# Organizing Committee

## General Chairs

Fabio Kon (Universidade de São Paulo)
Fernando Iazzetta (Universidade de São Paulo)

## Program Chairs

Technical papers chair: Geber Ramalho (Universidade Federal de Pernambuco)
Music papers chair: Mikhail Malt (IRCAM/Sorbonne-Paris IV)

## Local Arrangements Chair

Marcelo Queiroz (Universidade de São Paulo)

## Technical Papers Program Committee

Adolfo Maia Jr. (Universidade Estadual de Campinas)
Aluizio Arcela (Universidade de Brasilia)
Andrew Horner (The Hong Kong University of Science & Technology)
Chris Chafe (Stanford University)
Edilson Ferneda (Universidade Católica de Brasília)
Eduardo Miranda (University of Plymouth)
Emilios Cambouropoulous (Aristotle University of Thessaloniki)
Fabio Kon (Universidade de São Paulo)
Flávio Soares Silva (Universidade de São Paulo)
François Pachet (Sony Computer Science Laboratory)
Gérard Assayag (IRCAM)
Geber Ramalho (Universidade Federal de Pernambuco)
Giordano Cabral (Université Paris 6)
Henkjan Honing (University of Amsterdam)
Hugo de Paula (PUC Minas)
Ian Whalley (University of Waikato)
Jean-Pierre Briot (CNRS - Université Paris 6 & PUC-Rio)
Jônatas Manzolli (Universidade Estadual de Campinas)
Lelio Camilleri (University of Bologna)
Luis Jure (Universidad de la República)
Marcelo Pimenta (Universidade Federal do Rio Grande do Sul)
Marcelo Queiroz (Universidade de São Paulo)
Marcelo Wanderley (McGill University)
Marcio Brandao (Universidade de Brasilia)
Maurício Loureiro (Universidade Federal de Minas Gerais)
Oscar Di Liscia (Universidad Nacional de Quilmes)
Palle Dahlstedt (Göteborg University/Chalmers University of Technology)

Peter Beyls (Hogeschool Gent)
Petri Toiviainen (University of Jyvaskyla)
Regis R. A. Faria (Universidade de São Paulo)
Roger Dannenberg (Carnegie Mellon University)
Rosa Viccari (Universidade Federal do Rio Grande do Sul)
Sever Tipei (U. of Illinois School of Music)
Victor Lazzarini (National University of Ireland)

### Music Papers Program Committee

Mikhail Malt (chair) (IRCAM/Sorbonne-Paris IV)
Alexandre Lunsqui (Columbia University, NY -USA)
Damian Keller (Universidade Federal do Acre)
Daniel Teruggi (Ina-GRM, Paris-France)
Fernando Iazzetta (Universidade de São Paulo, Brazil)
Jônatas Manzolli (Universidade Estadual de Campinas, Brazil)
Karim Haddad (IRCAM, Paris-France)
Luiz Naom (Paris Conservatory/Geneva Music School)
Marc Battier (Sorbonne-Paris IV, France)
Martin Supper (Berlin University of the Arts, Germany)

### Webmaster

Carlos Eduardo Moreira dos Santos (Universidade de São Paulo)

### Proceedings Editor

Mário Henrique Cruz Torres (Universidade de São Paulo)

### Cover Art

{Stefano,Giuliano} Mega (Universidade de São Paulo)

# Letter from the Symposium Chairs

Once again, in 2007, the Brazilian Symposium on Computer Music (SBCM) is showing that it is reaching its maturity, growing both in the quality and quantity of papers submitted. The preparation for the 2007 edition of SBCM started in November 2006 with the constitution of the Program Committee (PC). As in previous editions, the PC is composed not only of top 16 Brazilian researchers in Computer Music but also of 20 world-class researchers from renowned institutions in the field such as IRCAM, Stanford, UIUC, and Carnegie-Mellon. The PC has members from 16 different countries.

We received a total of 45 submissions in three categories: full 12-page technical papers, full 12-page music papers, and poster (with 4-page extended abstract). The technical papers describe original research with scientific contributions; we received 31 full technical paper submissions, from which 13 where accepted for presentation during the conference, yielding an acceptance rate of 42%.

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 Sorbonne-Paris IV, Columbia University, Federal University of Acre, Ina-GRM, (France), University of São Paulo, University of Campinas, Paris Conservatory, Geneva Music School, Berlin University of the Arts, and IRCAM, selected 5 papers to be presented during the symposium. These selected works cover different approaches to music composition using computing technology.

We are grateful for the valuable help provided by Prof. Geber Ramalho, the technical program committee chair and by Mikhail Malt, the musical program committee chair, as well as for their respective program committee members. Carlos Eduardo Moreira dos Santos did a great job as Webmaster, Mário Henrique Cruz Torres prepared this elegant proceedings with the beautiful cover art by Giuliano and Stefano Mega.

We are very happy with the quality and diversity of the works presented in this Symposium. Besides the paper and poster presentations, we will have a strong concert program and excellent invited talks and a tutorial by Prof. Roger Dannenberg from Carnegie-Mellon University and Dr. Mikhail Malt from IRCAM/Sorbonne-Paris IV.

We sincerely hope that all participants enjoy these four days in São Paulo 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.

Fabio Kon, Fernando Iazzetta, and Marcelo Queiroz - São Paulo, August, 2007.

# Invited Talks

**Modeling and Simulation in the "Computer Aided Music" context**
Mikhail Malt, Ircam/Sorbonne, Paris IV, France

If, in the last years, the paradigm of computer use in music was embodied by the fields of "Computer aided composition", "Algorithmic Composition" or " Generative Music", nowadays we can see emerging forms of "computer aided performance" and "computer aided musical analysis". In this process, all musical users (performers, musicologists and composers) are asked, as a first step, to explicit and to formalize its own thought. In a subsequent stage, that is the execution phase, the user is asked to build operative models in a given computer environments, where the simulation is at the same time an implementation and a validation process and a definitive goal. Never before, the concepts of formalization, modelization and simulation were so present in the musical world. In this talk, we will try to investigate how these concepts are used en each field, and present the thesis that they are an extension of the activity "to think" and "to know".

**Music Understanding for Music Performance**
Roger B. Dannenberg, Carnegie-Mellon University, USA

Computers offer many opportunities for music performance, ranging from ordinary synthesizers to intelligent interactive systems. I will describe some early work on intelligent computer accompaniment systems that listen to, follow, and accompany live performers and on style classifiers that can detect different improvisational styles in a real time performance. I envision an interesting and diverse "symbiosis" between future computer music systems and human performers. I use the term "symbiosis" because the capabilities of computers and electronics already shape much of the music that we create, and the music we wish to create is shaping the design of future computer music systems. As we create and perfect automated systems for music listening, processing, and synthesis, new opportunities arise for computers in live performance. I will describe some new work in this direction and outline some research opportunities for the future.

# Workshop

**Sound Synthesis and Composition With Nyquist**
Roger B. Dannenberg, Carnegie-Mellon University, USA

Nyquist is a programming language and system that has been developed continuously since the early 90's. It is also the scriptiing language for the popular Audacity audio editor. In this workshop, I will give a brief introduction to Lisp, on which Nyquist is based, and then demonstrate how to use Nyquist for manipulating and creating audio. Participants will learn about the status of Nyquist, how to use the Nyquist IDE to develop Nyquist programs and compositions, and some common misconceptions and errors to avoid. Participants are welcome to try Nyquist in advance (nyquist.sourceforge.net) and bring questions to the workshop.

# Open Talk

**Writing Audacity Plugins with Nyquist**
Roger B. Dannenberg, Carnegie-Mellon University, USA

Audacity is popular, free, open-source, cross-platform audio editor. This talk will introduce Audacity and explain from a Computer Science perspective how a simple, free editor can perform many operations orders of magnitude faster than expensive "professional" editors. While most users see only the graphical interface, Audacity has an embedded scripting language based on the powerful, Lisp-based, Nyquist programming language. I will describe how Audacity interacts with Nyquist and show how to extend Audacity with new signal processing operations. Some examples of how users have extended Audacity include: support for developing podcasts, software to separate multiple songs from LP records into separate files, and new digital audio effects.

# Table of Contents



## Music Papers



## Technical Papers

## Posters

# Multimedia Performance & Concerts

## Multimedia Performance

### TEIA (M-U-R-O Ensemble)

**Saturday, 09/01, 12h00 – IME/USP**

*TEIA* is an interactive performance created by M-U-R-O Ensemble. The project is based on the concept of "capture", from de simple idea of capturing sounds via microphone to the imprisonment of an image, built under the watching of the public, and later "captured" in a screen in the background. The sound construction follows according to a specific script embracing unprocessed sounds to some textures with a great degree of density (using PD and Max), redefining and imposing ambiguities to the referential relation between image and sound result.

