders* no momento da performance. *anacris* apresenta dez *presets* que controlam valores como *window size*, *delay time*, *reverb time*, por exemplo.

4. Poucas linhas para um encerramento

Uma pergunta permanece ainda sem resposta. Qual a interferência da parte eletrônica na construção dos territórios e no seu desmantelamento? Uma pequena luz surge com as melodias em glissando, elemento que está também presente na parte escrita, no local que foi aqui denominado subterritório. Mas esta luz ainda está muito turva. Um estudo realmente aprofundado da função do *patch* em cada momento da música é necessário para responder a questão que permanece em aberto. Esta pesquisa pretende ainda abordar os aspectos de recepção da música tanto com o processamento quanto sem ele, abrangendo da composição à audição e observando como deve ser a postura do intérprete nesta intermediação. Espera-se assim contribuir com a performance instrumental de música eletroacústica mista e diminuir a distância entre os estágios de composição, performance e audição.

5. Bibliografia

- Boulez, P. Technology and the Composer. In: S. Emmerson (Ed.). *The Language of Electroacoustic Music*. London: MacMillan, p.5-14, 1986.
- Ferraz, S. *Livro das Sonoridades [notas dispersas sobre composição]*. Rio de Janeiro: 7 Letras, 2005.
- IRCAM. *Jimmies for MSP Handbook*. Primeira edição em Inglês. Centre Georges Pompidou, Paris, 1998.
- Menezes, F. *Atualidade Estética da Música Eletroacústica*. (Defesa Pública de Livre Docência). Departamento de Música, UNESP, São Paulo, 1999.
- Palombini, C. V. d. L. A Música Concreta Revisitada. *Revista Eletrônica de Musicologia*, v.4, 1999.

Impact of Distance in Pitch Class Profile Computation

Giordano Cabral¹, Jean-Pierre Briot¹, François Pachet²

¹Laboratoire d'Informatique de Paris 6 – Université Pierre et Marie Curie
8 Rue du Capitaine Scott 75018 Paris – France

²Sony Computer Science Lab Paris
6 Rue Amyot 75005 Paris – France

{Giordano.CABRAL, Jean-Pierre.BRIOT}@lip6.fr, pachet@csl.sony.fr

Abstract. *Pitch Class Profiles (PCP) [Fujishima 1999] are largely used for all applications involving harmonic content. Although the main steps of the PCP calculation are generally equal over all the scientific literature, some implementation details may vary, specially the impact of the distance between the frequency of the FFT bins and their closest note. This paper compares 6 ways of using this distance to weight the contribution of each FFT bin in the final PCP vector. We present the results of the 6 computed PCPs when used in a chord recognizer, a tonality estimator and a key detector.*

1. Introduction

Pitch Class Profiles (PCP) are vectors of low-level instantaneous features, representing the intensity of each of the twelve semitones of the tonal scale. They are largely used in all applications involving harmonic content, especially chord recognizers [Yoshioka et al. (2004)], tonality estimators [Gómez and Herrera 2004a] and key detectors [Pauws 2004]. The main advantages of the PCPs are the simplicity of calculation, the concision of the harmonic information and the power to unify various dispositions of a single chord class. Although the main steps of the PCP calculation are very similar over all the scientific literature, some implementation details may vary. Among them, it is especially imprecise how the distance from each FFT bin to its closest note may contribute to the quality of PCP. This paper intends to contribute to the discussion on the subject, comparing 6 functions (uniform, discrete, linear, anti-quadratic, exponential, and gaussian) using the above mentioned distance to change the PCP computation. To test the PCPs, we have implemented a chord recognizer, a tonality estimator, and a key detection system, as described in main publications: given a labeled database of sound recordings, the respective PCPs are computed, and a classification algorithm, such as k-nearest neighbors [Mitchell 1997] or hidden markov models [Sheh and Ellis 2003] is used. We assume that the higher the quality of a classifier using a specific PCP calculation method, the greater the precision of the PCP information. In other words, the number of correctly classified instances may serve as a comparative measure of precision for the PCP vectors, as illustrated in the end of this paper.

Next section describes how to compute the PCP vectors. Section 3 exposes the 6 proposed functions for weighting by distance. Section 4 explains the experiment in more details, the used datasets and the applied methodology. Section 5 presents the absolute and relative results, and section 6 draws some conclusions and future work.

2. PCP Computation

PCP vectors are computed by mapping each frequency bin of the spectrum to a pitch class (one of the 12 notes of the tonal scale). Figure 1 illustrates the main steps in the PCP computation. The sound in Figure 1a is converted to the frequency domain by means of some Fourier-Transform (FFT) [Orfanidis, S. 1995]. Each FFT bin is mapped to its closest note (e.g. FFT bins corresponding to frequencies like 433 Hz, 438 Hz, or 443 Hz are mapped to the A at 440 Hz) (see Equation 1). One may see such mapping as the division of the spectrum into regions, as shown in Figure 1c. Then, the amplitudes inside each region are summed up and divided by the number of bins inside the region, resulting in a histogram as in Figure 1d (see Equation 2). Finally, the histogram is folded, collapsing pure tones of the same pitch class, despite the octave, to the same chroma bin, resulting in a 12-sized vector, where each index represents the intensity of one note.

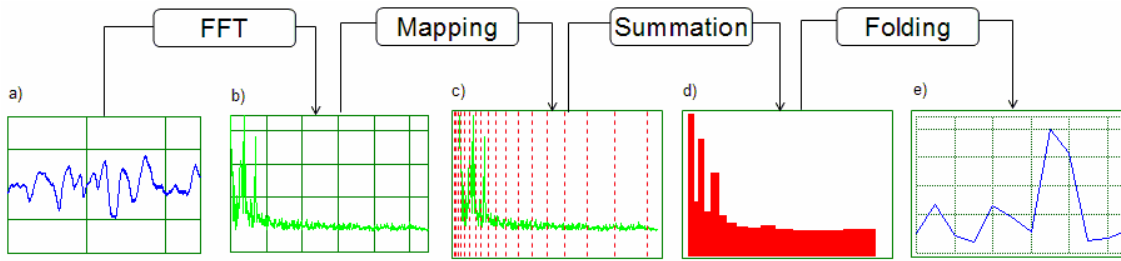


Figure 1. PCP computation steps.

Formally, we can formulate the PCP as in equation 1 [Sheh and Ellis 2003]. N is the number of samples in the sound (or the number of bins the FFT), k is a bin in the FFT (where $0 \leq k \leq N-1$), f_{ref} is the reference frequency corresponding to $PCP[0]$, and f_{sr} is the sampling rate. Normally, the value of each PCP element is calculated by summing the magnitude of all frequency bins that correspond to a particular pitch class (i.e. $p = 0, 1, \dots, 12$), as shown in Equation 2 [Sheh and Ellis 2003].

$$p(k) = \lfloor 12 \cdot \log_2(k/N \cdot f_{sr}/f_{ref}) \rfloor \bmod 12$$

Equation 1. Mapping from frequency bins to PCP bins.

$$PCP[p] = \sum_{k:p(k)=p} |X[k]|^2$$

Equation 2. Calculation of the values of the PCP elements.

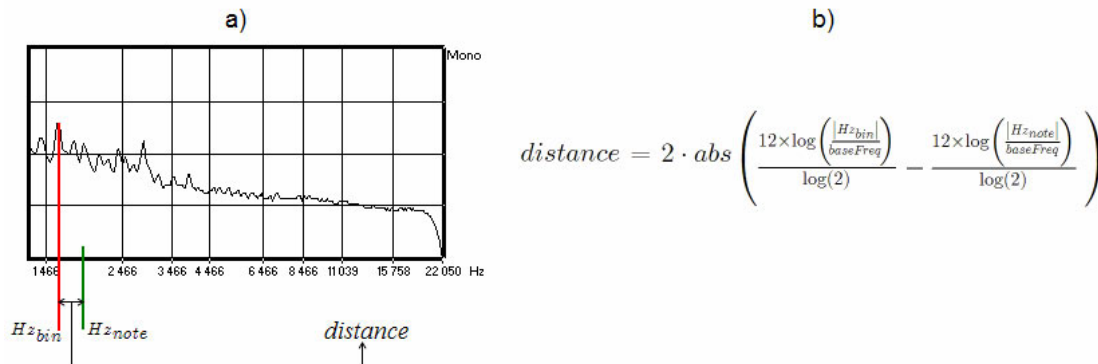


Figure 2. Distance between the frequency of the FFT bin (Hz_{bin}) and the frequency of its closest note (Hz_{note}).

However, as in most cases the frequencies of the FFT bins do not match the frequency of a note, one can imagine that this summation may take into account the distance between them (Figure 2). Figure 2b shows the formula of such distance, which is proportional to the size of the region. It varies from 0 (the FFT bin has exactly the same frequency of a note) to 1 (exactly in the middle of 2 notes). This work intends to evaluate the different methods using this distance to weight the summation, in order to improve the quality of the classifiers over such PCPs, with regards to precision and robustness.

3. Weighting functions

We are interested in the possible functions $f(\text{distance})$ (Equation 3) that could be used to weight the summation of values inside each region. These functions must be applicable for all values in the range 0:1, and must also return values in the range 0:1. For example, $f(x) = 1/x$ or $f(x) = \log(x)$ would not be allowed since they are not defined for $x=0$.

$$PCP_{[p]} = \frac{\sum_{k:p(k)=p} |X[k]|^2 \cdot f(\text{distance})}{\sum_{k:p(k)=p} f(\text{distance})}$$

Equation 3. Weighted summation using the distance to the closest note.

We took into consideration 6 functions: uniform, discrete, linear, anti-quadratic, exponential, and gaussian. The uniform function returns always 1. It means that the distance does not affect the result. This is the simpler and most used method. Discrete weighting will only consider the frequencies inside a narrower region (in Figure 3, distance must be smaller than 0.2). In the other 4 functions, the weight of the element will gradually decrease as the frequency goes farther, but the respective curves have different shapes. The graphs of these functions are shown in Figure 3, as well as their formula.

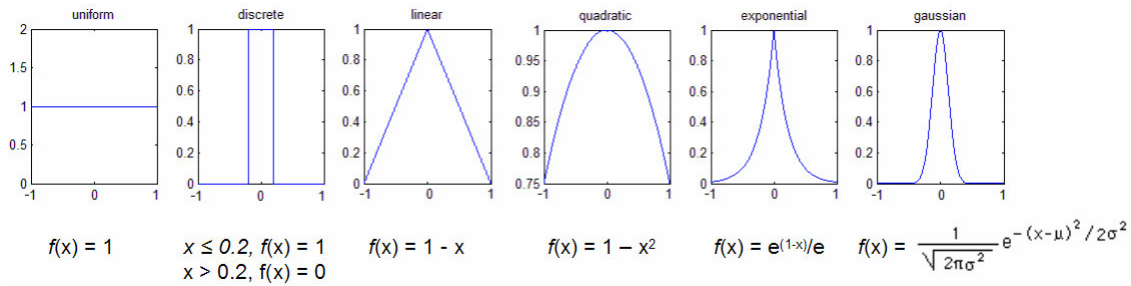


Figure 3. Weighting functions (uniform, discrete, linear, anti-quadratic, exponential, and gaussian).

4. Chord Recognition, Tonality Estimation, Root Detection

In order to evaluate the different weighting functions, we re-implemented some applications using PCP: a chord recognizer, a tonality estimator, and a key detector. All of them share the same architecture: a database of sound recordings is labeled with the expected class for each example. In the case of the chord recognizer, the class is the expected chord (key and type); for the tonality estimator, the expected tonality (key and

mode); for the key detector, just the key. Then, the PCP of each example is calculated, and used as input to some machine learning algorithm [Mitchell 1997] in order to automatically generalize a classifier. In other words, these algorithms try to automatically learn from the examples the patterns of the PCPs of each chord, so it will be capable to give answers to new examples.

As suggested by [Gómez and Herrera 2004b], we used a KNN classifier over the PCPs of the sound recordings. The data was taken from D'accord Guitar Chord Database [Cabral et al. 2001], a guitar midi based chord database. Each midi chord was rendered into a wav file using Timidity++ and a free nylon guitar patch. The richness of the symbolic information present (chord root, type, set of notes, key, position, fingers, etc.) allowed us to automatically label the data and create some databases. The first one (MajMin) is a tonality estimator-like database, restricted to deal only with Amaj and Amin chords. The second one (ChordC) constrained the root to be C, trying to automatically learn to classify the type (Maj, Min, Dom7, Min7, Dim). The third one (Root) tried to determine the root of a chord. The forth one (Chord) is the chord recognizer. It tries to learn what is the chord (i.e. which root and type simultaneously). The fifth one (RealTest) is the same chord recognizer, but tested with real sound recordings, instead of synthetic sounds. 80% of each database was settled on as the training dataset and 20% as the testing dataset.

Table 1. The databases corresponding to the 5 experiments.

Database	Root	Types
MajMin	Fix (A)	Maj, Min
ChordC	Fix(C)	Maj, Min, Dom7, Min7, Dim
Root	Variable (C, C#, ..., B)	Fix
Chord	Variable (C, C#, ..., B)	Maj, Min, Dom7, Min7, Dim
RealTest	Variable (C, C#, ..., B)	Maj, Min, Dom7, Min7, Dim

5. Results and Discussion

Table 2 and Figure 4 show the results of the classifiers by weighting function and by experiment. For the first, trivial, problem of separating the major and minor chords (with a synthetic database of fix root chords), all weighting functions worked satisfactorily. For the second experiment, such of finding the chord type, the simple discrete weighting surprisingly surpassed all others. For the third experiment, such of finding the root of a chord, regardless of its type, the exponential and gaussian worked slightly better than the others. For the fourth problem, such of chord recognition (the one we were most interested in), all but the uniform function got close values. Finally, the last experiment, dealing with the same problem of chord recognition but tested with a dataset of recorded audio, showed the most discrepant results. In fact, real recordings may have differences in tuning, which affect significantly the precision of the algorithms, especially those that abruptly increase or decrease, such as the discrete and gaussian functions. These functions leak in robustness, since they are extremely dependent to a good tuning.

On the other hand, a comparative analysis shows that a part from the dependency to the quality of the tuning, the weighting functions do not present significant disparities. Figure 5 shows such an analysis, in which the results of each experiment are proportional (i.e. each value is divided by the maximum). We can see, for example, that

the simpler and normally worst solution (uniform) is always at least 90% as good as any other method.

Table 2. Results of the classifiers using the different weighting functions in each experiment.

Name	MajMin	ChordC	Root	Chord	RealTest
Uniform	100,00 %	84,85 %	73,74 %	71,35 %	65,38 %
Discrete	100,00 %	93,94 %	78,51 %	77,45 %	34,62 %
Linear	100,00 %	84,85 %	77,98 %	76,39 %	69,23 %
Anti-Quadratic	100,00 %	84,85 %	77,72 %	76,39 %	65,38 %
Exponential	100,00 %	87,88 %	80,37 %	79,31 %	61,54 %
Gaussian	100,00 %	87,88 %	80,11 %	79,31 %	42,31 %

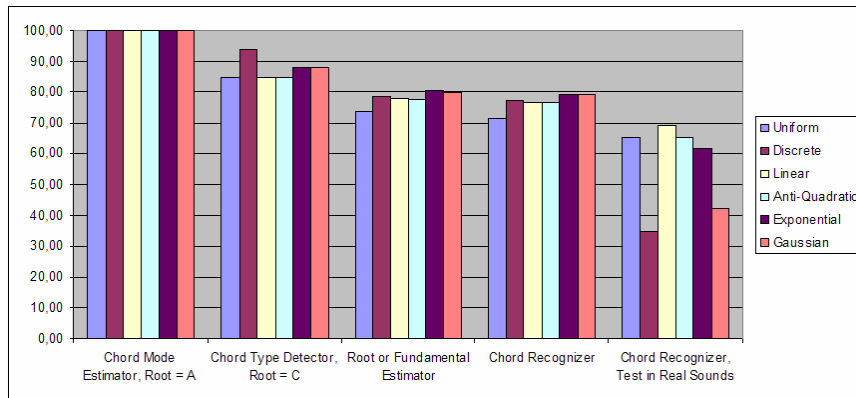


Figure 4. Results of the classifiers using the different weighting functions in each experiment.

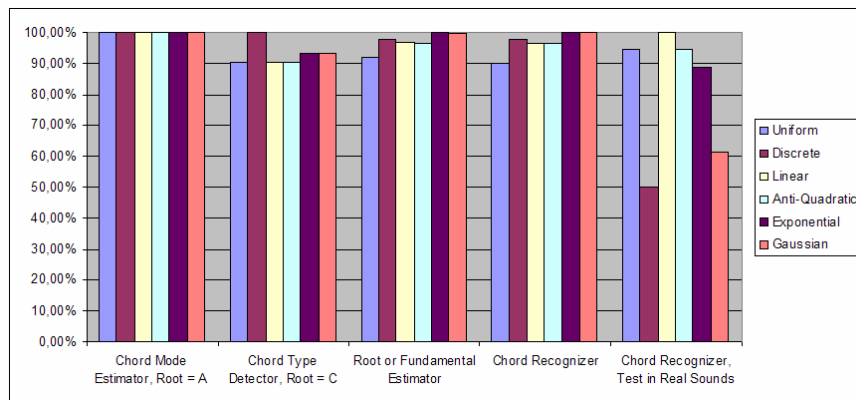


Figure 5. Comparative results (each value is divided by the maximum value in the experiment).

6. Conclusions and Future Work

Given the results, we can draw a few conclusions: 1) the weighting functions do not affect significantly the quality of the PCP; 2) discrete and gaussian weightings are not robust; 3) there is no absolute “winner”. Nevertheless, Table 3 suggests the use of some functions, depending on the interest of the developer. If he wants a simple solution, no weighting (uniform) works satisfactorily. Otherwise, the linear, discrete, exponential,

and gaussian works a little better. Additionally, if the tuning of the sound samples is not guaranteed, he should better choose the linear weighting.

Table 3. Best weighting function, given the interests of the developer.

	Robust	Good tuning guaranteed
Light, simple	Uniform	Discrete
Efficient	Linear	Discrete/Exponential

Some improvements can be done in future works, specially the conversion of the datasets to real recordings. The addition of hybrid methods, such as the discrete+linear, would also be interesting. Finally, the comparison with other existent variations of PCP extraction algorithms is strongly desirable, in order to delineate a standard, well defined algorithm.

7. Acknowledgements

We would like to thank all the team from CSL Sony in Paris.

This research is supported by CAPES/COFECUB, Brazil/France.

References

- Cabral, G., Zanforlin, I., Santana, H., Lima, R., & Ramalho, G. (2001) "D'accord Guitar: An Innovative Guitar Performance System", in Proceedings of Journées d'Informatique Musicale (JIM01), Bourges.
- Fujishima, T. (1999) "Real-time chord recognition of musical sound: a system using Common Lisp Music", Proceedings of International Computer Music Conference (ICMC99), Beijing.
- Gómez, E. and Herrera, P. (2004a) "Estimating the tonality of polyphonic audio files: cognitive versus machine learning modelling strategies", Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR04), Barcelona.
- Gómez, E. Herrera, P. (2004b) "Automatic Extraction of Tonal Metadata from Polyphonic Audio Recordings", Proceedings of 25th International AES Conference, London.
- Mitchell, T. (1997) "Machine Learning", The McGraw-Hill Companies, Inc.
- Orfanidis, S. (1995) "Introduction to Signal Processing", Prentice-hall.
- Pauws, S. (2004) "Musical key extraction from audio", Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR04), Barcelona.
- Sheh, A. and Ellis, D. (2003) "Chord Segmentation and Recognition using EM-Trained Hidden Markov Models", Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR03), Baltimore, USA.
- Yoshioka, T., Kitahara, T., Komatani, K., Ogata, T., and Okuno, H.-G. (2004) "Automatic Chord Transcription with Concurrent Recognition of Chord Symbols and Boundaries", Proceedings of 5th International Conference on Music Information Retrieval (ISMIR 2004), Barcelona.

Interações musicais em rede

Fábio Furlanete, Jonatas Manzolli

Núcleo Interdisciplinar de Comunicação Sonora, UNICAMP
Rua da Reitoria, 165 - Cidade Universitária "Zeferino Vaz" - Campinas - SP - Brazil

ffurlanete@nics.unicamp.br jonatas@nics.unicamp.br

Abstract. *This paper presents a proposal for the implementation of a distributed sound games platform aiming the creative exploration of the relations between memory, indeterminacy and self-organizing systems in the development of musical interaction models with computers and mobile devices. We suggest the formalization of the musical interaction process in a Semiotic Network as a way to the composer's control and interference in the system.*

Resumo. *Este artigo propõe a implementação de uma plataforma de jogos sonoros distribuídos visando a exploração criativa das relações entre memória, indeterminação e auto-organização no desenvolvimento de modelos de interação musical com dispositivos informáticos. Sugerimos ainda a formalização dos processos de interação musical nos termos de uma Rede Simiônica como meio de controle e interferência no sistema por parte do compositor.*

1. Introdução

Este trabalho propõe a exploração criativa das relações entre memória, indeterminação e auto-organização no desenvolvimento de modelos de interação musical com dispositivos informáticos. Para isso tomamos como ponto de partida a idéia de auto-organização como paradigma composicional [Manzoli, 1996]. Nesses modelos o compositor é o agente que estabelece o corte que dá início e possibilita a emergência de um processo de auto-organizado. Ao mesmo tempo, ele controla a interferência de ruído permitindo uma maior ou menor sedimentação de seus atratores.

A interferência e o controle em sistemas suficientemente complexos para permitir a emergência de comportamentos auto-organizados é um problema de difícil abordagem. Para tanto nos propomos a construção de um Modelo Semiótico [Silva e Gudwin, 2001] que permita tanto a descrição do sistema como o emprego do Controle Situacional Semiótico para a interferência e controle por parte dos compositores que utilizarem o sistema.

2. Motivação Estética

A formação musical tradicional, herdada do Romantismo e muito presente em nossa educação ainda hoje, atribui ao compositor o papel de agente central (quando não único) do processo criativo. Nesse paradigma [Zampronha, 2000, 1996][Furlanete, 2000] o compositor é o “dono da idéia”. Mesmo com as experiências realizadas pelos compositores das décadas de 1960 e 1970 com a abertura das composições à interferências mais efetivas por parte dos intérpretes e às possibilidades do acaso [Boulez, 1995], esse modelo do compositor que vai, *a priori*, determinar as possibilidades de sentido da obra continuou marcando forte presença, atribuindo ao compositor o papel de “demônio de Laplace” do discurso musical.

Com a introdução do computador como ferramenta de composição musical a necessidade de reformular esse modelo tornou-se mais evidente. Ao mesmo tempo que deslocou o processo criativo do *design* da música para o *design* do processo composicional [Laske, 1991], ele tornou mais explícita a dinamicidade da interação entre o compositor e suas ferramentas ao possibilitar a constante avaliação aural e reformulação do discurso. A idéia da identidade da obra como um todo orgânico começa a se dissipar em favor da idéia da obra como marca deixada por um processo criativo. Processo esse do qual o compositor é apenas mais um dos

agentes - o "mestre do jogo", que convida os demais (humanos ou sintéticos) e os orienta em um jogo cujas regras iniciais ele pode determinar mas cujo resultado final é desconhecido. Pensando composição musical como *design* de processos de interação, possivelmente auto-organizado, a obra deixa de ser dada *a priori* com relação à performance. O material musical passa a ser não mais um conjunto de objetos moldados pelo compositor - regiões de identidade pelas quais o discurso musical trafega. Mas como o resultado, *a posteriori*, de estratégias de interação, jogos, nos quais os agentes se engajam e a partir dos quais vai emergir não mais um *discurso*, mas um *decurso* musical.