**The members of M-U-R-O Ensemble are Lílian Campesato, Andrei Thomaz, Vitor Kisil, Valério Fiel da Costa, Alexandre Porres, Giuliano Obici and Alexandre Fenerich.**

# Electroacoustic Music Concert

**Sunday, 09/02, 20h00 - Centro Universitário Maria Antonia**
**In collaboration with Festival Música Nova**

1) Performance (2007), Dur: 10'
   Sax, violin, electronics and voice
   Akronon: Edson Ezequiel - violin, Rogério Costa - sax, Silvio Ferraz electronics, Annita Malufe – voice and poetry
   Collaborative performance

2) Tormenta em Campos Férteis (2006), Dur: 10'
   Six channel acousmatic composition
   Fernando Iazzetta

3) Klavibm II (1963), dur: 3'
   Piano, generated by in a IBM 1620 computer
   Damiano Cozzella and Rogério Duprat
   Piano: Horácio Gouveia

4) Interferências (2003), Dur: 9'
   Violoncello and electronics
   Daniel Barreiro
   Violoncello: Carol Joly

----- Intermission ---------

5) Pandora (2005), Dur: 7'
   Snare Drum with loudspeaker and electronics
   Sérgio Freire

6) Para Cima, Para Cima e Para Longe (2005), Dur: 8'
   Eight channel acousmatic composition
   Rodolfo Caesar

7) ...sofferte onde serene... (1976), Dur: 14'
   Piano e electronics
   Luigi Nono
   Piano: Horácio Gouveia

Book Release: "Notas.Atos.Gestos", Editora 7 Letras

# Music Papers

# Cyberock: uma programação para banda de rock

**Guilherme Augusto Soares de Castro[1], Sérgio Freire[2]**

Escola de Música – Universidade Federal de Minas Gerais (UFMG) Av. Antônio Carlos, 6627 – campus Pampulha 31270-010 Belo Horizonte – MG – Brazil

[1]Bolsista CAPES (mestrado), [2]Orientador

[1]somba.guilherme@gmail.com, [2]sfreire@musica.ufmg.br

***Abstract.*** *This article describes the general functioning of a Max/MSP patch developed to a rock band called Somba. This patch creates a virtual acoustic environment based in simulation by digital modeling. The acoustical mark reference to be achieved is the sound of a controlled studio environment that could be also obtained in live performances and therefore, passive to interaction and manipulation. Besides, this patch serves as a base to future sub-patches that explore interactivity as a musical composition element.*

***Resumo.*** *Este artigo descreve o funcionamento geral de uma programação, feita no ambiente Max/MSP, para a banda de rock Somba. Através dela cria-se um ambiente acústico virtual baseado em simulação por modelagem digital. A referência de marca acústica a ser alcançada é a sonoridade do ambiente controlado de estúdio e que seja possível de se obter em performances ao vivo e portanto, passíveis de manipulação e interação. Além disso, ela serve de base para futuras programações que explorem a interatividade como elemento de composição musical.*

## Introdução

O ambiente de programação Max/MSP é bastante versátil e tem sido utilizado em diversas aplicações e em vários contextos, de instalações interativas multimídia até concertos eletroacústicos. Apesar de sua versatilidade, há poucos exemplos de programação em Max voltados para a música popular, sobretudo em um contexto de concerto de rock. Este artigo traz a descrição do funcionamento de uma programação (*patch,* no jargão Max) desenvolvida especialmente para esse contexto - para ensaio e concerto de rock - através de um estudo de caso.

A metodologia utilizada se delineou pelas necessidades da banda em questão, o Somba[1]. Estas necessidades tornam-se aparentes à medida em que a experiência da banda com um sistema deste tipo se aprofunda. Além disto, foi utilizada a experiência dos autores deste artigo: como técnico de som de bandas de rock (como Cálix[2] e Cartoon[3], durante sete anos); como engenheiro de gravação de vários discos (como A Outra Cidade[4], Clube da Esquina dos Aflitos[5], entre outros); como co-produtor musical e membro da banda em questão. Aliado à experiência profissional, utilizamos dados recolhidos na pesquisa de mestrado sobre sistemas musicais interativos, em andamento.

## Breve contextualização

A banda Somba foi criada em 1998, inicialmente por Guilherme Castro (guitarra e voz) e por Avelar Júnior (baixo elétrico e voz). Logo depois, Carlos França (bateria) se juntou à banda para um trabalho sob o formato de um trio de rock. Em 2005, após uma pausa de um ano, a banda retornou com outra formação: além dos dois membros iniciais, entraram Vladmir Agostini (guitarra e voz) e Léo Dias (em substituição ao baterista Carlos França). Nesta volta, influenciados pela nascente pesquisa de Guilherme Castro em sistemas musicais interativos, a banda abraçou a idéia de se fazer rock através de meios eletrônicos e computacionais. Isto tornou viável a experimentação e teste das programações feitas, bem como uma resposta imediata às questões e impressões da banda em relação ao sistema em si e ao novo jeito de se fazer música que dali surgira.

As primeiras questões começaram a surgir: que material a banda dispunha para montar o ambiente sonoro? Como este poderia ser explorado? Como seria a interação musical dentro deste ambiente? Após um mapeamento, chegou-se à seguinte relação de equipamentos: 1 Macbook Intel Core Duo de 1.83 GHz com 1GB de memória RAM; 1 Powerbook G4 de 1GHz com 256 MB de memória RAM, 1 interface de áudio Motu 828 e 1 interface de áudio Motu Traveler; 1 pedaleira controladora MIDI Behringer modelo FCB1010 e 1 processador de efeitos e pedaleira MIDI Zoom modelo GFX-7; 1 processador de efeitos para contrabaixo Zoom modelo 504; 1 mesa Behringer modelo UB1002; 1 interface MIDI-USB Midisport Uno; 1 amplificador e mixer de fones de ouvido, da Behringer, modelo Powerplay HA4700. Aliam-se a esses equipamentos microfones diversos: microfones dinâmicos para as vozes, um par de microfones condensadores para uma captação geral da bateria e mais um kit de microfones especiais para bateria, também dinâmicos. Tais equipamentos foram conectados de acordo com o esquema mostrado na figura 1.

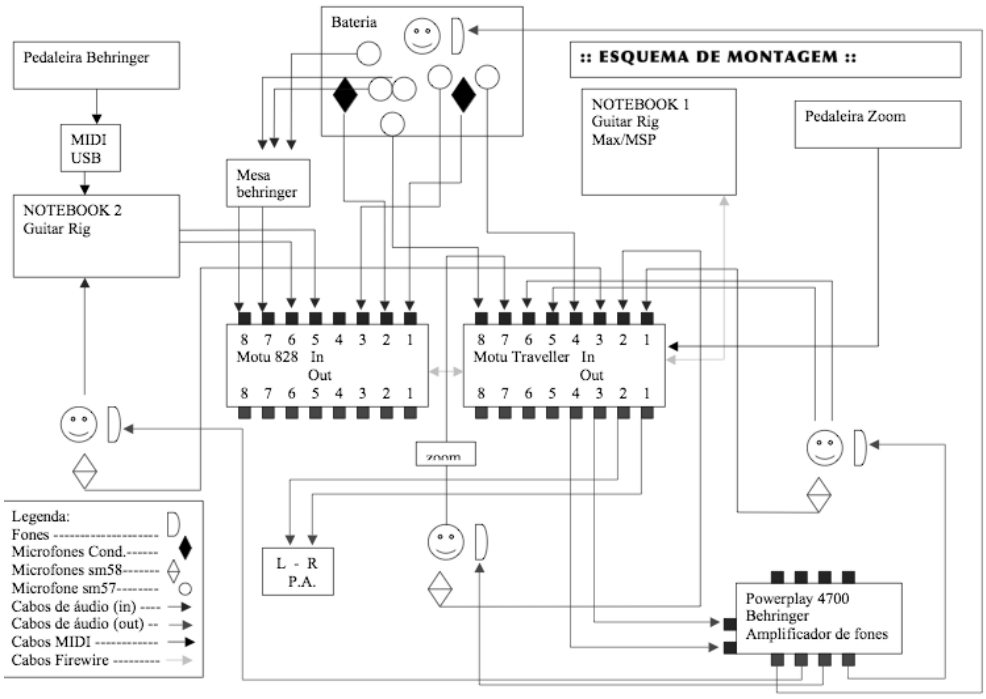



**Figura 1: Diagrama de conexões do ambiente Cyberock.**

A abordagem utilizada para a montagem do esquema acima partiu do ambiente de estúdio como paradigma referencial. Afinal, "diferentes gêneros musicais desenvolvem imagens acústicas específicas, derivadas de técnicas próprias de execução, microfonação, mixagem e pós-produção" [Freire 2004 p.54]. Em um ambiente de estúdio, tanto a acústica local como os processos de captação são mais controlados. O resultado é uma gravação com poucas distorções e vazamentos e, de alta qualidade sonora, fato que também proporciona uma marca acústica mais clara e definida. Em apresentações ao vivo, por outro lado, este controle dá lugar a uma administração de vazamentos, uma vez que a acústica do palco é mais imprevisível devido à presença de monitores de retorno e amplificadores. Isto aumenta a intensidade sonora no palco, fato que gera problemas com a captação feita pelos microfones, aumentando assim as possibilidades de realimentações, microfonias e a poluição da imagem sonora do palco.