3. Memória, Ruído e Auto-organização

Um dos aspectos dessa mudança de paradigma que mais nos interessa do ponto de vista poético é a relação entre memória e indeterminação para o entendimento do tempo nos processos de interação musical auto-organizados.

A importância da memória na temporalidade dos processos interativos auto-organizados é apontada por Debrun (1996, p. 16):

É a medida que um jogo complexo vai se estabelecendo entre uma memória real (isto é, não apenas reconstruída pelo observador, mas vivida pelo sistema em vias de constituição ou redefinição) e antecipações baseadas nessa memória que o processo poderá ao mesmo tempo "ir para frente" e se cristalizar numa forma. Ou seja: inventando aos poucos um atrator, e, logo em seguida, nele obedecendo – ou inversamente, contestando-o, até o amadurecimento de um atrator definitivo.

Porém,

à medida que o processo autoorganizado tende - quando bem sucedido – a se fechar sobre si, devido à consolidação de um atrator, sua temporalidade tende a definir. O processo se torna cada vez mais previsível, até, eventualmente, se transformar em "quisto". [Debrun, 1996, p. 54]

Se a memória é um elemento fundamental na formação dos atratores, a formação desses atratores por si só é muito pouco interessante do ponto de vista estético. Melhor seria a *quase* formação de atratores, ou ainda a formação de atratores dúbios, que mantivesse viva não apenas a temporalidade do sistema, como também o interesse dos participantes do processo e dos eventuais ouvintes. É aqui que a indeterminação, ou nas palavras de Atlan (1992), o ruído, adquire importância. Segundo Foerster (apud. Atlan, 1992, p. 38):

Os sistemas auto-organizadores não se alimentam apenas da ordem, mas também encontram o ruído em seu cardápio... Não é mau ter ruído no sistema. Quando um sistema se fixa num estado particular, ele fica inadaptável, e esse estado final pode ser igualmente ruim. Ele é incapaz de se ajustar a alguma coisa que constitua uma situação inadequada.

4. Modelo Semiótico

Várias aproximações foram experimentadas para o tratamento de interações complexas entre agentes humanos e/ou artificiais. Especialmente na área de *software* para jogos encontramos as que mais se aproximam de nosso problema. A modelagem de jogo de futebol com agentes artificiais realizada por Coelho *et al.* (2001) é um dos exemplos mais interessantes por se aproximar do grau de complexidade que postulamos aqui. Eles utilizam algoritmos genéticos e coevolução para coordenar as estratégias dos agentes durante o jogo e otimizar o comportamento de todo o grupo. Também nos chama a atenção a utilização de redes semiônicas no controle de agentes artificiais em jogos por parte de Tatai e Gudwin (2002), e a ferramenta por eles desenvolvida, o SNTTool, para modelagem e execução de redes semiônicas.

Um modelo de interação musical com as demandas descritas aqui, ocorrendo em ambientes distribuídos abertos, possui muito em comum com o que Pospelov (apud Silva e Gudwin, 2001) chamou de *sistemas abertos complexos de grande porte* (SACGP). Podemos dizer que das características listadas por Pospelov para esse tipo de sistema, o nosso possui: particularidades únicas, ausência de otimalidade, comportamento variável, descrição incompleta

e presença de livre-arbítrio. Apenas a característica da falta de um propósito formalizável de existência não faz parte de nosso sistema, uma vez que seu ponto de início é claro e planejado.

5. Plataforma de jogo

Para a realização deste trabalho propomos a elaboração de um Modelo Semiótico no qual a linguagem de controle situacional (SCL) e suas sub-linguagens descrevam tanto os elementos do domínio do discurso sonoro quanto as regras para sua conexão no tempo e para as possíveis transformações das regras. Ou seja, podemos modelar tanto o processo composicional (no sentido tradicional do termo) quanto as interações sonoras em um jogo de improvisação musical nos termos da SCL, de tal forma que a base de conhecimento semiótico do sistema constitua a “partitura” das possibilidades sonoras e musicais do processo (as regras do jogo *a priori*) as possibilidades de improvisação para os agentes (a transformação dinâmica das regras do jogo *in situ*) e a “partitura de escuta” registrando como o sistema evoluiu (a cristalização do discurso musical *a posteriori*).

É interessante a semelhança que encontramos entre algumas das sub-linguagens da SCL e algumas ferramentas formais já estabelecidas historicamente para a composição musical. Grande parte dos métodos composicionais existentes podem ser modeladas como linguagens de descrição de conhecimento sobre o objeto de controle (DSC) e linguagens para leis de transformação (LTR). É o caso dos métodos composicionais spectro-morfológicos descritos por Smalley (1986), e que serve de base para a elaboração de nosso modelo. Os resolvedores semióticos do sistema são mapeados para agentes que sejam capazes de gerar ou transformar as descrições do material sonoro e dos processos musicais formulados nas sublinguagens da SCL.

Esses agentes estão sendo desenvolvidos com o framework JADE para agentes distribuídos na plataforma Java™, de modo que possam rodar tanto em computadores pessoais de mesa como PDAs, telefones celulares e consoles de jogos, permitindo a formação de redes *peer-to-peer* amplas e heterogêneas. Existem dois tipos básicos de agente: *Player* e *Listener*. Os agentes do tipo *Player* são responsáveis pela geração e/ou transformação de processos musicais descritos pelos elementos da SCL. Eles podem servir apenas de interface para a interferência de agentes humanos no sistema, ou atuar como agentes autônomos interferindo no jogo de acordo com as regras estipuladas pelo compositor do processo de interação. Eles tem ainda a função de capturar os processos produzidos pelos agentes mais próximos na topologia da rede para processamento ou tomada de decisões, assim como a de publicar seus processos para a rede. Os agentes do tipo *Listener* são responsáveis por capturar os processos publicados pelos agentes *Player* mais próximos, misturá-los em um *buffer* comum e transformá-los em áudio.

Com essa plataforma o compositor poderá projetar os processos de interação a partir da criação de diferentes tipos de sub-linguagens da SCL que descrevam os processos musicais de seu interesse. Também através da criação de agentes artificiais autônomos ou agentes que são apenas interfaces para agentes humanos que vão atuar dentro do campo de ação pré-estabelecido para o jogo. O compositor inicia a performance dos agentes artificiais e disponibiliza os agentes de interface a partir de um servidor público.

6. Estado atual do trabalho

Nos últimos meses o trabalho tem sido concentrado na implementação de um protótipo do sistema de agentes. Em especial tem sido estudada a implementação do mecanismo de mistura de descrições dos processos descritos pela SCL e sua posterior transformação em som através de métodos de síntese. Uma SCL simplificada foi criada para a realização de testes.

7. Problemas a resolver e passos futuros

Algumas questões já aparecem como desafios a serem resolvidos após a implementação do primeiro protótipo:

- Como generalizar a relação entre a SCL e os mecanismos para transformá-la em som de modo a permitir a utilização de linguagens criadas pelos usuários do sistema.
- Que tipo de interface deve ser utilizada para a criação das SCLs e das regras de interação entre agentes.
- Que tipos de mecanismos devem ser oferecidos ao compositor para a interferência no processo de interação durante a performance.

Deve ainda ser adicionada ao sistema a capacidade de visualizar em tempo-real e registrar as ações dos agentes. Isso deve ser feito de modo a gerar dados para a análise do processo de interação para a possível detecção e análise de comportamentos auto-organizados.

8. Conclusões

Apesar de a pesquisa estar ainda em seu início, já é muito clara para nós a relação entre os problemas operacionais de nosso projeto e os problemas que o controle situacional semiótico se propõe a resolver. Acreditamos que o aprofundamento nesse campo pode nos permitir no futuro a análise e a síntese de sistemas que permitam a emergência de comportamentos auto-organizados. Também a atuação nesses sistemas de modo a interferir na formação, dissolução ou cristalização de seus atratores. Em termos musicais: interferir na formação de idéias musicais, temas e variações, durante a improvisação dos músicos, e ao mesmo tempo projetá-los em espaço topológico amplo e ao mesmo tempo não localizável que modula os parâmetros da música e determina a própria constituição do material sonoro e suas relações.

9. Referências

- ATLAN, Henri. Entre o cristal e a fumaça: ensaio sobre a organização do ser vivo. Rio de Janeiro, Jorge Zahar, 1992.
- COELHO, André L. V.; WEINGAERTNER, Daniel; GUDWIN, Ricardo; RICARTE, Ivan L. M. Emergence of multiagent spatial coordination strategies through artificial coevolution. *Computer & Graphics*. N. 25, p. 1013-1023. 2001.
- DEBRUN, Michel; GONZALES, Maria Eunice Q.; PESSOA Jr., Osvaldo (orgs). Auto-organização: estudos interdisciplinares em filosofia, ciências naturais e humanas, e artes. Campinas, UNICAMP, Centro de Lógica, Epistemologia e História da Ciência, 1996.
- LASKE, Otto. Toward an Epistemology of Composition. *Interface*, v. 20, p. 235-269. 1991.
- MANZOLLI, Jônatas. Auto-organização: um paradigma composicional. Auto-organização: estudos interdisciplinares em filosofia, ciências naturais e humanas, e artes, p. 417-435. Campinas, UNICAMP, Centro de Lógica, Epistemologia e História da Ciência, 1996.
- TATAI, Victor K.; GUDWIN, Ricardo. Using a Semiotic-Inspired Tool for the Control of Intelligent Opponents in Computer Games. 2002.
- SILVA, Mário Ernesto de S. e; GUDWIN, Ricardo R. Um Tutorial em Controle Situacional Semiótico. V Simpósio Brasileiro de Automação Inteligente. 2001.
- SMALLEY, Denis. Spectro-morphology and Structuring Processes. The Language of Electroacoustic Music. New York, Harwood Academic Publishers, 1986.

Rumo à formalização da Teoria das Árvores Harmônicas

Edilson Fernalda¹, Marcio Brandão², Fernando W. Cruz¹, Fernando S. Goulart Jr¹,
Luciênio de M. Teixeira³, Karen P. de Sousa¹, Marcio G. V. de Souza¹,
Paullus M. de S. N. Castro¹, Maria Carolina F. da Silva¹

¹ PRPGP – Universidade Católica de Brasília (UCB)
SGAN 916 – 70.790-160 – Brasília-DF

² CIC – Universidade de Brasília (UnB)
Caixa Postal 4466 – 70.910-900, Brasília-DF

³ DART – Universidade Federal de Campina Grande (UFCG)
Av. Aprígio Veloso, 880 – Bodocongó – 58.109-970 – Campina Grande, PB
eferneda@pos.ucb.br, brandao@unb.br, fwcruz@ucb.br, fgoulart@ucb.br,
sdart@dart.ch.ufpb.br, karen_ps@ibest.com.br, marcio@funcef.com.br,
paullus@terra.com.br, carollfs@terra.com.br

Abstract. *In this short paper we present the efforts that have been made towards a formalization of the Harmony Trees Theory, which tries to explain the harmonic structures in Brazilian Popular Music. This theory have been used by us in music analysis systems, Musical Harmony Intelligent Tutoring Systems and in the definition of mechanisms for information retrieval in digital music libraries.*

Resumo. *Neste trabalho, apresentamos os esforços que vêm sendo despendidos para a formalização da Teoria das Árvores Harmônicas, que busca explicar as estruturas harmônicas que ocorrem na MPB. Essa teoria vem sendo utilizada por nossa equipe na concepção de sistemas de análise musical, de sistemas tutores inteligentes em harmonia musical e na definição de mecanismos de recuperação de informação em bibliotecas digitais musicais.*

1. Introdução

Num país de dimensões continentais como o Brasil, pode-se facilmente encontrar músicos que detêm significativos conhecimentos acumulados, mas sujeitos ao desaparecimento, por falta de registros formais. É o caso de José de Alencar Soares, violonista autodidata que fez carreira em Brasília como músico e professor e que vem construindo, por mais de 15 anos, uma compilação das estruturas harmônicas mais comuns encontradas em música popular. Dessa compilação, ele construiu um corpus de canções populares e o utilizou numa análise estatística informal, fornecendo a probabilidade de ocorrência de seqüências harmonias específicas, e propôs uma teoria de harmonia musical por ele chamada *Teoria das Árvores Harmônicas*. Essa teoria tem sido utilizada com sucesso para explicar as estruturas harmônicas que ocorrem na MPB.

2. A Teoria das Árvores Harmônicas

A Teoria das Árvores Harmônicas é representada por diagramas que mostram graficamente as possibilidades harmônicas. O diagrama da Figura 1 representa as árvores harmônicas básicas (*a* e *c*) e estendidas (*b* e *d*), que contêm as progressões harmôni-

cas encontradas na maioria das músicas populares. Numerais romanos são usados para representar acordes relativos ao primeiro grau da tonalidade. Quando dois numerais romanos aparecem, o primeiro representa o grau relativo ao segundo. Por exemplo, $V7/II$ na tonalidade dó maior representa o acorde $A7$, que é o quinto grau relativo ao segundo grau (D) da escala de dó maior. Os quadrados representam acordes dominantes que preparam as resoluções, as quais estão representadas com círculos no diagrama.

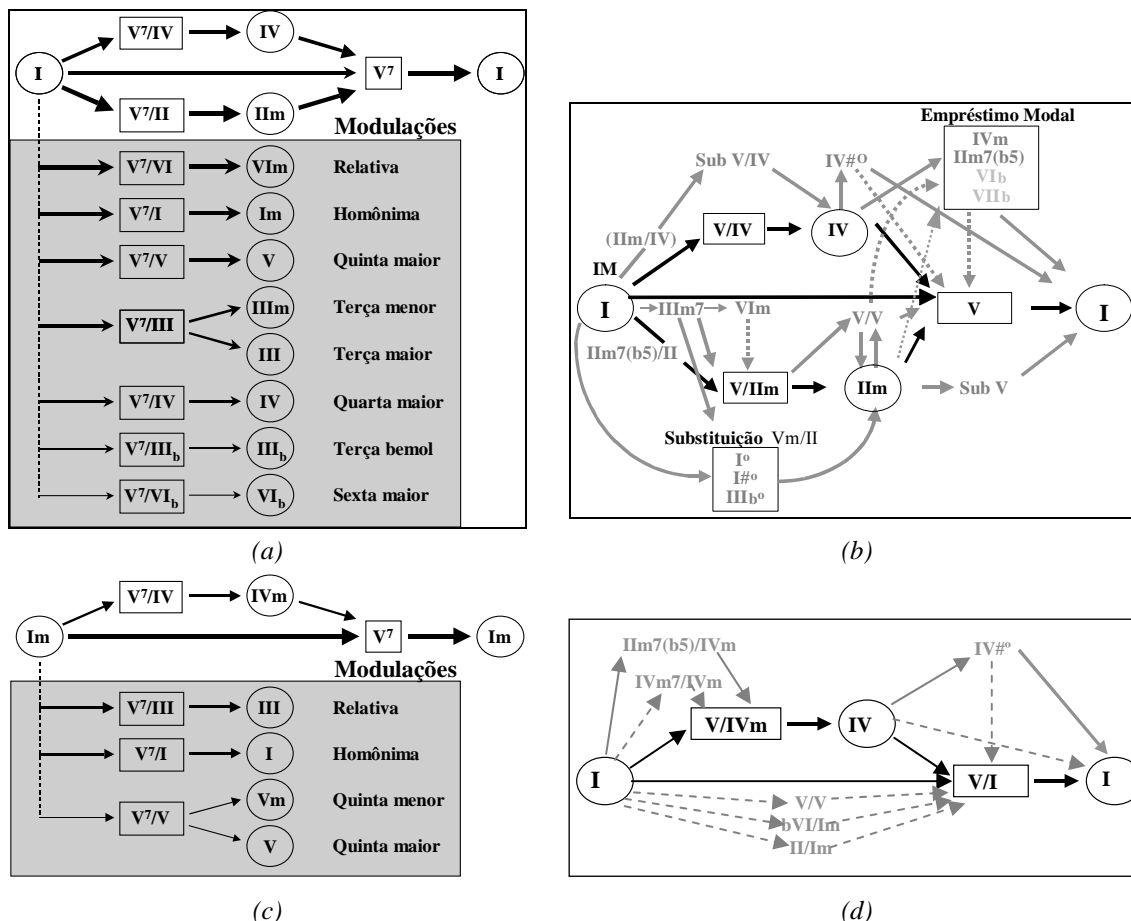


Figura 1: Árvore Harmônica maior, em sua forma geral simplificada (a) e estendida (b), e menor, também em sua forma geral simplificada (c) e estendida (d).

Na Fig. 1a, a área acinzentada representa as possíveis modulações presentes na MPB. As linhas tracejadas indicam caminhos com menor incidência de ocorrência. A espessura das setas está associada à frequência com que um caminho ocorre em músicas populares. Assim, o caminho $I - V7/II - II_m - V7 - I$ representa a progressão mais usada em tons maiores e a modulação mais comum é a que ocorre para uma tonalidade relativa, como, por exemplo, a sequência $C-E7-A_m$ para a tonalidade C . A Fig. 1b apresenta uma versão mais completa da árvore maior, que incorpora ao esquema funções envolvendo acordes dissonantes, substituições e empréstimos modais. Por exemplo, a sequência $C7M-A7/b13-Dm7/9-G7/9-C7M$ pode ser vista como uma instância dissonante do caminho $I-V7/II-II_m-V7-I$ na tonalidade C . Outra instância para esse caminho pode ser a sequência $C7M-C^\#o-Dm7-Ab7-G7-C7M$, que representa um dos caminhos alternativos ($I7M-I^\#o-II_m-VI_b-V7-I$).

3. Um sistema de apoio ao ensino de harmonia

O projeto HeART se propõe a desenvolver um sistema capaz não somente a facilitar o aprendizado da Harmonia segundo os preceitos da Teoria das Árvore Harmônicas, como também ser um recurso voltado para a análise musical em ambiente Web. Nesse sentido, um primeiro protótipo (chamado HART), ainda numa versão *standalone*, foi desenvolvido (Sousa, Souza & Castro, 2004). Na Figura 2, é apresentada a interface do sistema *HART* para árvores harmônicas simplificadas.

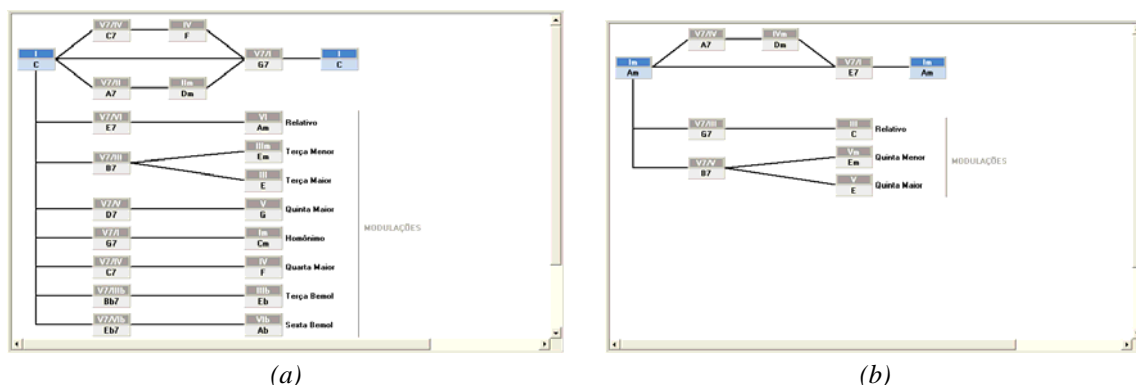


Figura 2: Interface do sistema *HART* para árvores harmônicas simplificadas maior (a) e menor (b)

A partir de um arquivo texto contendo uma sequência de acordes referentes a uma obra musical, o sistema produz sua análise. Por exemplo, para a sequência C-A7-Dm-G7-C, o sistema produz uma análise como mostrado na figura 3.

```
O tom da música é C
Acorde 001: A música iniciou-se pelo tom (I): C
Acorde 002: Preparação para a Segunda (V7/II): A7
Acorde 003: Segunda Menor (IIIm): Dm
Acorde 004: Dominante [Preparação para a Primeira] (V7/I): G7
Acorde 005: Tom (I): C
```

Figura 3: Exemplo de análise realizada pelo sistema *HART*

Nessa primeira versão, além das regras de formação das Árvore Harmônicas, as seguintes regras de análise harmônica foram inseridas dentro do sistema:

1. Não ocorre modulação quando não houver preparação. Entenda-se por preparação o quinto acorde relativo ao acorde da modulação a ser realizada. Por exemplo, quando uma música estiver na tonalidade *F*, para que ocorra modulação para o tom *C*, antes de ser tocado o acorde *C*, o seu quinto grau (*G*) deverá ser tocado.
2. A preparação de uma modulação pode ser precedida pelo segundo grau da tonalidade da modulação. No entanto, existem, na estrutura da árvore harmônica estendida, acordes que coincidem com os acordes que precedem a modulação. Nestas situações, o analisador verifica o próximo acorde e as seguintes regras se aplicam nestes casos:
 - 2.1 A segunda da tonalidade relativa é igual ao acorde de substituição do quinto grau da tonalidade atual. Caso o próximo acorde seja a preparação para a relativa (*V7/VIm*), o acorde atual será a segunda da relativa. Caso contrário, o acorde atual será o substituto da quinta.
 - 2.2 A segunda da quinta maior é igual ao acorde de substituição da tonalidade atual (*VIm*). Caso o próximo acorde seja igual à preparação para a quinta maior (*V7/V*), o acorde atual será a segunda da quinta maior. Caso contrário, o acorde será o substituto da tonalidade.

- 2.3 A segunda da terça bemol é igual ao quarto grau da homônima, que é, na verdade, um acorde de empréstimo modal da tonalidade atual. Caso o próximo acorde seja igual à preparação para a terça bemol ($V7 / IIIb$), o acorde atual é a segunda da terça bemol. Caso contrário, o acorde é de empréstimo modal.
- 3. Se o acorde atual corresponder à quarta maior, só ocorrerá modulação se o próximo acorde não for encontrado na árvore harmônica atual, na árvore estendida ou nos acordes de empréstimo modal.
- 4. Se o acorde atual corresponder à quinta da quinta da árvore estendida, as seguintes regras devem ser seguidas:
 - 4.1 Se o acorde logo após o acorde atual for a própria tonalidade, o acorde atual corresponde realmente à quinta da quinta e não ocorre modulação.
 - 4.2 Caso contrário, se o próximo acorde for a dominante, verificar se ocorre uma sétima no acorde:
 - 4.2.1 Caso ocorra uma sétima no próximo acorde, o acorde atual corresponde à quinta da quinta e não ocorre modulação.
 - 4.2.2 Senão, o acorde atual corresponde à preparação para a modulação para a Quinta Maior.