Com o esquema proposto, os monitores de palco são eliminados e a função de monitoramento de retorno fica a cargo do amplificador de fones, em um esquema de monitoramento *in-ear*, através de fones de ouvido. Desta forma, as únicas fontes sonoras presentes no palco são as vozes e a bateria, uma vez que os amplificadores de guitarra e baixo foram tratados através do conceito de simulação.

## Simulação por modelagem digital

Guitarra e baixo elétrico são instrumentos de corpo sólido que necessitam de algo que exerça a função de caixa de ressonância, de radiador sonoro. Esta função é delegada aos amplificadores de guitarra e de baixo. Estes amplificadores, diferentemente dos utilizados em sistemas *hi-fi*, são construídos de forma que ressaltem certas características dos instrumentos, delineando e definindo uma noção de sonoridade de guitarra e de baixo elétrico. Eles são constituídos de circuitos elétricos (que amplificam o sinal proveniente dos captadores) e de caixas acústicas com alto-falantes. Estas, também diferentemente das caixas acústicas dos sistemas *hi-fi* (que são construídas visando uma resposta plana do espectro sonoro), são construídas para que se ressaltem certas freqüências, colaborando também para a construção de sonoridade destes instrumentos. O problema aqui é, mais uma vez, a questão dos vazamentos sonoros destes instrumentos na captação feita pelo microfones presentes no palco, uma vez que o som que sai desses amplificadores costuma ser de grande intensidade (sobretudo em bandas de rock). A solução utilizada para isso veio da tecnologia de simulação desses amplificadores através da modelagem digital.

"Modelagem é uma ferramenta poderosa. Com ela, pode-se analisar, desenhar e operar sistemas complexos. ... Um modelo é uma descrição lógica de como um sistema, processo ou componente se comporta" [Diamond 2002]. Ao invés de interagir com o sistema real, pode-se criar um modelo que corresponda a ele em certos aspectos e onde se possa fazer experiências e testá-lo. Este é o conceito de simulação aqui utilizado.

Para as guitarras, foi utilizado o programa *Guitar Rig* (figura 2), do fabricante *Native Instruments*. Este é um *software* que simula todo o aparato pós-guitarra envolvido na gravação da mesma. Ele simula uma grande quantidade de modelos de pedais, amplificadores, caixas acústicas e microfonações destas caixas. O controle de seus componentes se dá através de mensagens MIDI provenientes de controladores (aqui neste caso, das pedaleiras) ou de interações com outros programas, como o Max/MSP, uma vez que o *Guitar Rig* funciona também como um *plugin* VST.


**Figura 2: Tela do *Guitar Rig***

Para o baixo elétrico, a simulação da amplificação se dá pelo processador de efeitos Zoom 504. Esse processamento também acontece por modelagem digital, simulando amplificador e caixa acústica.

## Limitação de processamento

Em qualquer sistema digital de áudio, há que se encontrar um equilíbrio entre capacidade de processamento, latência e estabilidade, sobretudo em sistemas de processamento em tempo real. Por esta razão, as tarefas de processamentos mais pesados (baixo e guitarras) foram divididas entre o Zoom 504 (para o baixo) e os dois *notebooks* (um para cada guitarra, com um *Guitar Rig* independente em cada *notebook*). Isto permitiu que fosse possível construir um sistema de gerenciamento de ações e processamentos para todos os instrumentos conectados ao computador principal, o Macbook.

Uma possibilidade para contornar esse problema de limitação de processamento pode vir do objeto *OpenSoundControl*, uma implementação do protocolo de comunicação OSC para o Max/MSP, desenvolvida no CNMAT [Wright 2004]. Através dele, máquinas diferentes podem realizar processamentos em paralelo e trocar sinais de controle e mensagens de maneira bem eficiente e interativa. Desse modo, os controles e processos de interação podem ser divididos entre máquinas diversas. Este é um caminho que ainda estou investigando em minha pesquisa.

Assim, a concepção da programação é, em parte, uma mistura entre mesa de som e unidade de efeitos, onde há um gerenciamento de informações, de fluxos de mensagens MIDI, de sinais de áudio e do processamento destes sinais. A abordagem

utilizada partiu do macro para o micro, com a construção de um *patch* "mãe", que serviria de interface para controle do funcionamento geral do ambiente sonoro. Este *patch* é o que foi denominado Cyberock.

## *Patch* Cyberock

Esta programação foi feita com intuito de funcionar como uma interface entre os processos internos de gerenciamento de informações e que, ao mesmo tempo, servisse de alicerce para a exploração das possibilidades de interação e de processamento que dali poderiam surgir. Seu principal objetivo é dar conta das demandas sonoras básicas de uma banda de rock (como equalização, compressão, reverberação, *gate*, etc.) sendo aberta o suficiente para se permitir uma maleabilidade de ajustes devido à variação de repertório, separando as necessidades de acordo com cada música. Assim, depois de várias adaptações, o *patch* Cyberock (figura 3) se definiu da seguinte forma:

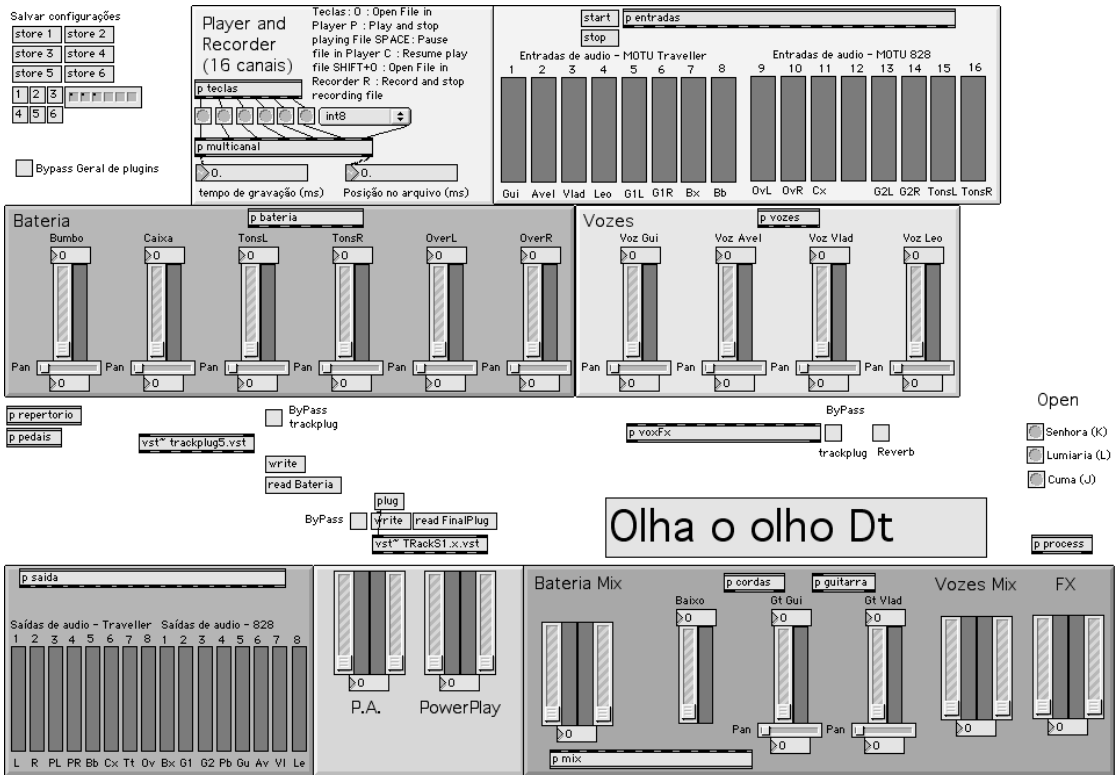



**Figura 3: Interface do *patch* Cyberock**

**1.** Uma seção de monitoramento das entradas de áudio nas interfaces de som, com ajustes de ganho (quando necessário). Esta seção é composta de medidores de sinal (*meters*) e de um *sub-patch* entradas (figura 4), que direciona os sinais de entrada através de objetos *send~* . Esta seção funciona também como um distribuidor de sinais, como um *patch bay*.

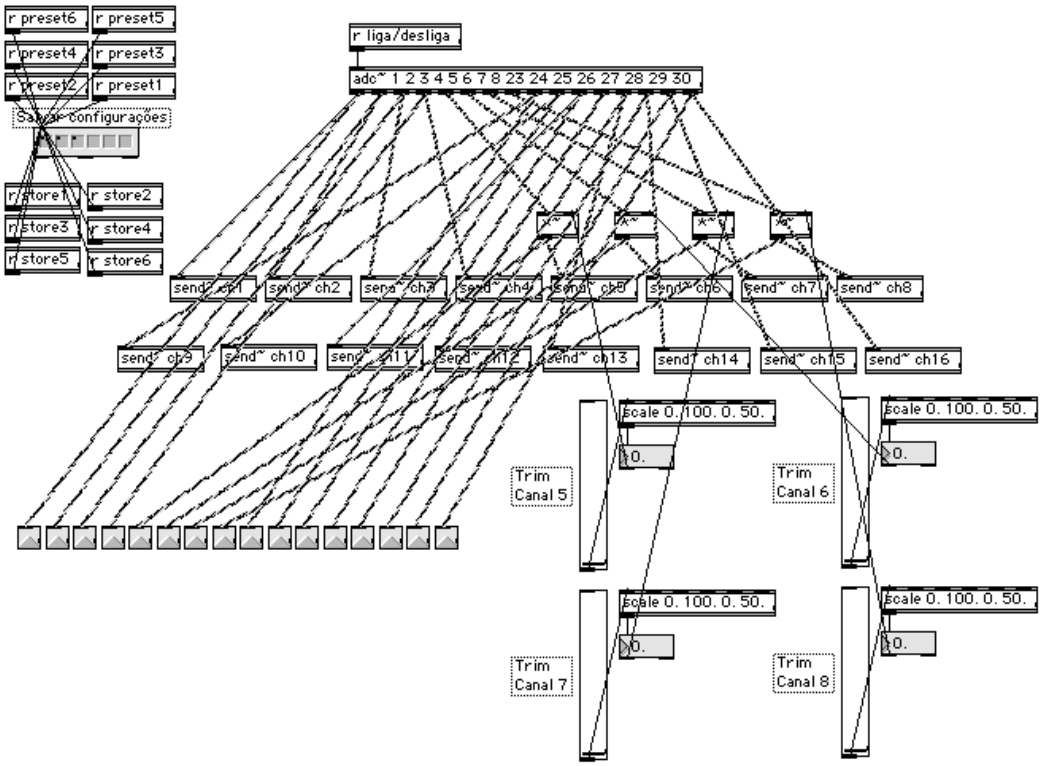



**Figura 4: *sub-patch* Entradas**

**2.** Uma seção de gravação e reprodução multipista de 16 canais, baseada no objeto *sfrecord~*. O arquivo aqui gravado captura todas as entradas de áudio do sistema, visando um posterior ajuste da música sem que haja a necessidade de uma execução da banda em tempo real. Isto poupa um tempo considerável, facilitando os ajustes dos trabalhos de pós-produção (mixagem e masterização), ajustes estes que retornarão para o contexto de apresentação ao vivo. Na prática, é um novo jeito de se "passar o som". Desse modo, com um posterior ajuste do som, de maneira mais refinada, em um ambiente de estúdio mais controlado e com mais calma e tendo os monitores de estúdio como referência para se alcançar um equilíbrio entre as partes da banda, a definição de uma marca sonora para a mesma fica mais fácil e definitiva.

**3.** Na parte central, encontram-se as primeiras seções de mixagem e processamento de sinais, dividida em seções de bateria e de vozes. A bateria é captada em seis canais, sendo os seguintes: bumbo, caixa-clara, *overs* (esquerdo e direito) e ton-tons (esquerdo e direito). Nos canais de ton-tons estão agrupados os dois ton-tons e o surdo, através de um mesa Behringer, anterior a entrada no sistema. Esta seção é composta de medidores de sinal, *faders*, *pans*, e um *sub-patch* Bateria. Neste *sub-patch* os sinais do bumbo, da caixa e dos tons são processados independentemente por um *plugins* VST. Estes *plugins* realizam o processamento de compressão, *gate*, equalização e *limiter* para cada sinal.

A seção de vozes é composta também por *faders*, medidores de sinal, *pans* e dois *sub-patchs* (vozes e vozfx). O *sub-patch* vozes é colocado antes dos *faders* para que se possa separar alguns sinais de vozes a serem processados individualmente através de objetos *gate~*, numa espécie de *insert*. O *sub-patch* vozfx é colocado depois dos *faders* e *pans* para processamento em conjunto das vozes, utilizando-se também um *plugin* VST (para compressão, *gate,* equalização e *limiter*) e um *plugin* VST de

reverberação. Além disso, neste *sub-patch* há ainda um controle dos parâmetros deste último *plugin* de forma interativa, feito através da pedaleira MIDI Zoom GFX-7.

**4.** Há ainda dois *sub-patches* nesta parte central que são responsáveis pela interação com o sistema a partir de um dos guitarristas, mais precisamente, pelo o que controla a pedaleira Zoom GFX-7. São os *sub-patches* repertório e pedais. No *sub-patch* pedais (figura 5), ocorre um mapeamento dos sinais MIDI enviados pela pedaleira e uma distribuição destes sinais através de objetos *send*. Os sinais enviados por ela são sinais de controle e de *program change*.

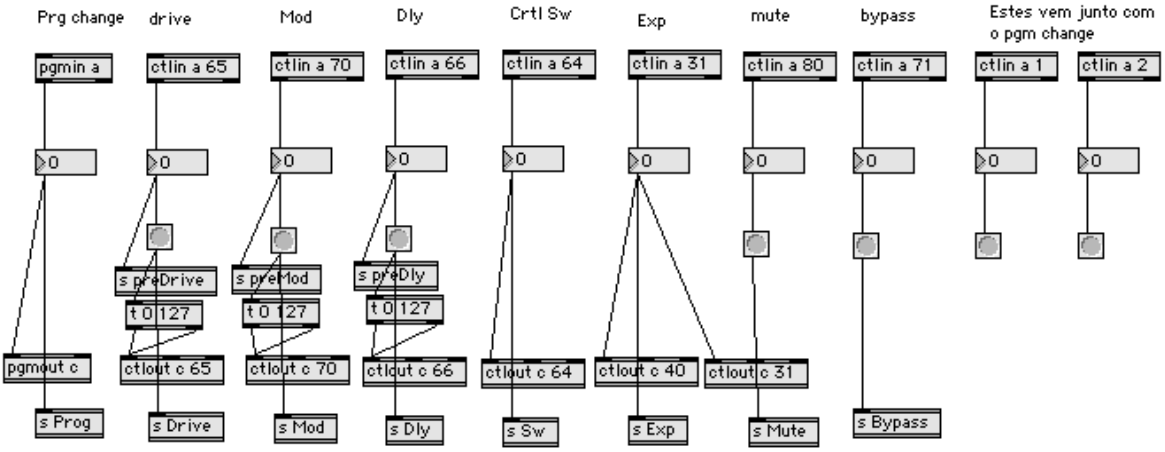
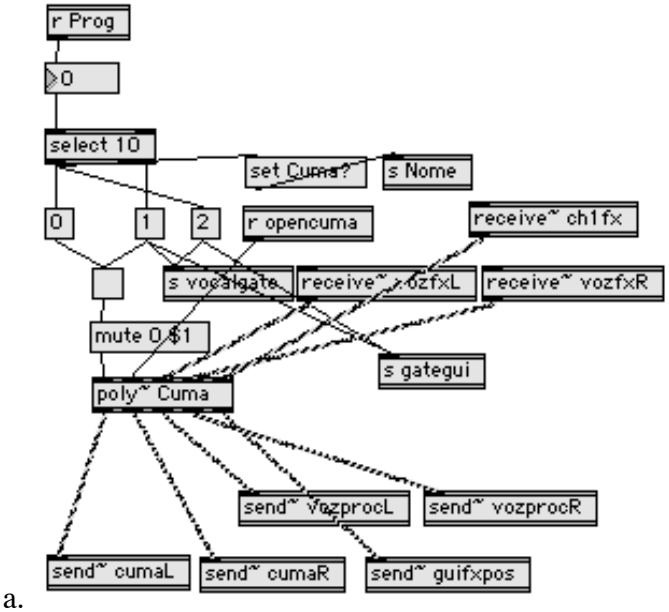
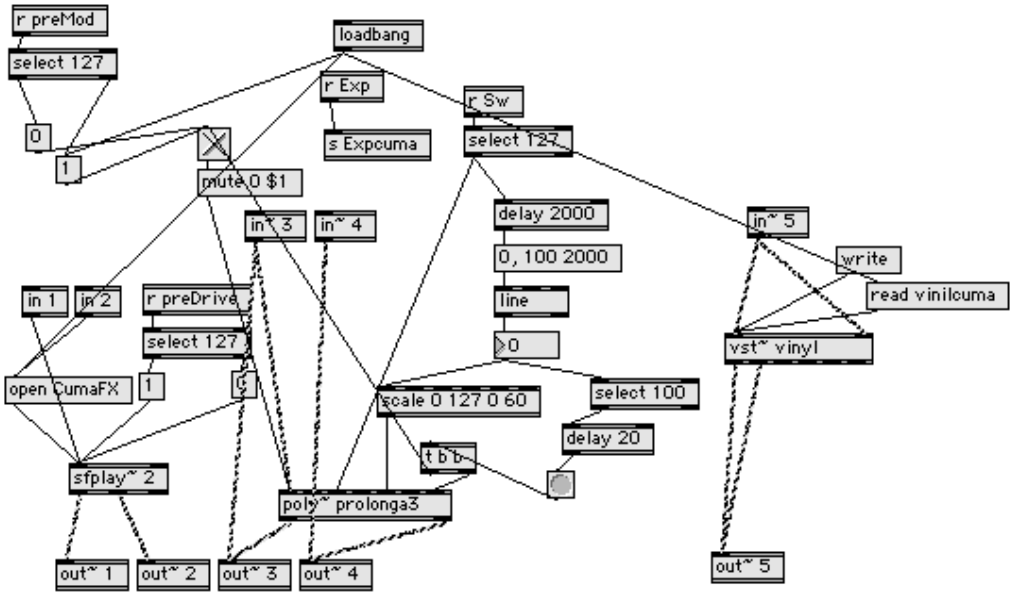