3. Conclusão

Atualmente, o grupo trabalha no aprofundamento da formalização da Teoria das Árvores Harmônicas para sua utilização em sistemas de análise musical, em sistemas tutores inteligentes (Ferneda et al, 2004a, 2004b) e na definição de mecanismos de recuperação de informação em bibliotecas digitais musicais (Cruz et al, 2004). Espera-se, assim, contribuir para a difusão, discussão e validação dessa teoria através da disponibilização dessas ferramentas em um ambiente para a criação de comunidades virtuais voltadas para a produção intelectual e artística e a troca de experiências em MPB (Ferneda et al, 2004c).

Referências bibliográficas

- Cruz, F. W. et al. (2004) "Brazilian Popular Music Oriented Digital Library For Musical Harmony E-Learning". Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04), p. 429-432. Barcelona (Espanha).
- Ferneda, E. et al. (2004) "A web-based cooperative e-learning environment for musical harmony domain". Proceedings of the 3rd IASTED International Conference on Web-Based Education (WBE 2004), pp. 43-47, Innsbruck (Áustria).
- Ferneda, E. et al. (2004b) "A cooperative virtual learning environment on the web for musical harmony". Advanced Technology For Learning, Vol. 1, nº 3, pp. 147-155.
- Ferneda, E. et al. (2004c) "A Virtual Community Environment for Brazilian Popular Music". Proceedings of the IEEE International Conference on Advanced Learning Technology (ICALT 2004), pp. 926-930. Joensuu (Finlândia).
- Sousa, K. P. de; Souza, M. G. V. de; Castro, P. M. de S. N. (2004) "HART - um sistema de apoio à análise musical baseado na teoria das árvores harmônicas". Projeto Final, Bacharelado em Ciência da Computação, Universidade Católica de Brasília.

Parsing Incremental para Acompanhamento em Tempo Real

Giordano Cabral¹, Jean-Pierre Briot¹, François Pachet²

¹Laboratoire d'Informatique de Paris 6 – Université Pierre et Marie Curie
8 Rue du Capitaine Scott 75018 Paris – France

²Sony Computer Science Lab Paris
6 Rue Amyot 75005 Paris – France

{Giordano.CABRAL, Jean-Pierre.BRIOT}@lip6.fr, pachet@csl.sony.fr

Abstract. *The incremental parsing (IP) algorithm has been successfully used in real-time applications, due to its efficiency and power of modeling musical style. Musical systems using IP can continue phrases played by a musician in a consistent way. This paper proposes some alterations in the original algorithm in order to allow the development of systems able to respect adaptation and continuity in musical accompaniment.*

Resumo. *O algoritmo de parsing incremental tem sido usado para o desenvolvimento de aplicações musicais em tempo real devido à sua eficiência e capacidade de modelar estilos musicais. Esta capacidade permite a tais sistemas continuarem frases tocadas por um músico, coerentemente ao estilo destas. Este artigo propõe uma alteração no algoritmo de modo a permitir o desenvolvimento de sistemas que além de guardar uma coerência estilística, se adaptem ao que está sendo tocado pelos outros músicos, características básicas do acompanhamento.*

1. Introdução e Problemática

A análise estatística de um corpus de material musical indica possibilidades de recombinação que se adequam às restrições e redundâncias típicas de um determinado modelo. Foi comprovado que a construção de cadeias de Markov [Dubnov et al. 1998] a partir de uma base de exemplos é um tipo de aprendizagem bastante eficaz para modelar o comportamento dinâmico de melodias. Este paradigma foi utilizado tanto na classificação de estilos musicais [Lartillot et al. 2001] quanto em sistemas interativos [Pachet 2002]. Estes últimos são capazes, por exemplo, de continuar frases tocadas por um músico, coerentemente ao estilo destas. Para implementar eficientemente as cadeias de Markov, é utilizado o algoritmo de *parsing* incremental (PI) [Ziv and Lempel 1977], projetado inicialmente como método de compressão.

O presente trabalho toma como ponto de partida o *Continuator*, o estado da arte de tais sistemas. O projeto explorou diversos modos de interação possíveis, como o de questão-resposta, mas mostrou-se particularmente deficiente no modo de acompanhamento. De fato, a dificuldade encontra-se em ser capaz ao mesmo tempo de gerar frases musicais coerentes a um determinado estilo, e de adaptar-se ao que está sendo tocado pelos outros músicos. A Figura 1 ilustra esse problema no caso de um

sistema destinado a acompanhar uma melodia cantada. Os acordes escolhidos para realizar o acompanhamento devem manter uma continuidade, mas também se encaixarem à melodia que está sendo tocada. No caso de um sistema em tempo real, uma dificuldade a mais é encontrada: não se sabe à priori o que vai ser cantado. Neste caso, o algoritmo deve ter a capacidade, além de adaptação e continuidade, de predição.

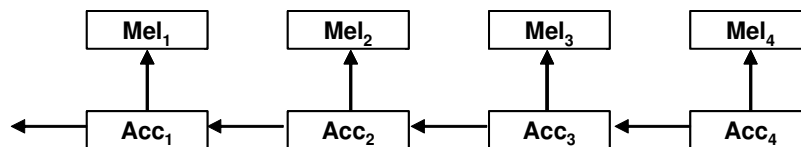


Figura 1 – relação de dependência no acompanhamento. Os acordes dependem não só dos acordes anteriores mas também da melodia que está sendo tocada.

Este trabalho propõe uma alteração no algoritmo de PI de forma que ele seja capaz de também levar em consideração a adaptação aos demais instrumentos. Com isso, acreditamos que o método será apropriado para ser utilizado em sistemas de acompanhamento musical. A próxima seção descreve o PI tradicional. A seção 3 apresenta os diversos compromissos possíveis entre adaptação, continuidade e predição apresentados por músicos no momento do acompanhamento. A seção 4 descreve as alterações possíveis no *parsing* incremental.

2. *Parsing* Incremental

Este algoritmo provém da fase de análise das técnicas de compressão Lempel-Ziv [Ziv and Lempel 1977]. Desde que a música seja interpretada como uma sequência de notas, a técnica é aplicável. O algoritmo original é dividido em duas partes: primeiro, ele lê a sequência de entrada e gera um modelo que captura a redundância. Em seguida, ele gera o código comprimido desta sequência com relação ao modelo. Em um sistema interativo, a segunda parte é substituída por uma simulação estocástica do modelo a fim de produzir novas sequências.

O PI lê a sequência incrementalmente. A cada ciclo, ele escolhe o padrão mais curto que ainda não existe no dicionário de sub-padrões, e o adiciona. Uma representação otimizada é uma árvore de sufixos onde os contextos são invertidos (i.e. da folha à raiz), e as continuações são armazenadas como índices associados a cada nó. Para pesquisar uma sequência, basta navegar na árvore de sufixos. A Figura 2 ilustra o funcionamento do algoritmo.

Suponhamos que queiramos construir o modelo relativo às sequências [C,A,F,E] seguida por [C,C,F,G]. Primeiro, o algoritmo lê a subsequência [C,A,F], cuja continuação tem índice 4. O ramo [C,A,F], onde cada elemento possui o valor 4 como continuação. Em seguida, é lida a subsequência [C,A], cuja continuação tem índice 3. Finalmente a subsequência [C] é lida, criando o último ramo (a árvore completa é a Figura 2a). Em seguida, uma nova sequência, [C,C,F,G], é observada. O mesmo processo é feito. O ramo [C,C,F], por exemplo, é anexado ao ramo [C,A,F], que tem a mesma raiz. A árvore final é mostrada na Figura 2b. Este mecanismo faz com que apenas uma nova informação seja adicionada. Para procurar uma continuação, basta navegar na árvore, procurando a maior subsequência. Por exemplo, para a sequência [E,A,F], a resposta é a continuação de índice 4 (E), referente à subsequência [A,F].

a) IP original, 1ª sequência

b) IP original, 2ª sequência e consulta

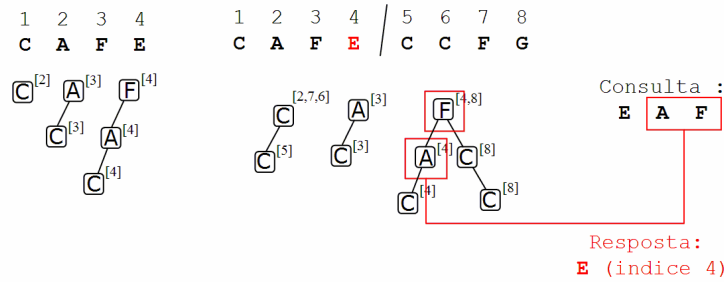


Figura 2 – árvores de sufixos geradas pelo IP para as sequências [C,A,F,E] e [C,C,F,G], e consulta à mesma. Em vermelho, a resposta.

3. Acompanhamento Musical

É extremamente difícil criar um sistema de acompanhamento musical interativo, em tempo real, sobretudo se a música não é conhecida à priori. Nós pretendemos modelar e comparar 3 comportamentos diferentes do acompanhador que nós julgamos particularmente pertinentes, combinando de maneiras diferentes predição, adaptação e continuidade.

1. Predição/Recuperação – de acordo com a melodia que foi tocada, prevê-se uma continuação. Em seguida, procura-se um acompanhamento para esta continuação. A continuidade é mantida à medida que a melodia segue o caminho previsto.
2. Recuperação/Predição – de acordo com a melodia que foi tocada, procura-se um acompanhamento. Em seguida, procura-se uma continuação para este acompanhamento. A continuidade do acompanhamento é naturalmente respeitada, mas a adaptação à melodia não é mais garantida.
3. Conjunta – a melodia e o acompanhamento são vistos como uma única entidade, onde a predição, a adaptação e a continuidade são pensadas simultaneamente.

4. Parsing Incremental Alterado

Para cada um dos comportamentos acima, nós previmos um algoritmo alterado. Para o comportamento 1, nossa proposição é alterar a estrutura das árvores de sufixos (Figura 3). Ao invés de armazenar as possíveis continuações, seriam armazenados os possíveis acompanhamentos. Para o comportamento 2, nossa proposição é criar duas árvores de sufixos separadas, uma para a melodia, outra para o acompanhamento (Figura 4). Cada folha da árvore referente à melodia apontaria para um nó na árvore relativa aos acordes. Para o comportamento 3, nossa proposição é mixar os estados. Cada elemento da sequência não seria mais uma única nota (da melodia), ou um único acorde (do acompanhamento), mas uma tupla <nota, acorde> (Figura 5).

Suponhamos que as sequências [C,A,F,E] e [C,C,F,G] são acompanhadas pelos acordes [Am,%F,E7] e [Am,F,E7,%], onde “%” indica que o acorde é mantido. Neste momento é observada a sequência [E,A,F], para a qual será procurado um acompanhamento.

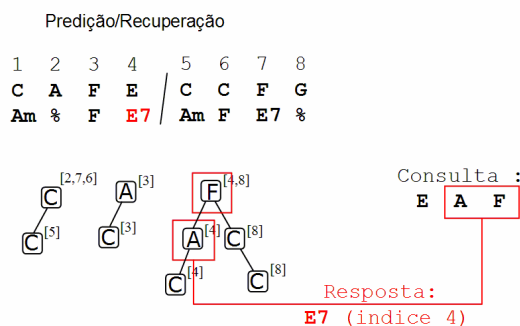


Figura 3 – exemplo de funcionamento com o algoritmo de predição/recuperação.

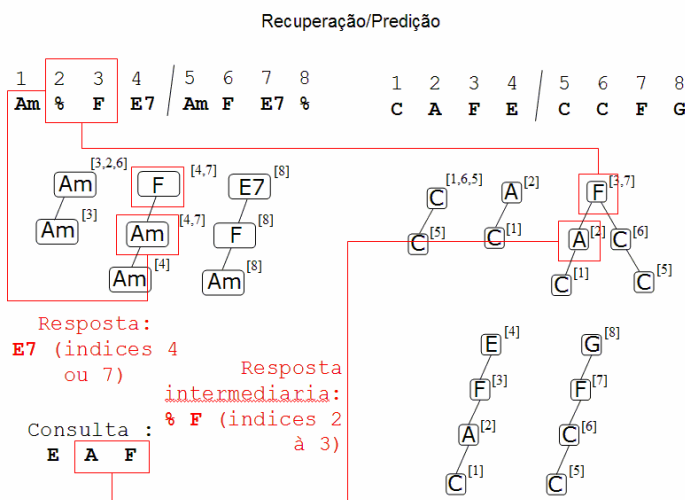


Figure 4 – exemplo de funcionamento com o algoritmo de recuperação/predição.

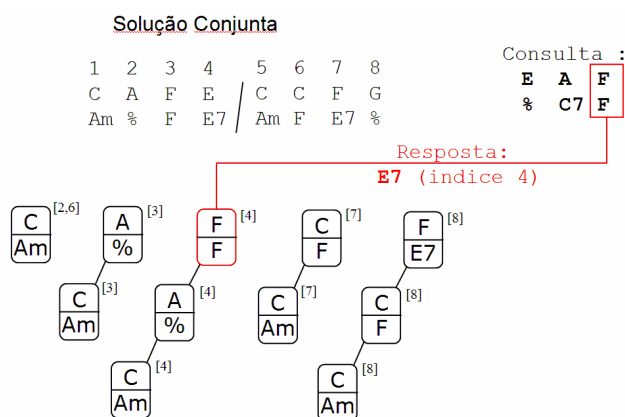


Figura 5 – exemplo de funcionamento com o algoritmo conjunto.

O caso 1 é ilustrado na Figura 3. A árvore é semelhante à anterior, apenas os índices referem-se desta vez aos acordes do acoplamento, e não à continuação. A árvore é navegada a partir da última nota (F), encontrando-se então a maior subsequência. Neste caso: [A, F].

No caso 2, a maior subsequência de notas encontrada retorna uma subsequência de acordes (no exemplo, os acordes entre os índices 2 a 3). É procurada então a continuação desta sequência de acordes (índices 4 ou 7), como visto na Figura 4.

No caso 3, os acordes que foram tocados também são considerados no momento da consulta. Digamos que os acordes previamente selecionados haviam sido [%C7F], a consulta torna-se então [E|%,A|C7F|F] (Figura 5). Neste caso, a maior subsequência encontrada não é mais de tamanho 2 (como o [A,F] anterior), mas de tamanho 1 ([F|F]), o que mostra que o algoritmo tem maior dificuldade em encontrar soluções.

Tais algoritmos estão em vias de implementação. Esperamos em breve efetuar experimentos com músicos e amadores, de forma a avaliar quais os pontos fortes e fracos de cada um, investigar em que pontos o sistema realmente emula uma aptidão do músico, e jogar um pouco de luz sobre o tema.

Nós gostaríamos de agradecer à equipe do Sony Computer Science Lab em Paris pelo apoio.

Esta pesquisa é patrocinada por CAPES/COFECUB, Brasil/França.

References

- O. Lartillot, S. Dubnov, G. Assayag and G. Bejerano. (2001) Automatic Modeling of Musical Style. In International Computer Music Conference, La Havana.
- S. Dubnov, G. Assayag, and R. El-Yaniv. (1998) "Universal Classification Applied to Musical Sequences". In Proceedings of International Computer Music Conference, pp. 332-340.
- Pachet. P. (2002) "The Continuator: Musical Interaction with Style". In ICMA, editor, Proceedings of ICMC, pp. 211-218.
- Ziv J., Lempel A. (1977) "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

Sistemas Dinâmicos Não-Lineares e Organicidade no Material Musical

Daniel Fils Puig

Colégio de Aplicação – Universidade Federal do Rio de Janeiro (UFRJ)
Rua J. J. Seabra, s/nº – 20000-000 – Rio de Janeiro – RJ – Brazil

daniel.puig@terra.com.br

Abstract. *After presenting briefly some important aspects of non-linear dynamic systems studied as part of Chaos Theory, we discuss two concepts brought forth by Malt and Xenakis concerning their use in musical composition, as for the pursuit of a certain organicity in the musical material generated. As examples, applied to Computer Aided Composition, we present three Max/MSP patches, with four different applications.*

Resumo. *Após apresentar sucintamente aspectos importantes dos sistemas dinâmicos não-lineares estudados pela Teoria do Caos, discutimos dois conceitos apresentados por Malt e Xenakis acerca da sua utilização na composição musical, tendo em vista a busca de uma certa organicidade no material musical gerado. Como exemplos, aplicados à Composição Assistida pelo Computador, apresentamos três patches construídos em Max/MSP, com quatro aplicações diferentes.*

A Teoria do Caos, como ressalta Wallin (1989), talvez tenha sido a primeira revolução científica disponível comercialmente e acompanhada passo a passo pelo grande público. Tendo sua divulgação tomado fôlego no início da década de 90, não demorou a exercer influência sobre o campo das artes. A partir das implementações para fins musicais de modelos matemáticos dela provenientes em diversos softwares de Composição Assistida pelo Computador (CAC), como *CSound*, *Max/MSP*, *PatchWork*, *OpenMusic*, *Pd*, entre outros, podemos facilmente inferir que a sua aplicação em Música não constitui uma novidade. O assunto já é conhecido dos pesquisadores da área e vários compositores a têm utilizado em seus trabalhos de criação.

Este artigo é um extrato de algumas questões levantadas em nossa dissertação de Mestrado em Composição junto ao Programa de Pós-Graduação em Música da Escola de Música da Universidade Federal do Rio de Janeiro (PPGM/EM/UFRJ), concluído em fevereiro do corrente, sob a orientação do Prof. Dr. Rodrigo Cicchelli Velloso. Nela abordamos a aplicação de sistemas dinâmicos não-lineares provindos da Teoria do Caos na composição de forma geral e na CAC especificamente.¹ Aqui, pretendemos mostrar algumas das aplicações desenvolvidas em Max/MSP durante esse curso e que vão de encontro à nossa busca por uma certa "organicidade" no material musical.

Sistemas Dinâmicos Não-Lineares e a Teoria do Caos

Sistemas dinâmicos não-lineares são sistemas cujas grandezas que os descrevem evoluem no tempo de forma não-periódica, irregular, aleatória e, portanto, imprevisível. Os modelos matemáticos que procuram representar seu comportamento apresentam características que têm renovadamente surpreendido os pesquisadores ao longo das últimas décadas.

O fato, por exemplo, de que modelos matemáticos simples são capazes de gerar comportamentos extremamente complexos, atirou por terra a concepção tradicional de que

¹ Puig (2005).

efeitos complexos têm necessariamente causas complexas. Isso se dá devido à característica de dependência hipersensível das condições iniciais, conhecida como o "Efeito Borboleta"². Tal dependência gera resultados qualitativamente muito diversos para um mesmo sistema, dada uma minúscula variação em suas condições iniciais. Outras características intrigantes desses modelos são a presença de "janelas de ordem" em meio ao caos (trechos em que os resultados parecem seguir uma sequência ordenada, para depois caírem novamente numa forma não-periódica), a possível ocorrência de uma persistente auto-semelhança em escalas (ligada à já bastante estudada geometria fractal) e a da universalidade de sua aplicação (ou seja, a possibilidade de descrever um comportamento caótico em dinâmica de fluidos, circuitos elétricos ou em biologia, a partir do mesmo sistema de equações não-lineares).

Não podemos perder de vista, porém, o fato de que esses modelos matemáticos são deterministas, ou seja, dadas *exatamente* as mesmas condições iniciais, eles irão reproduzir fielmente os mesmos resultados, o que não constitui um caos *stricto sensu*, onde a imprevisibilidade é total. Por esse motivo a Teoria do Caos também é chamada de o estudo do caos determinístico.

Organicidade com o Caos

Em primeiro lugar, o que mais nos interessa no uso de aspectos advindos da Teoria do Caos na composição musical, é a busca de uma qualidade quase que “orgânica” no material musical, de construções musicais que se aproximem da forma com que o homem vê o universo hoje: cheio de detalhes, mas ao mesmo tempo uno; possuidor de eventos caóticos, que tanto provém da ordem quanto podem gerá-la; com elementos auto-semelhantes, mas nunca iguais. Dois conceitos, extraídos de Malt e Xenakis, têm sido de grande valia nessa busca.

Malt (2000, 4.4 Conclusions, p. 1) refere-se à necessidade de evitar "conferir aos modelos e aos símbolos um privilégio em relação ao processo que eles representam", esclarecendo que "uma lógica formal não implica forçosamente em uma lógica musical", ou seja, "um modelo matemático não pode ser utilizado na composição musical se não for sustentado por uma conceitualização e um pensamento musicais". Tal afirmativa é fundamental para uma utilização consciente na composição. O pensamento e a conceitualização musicais devem ter ascendência sobre a utilização do modelo, que serve, então, de apoio àqueles.

Já Xenakis (1992), no capítulo X do seu livro "Formalized Music", expõe a idéia de que o universo se perpetua por seus eventos serem ao mesmo tempo singulares e semelhantes. A semelhança leva ao reconhecimento de regras e estruturas, assegurando a permanência, e a singularidade ao desaparecimento, à morte. Assim o universo se perpetua, ou seja, mantém a vida. Desta forma, Xenakis deixa claro o que parece ser um dos motivos primordiais para a aplicação de processos randômicos, sistemas estocásticos ou o caos determinístico na composição musical: fazer com que a música produzida acerque-se mais da visão que o homem tem hoje da natureza, de suas formas e ritmos, da maneira com que ela funciona, onde as coisas são “mais ou menos” iguais.

Para nós, essa qualidade de “auto-similaridade desviante”³ — se assim podemos nos expressar — onde as coisas são “mais ou menos” iguais, leva, portanto, a uma maior organicidade, refletindo um processo que a própria vida nos coloca frente aos olhos. Orgânico é tudo aquilo que é relativo a, ou próprio de organismos. Nesse sentido, os

² Esta denominação vem do título de uma palestra proferida por um dos pioneiros da Teoria do Caos, o meteorologista Edward Lorenz, em 1972, na American Association for the Advancement of Science: "Previsibilidade: Pode o bater das asas de uma borboleta no Brasil provocar um tornado no Texas?".

³ Cunhamos essa expressão unicamente para diferenciar essa qualidade de auto-similaridade daquela dos fractais, cuja característica marcante é apresentar-se estritamente igual.

sistemas dinâmicos não-lineares são capazes de ajudar a gerar e transformar material musical que se aproxime dessa organicidade, por terem a auto-similaridade como característica básica dentro de um comportamento caótico, o qual por sua vez provoca singularidades. Tanto Wallin (1989)⁴, quanto Slater (1998), apontam ainda para o fato de que o acoplamento de dois ou mais sistemas dinâmicos não-lineares resulta em uma maior auto-similaridade desviante (embora não se utilizem desta expressão). Apesar deste não ter sido o foco de nossa pesquisa no Mestrado, encontramos poucos autores que se referissem a essa utilização como uma forma de aumentá-la. Acreditamos que essa questão mereça maior atenção e estudos pormenorizados e é com esse intuito que apresentamos os resultados práticos obtidos com os *patches* descritos abaixo.

Exemplos em CAC

Os três *patches* aqui apresentados (*mar.pat*, *pri.pat* e *circpan-caospan.pat*) foram construídos para serem difundidos através de um sistema quadrafônico, cujos alto-falantes estejam colocados na posição das arestas de um quadrado no espaço de difusão.

"mar"

Utilizado em nossa composição "Revoada"⁵, este *patch* (*mar.pat*) implementa uma modificação na velocidade de reprodução de um pequeno trecho gravado (21 segundos) do som de ondas do mar, a partir dos resultados numéricos do sistema de equações não-lineares de Hénon-Heiles⁶. O arquivo original é acessado a partir de um *buffer~* por dois objetos *groove~* diferentes. Cada um destes últimos reproduz o som do mar variando sua velocidade de acordo com o fluxo dos resultados de uma das variáveis escalonadas (x e y' escalonados entre 0,1 e 2,5; onde 1,0 corresponde à velocidade original). Desta forma obtemos duas reproduções diferentes do mesmo arquivo de som. Estas são difundidas, por sua vez, em alto-falantes diametralmente opostos no espaço de difusão.

⁴ Em nossa dissertação de Mestrado refizemos, com o auxílio de um *patch* em Max/MSP, a comparação entre os resultados musicais obtidos pelo mapeamento direto para alturas dos resultados da Equação Logística e de três sistemas não-lineares simetricamente acoplados, ao qual Wallin se refere em seu artigo. Ver Puig (2005), Capítulo 3: "Mexendo no Caos".

⁵ As partituras e os *patches* de "Revoada" e "inconfidências" podem ser encontradas como anexos de nossa dissertação de Mestrado. Ver Puig (2005).

⁶ Esse modelo caótico descreve o movimento de uma estrela no potencial gravitacional de uma galáxia e seus resultados numéricos são apresentados em quatro dimensões (x , y , x' e y'). Cada conjunto de resultados refere-se a uma iteração do cálculo do sistema e à condição do sistema decorrido um pequeno intervalo de tempo. A implementação dele para Max/MSP 4.x através do objeto *a-henon-heilles* foi feita por André Sier como parte do conjunto de objetos "A-Chaos Lib 1.0", baseado nos trabalhos de Richard Dudas, Mikhail Malt e Paul Bourke.

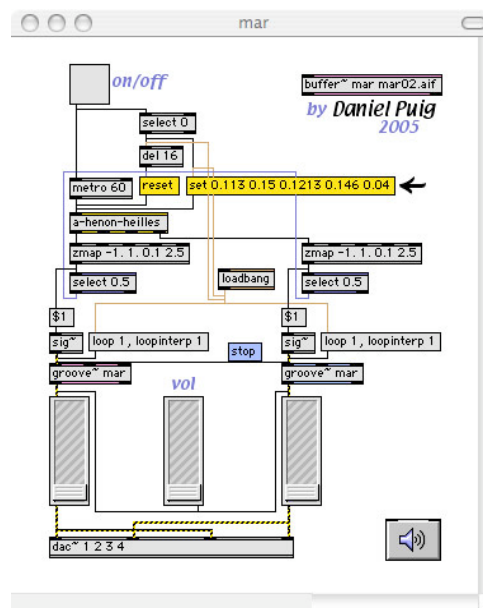


Figura 1. O patch *mar.pat* em Max/MSP. O conjunto de condições iniciais encontra-se destacado por uma seta.

O resultado auditivo é intrigante, uma vez que se sabe que se trata apenas de uma gravação de 21 segundos, pois a reprodução é constantemente variada e não se repete. Isto se dá devido ao conjunto de valores dados como condição inicial do sistema (ver Figura 1), que provoca um comportamento caótico, resultando em uma sequência de valores não-periódica para o par (x, y') .

A organicidade do material musical assim obtido fica clara na emulação bastante próxima de um ambiente sonoro natural. Embora esta não seja perfeita, tem a capacidade de passar ao ouvinte tal impressão. Na nossa experiência, ouvintes atentos são capazes de perceber que há uma repetição, mas não sabem precisar onde ou se esta não é decorrente do próprio comportamento da fonte sonora original.

"pri"

No segundo exemplo (*pri.pat*) o mesmo sistema de equações não-lineares de Hénon-Heiles é utilizado para perturbar quatro osciladores em relação à sua frequência central e sua amplitude. Estes osciladores estão combinados dois a dois, possuindo cada par os mesmos valores de frequência (960Hz e 925Hz; ver Figura 2, no objeto *cycle~*). São utilizados dois sistemas de Hénon-Heiles submetidos a condições iniciais bastante próximas (diferindo apenas em um valor, na ordem da quarta casa decimal; ver Figura 2, em destaque), cujas trajetórias vão divergindo aos poucos, diferenciando-se em movimento. Subitamente parecem sincronizar-se novamente, para logo seguirem trajetórias diversas. Nasce a impressão auditiva do canto de insetos à noite.

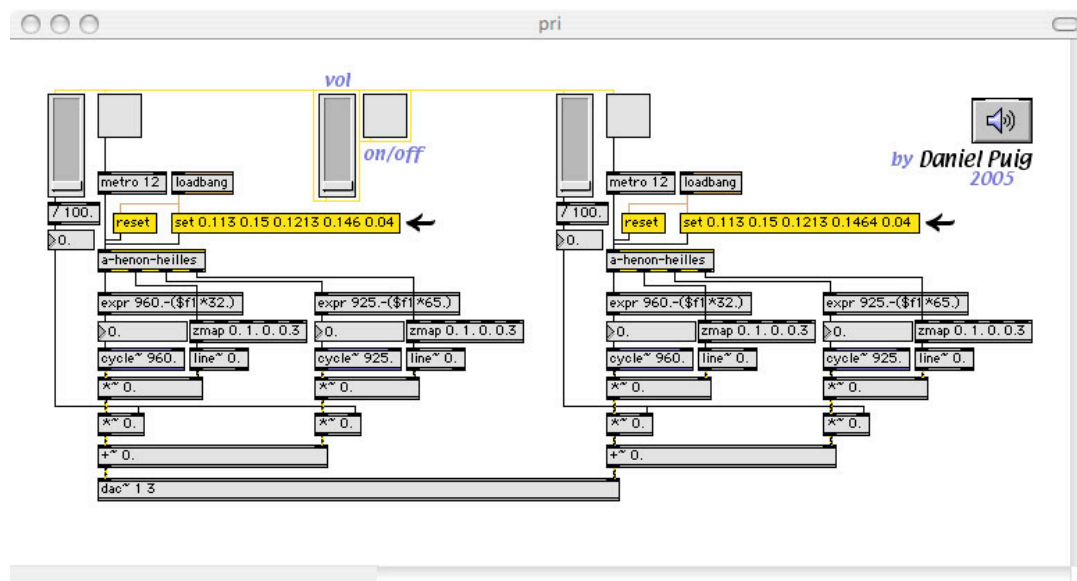


Figura 2. O patch *pri.pat* em Max/MSP. Note-se a pequena diferença entre os dois conjuntos de condições iniciais dadas (destacados por setas).

Esta aplicação foi inspirada na referência que o Prof. Dr. Rodolfo Caesar fez em um dos cursos durante o Mestrado, ao fato de que insetos noturnos poderiam produzir suas "vozes" a partir de uma síntese que utilizasse osciladores de baixa frequência, os chamados LFOs. Daí a imaginar essa síntese a partir da oscilação vagarosa dos valores produzidos por um sistema de equações não-lineares como o de Hénon-Heiles, cujo gráfico apresenta uma curva contínua e ondulada (ver Figura 3), não custou muito.

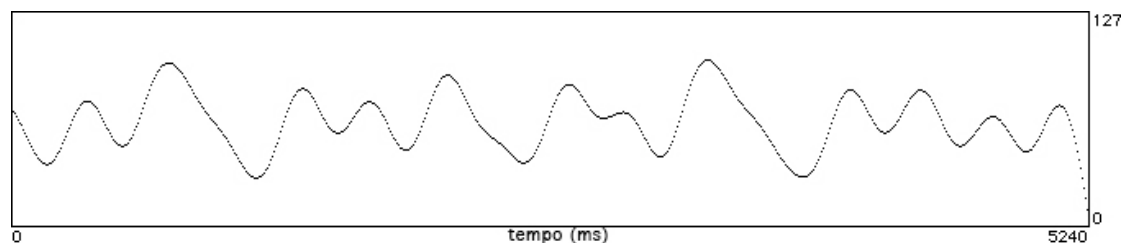


Figura 3. Representação gráfica dos valores obtidos (escalonados para valores entre 0 e 127) para uma das variáveis do sistema de equações não-lineares de Hénon-Heiles, submetido a condições iniciais quaisquer, como exemplo do seu comportamento ondulado e contínuo.

Não foi nosso intuito aqui emular exatamente uma paisagem sonora natural (embora acreditemos que isso seja possível a partir deste mesmo princípio), mas apenas causar a impressão dela através do uso da auto-similaridade desviante, remetendo o ouvinte a essa lembrança. Tal gesto musical é utilizado para esse fim no início de nossa composição "inconfidências".

"circpan" e "caospan"

Os algoritmos de espacialização apresentados neste exemplo fazem parte de nossa composição "Revoada". "circpan" foi uma criação nossa e "caospan" foi idealizado e criado em conjunto com o Prof. Dr. Rodrigo Cicchelli Velloso.

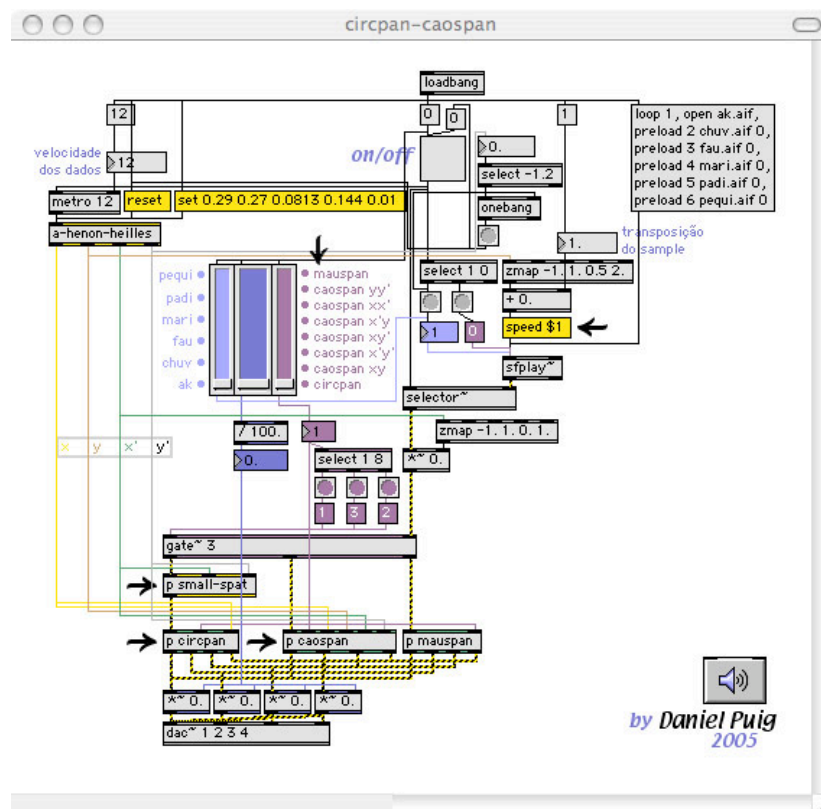


Figura 4. O *patch circpan-caospan.pat* em Max/MSP. Em destaque, indicados por setas, o comando "*speed*", a lista de possibilidades para a espacialização e os *subpatches* referentes aos algoritmos de espacialização.

"Revoada" surgiu a partir da observação de revoadas de pombos, onde todo o grupo de aves mantém simultaneamente trajetórias parecidas, mas não iguais. A utilização do sistema de Hénon-Heiles para emular esse movimento é um exemplo curioso da universalidade como característica de modelos matemáticos de sistemas não-lineares, pois os gráficos produzidos por esse modelo específico, e observados por nós pela primeira vez no software *PatchWork*, retornavam curvas que poderiam ser usadas para descrever essas trajetórias da revoada das aves em um espaço tridimensional, chegando a reproduzir detalhes, para certos conjuntos de condições iniciais, como o pequeno movimento ascendente característico antes do pouso (ver Figura 5).

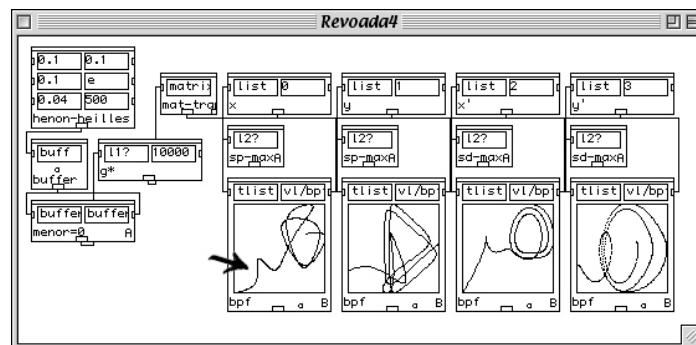


Figura 5. *Patch* construído no software *PatchWork* mostrando os gráficos no espaço de fases para as variáveis do sistema de Hénon-Heiles combinadas duas a duas ((x,y) , (x,y') , (x',y) e (x',y')). Note-se o

destaque feito com a seta para o movimento ascendente antes do "pouso".

A idéia por trás de "cirspan" é a de fazer com que o som execute esse "vôo" ao redor do público. Já em "caospan" o som percorre uma trajetória caótica dentro do espaço de difusão. Para isso utilizamo-nos da reprodução ininterrupta de um pequeno arquivo de som (um *sample* em *loop*), cuja altura sofre uma transposição contínua (glissando) segundo os valores escalonados da variável *y*. Isto é feito através da variação da velocidade de reprodução do *sample* por meio do comando *speed* enviado ao objeto *sfplay~* (ver Figura 4). A essa síntese são aplicados três algoritmos de espacialização controlados pelos valores resultantes do sistema de Hénon-Heiles.

O primeiro algoritmo de espacialização encontra-se encapsulado no *subpatch* "small-spat" (pequeno espacializador) e seu resultado é utilizado apenas por "cirspan". Ele submete a reprodução contínua do *sample* a um pequeno atraso (*delay*) com realimentação (*feedback*), controlado pela variável *x'*, para emular uma maior ou menor distância do centro do espaço de difusão, através do efeito de reverberação gerado, e a uma filtragem das suas componentes espectrais agudas, para dar a idéia auditiva de maior ou menor distância do solo (movimento vertical). Para a filtragem, são utilizados os valores da variável *y'*, que controlam a frequência de corte de um filtro passa-baixos (*lpass~*): quanto menor o valor de *y'*, mais aguda a frequência de corte, portanto, maior o número de componentes espectrais e maior a impressão de que o som se encontra próximo ao solo; quanto maior o valor de *y'*, mais grave a frequência de corte e provoca-se a sensação contrária. É bom ressaltar que este procedimento não tem por finalidade posicionar de forma acurada o som no espaço de difusão, apenas dar uma impressão um tanto quanto vaga desse movimento, o que está relacionado à poética dessa composição.

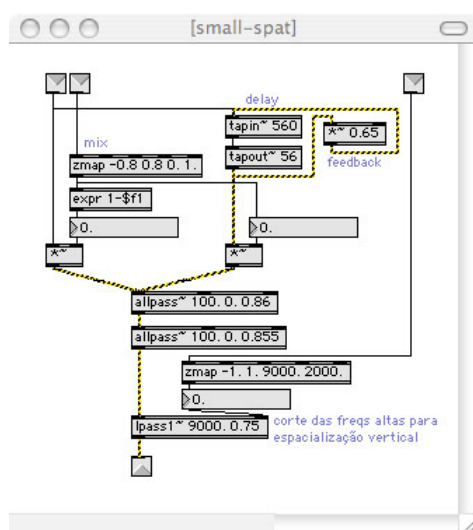


Figura 6. Subpatch do small-spat.

O resultado dessa síntese é submetido a "cirspan", onde a variável *x* fornece o tempo de duração de uma revolução do som ao redor do público, numa espacialização circular no sistema de difusão quadrafônica. Ao término de cada revolução, novo valor é tomado do fluxo de resultados, gerando nova revolução com duração distinta da anterior. Combinados, estes dois algoritmos criam uma trajetória espacial cujos movimentos são sempre similares, mas nunca idênticos. Ao se mudar as condições iniciais do sistema de Hénon-Heiles, novas trajetórias surgem. A associação de três ou mais destes gestos sonoros — através, por exemplo da replicação do mecanismo do *patch* várias vezes em um novo *patch* (o que fizemos em "Revoada") — submetidos a condições iniciais diferentes faz com que o ouvinte ouça essas trajetórias se ramificando pelo espaço de difusão: uma revoada de sons.

O princípio por trás de "caospan" é o de combinar diferentes variáveis do sistema de Hénon-Heiles duas a duas, a fim de gerar trajetórias a serem percorridas pelo som no plano horizontal formado pelo sistema quadrafônico de difusão. Ou seja, o som do *sample* em glissandos irá deslocar-se de acordo com a curva formada pelos pares ordenados de duas das variáveis do sistema, dentro do quadrado formado pelos alto-falantes. O *patch* que construímos possibilita a escolha da combinação de variáveis a ser utilizada através de uma lista de possibilidades para a espacialização (ver Figura 4).

As divergências e convergências das trajetórias nestes algoritmos são ricas em auto-similaridade desviante, provocando um contraponto espacial de grande riqueza formal e variedade.

Conclusão

É possível, tanto teoricamente, quanto pelos exemplos práticos aqui apresentados, concluir que o uso de sistemas dinâmicos não-lineares na composição musical pode ajudar a gerar material musical possuidor de uma auto-similaridade desviante, onde suas componentes apresentem-se "mais ou menos" iguais e que, portanto, possua algo da qualidade de organicidade que buscamos.

Esta conclusão pode ainda ser enriquecida por uma melhor definição do que seja organicidade, bem como estudos mais aprofundados de como ela se manifesta no discurso musical e para o ouvinte.

Referências

- Malt, Mikhail. (2000) "Les Mathématiques et la Composition Assistée par Ordinateur (Concepts, Outils et Modèles)", Tese (Doutorado em Música e Musicologia) - Ecole Des Hautes Etudes En Sciences Sociales, Paris.
- Puig, Daniel F., (2005) "Música e Sistemas Dinâmicos Não-Lineares: uma abordagem composicional" Dissertação (Mestrado em Música) - Universidade Federal do Rio de Janeiro - UFRJ, Escola de Música - PPGM, 2005.
- Slater, Dan. "Chaotic Sound Synthesis", In: Computer Music Journal. Massachussets: MIT, Summer, 1998.
- Wallin, Rolf. (1989) "Fractal Music - Red Herring or Promised Land?", Palestra proferida no Nordic Symposium for Computer Assisted Composition, Estocolmo, 1989. Disponível em: <http://www.notam02.no/~rolfwa/Fractalarticle.html> Acesso em: 14 de maio de 2002.
- Xenakis, Iannis. (1992) "Formalized Music. Thought and Mathematics in Music", Nova Iorque: Pendragon.

A Brief History of Auditory Models

Leonardo C. Araújo¹, Tairone N. Magalhaes¹, Damares P. M. Souza¹,
Hani C. Yehia¹, Maurício A. Loureiro¹

¹CEFALA - Center for Research on Speech, Acoustics, Language and Music
Universidade Federal de Minas Gerais (UFMG)
CPDEE-Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica, room 214
Av. Antônio Carlos 6627 – 31270-010 Belo Horizonte, MG

leoca, tairone, damares, hani, mauricio @cefala.org

Abstract. *This work presents a brief description of the human auditory system together with the history of human comprehension of the auditory function, its main features, and classic models used to represent it. First, a historical view of the hearing apparatus is presented. After that, the physiology of the peripheral auditory system is described. The process of acoustic propagation through the outer, middle and inner ear, as well as the mechanism of transformation of cochlea inner hair cell motion into neuron spikes are explained. Next, Flanagan's mathematical representation (based on physiological data acquired by von Békésy) of the passive relation between the sound that reaches the outer ear and the motion of the cochlea basilar membrane. Flanagan's model is followed by Lyon's model of the cochlea, Meddis' model of the inner hair cell, and Patterson's Auditory Image Model. Finally, the IPFM Toolbox is introduced as an example of music analysis system that incorporates an auditory model to perform acoustic analysis of sound based on human perception.*

1. Introduction

Over the past half century, the auditory system has been the focus of intensive research. Knowledge on physiology, psychology and engineering has provided the possibility for creation of models which until a certain extension tries to describe and mimic the hearing mechanisms. Mathematical and computational models are used to build quantitative simulations that describes a system and bring insights about the system itself or its behavior under certain conditions. Auditory models might be used to several purposes, such as help on tracking problems on hearing or speech fields. They can be also very useful on music research since these systems can simulate the mechanisms of sound perception, so it is possible to use it in musical timbre research, as has been done by a few [Cosi et al., 1994], [Toivainen et al., 1995].

2. Hearing Physiology

Basically, the peripheral auditory system has three main parts: the outer, middle and inner ear. The outer ear is the visible portion of the ear. It includes the pinna (also called auricle), the ear canal and the eardrum. The pinna collects the sounds and directs them down into the ear canal. Its shape also help us to extract directional information from sounds. The ear canal is a tube about 2.5 cm long, and it acts as a quarter-wavelength resonator. So it enhances frequencies around 3400 Hz, the maximum sensitivity regions of human hearing. This region can be easily observed at the famous equal loudness curves.

Acoustic sound waves impinging on the outer ear is propagated down the external meatus leading to the eardrum which is set into vibration. These vibrations are then transmitted to the tiny ossicles of the middle ear. The ossicle chain is compounded of three units: the incus, the malleus and the stapes. The malleus, or hammer, is fixed on the eardrum. It makes contact with the incus, or anvil, which in turn connects with stapes, or stirrup, through a small joint.

The main function of the ossicles is to make an impedance transformation from the air medium of the outer ear to the liquid medium inside the cochlea (inner ear). As there is a great change in the propagation medium density, and so in its impedance, it is necessary an impedance transformation, trying to match those two far distant impedances, comprising their respective disparate realms. The lever action of the ossicles alone provide a force amplification of about 1.3 [von Békésy, 1960]. Another amplification arises from the change in effective area, the eardrum area is much greater than that of the stirrup on the order of 15:1, according to Békésy's measurements [von Békésy, 1960].

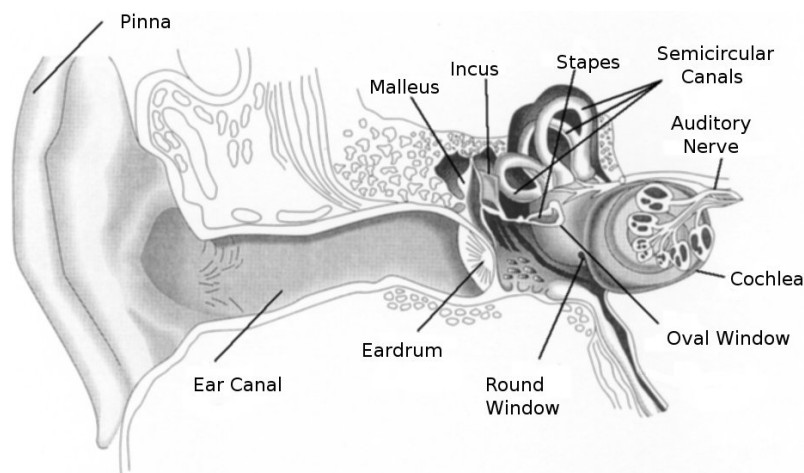


Figure 1: Outer, middle and inner ear

The inner ear is the part where the cochlea is located, just behind the oval window, which is connected to the stapes footplate. The inner ear also comprises the vestibular apparatus and the auditory nerve terminations. The cochlea is a snail-shaped organ filled with almost incompressible fluids. It is divided by two membranes: Reissner's membrane and the basilar membrane.