**Figura 5: *Sub-patch* pedais.**

O *sub-patch* repertório é composto de uma série de *sub-patches* relativos a cada música do repertório da banda. Em cada *sub-patch* destes existe uma programação relativa às demandas de interação necessárias em cada peça. Para citar um exemplo, a programação da música Cuma? (figura 6a) envolve o desvio de alguns sinais de áudio das vozes através de objetos *send~*, *receive~* e *gate~*, isolando o vocal principal e o coro. Para o coro, é aplicada uma mixagem de reverberação variante, sendo esta controlada pelo pedal de expressão da pedaleira, enquanto o vocal principal é desviado para um processamento pelo *plugin* VST *Vynil*, do fabricante *Izotope*. Este *plugin* simula as impurezas características de um disco de vinil (arranhados, sujeiras, etc.) no sinal de áudio. Além disso, houve uma adaptação do *patch* prolonga, do compositor Sérgio Freire, para uso nas vozes agrupadas. Este *patch* foi originalmente desenvolvido por ele para prolongar as notas de um cavaquinho através do acionamento de um pedal, em uma peça para cavaquinho e computador [Freire 2003]. Este *patch* foi adaptado para prolongar as vozes do coro por um determinado tempo e depois, para que este coral suba em glissando até sumir, em um efeito análogo ao que seria se, em uma gravação em fita magnética, fossemos acelerando a velocidade da rotação até o ponto de ruptura da fita. Outro evento que ocorre nesta mesma música é a utilização de um som pré-gravado de um foguete (de fogos de artifício) que é acionado também através da pedaleira. Todas estas programações ficam encapsuladas em um objeto *poly~* (figura 6b) que pode ser ligado e desligado conforme o repertório, poupando assim um pouco da capacidade de processamento do computador. O *sub-patch* repertório é também responsável pelo envio de uma mensagem com o nome da música para a qual o sistema está configurado, mensagem esta exibida em um visor na parte central do *patch* Cyberock.




a.




b.

**Figura 6: a.*Sub-patch* Cuma; b.Objeto *poly~* Cuma.**

**5.** Na seção inferior é onde se dá a mixagem e masterização do sinal de audio geral. Os canais de bateria e vozes são agrupados, o que facilita uma mixagem da banda por funções dos instrumentos. Então, assim podemos mixar o baixo, as guitarras (separadamente), as vozes, a bateria e efeitos ou partes pré-gravadas. Esta seção é composta também de *faders*, medidores de sinal e os três *sub-patches*: mix, cordas e guitarra. Neste último, é onde foi colocado o *plugin* VST *Guitar rig*. Os outros dois *sub-patches* ainda têm a função de distribuição de sinais, tanto para a saída geral do sistema como também para uma espécie de *insert,* de onde o sinal é retirado, processado em outro *sub-patch* e retornado a este módulo. Ao lado, há a seção de masterização, aonde o sinal geral mixado passa por um processamento que envolve uma compressão,

equalização e *limiter* através de um *plugin* VST próprio para masterização. O sinal de áudio masterizado é enviado para duas saídas: P.A. (*Public Address*) e para o amplificador de fones. A diferença entre as duas saídas é que na saída para os fones pode-se mandar ainda alguns elementos de referência, como um metrônomo (necessário em uma das músicas). Mixagens diferentes poderiam ainda serem enviadas aos fones de ouvido (individualmente para cada integrante), como acontece comumente nos monitores de palco em um contexto convencional. Mas a qualidade sonora alcançada com esta abordagem foi preferida por todos os integrantes. Ou seja, todos escutam o som como se estivesse escutando um CD, porém com a diferença de que a música aqui está sendo executada em tempo real. Muda-se completamente a referência sonora para a performance.

**6.** Por fim, há a seção de endereçamento de saídas. Esta seção é composta de medidores de sinal e um *sub-patch* saída. Este *sub-patch* apenas recolhe sinais diversos vindos de objetos *send~* e os direciona para as dezesseis saídas das interfaces de áudio. As saídas 1 e 2 são para P.A., as saídas 3 e 4 são para o monitoramento (retorno) através do amplificador de fones, e as saídas restantes são para elementos individuais da banda (bumbo, caixa, vozes separadas, guitarra, etc.). Isto é para o caso de ainda haver a necessidade de algum ajuste em separado a ser feito pelo técnico de som, no decorrer de uma apresentação.

## Considerações finais

Apesar da programação aqui apresentada ser bastante individualizada para o caso específico da banda Somba, a concepção de tal sistema pode ser adaptada a diversas situações de banda. As vantagens de um sistema deste tipo são muitas: elimina-se em grande parte o fator de imprevisibilidade da resposta acústica do ambiente, escutando-se sempre uma mesma sonoridade da banda, independente do local onde a apresentação seja realizada. A banda ensaia e toca embalada sempre pela mesma marca acústica, possibilitando o desenvolvimento de uma musicalidade ao mesmo tempo dependente e influenciada pelo o próprio sistema. Além disso, poupa-se um tempo considerável de passagem de som, que nada mais é do que uma adaptação da banda à acústica e ao equipamento do local da apresentação. A passagem de som fica reduzida ao tempo de montagem do equipamento e a ajustes do P.A.

Outro fator bastante interessante em termos de possibilidades para exploração de interatividade é a abertura que o ambiente Max/MSP dá para a interligação entre audio, *plugins* VST, MIDI, Rewire, rede (através do OSC - OpenSoundControl), etc. Esta característica merece uma pesquisa musical mais aprofundada sobre a utilização destas interligações como ferramentas composicionais. Na presente pesquisa apenas começou-se a trilhar e investigar alguns destes caminhos certo de que, com uma maior familiarização dos procedimentos, há muito que se explorar em termos de interatividade e de utilização dos resultados no contexto descrito neste artigo.

Dentre os fatores contrários a esta abordagem está o grande numero de cabos para as conexões necessárias, fator este que pode ser minimizado com equipamentos *wireless*. Essa solução, porém, ainda se encontra fora da realidade econômica da banda. Além do grande número de cabos, o fato de que todos eles convergem para um mesmo ponto - as interfaces de áudio - contribui para a sensação de poluição visual no palco.

A montagem, concepção e programação deste sistema leva em conta a capacidade de processamento dos computadores disponíveis e a latência, que ficou em um nível satisfatório para a banda. Porém, com computadores ainda mais potentes, maiores possibilidades de interação poderão surgir, incluindo a utilização de vídeo. Além disso, a banda agora se encontra em uma fase de exploração e investigação da musicalidade desse sistema. Alguns resultados musicais desta pesquisa podem ser ouvidos no próximo CD Cuma? [6].

## Notas de referência

1. http://www.somba.com.br

2. http://www.calix.art.br

3. http://www.bandacartoon.com.br

4. A Outra Cidade é um CD resultante de um projeto conjunto de três compositores: Kristoff Silva, Pablo Castro e Makely Ka, lançado em novembro de 2003.