When the oval window is set into movement, a pressure difference on the fluid propagates down the cochlea. Thus inward motion of the ossicles results in an outward movement in the round window. The basilar membrane (Figure 2) is set in motion by this pressure difference, but the pattern of its movement varies, depending of its mechanical properties at a certain position. At its basal end, the basilar membrane is narrow and stiff, while at its apical end it is wider and more compliant. Because of these differences, the position of the peak in the pattern of vibration differs according to the frequency of stimulation. The basis responds best to high frequencies, while the apex responds best to low frequencies.

Mechanical motion of the basilar membrane leads to displacements of the inner hair cells stereocilia, which are located between the basilar membrane and the tectorial membrane, in a structure called organ of Corti.

3. Auditory Models

The crucial problem in modelling is deciding what is the optimal level of details and what are the possible simplifications. The amount of details and the computational (and why not, mathematical)

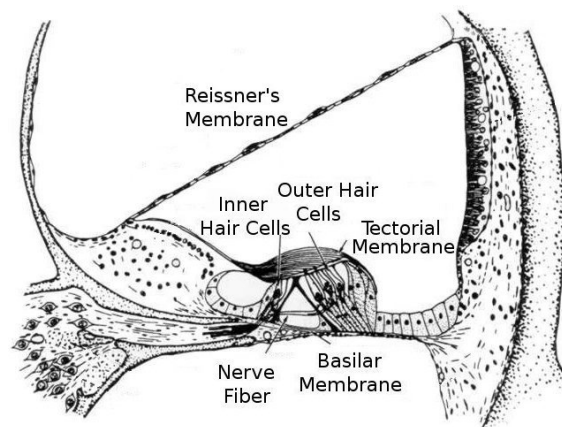


Figure 2: Section of the cochlea. (Adapted from Moore,1995).

burden go against each other. Increasing the level of details, leads us to complex and computational expensive models. Furthermore, all the effort put on those details modelling might be wasted if the final user of the models does not care about them.

This kind of representation subdue the dynamic characteristics and the nonlinearity of the signal. It also incorporates other phenomena from the perception mechanism such as critical bands, frequency masking and temporal masking.

Auditory models are being continually upgraded, and there are many different models available on the internet. The principal models will be described here.

3.1. Flanagan's Model

Flanagan, based on the physiological data measured by Békésy, has proposed a mathematical and also a computational model for the auditory mechanism [Flanagan, 1960] [Flanagan, 1962] [Flanagan, 1972]. His model is divided into two parts, one comprising the middle ear and the other the basilar membrane.

The first block implements the physiological function embraced by the middle ear and the second the basilar membrane. The input in the first block, $p(t)$, is the pressure at the eardrum, its output, $x(t)$, is stapes displacement which will be the input to the next stage. The final output, $y_l(t)$, is the basilar membrane displacement at a distance l from the stapes.

The approximating functions are indicated in its frequency-domain (Laplace) representation by the functions $G(s)$ and $F_l(s)$. Those functions must be fitted to available physiological data. Flanagan assumes the ear to be mechanically passive and linear over the frequency and amplitude ranges of interest, then, rational functions of frequency with left half-plane poles (stable) can be used to approximate the physiological data. The fact of being modeled by rational functions make it feasible of lumped-constant electrical circuit implementation.

Flanagan uses as an approximation to $G(s)$ a third degree function, composed of a real pole and a pair of complex conjugated poles. $F_l(s)$ is approximated by a second order pair of complex conjugated poles, a real pole, a real zero and a delay factor.

This transfer-function is created this way to resemble the resonant properties of the membrane, approximating it as a constant-Q (constant percentage bandwidth) filter bank in character.

3.2. Lyon's Model

Richard F. Lyon [Lyon and Mead, 1988] developed a model of analog electronic cochlea based on the knowledge of how the cochlea works. His approach was to model the fluid-dynamic wave

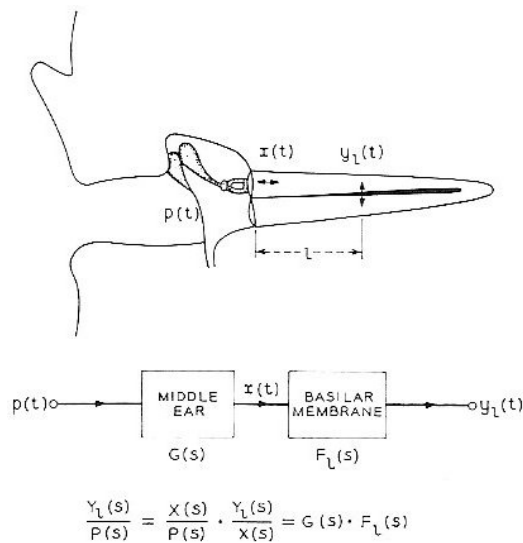


Figure 3

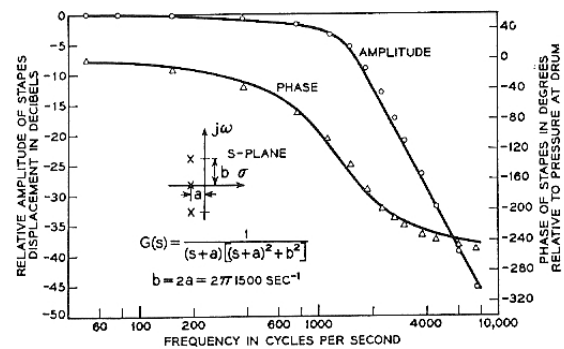


Figure 4

medium of the cochlea by a cascade of filters based on the observed properties of the medium. The action of the active outer hair cells was modelled by a set of automatic gain controls (AGC) which simulated the dynamic compression of the intensity range on the basilar membrane.

A neural spike at the base of auditory nerve is only generated when the stereocilia of the inner hair cell are bent one way. When the cilia are bent the other way, no spikes are generated. So the inner hair cells acts like half-wave-rectifiers, and this unit was used to model them in the Lyon's cochlear model. The output of the model is the probability of firing along time for the neurones of the auditory nerve. This data received the name cochleagram.

Malcolm Slaney implemented an auditory model based on Lyon's model. He developed a toolbox for MatLab called auditory toolbox which includes several functions for auditory modelling including different models for cochlear processing, hair cell transduction and additional functions for spectral analysis and correlogram generation. The correlogram is a function that summarizes periodical information of the cochleagram to give us a better representation of an auditory image. It can be visualized as a movie, and there is also a function for their visualization in the auditory toolbox.

3.3. Meddis' Inner Hair Cell

Ray Meddis [Meddis, 1986] developed a model of inner hair cell based on its physiology. In this model, the permeability function controls the neurotransmitter release into the synaptic cleft. The spike probability on specific neurones of the auditory nerve is a function of the amount of neurotransmitter in the cleft. Interspike intervals are also taken into account. Spikes cannot occur in intervals less than 1 ms.

His inner hair cell model became popular between the auditory researchers. The auditory image model (AIM) (see section 3.4), a famous model of auditory processing developed by Roy Patterson et al. [Patterson et al., 1995] includes a stage of neural transduction using the Meddis hair cell.

This model was upgraded by Sumner et al [Sumner et al., 2002], creating a revised model of the inner hair cell and auditory-nerve complex. This model improves the previous in terms of the range of phenomenon simulated and is more consistent with recent developments in hair-cell physiology. It represents better the response of medium and low spontaneous rate fibers. The

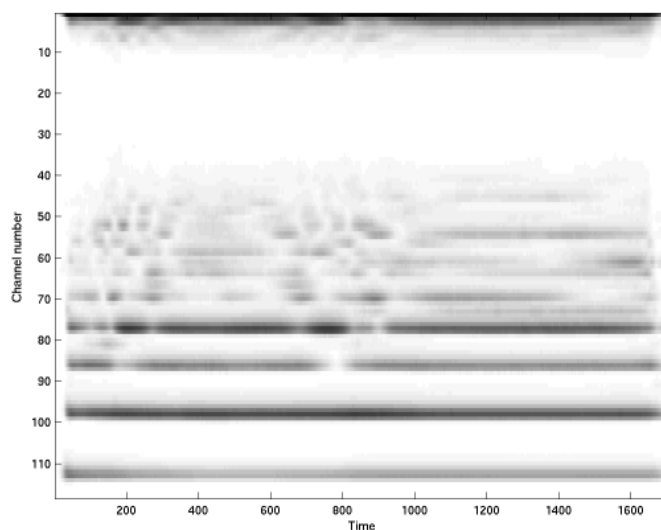


Figure 5: Example of cochleagram using a recorded clarinet sound (C4) as input to the model.

former only simulate the response of high spontaneous rate fibers at the fiber's best frequency.

3.4. Auditory Image Model (AIM)

The auditory image model includes several alternative modules comprising the stages of processing by the peripheral auditory system. These stages are (1) middle ear filtering, (2) spectral analysis, (3) neural encoding and (4) time-interval stabilization. The middle ear filtering is a simple linear filter that enhances middle frequencies. The spectral analysis can be done with a functional (gammatone¹) or a physiological auditory (nonlinear transmission) filter. The output of this stage is the estimated basilar membrane motion (BMM) obtained in the presence of the input signal.

The neural encoding stage converts the BMM into a neural activity pattern (NAP). Two modules are provided for generating the NAP: a bank of meddis inner hair cells and a bank of two-dimensional adaptive threshold units, which rectify and compress the BMM, then apply adaptation in time and suppression across frequency.

The last stage summarizes the temporal activity at the output of the NAP stage based on the idea that periodic sounds give rise to static perceptions by the human listener. There are two modules included: an strobed temporal integration (STI) and a correlogram.

3.5. IPEM Toolbox

The IPEM Toolbox is an example of music analysis system which incorporate an auditory model to perform the perception-based acoustic analysis of the sound. The auditory model (Auditory Peripheral Module) was developed by Van Immerseel and Martens (1992) and implement some features that are important for music perception.

There is an outer ear filtering, an array of band-pass filters for simulate the cochlea and a hair cell model (HCM) that incorporates half wave rectification and dynamic range compression. The HCM also introduces distortion products that corresponds to the beating frequencies. There is also a low pass filter that extracts the envelope of each channel. This filter agree with the loss of synchronization at the primary auditory nerve.

¹Gammatone filter bank is a constant bandwidth filter bank. The gammatone auditory filter can be described by its impulse response: $y_{tone}(t) = at^{n-1}e^{2\pi bt} \cos(2\pi f_c t + \phi)$, for $t > 0$, where the parameters a and b determines the duration of the impulse response and n the filter's order.

This toolbox contains modules which deal with different aspects of perception and can provide good estimations of sensory, perceptual and cognitive parameters of the sound.

4. Conclusions

The research carried out up to now has shed light on the mechanisms that govern the human auditory sense. However many issues involving the function of the auditory periphery are still unresolved. New publications suggest that factors such as displacements of the stereocilia due to the influence of Brownian motion and stochastic resonance enhances the detection of weak signals in the middle frequency range. The ultimate mechanism of perception and the transmission of neural information to the brain is still a cloudy subject. These processes are dealt as *black box* systems (we are unaware of what is going inside) but, still, we might be able of observing and measuring subjects behavior, in response to prescribed auditory stimuli. This approach together with new measurement techniques might give us the information necessary to draw conclusions and understand to which extent auditory physiology and psychology behavior are brought into harmony.

References

- Cosi, P., Poli, G. D., and Lauzzana, G. (1994). Auditory modelling and self-organizing neural networks for timbre classification. *Journal of New Music*, 23:71–98.
- Flanagan, J. L. (1960). Models for approximating basilar membrane displacement. *Bell System Technology Journal*, 39:1163–1191.
- Flanagan, J. L. (1962). Models for approximating basilar membrane displacement ii. *Bell System Technology Journal*, 41:959–1009.
- Flanagan, J. L. (1972). *Speech Analysis, Synthesis and Perception*. Springer-Verlag, end edition.
- Lyon, R. F. and Mead, C. (1988). An analog electronic cochlea. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(7):1119–1134.
- Meddis, R. (1986). Simulation of mechanical to neural transduction in the auditory receptor. *Journal of the Acoustical Society of America*, 79(3):702–711.
- Patterson, R. D., Allerhand, M. H., and Giguere, C. (1995). Time-domain modelling of peripheral auditory processing: A modular architecture and a software platform. *Journal of the Acoustical Society of America*, 98:1890–1894.
- Sumner, C. J., Lopes-Poveda, E. A., O'Mard, L. P., and Meddis, R. (2002). A revised model of the inner-hair cell and auditory-nerve complex. *Journal of the Acoustical Society of America*, 111(5):2178–2188.
- Toivainen, P., Kaipainen, M., and Louhivuori, J. (1995). Musical timbre: Similarity ratings correlate with computational feature space distances. *Journal of New Music Research*, 24:282–298.
- von Békésy, G. (1960). *Experiments in Hearing*. McGraw-Hill, 1st edition.

Revisitando *Waveshaping*: implementando um *plugin* VST para distorcer sons de guitarra

André Luiz Luvizotto^{1,2}, Ricardo Ichizo^{1,3}, Jônatas Manzolli^{1,2}

¹Núcleo Interdisciplinar de Comunicação Sonora - NICS

²Departamento de Música – IA

³Instituto de Computação -IC

(andre@nics.unicamp.br, ichizo@nics.unicamp.br,
jonatas@nics.unicamp.br)

Abstract. *Chebyshev Polynomials were used to create a VST plugin. Waveforms with a spectrum close to sounds recorded with the analogical distortion were obtained. The results were satisfactory with low processing cost.*

Resumo. *Polinômios de Chebyshev foram utilizados para implementação de um plugin VST. Obteve-se formas de ondas com conteúdo espectral próximo de sons gravados com distorção analógica. Os resultados sonoros foram satisfatórios com baixo custo de processamento.*

1. Motivação

A busca por sonoridades com características provenientes de métodos de síntese clássica utilizando-se de modelagem digital e, eventualmente, de *plugins* VST é um campo de pesquisa em grande expansão na atualidade. Implementar um sintetizador em software, manipulá-lo como uma interface gráfica e obter resultados em tempo real, são as principais características deste conjunto de procedimentos. Neste sentido, tem portabilidade, robustez e acessibilidade. Por outro, lado há uma infinidade de dispositivos que poderiam ser emulados em *software* e se tornarem *plugins* VST. Dentro desta infinidade de possibilidades, a pesquisa que apresentamos neste artigo, discute o processo de implementação de um *plugin* VST para distorção de guitarra .

2. Método Waveshaping

O método waveshaping, foi proposto em [Le Brun 1979]. A Waveshaper é uma função $F(x)$ que gera distorção quando aplicada a uma amostra sonora [Dodge 1985]. Há diversas funções que podem ser utilizadas como Waveshaper. No caso em estudo, desenvolvemos um aplicativo VST onde o conteúdo espectral resultante do processo de distorção foi associado aos graus e aos coeficientes de polinômios denominados *Chebyshev*. Portanto, o objetivo foi relacionar os coeficiente dos termos do polinômio com controles deslizantes numa interface gráfica.



$$F(x) = \sum_{k=1}^{\infty} h_k T_k(x)$$

Figure 1. O diagrama acima apresenta a relação entre os termos do polinômio e os controles da interface gráfica.

3. Experimento e implementação do VST

Utilizamos o *software Mathematica* para avaliar o potencial sonoro dos polinômios de Chebyshev quando aplicados no processo de distorção de uma amostra sonora de guitarra. Obtivemos espectros ricos com colorações timbrísticas muito variadas. Estudamos também as características timbrísticas da paleta de sons distorcidos da guitarra. O nosso objetivo foi cruzar as duas informações, pois conhecendo as características espectrais da paleta estudada, poderíamos recriá-la através da manipulação dos polinômios.

Escolhemos um overdrive da marca Ibanez modelo Tube Screamer (Ts), como referência experimental. O nosso interesse foi vinculado às características timbrísticas deste dispositivo, pois o mesmo é freqüentemente utilizado como sonoridade de referência para muitos guitarristas.

Para criar o software de controle de distorção descrito acima, utilizamos o pacote de desenvolvimento de VST disponibilizado pela *Steinberg*, empresa criadora da tecnologia. Tal pacote inclui um *kit* de APIs que possibilitam a implementação de *plugins* e de instrumentos virtuais em tempo real para códigos escritos em C++ e para as plataformas *Macintosh* e *Windows*.

4. Conclusão e Resultados

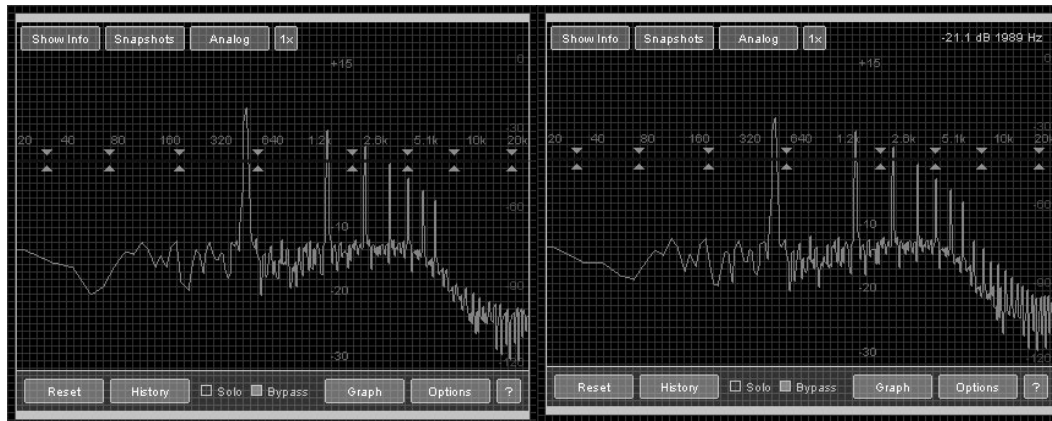


Figura 2: A esquerda temos a análise spectral do plugin e a direita do efeito analógico gravado. Observa-se uma maior densidade de energia nas altas frequências por parte do efeito analógico.

References

Dodge, C. and Jerse, T. A. (1985) “Computer Music: Synthesis, Composition and Performance”, Schirmer Books – New York.

Le Brun, M. (1979) “Digital Waveshaping Synthesis”, Journal of the Audio Engineering Society Volume 27.

Oppenheim, A. V and Willsky A. S (1975) “Signals and Systems”, 2nd Edition, Prentice Hall – New Jersey .

Classification of Triadic Chord Inversions Using Kohonen Self-organizing Maps

Luis Felipe de Oliveira¹, Luis Guilherme Pereira Lima², André Luiz Gonçalves de Oliveira³, Rael Bertarelli Gimenes Toffolo¹

¹ Departamento de Música – Universidade Estadual de Maringá (UEM)
Maringá – PR - BR

² Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – BR

³ Departamento de Teoria e Prática da Educação – Universidade Estadual de Maringá (UEM)
Maringá – PR – BR

luisfol@bol.com.br, lglima@gmail.com, alguns@gmail.com,
rael_gimenes@uol.com.br

Abstract. *In this paper we discuss the application of the Kohonen Self-organizing Maps to the classification of triadic chords in inversions and root positions. Our motivation started in the validation of Schönberg's hypotheses of the harmonic features of each chord inversion. We employed the Kohonen network, which has been generally known as an optimum pattern classification tools in several areas, including music, to verify that hypothesis. The outcomes of our experiment refuse the Schönberg's assumption in two aspects: structural and perceptual/functional.*

1. Schönberg's Hypothesis

In the *Harmonielehre*, Schönberg (1974) discuss the peculiar harmonic features and rules of the first and second triadic inversions at the relation of root position triads (53). The first inversion (6) does not require any kind of special treatment than the second inversion (64) does. The second inversion, as he argues, has an ambiguity constitution, being it related to its root position chord and to a chord a fifth above. This ambiguity has been lead to specific harmonic rules in the attempt to characterize the function of this chord. For instance, the tonic second inversion chord can be analyzed as I64 or V64, and in this case, the cadential 64, there are no specific voice leading procedures in bass. In Schönberg's assumption there is a contrasting forces involved in the 64 chord due by the derived harmonic partial structure. There are harmonic partials that emphasize the bass note as fundamental (they match the bass harmonic series) and there are partials that attenuate the strength of the bass as fundamental (they do not match the bass harmonic series). In the first inversion chord, the harmonic partials that match the bass note (in this case, the third) are higher than the ones that match the fundamental. In the same time, at the second inversion chord, the harmonic partials that matches the bass note (in this case, the fifth) are closer than ones that matches the fundamental. One can resume the Schönberg's treatment of second inversion chord as moving the bass voice by step wise or staying on the same note, after and before this chord, producing an aural perception of a chord over a passing note, solving the ambiguity. For the sake of clarity, the above statement can be illustrated in the following table:

Table 1. Harmonic partials set in a C major root position and inverted chords (the played notes in capitals letters; the harmonic partials in lower case; the subscript letters are not considered by Schönberg; the notes that matches the bass in blue and the harmonic partials that contradict the bass in red).

Harmonic partials of triadic chords									
Root position (53)									
C	<i>c</i>	<i>g</i>	<i>c</i>	<i>e</i>	<i>g</i>	<i>b_b</i>			
<i>E</i>	<i>e</i>	<i>B</i>	<i>e</i>	<i>g#</i>	<i>b</i>				
<i>G</i>	<i>g</i>		<i>D</i>	<i>G</i>	<i>b</i>	<i>d</i>			
First inversion (6)									
<i>E</i>	<i>e</i>	<i>B</i>	<i>e</i>	<i>g#</i>	<i>b</i>	<i>d</i>	<i>e</i>		
<i>G</i>		<i>g</i>	<i>d</i>	<i>G</i>	<i>b</i>	<i>d</i>			
	<i>C</i>		<i>c</i>	<i>G</i>		<i>c</i>	<i>e</i>	<i>G</i>	
Second inversion (64)									
<i>G</i>		<i>g</i>	<i>d</i>	<i>G</i>	<i>b</i>	<i>d</i>	<i>f</i>	<i>g</i>	
	<i>C</i>		<i>c</i>	<i>G</i>		<i>c</i>	<i>e</i>	<i>G</i>	
	<i>E</i>		<i>e</i>		<i>b</i>	<i>e</i>	<i>g#</i>	<i>b</i>	

Analyzing the table and considering the contradictory harmonic partials, which are not considered by Schönberg, one can note that the root position presents problems in the same way that the second inversion does. We must stress that Schönberg do not note the harmonic sets to the root position chord - in this sense, his argument became fragile and naive. Otherwise, he neither evaluate this explanation to the minor tonalities, what seems to be a recurrent conduct of music theorist in general.

In Oliveira *et al* (2005), we demonstrated the fallacy of Schönberg's argumentation for a single major tonality using a Kohonen Self-organizing Maps. The experiment results, demonstrated that the root position chords and its inversions were classified in near clustering neighborhood. If the Schönberg's hypothesis were correct, the 64 chords should be placed in a fifth above chord neighborhood.

In this paper, our inquiry is motivated by the historical lack in music theory of a solid argumentation explaining the minor chords. The supposed perfection of major triad was explained by the relations of the harmonic partials, especially the six foremost ones¹. If there are difficulties to justify the major chord "perfection" in this argument, in the minor chords, the argumentation problems about its constitution and tonal origins, will be more prominent. To escape from this traditional lack, present in harmony studies, we must verify the results of a SOM working on minor, major and diminished chords as input values. Our input patterns set consists in the 252 chords from the twelve major tonalities (21 chords for each tonality). The use of a SOM to classify chords is well known in the literature about cognitive (and computer assisted) musicology [LEMAN

¹ If the perfection is derived from the harmonic series, one can note that the seventh partial brings troubles to this description.

2000, 1995, 1991, 1990]. However, usually this sort of research does not consider any aspect of chord inversion. That is the main objective of our investigation. We expect that the clustering map will create the same categorization for the inverted chords for all major, minor and diminished triads. The results should appoint that the problem explanations based in the harmonic series, that justify the structure and the harmonic treatments of the inverted chords, are misleading for major, minor and diminished chords.

2. The Kohonen's Self-organizing Maps.

2.1. The Self-organizing Maps.

The main objective of a Kohonen Self-organizing Map (SOM) (1997) is to determine the mapping of a n dimensional input signals set into a bi-dimensional grid. This mapping occurs in an adaptative and topologically ordered fashion. The SOM architecture consists in two layers: the input layer of n dimensionality (n equal to the dimensionality of the inputs set) and the output layer characterized as a bi-dimensional grid of neurons. Each input neuron is fully connected with the bi-dimensional grid, where each of the connections is represented by an associated synaptic weight. The weight vector of a grid unit consists in the set of weight values of all connections.

The algorithm responsible to the map formation initializes the grid connections weight with small random values. We can stand out three processes when the network is initialized: competition; cooperation; synaptic adaptation. These processes are responsible to the formation of the map in an unsupervised learning method [HAYKIN 2001].

Competition:

When an input pattern is presented to the network, just one grid unit must be activated by that pattern. This activated unit is called the winner neuron (winner-take-all). A discriminant function provides the competition bases in this process.

Cooperation:

The winner neuron determines the center of the topological neighborhood region of laterally excited neurons, furnishing the bases of cooperation. The neighborhood area can be expressed as a rectangular field or a hexagonal one, as showed in the figure bellow.

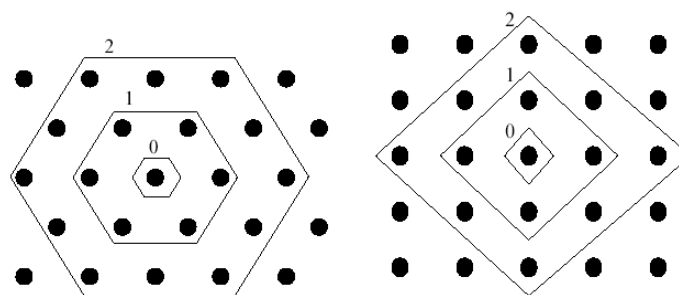


Figure 1. Rectangular and hexagonal neighborhood.

The lateral interaction among the winner neuron and its neighbors is represented by the neighborhood topological function. The maximal function value is reached when the distance among the winner neuron and its neighbors is equal to zero. Otherwise, the minimum function value occurs when the distance among the winner neuron and its neighbors tends to the infinity. The gaussian function satisfies the above necessities and is widely employed in SOM networks:

$$f_{j,i(x)} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right)$$

where the d_j, i represents the lateral distance between the winner neuron i and the excited neuron j . The σ represents the effective width of neighborhood as showed in the Figure 1.

Synaptic adaptation

In this phase, the synaptic weight are adjusted to that the winner's weight vector approximates to the input vector x . Once the network is continuously feed with the input set, the algorithm produces a topological map ordination of the features, leading to similar values of the weight vectors for the adjacent neurons.

2.2. The SOM Algorithm

The SOM Algorithm could be summarized in five steps:

1. Initialization

Attribute random small values to the synaptic weights of each neuron. This step ensures that the map will have no previous organization at all.

2. Sampling

Present a random choose sample X to the network, from the input space with n dimensionality, being $X = [x_1, x_2, x_3, \dots, x_n]^t$.

3. Matching

Find the winner neuron $i(X)$, with the weight vector $W = [w_1, w_2, w_3, \dots, w_n]^t$, in the epoch t , closer to the vector X presented to the network, adopting the minimum Euclidian distance criterion:

$$i(X) = \arg \min_j \|X - W_j\|, j = 1, 2, \dots, n$$

being n , the total number of grid neurons.

4. Updating

Adjust the synaptic weights to every grid neuron by the actualization formula

$$w_j(t+1) = w_j(t) + \eta(t)f_{j,i(x)}(t)(x(t) - w_j(t))$$

where $\eta(t)$ is the learning rate, $f_{j,i(x)}(t)$ is the neighborhood topological function surrounding the winner neuron. Both parameters are dynamically varied to ensure better results.

5. Repetition

Return to step 2 until no significant changes occur in the features map.

The competitive learning happens in the second step when the weight vectors are updated. To each input presented to the network only one neuron must be active in a specific instant. In this sense, the competitive learning has a priority to define the statistical features more outstanding for the classification of an input pattern set [HAYKIN 2001]. In the third step, the cooperative process among the grid neurons is determined by the neighborhood area of the winner unit ($f_{j,i(x)}$).

The network training consists in two distinct phases: the rough phase and the fine-tuning phase. The first is characterized by the topological ordination of the weight vectors. In this phase, the learning rate should be set in value close proximity to 0.1 and the neighborhood area of the winner neuron should take almost all neurons of the grid. During the rough phase the learning rate decreases smoothly until it reaches the value of 0.01. The second phase is necessary to achieve a fine-tuning ordination of the features map. To make the statistical precision as good as possible, the learning rate should be maintained closer to 0.01; it should not take zero to avoid a meta-stable state of the grid (a topological impairment). The neighborhood area must contain only the next neighbors of the winner unit, possibly reducing its area to one or zero in the fine-tuning phase.

3. Method

The network input dataset is a group of vectors with dimensionality $n=63$, being n the necessary number of pitch-classes to represent the sixth first harmonic partials of each chord note, without octave reduction. The value of the partials in each vector was set up in two fashions: decreasing Fibonacci scaling and an unique value of 1 for all the six partials. Both alternative scaling procedures have leaded the network to similar results [OLIVEIRA et al 2005], and we adopted the Fibonacci scaling for the experiments described in this paper. The SOM network employed has a total of 2500 units, as a square matrix of 50 x 50 cells.

To make the topological error as minor as possible, we adopted the optimum-learning rate of $\alpha_1=0.1$ and $\alpha_2=0.037$ for the rough and fine-tuning training epochs, respectively. The rough epoch consisted in 500 iterations and the fine-tuning in 1500 (the network training employed a learning rate power function in sequential input presentation mode). The neighborhood radius was started with 30 units (rough phase) and decreases to 1 (fine-tuning phase).

4. Results and Discussion

During the network training, the quantization error was reduced to zero, as shown in figure 2, in the fine-tuning training phase. After the training period we can obtain the topological representation of the network response to the input set, as figure 3 demonstrates.

The topological map shows the clusters of each pattern of the input set. In a very first observation one can see that the map demonstrates that similar chords are grouped together. This fact proves that the outcome is coherent over the network behavior. Similar chords have common values in the input vector, so it's natural that they are placed in near positions of the topological map. For instance, root position chords were classified as very similar categories than the inverted ones. One can ask why a chord of C6 is classified closer to C53 than Em53 if it has two common notes with both chords. Could be argued that the fact it was classified near to C53 than Em53 because the octave interval has common values in the input vector but the minor second (obviously!). In this sense, the octave similarity is due to the equivalence of some input vector values.

Regarding Schönberg's argumentations about the 64 chords, the network results should put this sort of chord far from the root position ones. The result, as in the figures bellow, clearly dismisses such an argumentation. The argument of the author came from the fact that there are harmonic partials that stress the bass note and others that emphasize the fundamental note. One can take this statement as attempt to include perceptual properties in the scope of the explanation about chord similarities. On the other hand, one can take that Schönberg's assertion is an attempt to include functional statements into the

justification. But, besides his statements, the SOM network placed the 64 chord generally in the same cluster that the 53 chord, as being the same category. The topological map grouped the chords and its inversions in the same neighborhood in 97.22% of the cases (see figure 5). The isolated chords, beside the fact they are placed distantly from its similes, they are distant from the neighbor chords as should be noted in a three-dimensional representation of the topological map (figure 4). The presence of the hills on this map indicates the growing of the Euclidian distance to the non-similar neighbors. This statement reveals that the SOM network classified those chords in localizations far away from their similes, but categorized they as being different from the other chords in the same topological region.

Recalling that the 64 inversion was classified as the same category of the root position chord, the overall outcome corroborates the Enlightenment theorist Rameau (1971), in his argumentation of the chords inversions. Rameau did come to his results inside a deep reflection and sometimes an unusual mathematics over the monochord². Our results came from the proximal values of the input vectors in octave interval (see table 2).

Table 2. Input vectors of a note C and octave above C. The boldface items show the coincident partials.

C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
1	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0	0.5	0	0	0	0	0.3	0	0	0	0.2	0	0	0.1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0	0.5	0	

A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.3	0	0	0	0.2	0	0	0.1	0	0	0	0	0	0	0	0

Based on the results of the SOM network, we can say that Schönberg's affirmations and our refutations are founded on two points: structural and perceptual/functional. Firstly, we will analyze the structural point in the bases of three aspects: the temperament; the consideration of only the sixth firsts harmonic partials; the dynamic behavior of harmonic partials. Secondly, the argumentation about the perceptual/functional considerations of inverted chords should be took in a different way that Schönberg considers it, if he does at all (in *Harmonielehre* e specially³).

In the structural consideration, the non-tempered harmonic series used in the Schönberg's statement, does not seems to be the better way to explain the emphasis on the bass notes by the coincident harmonic partials generated by the other chords notes. To evaluate his hypothesis we adopted the same representation in the construction of the inputs patterns, i.e. non-octave reduced pitch class. In this sense, we had model the experiment using the same presuppositions provided by Schönberg, even so his hypothesis was not confirmed. Other issue that must be considered is the use of only the six first partials. Schönberg uses only this set of partials to avoid the problems that the seventh partial brings to the scene⁴. The non-consideration of the seventh partial is misleading in the sense that one uses just the corroborative aspects of a natural phenomenon for the sake of argument. We can suppose that this fact is justified by the necessity to establish a natural foundation to the tonal system. This natural foundation is as artificial as any abstract and arbitrary possible syntactic system (twelve-tone scale or

² For furthers stances see Rameau (1971) and specially the translator's introduction.

³ Even in Structural Functions of Harmony, Schönberg does not talk about chord inversions.

⁴ Rameau, in his treatise, spares the very same problem skipping the seventh partial from his explanations (1971, p.7).

microtonal scale etc) because they did not take the nature phenomena as whole coherent instance but just some parts of it. The third point in the structural aspects concerns the dynamic behavior of harmonic series (in nature). Beside the above considerations the harmonic series is not a static and stable set of frequencies over the time independent from its physical instantiation. If the explanation for the inverted chords bears on the harmonic partials, a chord structure played in some of the musical instruments or electronic devices should be very confusing for a listener. For instance, a second inversion chord played by the clarinetists does not presents the same set of harmonics as an idealized harmonic series. Consequentially, the explanation of chord inversions would not apply to such a physical instantiation. Could the relation of a chord and its inversions be accounted in the perspective of an idealized and incomplete harmonic series? The present experiment does not take this fact into consideration (our goal was just test the Schönberg's hypothesis), but further developments we are designing operate over a non-tempered harmonic series with spectral envelope behavior extracted from various musical instruments.

To conclude our divagation we should state some perceptual (and functional) considerations. Perhaps, the better justification for the closer relation of a 64 chord with a 53 chord a fifth above came from a perceptual *cliché* and some counterpoint procedures, as the *retardo* or suspension (the dissonance should ever descend like all the things fall to the ground, as stated by Fux (1965) in the XVIII century). The perceptual justification does not necessarily require an acoustic correlation as Schönberg tried to do, because sometimes it is more a subject of cultural stereotyped expectations, that are meaningful just to a part of the human beings, and is not absolutely a property of nature. Being that, perhaps the some music theorists, in an attempt to justify the naturalness of some abstract system, attaches a physicalist reasoning to some properties and procedures coming from the level of language and art, that are not just a physical phenomena. Finally, we can recall that the Schönberg's physical hypothesis to the second inversion chord, based on a static and idealized harmonic series, was not confirmed by the experiment described in this paper. In further researches we intend to examine the limits of the relation of tonal functionality and structural acoustics aspects of triadic chords, using a representational system closer to the acoustic phenomena and capable to account for harmonic progressions.

4.1. Graphic Results

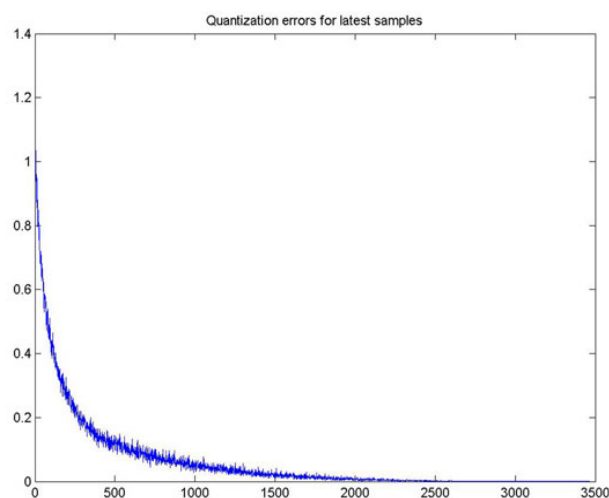


Figure 2. Quantization error.

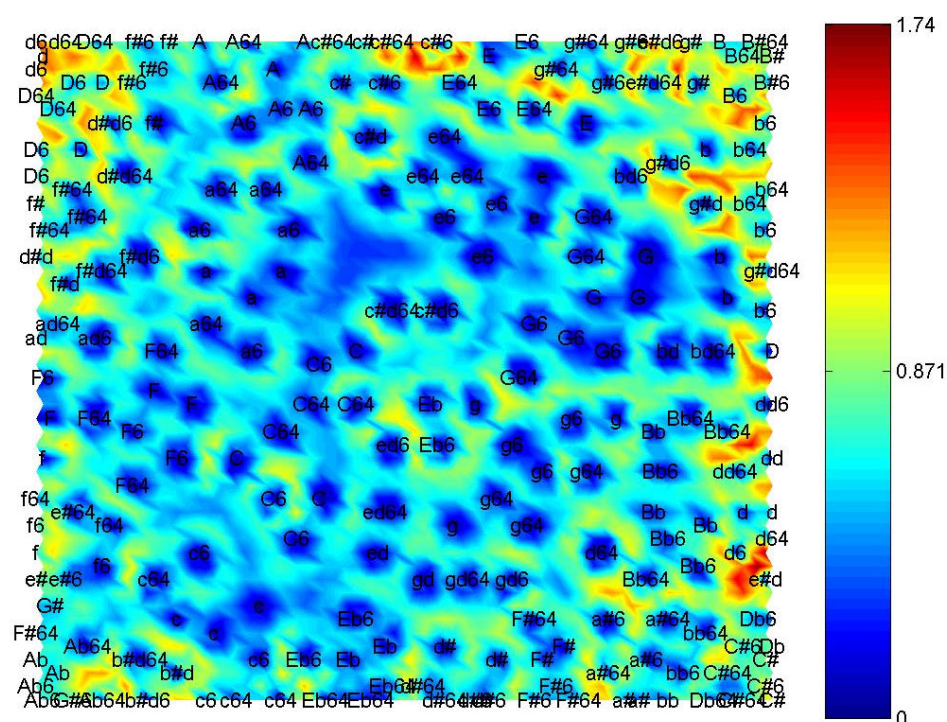


Figure 3. Topological Map. The color bar indicates the Euclidian distance among clusters.

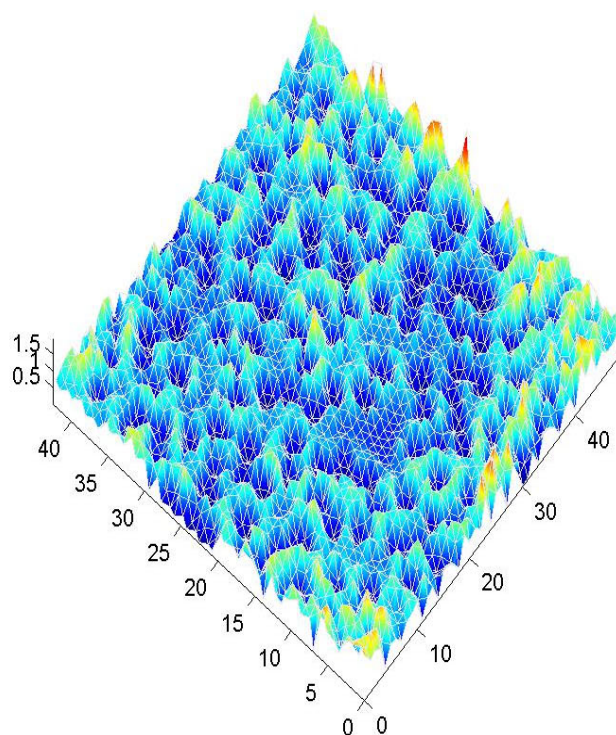
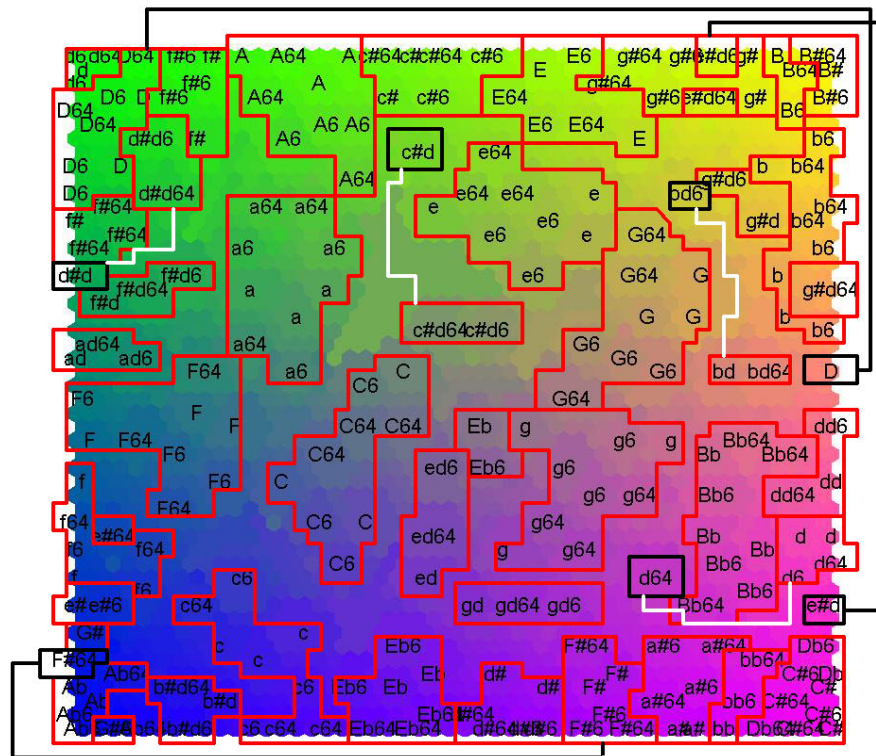


Figure 4. The three-dimensional representation of the topological map.



References

- Arbib, Michael A. (1998). The Handbook of Brain Theory and Neural Networks. Cambridge, MA: The MIT Press.
- Fux, Johann J. (1965). The Study of Counterpoint From Johann Joseph Fux's Gradus ad Parnassum. New York: W. W. Norton & Company.
- Leman, Marc (2000). An Auditory Model of the Role of Short-Term Memory in Probe-Tone Ratings. *Music Perception*, Vol.17(4), 481-509.
- Leman, Marc (1995). A Model of Retroactive Tone-Center Perception. *Music Perception*, Vol.12(4), 439-471.
- Leman, Marc (1991). The Ontogenesis of Tonal Semantics: results of a computer study. In: P. Todd & G. Loy (Eds.), *Music and Connectionism*. Cambridge, MA: The MIT Press.
- Leman, Marc (1990). Emergent Properties of Tonality Functions by Self-organization. *Interface*, Vol.19(2-3), 85-106.
- Haykin, Simon (2001). Redes Neurais: princípios e prática. Porto Alegre: Bookman.
- Kohonen, Teuvo (1997). Self-organizing Maps. Berlin: Springer-Verlag.
- Oliveira, Luis F, Lima, Luis G. P., Oliveira, André L. G. and Toffolo, Rael B. G. (2005). Classificações de Acordes Invertidos por Mapas Auto-organizados de Kohonen, In: M. Dottori, B. Ilari & R. C. Souza (Eds.), *Anais do Primeiro Simpósio Internacional de Cognição e Artes Musicais*. Curitiba: UFPR.
- Rameau, Jean-Philippe (1971). Treatise on Harmony. New York: Dover Publications.
- Schönberg, Arnold (1974). Armonia. Madrid: Real Musical.

Estudo e Implementação de Métodos de Medição de Resposta Impulsiva em Salas de Pequeno Porte

Bruno Sanches Masiero¹, Fernando Iazzetta¹

¹ Laboratório de Acústica Musical – Escola de Comunicação e Artes da Universidade de São Paulo (ECA/USP)

Av. Prof. Lúcio M. Rodrigues, 443 – 05.508-900 – São Paulo – SP – Brazil

bruno.masiero@poli.usp.br, iazzetta@usp.br

Abstract. *This paper describes the results obtained in an under-grad project in the area of acoustical measuring. In this project, a research about the various acoustic impulse response measurement systems for small room was made. A research about room acoustical parameters, as well as about impulse response processing methods for its derivation, was also done. As a result of this project, a acoustic impulse response measurement system was developed.*

Resumo. *Neste projeto foi feita uma revisão das diferentes técnicas de medição da resposta impulsiva acústica para salas de pequeno porte. Foi feita também uma revisão dos parâmetros acústicos para salas, assim como métodos de processamento da resposta impulsiva para sua obtenção dos mesmos. Como resultado final do projeto, foi desenvolvido um sistema de medição da resposta impulsiva acústica.*

1. Métodos de Obtenção da Resposta Impulsiva

A resposta impulsiva acústica é uma função temporal da pressão sonora de um espaço acústico, que resulta da excitação desse espaço por uma função que se aproxima da função delta de Dirac [1]. A resposta impulsiva (IR) de um espaço acústico fornece uma descrição precisa desse sistema. Todos os parâmetros acústicos definidos pela norma ISO 3382 [2] são derivados diretamente da IR acústica.