5. Clube da Esquina dos Aflitos é o primeiro CD comercial da banda Somba, lançado em agosto de 2003.

6. O CD Cuma? da banda Somba foi lançado em julho de 2007. Mais resultados desta pesquisa também poderão ser conferidos na série de shows que está por vir em seqüência a este lançamento.

## Bibliografia

Chadabe, J. Electric Sound: The Past and Promise of Electronic Music, Prentice Hall, 1997.

Dean, R. Hyperimprovisation: Computer-Interactive Sound Improvisation, A-R Editions Inc., Middleton/Wisconsin, 2003.

Diamond, B. (2002), "Simulation Overview", In: http://www.imaginethatinc.com/sols_simoverview.html, acessado em 28/05/2007.

Freire, S. Alto-, alter-, auto-falantes: concertos eletroacústicos e o ao vivo musical, Tese de Doutorado em Comunicação e Semiótica, PUC/SP, 2004.

Freire, S. (2003),"cvq: entre o meta-instrumento e a pseudo-obra", In: IX Simpósio Brasileiro de Computação e Música, 2003, Campinas. Anais do XXIII Congresso da Sociedade Brasileira de Computação. Campinas : Editora da Unicamp, 2003. v. 9. p. 271-276

Iazetta, F. Sons de Silício: corpos e máquinas fazendo música, Tese de Doutorado em Comunicação e Semiótica, PUC/SP, 1996.

Rowe, R. Interactive Music Systems: Machine Listening and Composing. Cambridge, MA: The MIT Press, 1993.

Wright, M. (2004), OpenSound Control Home Page In: http://www.cnmat.berkeley.edu/OpenSoundControl/, acessado em 01/06/2007.

Zicarelli D. Taylor G., Clayton J.K., Dudas R., Nevile B. and jhno, Max 4.6 Reference Manual, documento PDF distribuído por Cycling'74, 2006.

# Composição Assistida por Computador na Obra Klavibm II de Duprat e Cozzella

**Denise Garcia[1], Jônatas Manzolli[2]**

[1]Departamento de Música – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6159 - 13083-970– Campinas – SP – Brasil

[2]Núcleo Interdisciplinar de Comunicação Sonora (NICS) – Universidade Estadual de Campinas (UNICAMP) Caixa Postal 6166 - 13092-970

d_garcia@iar.unicamp.br, jonatas@nics.unicamp.br

***Abstract.*** *After a musical historical research on works of Rogério Duprat e Damiano Cozzella, this paper describes Klavibm II (1963). Given the historical period it could be the first composition using algorithmic processes in the Brazilian contemporary music history. Here we present an analysis of the compositional process in the context of computer music, since it is a historical reference and also a work that, probably, used stochastic processes and computer processing in an ancient IBM-1620 mainframe. First of all, this article describes the process of historical searching of this work, including its musical structures and, finally, it presents a hypothesis of the algorithmic process developed by Duprat and Cozzella, probably.*

***Resumo.*** *Trata-se de um artigo que, após um levantamento histórico musical da obra de Rogério Duprat e Damiano Cozzella, traz a toma a obra Klavibm II (1963) que, situada no tempo, parece-nos a primeira composição que utilizou processos algorítmicos na produção contemporânea brasileira. Apresentamos uma análise do processo composicional no contexto de computação musical, pois trata-se de um achado histórico e também de uma obra que, provavelmente, utilizou processos estocásticos e processamento computacional num antigo mainframe IBM-1620. Primeiramente, o artigo faz uma descrição do processo de levantamento histórico da obra, das suas estruturas musicais e, finalmente, apresenta uma hipótese do processo algorítmico, que possivelmente, thenha sido utilizado por Duprat e Cozzella.*

## 1. Introdução

O trabalho de busca de fontes etnográficas e o estudo do processo histórico vinculado ao desenvolvimento da música contemporânea brasileira é um trabalho de folego. No contexto da produção eletroacústica, torna-se um desafio ainda maior. Este artigo descreve uma obra específica encontrada, justamente, durante a pesquisa que foca a produção do Grupo Música Nova e seu pioneirismo na utilização de recursos tecnológicos. Pareceu-nos oportuno apresentar a análise de uma obra que poderia ser a primeira composição brasileira a utilizar o computador no processo de organização musical.

A história da composição musical assistida por computador, no que tange ao seu desenvolvimento no século passado, vem do trabalho pioneiro de diversos compositores dentre os quais poderíamos citar a Illiac Suite composta por Hiller (1959), o trabalho de formalização e serialização de Babbitt (1965), a utilização de processos estocásticos desenvolvida por Xenakis (1971), as gramáticas generativas utilizando-se de Cadeias de Markov desenvolvida por Jones (1981), os sistemas composicionais PR1 e PR2 desenvolvidos por Koenig (1978). Certamente, esta lista não é extensiva, mas representa o foco e as tentências iniciais que pulverizaram o universo composicional a partir da década de 60.

Parece-nos importante mostrar que, de alguma forma, no início da década de 60 e acompanhado o mesmo desenvolvimento, já existiram processos composicionais assistidos por computador no Brasil. Se no contexto histórico da computação musical, o trabalho dos compositores mencionados anteriormente, foi pioneiro e inovador, é importante também apresentar uma obra que, justamente, na mesma época foi composta por Duprat e Cozzella usando os recursos tecnológicos da época.

Este artigo tem o objetivo de descrever a estrutura da obra para piano solo *Klavibm II*, levantar a hipótese de que o processo algorítmico utilizado foi vinculado a três estratégias composicionais: a) a utilização do teclado do piano como espaço amostral e a geração de notas do piano sem repetição, b) a serialização das durações em 17 valores e as dinâmicas em 13. Levanta-se também a hipótese que a participação de Duprat e Cozzella do Festival de Darmstad tenha favorecido a utilização de um processo composicional derivado do conceito de Serialismo Integral. Neste caso, as alturas foram consideradas como campo variacional e as durações e dinâmicas foram serializadas de acordo com uma derivação própria dos autores do processo serial.

Nas próximas seções apresentamos uma breve descrição do contexto histórico no qual a obra foi composta, fazemos um levantamento da estrutura da obra, segundo as hipóteses apresentadas no parágrafo anterior e, finalmente, descrevemos um algorítmo de complexidade mínimo que, segundo a nossa análise, poderia ser capaz de gerar uma organização musical com as características de *Klavibm II*.

## 2. Histórico da Pesquisa

Rogério Duprat e Damiano Cozzella foram ativos compositores na área da música erudita contemporânea no início da década de sessenta. Nesse período foram atualizar-se nos Festivais de Darmstadt (*Neue Musik Tage* de 1962), assim como em outros centros de pesquisa musical na Europa, [Gaúna, 2001 p. 51] onde tiveram contato com as produções mais recentes da nova música e principalmente vivenciaram o choque que Darmstadt sofreu com a participação de John Cage no festival, que abalou as certezas do serialismo integral promulgado pelos compositores europeus.

Em 1963, foi o próprio Duprat que redigiu o famoso "Manifesto" [Cozzella, 1963], assinado por ele, Cozzella, Gilberto Mendes, Willy Corrêa de Oliveira, Júlio Medaglia e Alex Pascoal, publicado pela Revista *Invenção* e que ficou conhecido como

"Manifesto Música Nova" e indiretamente atribuiu ao grupo de compositores o nome de "Grupo Música Nova" (Neves, 1981)[1].

A composição de *Klavibm II* data exatamente do mesmo período (jan. 1963). No entanto não sabemos se essa obra permaneceu inédita, pois não foram encontrados registros de sua difusão em concerto. Tampouco foram encontradas referências dela em bibliografia mais corrente sobre o tema. Dos autores que documentaram a história da música brasileira do século XX, encontramos citações sobre a realização pioneira de composição com assistência de computador da parte de Duprat e Cozzella apenas no livro de Gérard Béhague (1979), na dissertação de mestrado de Igor Linz Maués (1989) e em artigo de Vânia Dantas Leite (2000). Porém esses autores trazem como título de obra resultante desse trabalho com computador uma "Música Experimental", com data também de 1963 (Maués, p.14, Béhague, p. 346). Apenas no livro de Regiane Gaúna consta o nome *Klavibm II* [Gaúna, 2001, p.183], comprovado pela entrega por parte dessa pesquisadora do próprio manuscrito da partitura[2].

Graças então à generosidade de Gaúna em nos ceder cópia da partitura manuscrita é que podemos trazer a público maiores esclarecimentos sobre essa atividade pioneira na área da composição assistida por computador no Brasil.