A norma ISO 3381 faz algumas considerações importantes quanto à medição da IR acústica. Por definição, a IR acústica é medida com pares de emissor-receptor. Na acústica de salas, a IR obtida entre um receptor e um emissor caracteriza o sistema acústico entre a localização exata desses dois itens, mas essa IR não pode usada para caracterizar a resposta da sala como um todo. Para a obtenção dos parâmetros acústicos de uma sala, essa norma recomenda a IR de uma sala seja medida em um mínimo de dezoito posições distintas, posteriormente obtendo-se a média energética destas IR. Microfones e alto-falantes usados para este fim devem ser omnidirecionais.

Existem diversos métodos de medição acústica de salas entre eles a excitação por impulsos, a excitação estática por senóides, Time Delay Spectrometry (TDS) e

gravador de intensidade, mas atualmente os principais métodos são a análise por FFT, as Maximum Length Sequence (MLS), e o método da varredura, que serão descritos a seguir.

1.1 Maximum Length Sequences (MLS)

O método MLS utiliza um grupo especial de ruídos como sinal de excitação, as seqüências pseudo-aleatórias de máximo comprimento – dentre os ruídos, este é o que fornece a melhor SNR de acordo com [3]. Este método é bastante popular em todo o mundo.

A obtenção da IR a partir da MLS recebida pode ser realizada eficientemente por meio da Transformada Rápida de Hadamard (FHT). Devido a restrições de quantidade de memória e tempo de processamento, era de vital importância nos anos 70 e 80 o uso de métodos eficientes como a FHT. Com a velocidade dos computadores atuais, a técnica MLS já não apresenta vantagem significativa frente às demais técnicas.

O método MLS é bastante vulnerável à não-linearidades do meio, tornando seu uso desaconselhável para medição de grandes espaços abertos.

O MLS, assim como o TDS, possui um espectro branco, ou seja, menos energia nos graves que nos agudos, o que pode ser contornado com a pré-ênfase do sinal [4].

1.2 Análise por FFT

Esta é a única técnica que permite realizar medições acústicas durante uma apresentação musical, usando como sinal de excitação a própria música que está sendo executada (desde que esteja sendo reproduzida eletronicamente). A obtenção da IR é praticamente igual à do método anterior, baseando-se em comparar o espectro do sinal antes de ser enviado ao ambiente com um sinal capturado na sala em questão. Divide-se o espectro (deconvolução no domínio do tempo) do sinal capturado pelo espectro do sinal enviado.

O uso de música como sinal de excitação não é recomendável, devido ao comportamento espectral inconsistente, bastante rugoso. Para este tipo de sinal é necessário considerar um longo período de tempo para a obtenção do espectro, e ainda assim é necessário realizar uma média com diversas realizações de medida, para obter-se um resultado consistente.

1.3 Varredura Logarítmica

Na análise por FFT, qualquer tipo de sinal de excitação pode ser usado. Recomenda-se então usar sinais de excitação que apresentem comportamento espectral favorável para medições, em especial ruído ou varredura.

Atualmente, existe a tendência de usarem varreduras logarítmicas como sinal de excitação, por apresentarem boa distribuição da energia em seu espectro (espectro rosa) para aplicações acústicas [4] e serem bastante robustas a não-linearidades.

Quando uma MLS é usada como sinal de excitação, não-linearidades do sistema podem ser notadas na IR como rugosidades ou repetições do sinal deslocadas no tempo de menor amplitude. Quando a varredura logarítmica é usada como sinal de excitação, é possível isolar completamente a resposta impulsiva desejada das componentes presentes no sinal devido à não-linearidades. Isto é possível porque as harmônicas geradas por

não-linearidades do sistema, após a deconvolução, aparecem em tempos negativos da IR.

2. Processamento da IR para análise acústica de salas

Respostas impulsivas reais diferem em basicamente três aspectos das respostas impulsivas teoricamente esperadas:

1. Uma IR real apresenta um atraso antes da chegada do som direto, devido à velocidade de propagação do som. A norma ISO 3382 fornece a seguinte recomendação: “Determina-se o início do som direto a partir da IR de banda larga, como o ponto onde o sinal é 20dB menor que valor máximo da IR, mas significativamente maior que o ruído de fundo”.
2. O decaimento pode conter várias partes com diferentes taxas de decaimento, ou nem mesmo ser exponencial.
3. A IR possui ruído de fundo, o que limita o decaimento a um certo patamar. A subestimação do efeito do ruído pode causar um erro sistemático na obtenção dos parâmetros acústicos. A norma ISO 3382 propõe que o limite de integração superior da curva de decaimento seja tomado num ponto 10dB acima do ponto de cruzamento entre a assíntota de decaimento da IR e o nível de ruído. Desta forma, minimiza-se a energia de ruído presente na curva de decaimento.

2.1 Minimização da influência do ruído

Existem alguns métodos para minimizar a influência do ruído no cálculo da curva de decaimento, entre eles o método de Chu [5] e o método de Hirata[6]. O método de Lundebj[7] se propõe a minimizar o ruído conjuntamente com a minimização do efeito de truncamento da integral inversa.

3. Comparação entre Métodos de Medida

O uso da FFT é provavelmente o método mais interessante nos dias de hoje. Sendo assim, cabe apenas selecionar o sinal de excitação mais adequado, ou seja, aquele que oferece a melhor SNR. De acordo com a teoria do fator de crista, os melhores sinais para esta aplicação seriam a varredura logarítmica e a MLS.

Em primeira instância, um sinal MLS bipolar seria o sinal ideal no sentido de se extrair o máximo de energia de uma medição, já que apresenta um FC=0dB. Mas, o fator de crista das MLS's usadas na prática é sensivelmente maior que 0dB. Na placa de áudio, após o conversor D/A, o sinal passa por um filtro “anti-aliasing”, que acaba por alterar a forma de onda da MLS. Verifica-se então a presença de picos no sinal de saída, o que degrada o fator de crista do sinal. Para evitar que estes picos sejam saturados, o sinal MLS deve ser enviado ao conversor D/A com nível entre 8dB e 5dB menor que o nível máximo do conversor.

A varredura de senóide apresenta um FC teórico de 3dB. Por este sinal conter energia apenas nas frequências de interesse, ele não será afetado pelos filtros da placa de áudio, e poderá na prática ser reproduzido com uma intensidade maior que uma MLS. Uma varredura utilizada para a excitação de salas (após passar pelo filtro “anti-aliasing”) apresentava FC=3.6dB, valor bastante próximo do FC teórico.

Verificou-se que o método de excitação por varredura logarítmica com deconvolução via FFT – sugiro o nome “*Log-sweep FFT method*” (LSF) – mostra-se o método mais indicado para medição acústica de salas nos dias de hoje.

4. Sistema de Medição

A arquitetura de um sistema de medição acústica é geralmente constituída por dois módulos. O primeiro módulo responsável pela geração do sinal, obtenção da IR e cálculo dos parâmetros acústicos, é implementado por um microcomputador. O segundo módulo responsável pela reprodução e aquisição de sinais sonoros é implementado por uma placa de áudio e um conjunto de transdutores.

4.1 Reprodução e Aquisição de Áudio

Uma placa de áudio de boa qualidade é um requisito chave para a qualidade de reprodução e gravação dos sinais. Os requisitos básicos para que uma placa de áudio possa ser usada para medição acústica são:

1. Apresentar linearidade e boa SNR;
2. trabalhar com taxas de amostragem superiores a 40kHz;
3. funcionar em modo “stereo full duplex” (produzir um sinal estéreo e gravar outro sinal estéreo simultaneamente).

Como transdutor de entrada recomenda-se usar um microfone de medição com resposta praticamente plana e omnidirecional. O mesmo vale para o transdutor de saída.

A IR de uma sala obtida por meio destes equipamentos contém não só a resposta da sala, mas também a resposta de todos os elementos deste sistema. Para obter uma medição o mais fiel possível, é necessário que os elementos deste sistema possuam função de transferência o mais lineares possível. Como esses elementos estão todos ligados em série, a qualidade do sistema é limitada pela resposta do elemento de qualidade mais baixa.

Caso o sinal esteja sendo reproduzido por uma fonte externa, como um CD-player, e sendo apenas gravado pela placa de áudio, é necessário atentar ao sincronismo entre estes dois equipamentos. Mesmo uma diferença mínima entre as frequências de amostragem acarreta uma diferença no número de amostras, o que para sinais como o MLS é inaceitável.

4.2 Geração e Tratamento do Sinal

A geração e, principalmente, o tratamento dos sinais usados para as medições acústicas requer uma elevada taxa computacional. Portanto é recomendável o uso de um bom computador para acelerar a realização desses cálculos. Também, como se trata de arquivos de áudio não comprimidos, é necessária disponibilidade de memória física e memória RAM. Estes requisitos são facilmente atingidos pelos microcomputadores disponíveis atualmente no mercado.

4.2.1 Geração do Sinal

O primeiro passo para a realização da medição é a criação de um sinal de excitação para a sala. Os sinais recomendados para medição acústica são varreduras e seqüências MLS.

Tanto varreduras senoidais quanto seqüências MLS podem ser geradas por algoritmos relativamente simples. A varredura linear ou logarítmica é geralmente gerada no domínio do tempo, mas pode também ser gerada no domínio da frequência, como explicado em [4]. Para as seqüências MLS, além da seqüência MLS propriamente dita, a função deve também retornar os vetores de permutação de linha e coluna necessárias para o uso da FHT, conforme especificado por Chu [5].

O sinal gerado de forma digital deve ser gravado num formato adequado para sua reprodução.

4.2.2 Recepção do Sinal

A placa de áudio e o sistema operacional do PC devem permitir a reprodução e aquisição simultânea de som. Um arquivo estéreo com a resposta da sala ao sinal de excitação em um canal e o sinal de referência (curto elétrico) no outro canal deve ser gerado

Na fase de aquisição do sinal, é importante que o aplicativo de aquisição avise caso ocorra saturação do sinal, ou seja, quando o nível do sinal está acima do nível máximo de amostragem do conversor A/D.

4.2.3 Deconvolução

De posse da resposta da sala ao sinal de excitação, é necessário deconvoluir o sinal, de forma a obter-se a IR. As MLS devem ser deconvoluídas através da FHT, conforme Peltonen [8].

Para a deconvolução via SLF, usá-se uma função FFT. Primeiramente obtém-se o espectro do sinal de excitação e do sinal de resposta. O espectro do sinal de resposta é então dividido pelo espectro do sinal de excitação, o que fornece a função de transferência da sala. A IR é obtida por meio da transformada inversa de Fourier (IFFT) da função de transferência. Este método para a obtenção da IR é bastante suscetível ao ruído. Existem diversos métodos de estimação espectral que tentam tornar esta operação mais confiável, mas que não foram tratados neste projeto.

4.3 Tratamento da IR

Os parâmetros acústicos são usualmente calculados por faixas de frequência. Deve-se então filtrar a IR de banda larga por um banco de filtros de oitava ou terço-de-oitava.

Após a filtragem, ainda é necessário um tratamento da IR, como já comentado anteriormente, antes de se calcular os parâmetros acústicos.

4.4 Cálculo dos Parâmetros Acústicos

Uma vez que o sinal já foi tratado, e já está filtrado na banda de interesse, resta então calcular os parâmetros acústicos detalhados na norma ISO 3382, entre eles: Tempo de Decaimento, Força Sonora, Clareza, Definição e Tempo Central.

5. Conclusão

O método MLS, baseado na *transformada rápida de Hadamard* (FHT), calcula muito eficientemente a resposta impulsiva de sistemas lineares e invariantes no tempo.

Verificou-se que para sistemas acústicos que não respeitam as premissas de linearidade ou invariância no tempo o resultado apresenta presença elevada de ruído. Já o método baseado na FFT apresenta relativa imunidade à distorção harmônica e certa tolerância à variação temporal, mostrando-se mais apropriado para medições acústicas. Com o constante avanço da capacidade de memória e de processamento dos computadores pessoais, o método FFT pode ser facilmente realizado com qualquer computador doméstico.

Os parâmetros acústicos abordados neste projeto são definidos em função de um decaimento idealmente exponencial. A resposta impulsiva medida terá sempre um comportamento não-ideal com a presença de ruído de fundo e atraso devido ao caminho acústico. Três destes métodos foram implementados visando reduzir este efeito, mas não foi realizado um estudo objetivo para a escolha de um desses métodos como o método ideal para o projeto.

Os parâmetros acústicos devem ser calculados por bandas de frequência. Para isto, foram implementados filtros de oitava de acordo com normas internacionais. Para reduzir o efeito de distorção causado pela fase não linear dos filtros IIR, utilizou-se a técnica da filtragem causal.

As vantagens da medição da resposta impulsiva baseada na excitação da sala por uma varredura logarítmica deconvoluída via FFT frente à técnica da MLS ficou clara através das medições realizadas. Este é, portanto, o método indicado para ser implantado no aplicativo de medição do software que está sendo desenvolvido pelo projeto AcMus.

Referências Bibliográficas

- [1] Vorländer M., Bietz H. 1994. Comparison of Methods for Measuring Reverberation Time. *Acustica* Vol. **80**. pp. 205–215.
- [2] ISO 3382:1997. Acoustics – Measurement of the reverberation time of rooms with reference to other acoustical parameters.
- [3] Burt, Phillip M. S. “Measuring Acoustic Responses with Maximum-Length Sequences”. In ITS Proceedings, pp 284-289. Agosto 1998.
- [4] Müller, S., Massarani P., “Transfer Function Measurements with Sweeps”. *J.AES*, Vol. 49, number 6, pp.443. 2001.
- [5] Chu W. T. “Comparison of reverberation measurements using Schroeder's impulse method and decay-curve averaging method”. *J. Acoust. Soc. Am.* **63**(5), May 1978. pp. 1444–1450.
- [6] Hirata Y. “A Method of Eliminating Noise in Power Responses” *J. Sound Vib.* vol. **82** pp. 593–595. 1982.
- [7] Lundeby A., Vigran T.E., Bietz H., Vorländer M. 1995. Uncertainties of Measurements in Room Acoustics. *Acustica* Vol. **81** (1995). pp. 344–355.
- [8] Peltonen, Timo. A Multichannel Measurement System for Room Acoustics Analysis. This thesis has been submitted for official examination for the Degree of Master of Science, in Espoo, on October 23rd, 2000.

ESTUDO EXPERIMENTAL DA SONORIDADE “CHALUMEAU” DA CLARINETA ATRAVÉS DE PROJETO FATORIAL

Luís Carlos Oliveira¹, Ricardo Goldemberg², Jônatas Manzolli³

^{1,2,3}Instituto de Artes – Universidade Estadual de Campinas (UNICAMP) e Núcleo Interdisciplinar de Comunicação Sonora (NICS-UNICAMP)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{luis,ricardo,jonatas}@nics.unicamp.br

Abstract. *The sonority **chalumeau** of a clarinet is investigated empirically. A Factorial Design of Experiments was used as an important tool of analysis. In this paper the attention is on the effect of the several factors on the intensity and frequency of the sound produced. As a secondary result we have investigated if there are interactions between the factors.*

Resumo. *A sonoridade **chalumeau** da clarineta é investigada de modo empírico através de um método de otimização de experimentos conhecido como Projeto Fatorial de Experimentos. Neste trabalho a atenção é voltada para a determinação do efeito de diversas variáveis sobre a intensidade e frequência do som emitido nesta região. Como resultado paralelo foi investigado se existe interação entre as variáveis estudadas.*

1. Introdução

O processo sistemático de investigação científica a respeito dos instrumentos de sopro teve seu início no final do século XIX. Desde então o timbre desses instrumentos é uma preocupação constante. Dentre os autores representativos, destacam-se Helmholtz (1887), e mais recentemente, Backus (1974, 1978, 1985) e Benade (1976, 1985, 1988). Ainda que de maneira limitada, as contribuições desses e outros investigadores permitem afirmar que, em nossos dias, existe um corpo de conhecimento teórico e experimental que possibilita descrever e simular razoavelmente o comportamento dos instrumentos musicais de sopro, Fletcher e Rossing (1998), Nederveen (1998) e Hall (1990).

Entretanto, devido à complexidade do problema em estudo, várias simplificações são impostas. Deste modo, os resultados obtidos divergem consideravelmente das condições reais, tanto do ponto de vista teórico quanto do experimental. Com relação aos trabalhos empíricos, notamos um elevado grau de preocupação com o sistema oscilador composto pelo conjunto formado pela boquilha e palheta.

Nossa proposta consiste primeiro, em eliminar a variável subjetiva do músico, segundo, trabalhar simultaneamente com um conjunto maior de variáveis e finalmente, estabelecer se há interação entre estas variáveis, Oliveira et alii (2005). Para tal objetivo

utilizaremos uma montagem experimental instalada no estúdio do NICS (Núcleo Interdisciplinar de Comunicação Sonora). Em seguida, vamos analisar o efeito das variáveis na sonoridade (energia sonora e frequência) *chalmereau* (região grave) da clarineta.

2. Aparato e Procedimento Experimentais

O aparato consiste basicamente de cinco unidades: 1) compressor; 2) tanque “pulmão” que simula o reservatório de ar no corpo humano; 3) unidade de contato com a palheta, que daqui por diante denominaremos por “mordedura”; 4) unidade formada pela clarineta e 5) unidade de captação de dados. Com exceção da unidade 5, as demais podem ser visualizadas nas figuras 1 e 2.

Instrumentos de medida como um rotâmetro (medidor da vazão volumétrica de ar que passa pela clarineta) e dois manômetros (para medir a pressão na entrada do tanque e no interior do tanque) complementam o sistema.



Figura 1. Visão Geral do Aparato Experimental.

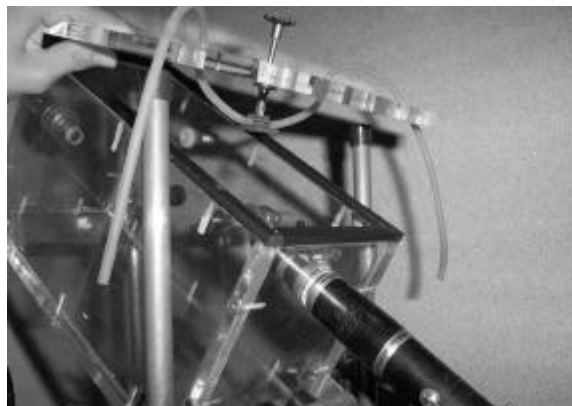


Figura 2. Detalhe da Unidade de Contato com a Palheta: “Mordedura”.

Primeiro enche-se o compressor. As chaves da clarineta são fechadas com pequenas borrachas que podem alterar a nota desejada. No tanque pulmão o contato com a palheta é mantido fechado, impossibilitando a passagem de ar através da palheta. Abre-se lentamente a válvula de saída do compressor de modo que os manômetros indiquem o aumento gradativo da pressão. Quando os manômetros indicarem a pressão de fundo de escala (9808 Pa) abre-se muito lentamente o contato com a palheta de

modo que uma pequena quantidade de ar é injetada no interior da clarineta. As pressões indicadas pelos manômetros começam a diminuir como consequência.

Deste momento em diante a clarineta está em estado de emissão sonora. Uma vez emitido o som a gravação é acionada no momento que se percebe que sua intensidade é máxima e constante. A gravação é efetuada por volta de três minutos. Desta gravação apenas 15 segundos são selecionados para a análise.

Depois de uma bateria de testes, estabelecemos que as variáveis que poderiam estabelecer certa influência na sonoridade da clarineta foram: i) volume do tanque pulmão; ii) dureza da palheta; iii) posição de contato da mordedura na palheta; iv) ângulo de abertura da boquilha; v) Tipo de mordedura (área de contato com a palheta) vi) quantidade de material absorvente sonoro (estopa) no tanque pulmão. Para o estudo destas variáveis utilizamos um método estatístico conhecido por *Projeto Fatorial de Experimentos*. Em particular, foi aplicado um *Projeto Fatorial Fracionado com Resolução III*, Box et alii (1978). Os níveis dos respectivos fatores (variáveis independentes) podem ser conferidos na tabela 1. A boquilha **A** tem a menor abertura enquanto a **C**, a maior.

Tabela 1. Níveis dos Fatores Envolvidos na Experiência

FATORES	-1	0	+1
1)Volume Vazio do Tanque Pulmão(%)	60	65	70
2)Dureza da Palheta (N ^o)	2	2,5	3
3)Posição da Mordedura na Palheta	Interna	Centro	Externa
4)Boquilha	A	B	C
5)Área de Contato com palheta (m ²)	3x10 ⁻⁵	1x10 ⁻⁴	1,4x10 ⁻⁴
6)Quantidade de Estopa (kg)	0	0,015	0,030

3. Análise dos Resultados e Conclusões

Observamos que a área de contato com a palheta mostrou ser a variável de maior influência, tanto sobre a energia sonora emitida (intensidade) como sobre as frequências dos componentes espectrais. No entanto, neste experimento a variação da área de contato (mais de 450%) é muito grande com relação às demais variáveis. Como consequência, a magnitude de seu efeito é bastante superior. Por isso não devemos descartar a influência das outras variáveis, pelo menos por enquanto. Vale mencionar que o material para absorção sonora não teve muita relevância na sonoridade.

Notamos que as variáveis agem de modo homogêneo ao longo dos harmônicos. Na região *chaleur*, o 2^o e o 4^o harmônicos têm intensidades bastante reduzidas quando aumentamos a área de contato com a palheta.

O aumento do volume vazio, de 60% para 70% do volume total provoca uma diminuição da intensidade da fundamental e de seu 8^o harmônico. Porém ele provoca o aumento da intensidade do 3^o, 6^o, 10^o e do 12^o harmônicos. Os valores envolvidos apresentam magnitudes relativamente altas para a pequena variação de apenas 10% do volume total.

Quando passamos a mordedura de uma posição mais interna para uma mais externa notamos a tendência em enriquecer o 2^o harmônico e empobrecer o 4^o. No

entanto notamos que os harmônicos superiores também são empobrecidos quando a posição da mordedura é mais externa.

O aumento da área de contato de $0,3 \text{ cm}^2$ para $1,4 \text{ cm}^2$ diminui a frequência dos componentes espectrais. A dureza da palheta é o fator com a segunda maior influência sobre a frequência nesta região. O aumento da dureza da palheta de No 2 para 3, faz aumentar o valor das frequências dos componentes espectrais.

4. Referências

- Helmholtz, H.L.F. (1877). On the Sensations of Tone as a Physiological Basis for the Theory of Music. 4th ed., trad. ELLIS, A.J. (Dover, New York, 1954)
- Backus, J. (1985). The effect of the player's vocal tract on the woodwind instrument tone. J. Acoust. Soc. Am. 78, 17-20.
- Backus, J. (1978). Multiphonic tones in the woodwind instruments. J. Acoust. Soc. Am. 63, 591-599.
- Backus, J. (1974). Input impedance curves for the reed woodwind instruments. J. Acoust. Soc. Am. 56, 1266-1279.
- Benade, A.H. e Kouzoupis, S.N. (1988). The clarinet spectrum: Theory and experiment. J. Acoust. Soc. Am. 83, 292-304.
- Benade, A.H. e Larson, C.O. (1985). Requirements and Techniques for measuring the musical spectrum of the clarinet. J. Acoust. Soc. Am. 78, 1475-1498.
- Benade, A.H. (1976). Fundamentals of Musical Acoustics. Oxford University Press, New York.
- Nederveen, C.J. (1998). Acoustical Aspects of Woodwind Instruments. Northern Illinois University Press, DeKalb, Illinois.
- Box, G.E.P.; Hunter, W.G.; Hunter, J.S. (1978). Statistics for Experimenters – An Introduction to Design, Data Analysis and Model Building. John Wiley & Sons, NY.
- Hall, D.E. (1990). Musical Acoustics. Pacific Grove, CA: Brooks/Cole Publishing Monterey: Brooks/Cole.
- Fletcher, N.H., Rossing, T.D. (1998). The Physics of Musical Instrument. Springer-Verlag, 2nd ed., NY.
- Oliveira, L.C., Goldemberg, R., Manzolli, J. (2005) Estudo Experimental da Sonoridade “Chalumeau” da Clarineta através de Projeto Experimental. Anais da IX Convenção Nacional da Sociedade de Engenharia de Áudio, São Paulo, SP.

Simulação de Performances de Violão por Agentes Artificiais

Leandro L. Costalonga, Luciano V. Flores, Evandro M. Miletto, Rosa M. Vicari

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{l1costalonga,miletto,rosa}@inf.ufrgs.br, lflores@cpovo.net

Abstract. *This work describes, from a user perspective, a multi-agent system able to simulate a guitar player. In this system the elements involved in a guitar musical performance were modeled as artificial agents. Harmony and rhythm musical elements are separated and defined independently, respectively as Left-Hand and Right-Hand agents. We also summarize and briefly discuss here the implemented rules for chord shape calculus. Finally, we outline plans to include affective computing in this work, giving emotional skills to the Right-Hand agent, aiming at improving its expressiveness.*

Resumo. *Este trabalho apresenta, do ponto de vista do usuário, um sistema multiagente capaz de simular um violonista. No sistema, os elementos identificados em uma performance de violão foram modelados como agentes artificiais. A harmonia e o ritmo são definidos separadamente e estão respectivamente representados pelos agentes Mão-Esquerda e Mão-Direita. Também resumimos e discutimos brevemente os critérios implementados para o cálculo dos desenhos de acordes. Por fim, esboçamos os planos de direcionar o trabalho à computação afetiva, pela inclusão de habilidades emocionais ao agente Mão-Direita, aumentando sua expressividade.*

Resumo Estendido

Este trabalho apresenta um sistema multiagente capaz de simular performances musicais em violão. São consideradas as limitações, tanto do instrumento como do violonista, tendo como entrada somente a(s) linha(s) harmônica(s) e o(s) padrão(ões) rítmico(s). Apresentamos aqui as funcionalidades do sistema do ponto de vista do usuário, sem entrar em detalhes técnicos de implementação.

O principal objetivo deste trabalho é demonstrar que a abordagem multiagente é eficaz para simular performances musicais de violão. Outro objetivo é o desenvolvimento de uma ferramenta que permita a criação da parte de violão em uma composição, sem que seja necessário conhecimento da execução do instrumento e ressaltando principalmente os aspectos rítmicos. O uso desta ferramenta está voltado para a composição musical, porém pode ser útil também nas seguintes atividades:

- Criação da parte de violão em uma composição por compositores que não tocam o instrumento.
- Execução de músicas para violão com alto grau de dificuldade.
- Auxílio na educação musical, principalmente de conceitos rítmicos e de formação de acordes.

- Experimentos musicais diversos, principalmente trabalhando a polirritmia, características do instrumento e extrapolação das limitações de execução humanas. Este foi o ponto predominantemente avaliado nesta pesquisa.
- Sugestão dos desenhos de acordes e suas transições, dada uma harmonia e o padrão rítmico, como uma espécie de dicionário de acordes dinâmico.

Em uma execução musical em violão observam-se elementos distintos que, trabalhando em conjunto, produzem a sonoridade desejada pelo músico. Nossa ferramenta trata estes elementos como agentes artificiais capazes de se comunicarem e tomarem decisões relativas à execução da música neste instrumento específico, tal qual um violonista humano o faria. Buscando facilitar a compreensão em software musical, é comum haver a separação dos elementos musicais em harmonia, melodia e ritmo. Esta separação é traduzida neste sistema pelos papéis dos agentes modelados sendo, respectivamente, Agente Mão-Esquerda, Agente Solista e Agente Mão-Direita. Além destes foi ainda modelado o Agente Caixa de Som, responsável por fazer soar as notas (correspondente, na modelagem, ao elemento “instrumento”).

O Agente Solista, cuja implementação encontra-se em andamento, será responsável por tocar uma melodia que influenciará na escolha dos desenhos dos acordes e possivelmente eliminará a necessidade de se informar a harmonia (geração de ritmo e harmonização automáticas).

O Agente Mão-Esquerda (ME) agrega algumas das tarefas mais trabalhosas e cognitivas em suas responsabilidades. As principais tarefas do Agente ME são:

- Reconhecimento e validação das cifras do acorde baseado em uma notação que pode ser personalizada pelo usuário.
- Cálculo do acorde informando ao usuário as notas e intervalos que compõem aquele acorde.
- Cálculo do desenho do acorde considerando as propriedades do instrumento virtual utilizado (quantidade de cordas, afinação, quantidade de trastes) e características do instrumentista, como: quantidade de dedos da mão esquerda, quantidade de dedos da mão direita, abertura máxima de dedos.
- Escolha do melhor desenho de acorde considerando-se: facilidade na transição do desenho de acorde anterior, facilidade na execução do desenho de acorde (quantidade de dedos, proximidade da cabeça do violão, abertura de dedos, uso de pestana), e o padrão rítmico. O primeiro desenho de acorde é escolhido somente pela proximidade em relação à cabeça do violão, garantindo que soe em uma região mais agradável.

O Agente Mão-Direita (MD) permite a definição de um padrão rítmico para violão através de uma interface desenvolvida para este fim. Após obter as notas do acorde do Agente ME, e com base no padrão rítmico, escalona-as no tempo e envia para o Agente Caixa de Som.

É responsabilidade do Agente MD:

- Permitir a definição de padrões rítmicos para violão, considerando:

- Polifonia do padrão: quantidade de cordas necessárias para executar o padrão e, conseqüentemente, quantidade de vozes do acorde.
- Quantidade de tempos de todo o padrão.
- Figura de tempo: duração de cada um dos tempos do padrão.
- Associação do padrão rítmico à harmonia do agente ME ao qual está vinculado.
- Permitir audição da composição com o ritmo definido neste agente.

O Agente Caixa de Som (CS) sintetiza e *mixa* as notas provenientes das interações entre todos os agentes ME e MD. É usado quando se deseja ouvir a música fazendo pequenas alterações nos agentes, tais como volume, timbre, mudo ou solo. É o único agente que possui uma interface gráfica e executa sobre o computador cliente. Todos os demais agentes são configurados através de sua interface, mas podem executar em outras máquinas. Podem-se ter diversos Agentes CS na composição e com isso experimentar diversas configurações dos parâmetros de execução.

Atualmente o protótipo implementado permite uma única composição por vez e todos os agentes são criados no contexto desta composição. O sistema é mono-usuário, executa localmente e, apesar de tecnicamente possível, os agentes não estão distribuídos.

Após a tela inicial do sistema, onde o usuário pode criar ou recuperar uma composição, o próximo passo é a criação da(s) linha(s) harmônica(s) da música, representada(s) pela criação de Agente(s) Mão-Esquerda. O método de inserção da harmonia é baseado no método de composição binário e ternário, onde partes são criadas e repetidas durante a composição. Estas partes são compostas pelas cifras representando os acordes. A não ser pela ordem com que os acordes serão tocados, nenhuma informação rítmica, de andamento ou duração, é definida neste momento. A Figura 1 mostra a criação de uma linha harmônica, para a qual o Agente ME irá propor os desenhos dos acordes, a princípio sem interferência direta do usuário.

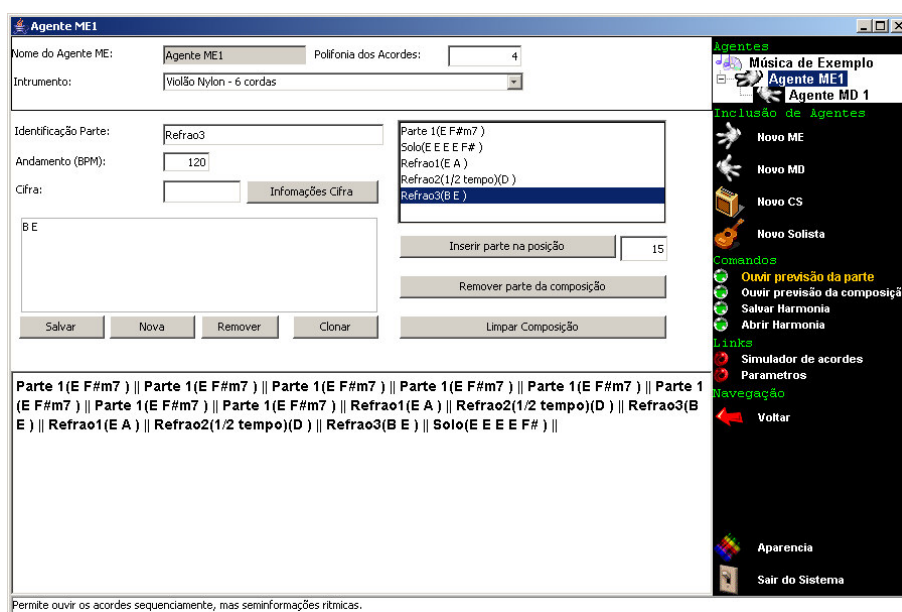


Figura 1. Definição da harmonia no Agente ME.

O usuário pode determinar o desenho para um acorde específico que deseja usar na composição. Essa decisão implica diretamente na escolha dos desenhos de acordes posteriores a ele, pois o sistema calcula o desenho do acorde primordialmente pela semelhança com o anterior, favorecendo uma digitação no violão sem grandes sobressaltos. Para que o usuário escolha um desenho de acorde basta que ele informe as cordas e casas sequencialmente após a cifra, por exemplo, para o acorde de Dó maior poderíamos inserir **C<53-42-30-21-10>**, onde lê-se 53 como quinta corda, terceira casa. Já para o cálculo automático dos desenhos de acordes, vários parâmetros podem ser configurados no sistema como, por exemplo, dobrar ou não a nota fundamental. Ainda na tela do Agente ME, também é possível solicitar ajuda para entender uma determinada cifra, ou ver os desenhos de acordes possíveis e qual seria o próximo na escolha do sistema.

Para a escolha de um desenho de acorde adequado o sistema considera:

Agradabilidade sonora: A escolha do primeiro desenho de acorde pode ditar toda a região com que os acordes subsequentes são tocados (altura), por isso, nem sempre o desenho de acorde mais fácil é o melhor a ser escolhido como primeiro, pois o mesmo pode estar em uma região muito aguda. Portanto, para o primeiro desenho de acorde foi implementado um algoritmo que escolhe o desenho cuja média da soma das casas seja a menor, com isso escolhe-se os desenhos mais próximos da cabeça do violão.

Facilidade de execução: Menor abertura de dedos, menor quantidade de dedos, maior número de cordas soltas e proximidade da cabeça do violão.

Similaridade com o acorde anterior: A partir do segundo acorde, o sistema automaticamente procura por desenhos de acorde que mais se equivalem ao desenho anterior, que satisfaçam aos requisitos do padrão rítmico e restrições do usuário. Neste caso os desenhos possíveis são ordenados pela facilidade de execução. Não é considerada a condução de vozes [Forte e Gilbert 1982].

Adequação ao padrão rítmico: Polifonia (máxima e mínima), quantidade de dedos da mão direita (1 representando palheta) e necessidade de cordas contíguas (*strum* ou *palhetada*).

Características do instrumentista: Quantidade de dedos da mão esquerda e abertura de dedos (medida em trastes).

Características do instrumento: Quantidade de trastes e cordas; afinação.

A consideração de todos esses parâmetros torna o cálculo complexo e até imprevisível do ponto de vista do usuário, podendo até ocorrer de um mesmo acorde ter diversos desenhos em uma mesma música.

Nesta etapa da criação de uma composição, somente com as informações inseridas e geradas pelo Agente ME, ainda não é possível tocar a sequência de acordes designada. Para que seja possível executar a música é necessário que pelo menos um padrão rítmico seja associado à harmonia e isso é feito através do Agente Mão-Direita.

Para se criar um Agente Mão-Direita é necessário associá-lo a um Agente ME. No protótipo, a cardinalidade é de um Agente ME para muitos Agentes MD (no mínimo um para conseguir algum resultado sonoro). Após a criação do agente MD podemos criar vários padrões rítmicos. Cada padrão rítmico pode ser associado a nenhuma ou a

várias partes harmônicas definidas no agente ME. A Figura 2 mostra a tela para definição do padrão rítmico.

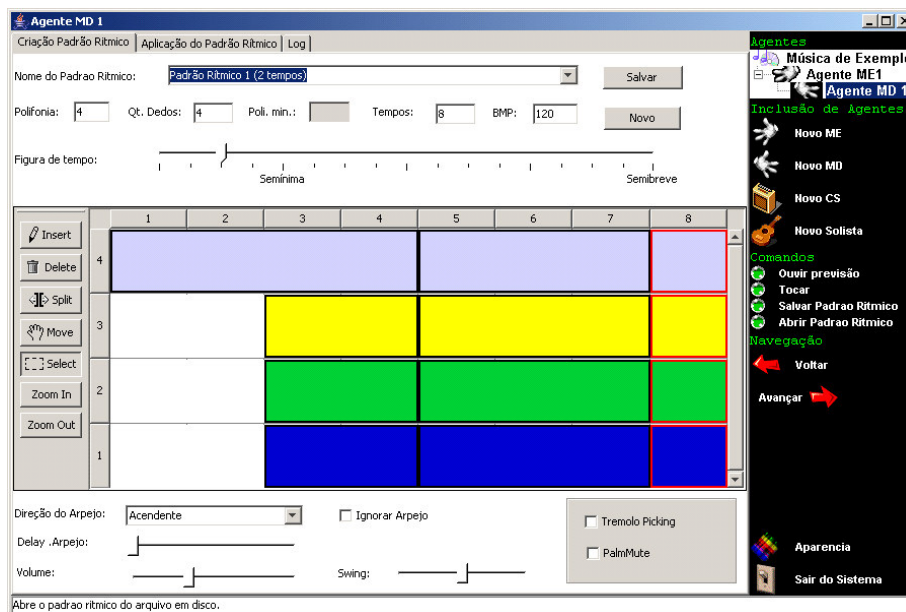


Figura 2. Interface de definição do padrão rítmico.

A definição do padrão rítmico é o grande diferencial desta proposta merecendo considerações detalhadas. O principal componente desta interface de criação do padrão rítmico é a tabela representada por polifonia (linhas) versus tempo (colunas), onde cada marca gráfica, chamada de evento musical, representa o som com o ataque no tempo onde começa durando até a tempo onde se encerra. É importante salientar que no ato da definição do padrão rítmico não é conhecido quais são as notas que representam cada evento musical. Por isso, cada linha da tabela representa uma das notas que o agente ME vai calcular, sendo as mesmas ordenadas da mais grave (linha superior) para a mais aguda (linha inferior).

A partir da definição de, ao menos, uma parte harmônica e um padrão rítmico através dos respectivos agentes ME e MD, é possível solicitar a execução da música, que pode ocorrer de duas maneiras: através do próprio agente MD que tocará somente os seus padrões rítmicos ou através do agente Caixa de Som (CS) que toca todos os agentes simultaneamente.

O sistema vem sendo testado e avaliado por alunos do curso de música da UFRGS e entusiastas da música em geral (16 pessoas no total). A maior parte dos avaliadores conhece e toca violão com diferentes graus de habilidade, porém todos têm um bom conhecimento de informática. Esses avaliadores relataram algumas limitações do sistema que foram sendo corrigidas nas últimas versões e por isso não serão mencionadas. Citaremos somente as limitações mais conceituais, cujas soluções estão previstas nos trabalhos futuros:

- Visualização dos desenhos de acordes sobre a imagem do instrumento.
- Externalização das partes configuráveis do sistema para que o usuário possa fazê-lo a qualquer momento.

- Captura do padrão rítmico por instrumentos MIDI.
- Melhoria do mecanismo de associação dos padrões rítmicos à harmonia.
- Criação de bibliotecas de padrões rítmicos.
- Consideração da melodia na escolha dos desenhos de acorde.

Quanto ao cálculo de acordes, o sistema não pode garantir que o desenho escolhido é o melhor para uma determinada situação, mas simplesmente que ele é adequado. Nem sempre o desenho do acorde escolhido traz um bom resultado sonoro, talvez por não ter sido considerada a condução de vozes [Forte e Gilbert 1982]. Nota-se também uma certa tendência do sistema a levar a música para regiões mais agudas do violão, devido aos pesos definidos no algoritmo que calcula a similaridade entre os desenhos de acorde. Soluções para contornar esse problema ainda estão em estudo.

No aspecto rítmico a interface gráfica implementada mostra-se eficiente para definir arpejos simples (dedilhados), entretanto pode ser muito trabalhoso definir certos padrões rítmicos com figuras de tempo muito pequenas e muitos tempos, como ocorre em *rasgueados*.

Nossa meta futura é o aumento da capacidade cognitiva dos agentes, em especial do agente MD. O trabalho terá continuidade visando permitir que os agentes possam não somente executar uma peça musical no violão, mas também demonstrar expressividade ao executá-la. Para isso, pretende-se dotar o Agente MD com a noção de emoção.

Pesquisas na área da neurociência demonstraram que as emoções reforçam e agilizam o mecanismo de tomada de decisão quando a situação demanda ações urgentes. Em situações onde há tempo para decidir, geralmente, as decisões são baseadas em processos que envolvem raciocínio e dedução [Ventura et al. 1999]. Segundo Damasio, quanto maior a urgência e seriedade da situação menos racional e mais emocional serão nossas decisões. Esta é a forma dos humanos lidarem com a complexidade [Damasio 1994]. A música é uma atividade dependente do tempo, logo susceptível a tais ações emocionais. A música também pode ser vista como uma forma de comunicação e, assim, pode servir como uma ponte mediadora na captura e expressão da emoção pelo computador [Nemirovsky e Davenport 1999].

Ainda são necessários experimentos para determinar as potencialidades e restrições do sistema, entretanto pode-se afirmar que o mesmo atinge seus propósitos iniciais. A ferramenta encontra-se em estado experimental, relativamente estável e disponível para uso e teste por toda a comunidade interessada.

Referências

- Damasio, A. R. (1994) *Descartes' Error: Emotion, Reason and the Human Brain*. [S.l.:] Avon Books.
- Forte, A.; Gilbert, S. (1982) *Introduction to Schenkerian Analysis*. New York: Norton.
- Nemirovsky, P.; Davenport, G. (1999) "GuideShoes: Navigation Based on Musical Patterns". In: CHI99 Extended Abstracts. New York: ACM. p.266-267.
- Ventura, R.; Custódio, L.; Pinto-Ferreira, C. (1999) "Artificial Emotions - Good Bye Mr. Spock!" In: *Cognitive Science*. Tokyo: [s.n.]. p.938-941.

Author index

A

Agon, Carlos	1
Álvaro, Jesus L.	36
Araújo, Érika P.	303
Araújo, Leonardo C.	346
Assayag, Gerard	1

B

Barbedo, Jayme G. A.	271, 291
Barbosa, Virgínia	303
Barros, Beatriz	36
Brandão, Marcio	329
Bresson, Jean	1
Briot, Jean-Pierre	319, 333

C

Cabral, Giordano	319, 333
Cabrera, Andrés	237
Caetano, Marcelo	94
Castro, Paullus M. de S. N.	329
Costa, César	94
Costalonga, Leandro	70, 82, 374
Cruz, Fernando W.	329

D

Da Silva, Andrey R.	295
Da Silva, Flávio F.	311
Da Silva, Maria Carolina F.	329
Dahia, Márcio	154
Dantas, Vítor	287
De Almeida, Anselmo G.	207
De Araújo, Leonardo C.	275
De Campos, Ignácio	256
De Lima, Ernesto T.	154
De Oliveira, André Luiz G.	355
De Oliveira, Luis Felipe	355
De Oliveira, Rafael	249, 299
De Paula, Hugo B.	195

De Sousa, Karen P.	329
De Souza, Marcio G. V.	329
Dias, Luciana	130

F

Faria, Regis R. A.	106
Ferneda, Edilson	329
Figueiredo, Fábio L.	118
Figueiró, Cristiano	207
Flores, Luciano V.	70, 374, 299
Fornari, José	283
Freire, Sérgio	219
Fritsch, Eloi Fernando	249, 299
Furlanete, Fábio	325

G

Garcia, Denise	264
Gimenes, Marcelo	13, 164
Gois, Matheus C.	303
Goldemberg, Ricardo	370
Goulart Jr, Fernando S.	329
Guigue, Didier	279

I

Iazzetta, Fernando	118, 364
Ichizo, Ricardo	352

J

Jefferson, Bruno	279
Johnson, Chris	13

K

Kaestner, Celso A. A.	48
Koerich, Alessandro L.	48
Kon, Fabio	118
Kröger, Pedro	307
Krotoszynski, Andrea C. B.	226

L

Lima, Luis Guilherme P.	355
Lopes, Amauri	271, 291
López ,Ernesto	142
Loureiro, Maurício A.	195, 311, 346
Luvizotto, André Luiz	352

M

Madsen, Søren T.	154
Magalhaes, Tairone N.	346
Maia Jr. , Adolfo	58, 164, 283
Mamedes, Clayton R.	264
Manzoli, Jônatas	94, 164, 226, 283, 325, 352, 370
Maranhão, Suzana M.	303
Martins , João M.	164
Masiero, Bruno S.	364
Miletto, Evandro M.	70, 82, 374
Miranda, Eduardo R.	13, 36, 58
Mitre, Adriano	174
Moraes Neto, Jarbas	226
Muniagurria, Rodrigo A.	249

O

O'Maidin, Donncha	186
Oliveira, Luís Carlos	370

P

Pachet, François	319, 333
Pimenta, Marcelo S.	82
Puig, Daniel F.	338

Q

Queiroz, Marcelo	130, 174
------------------------	----------

R

Ramalho, Geber	24, 154, 287, 303
Rocamora, Martín	142

Rolim, André L.	279
----------------------	-----

S

Sampaio, Pablo	24
Santos, Breno T.	106
Scavone, Gary	295
Scholz, Ricardo	287
Silla Jr., Carlos N.	48
Soares, Luciano	106
Souza, Damares P. M.	346

T

Tedesco, Patricia	24
Teixeira, Luciênio de M.	329
Thomaz, Leandro F.	106
Toffolo, Rael B. G.	355
Trevilatto Junior, Narciso	291

U

Ueda, Leo K.	118
-------------------	-----

V

Vicari, Rosa	70, 82, 374
--------------------	-------------

W

Walsh, Riana	186
Widmer, Gerhard	154

Y

Yehia, Hani C.	195, 346
---------------------	----------

Z

Zuben, Fernando V.	94
Zuffo, João Antônio	106
Zuffo, Marcelo K.	106