### 2.1 A obra: *Klavibm II*

Gaúna registra que a obra *Klavibm* II resulta da experiência de Duprat e Cozzella adquirida na França e na Alemanha, associada ao seu contato com a área de programação de dados [Gaúna, p. 56]. Por sua vez, Maués comenta que para essa obra pioneira, os compositores aplicaram cálculos efetuados por computador na realização da montagem da obra, similares às experiências também realizadas com computador pelos poetas concretistas paulistas [Maués, p.15]. Além desses dois rápidos testemunhos, não há qualquer registro de procedimentos realizados para a composição desta obra. Nosso trabalho segue rastros e se desenvolve portanto a partir apenas da própria partitura.

De início constatamos que, não existindo nesse período o conceito de 'composição assistida por computador', a imagem que os compositores tinham do trabalho era a de que a música tenha sido 'composta' pelo próprio equipamento, claramente enunciado na capa da partitura, que tem os seguintes dizeres: "Klavibm II (no record) - composta por uma computadora IBM-1620 - São Paulo, 14-1-1963 - Programação: Damiano Cozzella e Rogério Duprat".

As perguntas que motivaram este estudo e que surgiram espontaneamente no confronto com a partitura foram duas: como teria sido feita a programação? É possível deduzi-la apenas pela análise da partitura? Para uma possível dedução de sua programação, partimos então para uma análise quantitativa, descrita a seguir na seção 3.

---

[1] não podemos deixar de mencionar a importância de dois regentes para a divulgação da música desses compositores, os maestros Klaus Dieter Wolf e Olivier Toni, que regeram nesse período a maior parte de suas obras em concertos na cidade de São Paulo.

[2] Deve-se ressaltar que o nome "Música Experimental" inexiste no catálogo de obras de Duprat publicado nesse livro e que, segundo testemunho pessoal da autora, esse catálogo foi realizado pelo próprio Duprat em caderno manuscrito, que se encontra no acervo pessoal da pesquisadora.

Todavia, o primeiro aspecto evidente na partitura é que as notas não constroem aglomerados de qualquer espécie (melodias ou acordes), sendo a música uma sequência de notas descritas por três parâmetros: **altura, duração e dinâmica**. A relação entre eles aparentemente é autônoma, sequencial e as alturas estão distribuídas por todo o campo de tessitura possível no piano.

A segunda observação simples da partitura mostrava que não se tratava de obra dodecafônica, uma vez que logo no início notas se repetem. Foi então levantada a hipótese de tratar-se de distribuição serial de todas as notas do teclado do piano. De fato a obra é constituída de 88 notas, exatamente o número de notas do piano (as notas neste caso, seriam valores absolutos e não relativos).

Para confirmar essa hipótese foi realizada uma tabela com todas as notas do piano para comprovar as ocorrências na partitura, apresentada na figura 1.

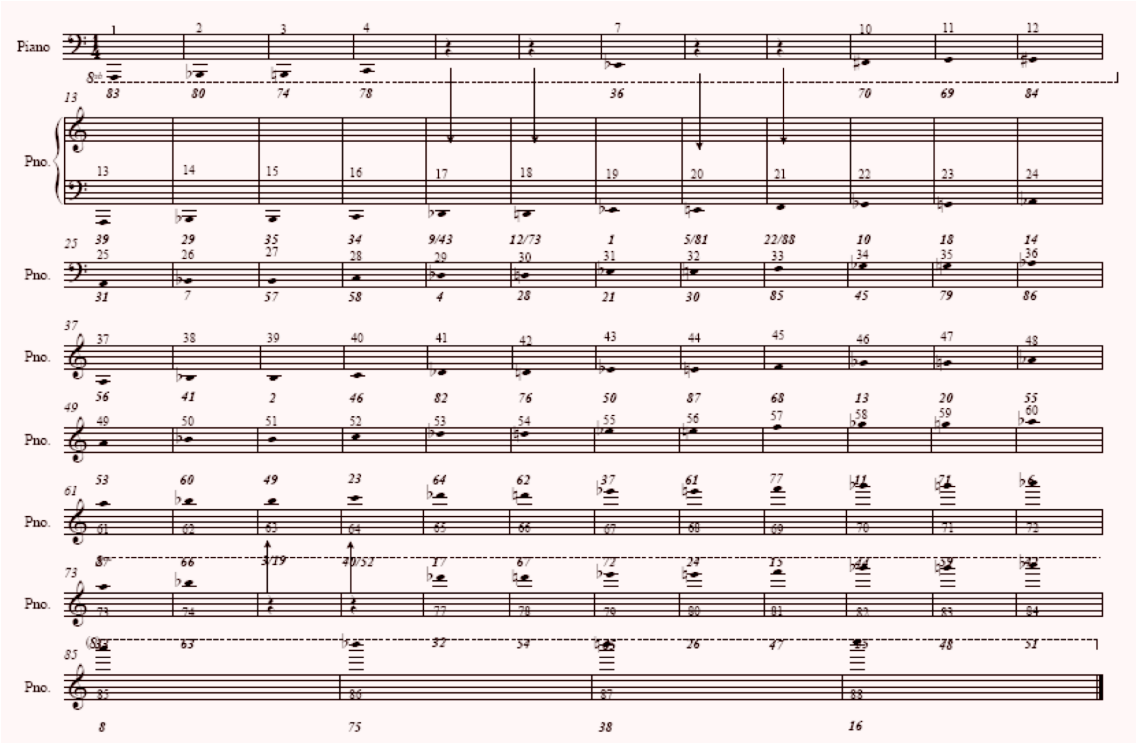



**Figura 1. Apresenta a distribuição das notas que aparecem na obra, na ordem sequencial do instrumento. Na linha superior de números, se configura a ordem de acordo com a sequência do teclado do piano. Os índices numéricos na linha inferior indicam a posição da mesma nota na obra. As setas e as pausas indicam as notas que faltam na respectiva oitava e foram repetidas nas oitavas superiores ou inferiores.**

## 3. Análise de Klavibm II

Podemos ver na figura 1 que quase todas as notas do piano estão presentes na obra, com excessão de 6 indicadas nas pausas e que, coincidentemente, as seis repetições de notas que completam o total de 88 notas da obra são justamente oitava acima ou abaixo das faltantes. Neste sentido levantou-se duas hipótese: foi um erro de notação na partitura, ou uma excessão voluntária, pois trata-se de um composição assistida por computador e os autores tiveram um certo grau de liberdade na transcrição da obra.

Seguindo esse raciocínio - se por um lado não se pode saber se foi erro ou decisão voluntária a ausência dessas 6 notas e a repetição das oitavas - há um outro aspecto que evidencia certamente um lapso de notação de clave no manuscrito: trata-se dos compassos 21 e 22. Na partitura manuscrita, se as duas notas continuassem a ser interpretadas sob a clave de fá assinalada no compasso 19, quebraria-se completamente o raciocínio da hipótese da distribuição estocástica por todas as notas do piano. Além disso, a própria notação não deixa margem de dúvidas: em nenhum outro ponto da partitura os compositores utilizaram notas brancas do piano como notas alteradas (o mi# seria então notado como fá) e tampouco teria razão a linha indicativa de 8a acima para tal nota (seria correspondente ao fá4 e portanto sem razão de ser notada em clave de fá, sequer por uma questão de leitura no piano). Portanto, haveria neste caso um lapso de notação de clave de sol, as notas correspondentes seriam do#6 e la5, que ocorrem apenas neste trecho da obra (vide partitura na figura 1).

Em seguida elaborou-se uma segunda tabela da ordem das notas na sequência da música para observar se haveria uma lógica de distribuição possível. Pudemos observar o seguinte: 88 notas distribuidas utilizando-se a distribuição estocástica uniforme, na qual se observa a estratégia da não repetição (com as 6 excessões que podem ser erros de notação na partitura), com o fato de as notas da primeira oitava do piano estarem presentes mais para o final da peça (acaso?).

Junto com os elementos levantados na análise das estruturas de *Klavibm II* e sob a ótica do uso de três parâmetros musicais: *alturas, durações e dinâmicas*, apresentamos também os gráficos relativos a cada tabela de análise. O objetivo aqui é mostrar a forma geral de cada distribuição. Não é feito nenhum processo de inferência estatística o que foge ao escopo deste artigo.

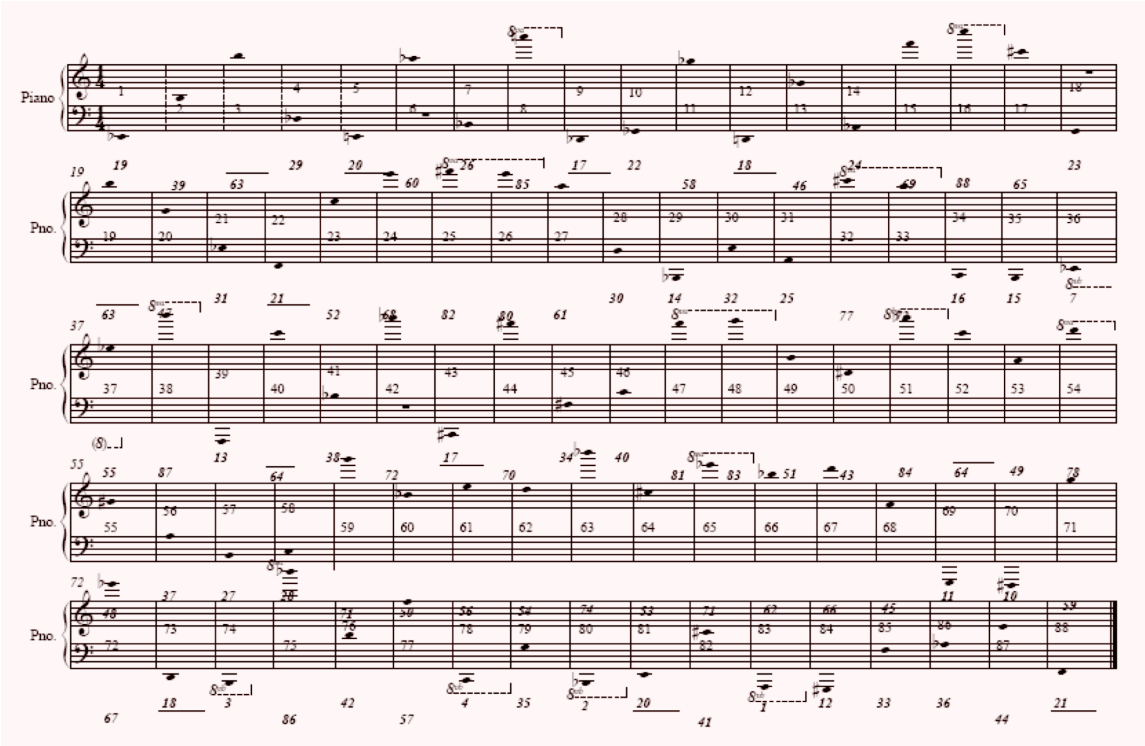



**Figura 2: sequência das notas na ordem da partitura de Klavibm II. Numeração superior denota a sequência da ordem da partitura. Numeração inferior denota a numeração dessas notas na ordem da sequência do instrumento do grave ao agudo.**

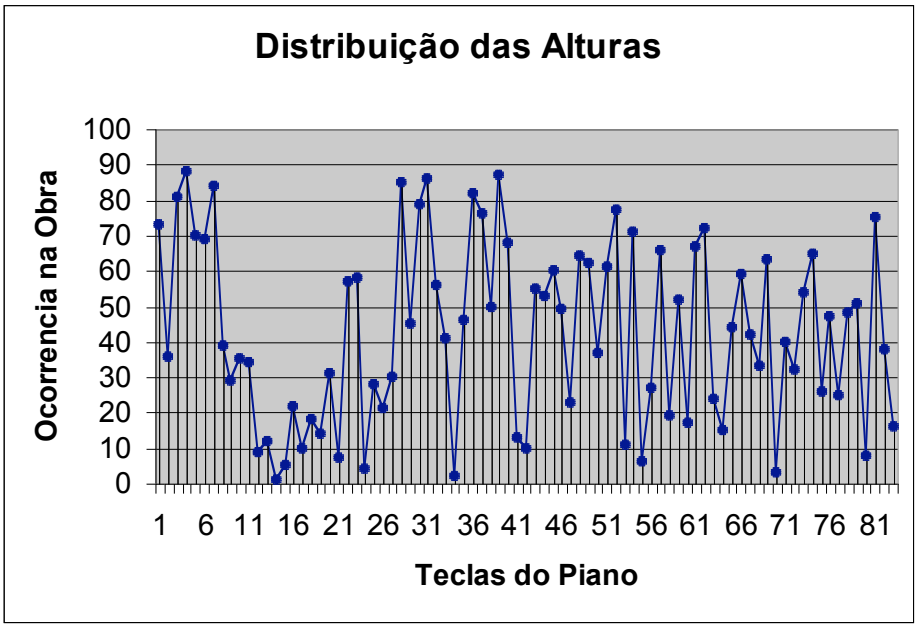



**Figura 3: Distribuição das Alturas de acordo com o seu aparecimento em Klavibm II. Exceto por 6 notas, como já foi comentado no parágrafo anterior, não há repetição. As notas repetidas aparecem no gráfico na suposta oitava original.**

### 3.1 Distribuição das Durações

Apresenta-se a seguir uma tabela de durações, na sequência das notas em número de colcheias (a colcheia como valor de unidade):

**Tabela 1: durações por número de colcheias na sequência da música.**

| Notas (ordem) | Duração | Notas (ordem) | Duração | Notas (ordem) | Duração | Notas (ordem) | Duração |
|---|---|---|---|---|---|---|---|
| 1 | 10 | 23 | 5 | 45 | 8 | 67 | 7 |
| 2 | 9 | 24 | 10 | 46 | 15 | 68 | 2 |
| 3 | 7 | 25 | 6 | 47 | 7 | 69 | 4 |
| 4 | 4 | 26 | 4 | 48 | 6 | 70 | 6 |
| 5 | 4 | 27 | 7 | 49 | 2 | 71 | 3 |
| 6 | 9 | 28 | 10 | 50 | 14 | 72 | 1 |
| 7 | 7 | 29 | 12 | 51 | 5 | 73 | 9 |
| 8 | 4 | 30 | 10 | 52 | 4 | 74 | 12 |
| 9 | 8 | 31 | 13 | 53 | 17 | 75 | 10 |
| 10 | 1 | 32 | 9 | 54 | 12 | 76 | 10 |
| 11 | 9 | 33 | 5 | 55 | 10 | 77 | 7 |
| 12 | 6 | 34 | 6 | 56 | 4 | 78 | 8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 13 | 3 | 35 | 2 | 57 | 1 | 79 | 9 |
| 14 | 6 | 36 | 8 | 58 | 11 | 80 | 9 |
| 15 | 6 | 37 | 8 | 59 | 5 | 81 | 8 |
| 16 | 2 | 38 | 3 | 60 | 3 | 82 | 5 |
| 17 | 7 | 39 | 15 | 61 | 5 | 83 | 9 |
| 18 | 2 | 40 | 7 | 62 | 1 | 84 | 10 |
| 19 | 11 | 41 | 6 | 63 | 1 | 85 | 8 |
| 20 | 1 | 42 | 7 | 64 | 2 | 86 | 11 |
| 21 | 1 | 43 | 8 | 65 | 3 | 87 | 13 |
| 22 | 2 | 44 | 5 | 66 | 2 | 88 | 8 |

Contando com a colcheia como unidade mínima, vemos na obra a ocorrência de durações de 1 a 17 colcheias (sendo que não há nenhuma duração com 16 colcheias). Essas 16 durações são distribuídas nas 88 ocorrências sonoras (notas) sem um padrão determinado de repetições (há diversas repetições de durações sequencialmente) com uma média de ocorrência uniforme as durações de 1 a 10 colcheias e as mais longas com um número menor de ocorrências, como se pode observar na tabela abaixo:

**Tabela 2: Quantidade de figuras de durações em ocorrência na obra, frequência de durações acumulada**

| Durações | Notas em que ocorrem (ordem na partitura) | Número de ocorrências |
|---|---|---|
| **1 colcheia** | **10 - 20 - 21 - 57 - 62 - 63 - 72** | **7** |
| **2 colcheias** | **16 - 18 - 22 - 35 - 49 - 64 - 66 - 68** | **8** |
| **3 colcheias** | **13 - 38 - 60 - 65 - 71** | **5** |
| **4 colcheias** | **4 - 5 - 8 - 26 - 52 - 56 - 69** | **7** |
| **5 colcheias** | **23 - 33 - 44 - 51 - 59 - 61 - 82** | **7** |
| **6 colcheias** | **12 - 14 - 15 - 25 - 34 - 41 - 48 - 70** | **8** |
| **7 colcheias** | **3 - 7 - 17 - 27 - 40 - 42 - 47 - 67 - 77** | **9** |
| **8 colcheias** | **9 - 36 - 37 - 43 - 45 - 78 - 81 - 85 - 88** | **9** |
| **9 colcheias** | **2 - 6 - 11 - 32 - 73 - 79 - 80 - 83** | **8** |
| **10 colcheias** | **1 - 24 - 28 - 30 - 55 - 75 - 76 - 84** | **8** |
| **11 colcheias** | **19 - 58 - 86** | **3** |
| **12 colcheias** | **29 - 54 - 74** | **3** |
| **13 colcheias** | **31 - 87** | **2** |
| **14 colcheias** | **50** | **1** |

| | | |
|---|---|---|
| **15 colcheias** | **39 - 46** | **2** |
| **16 colcheias** | **-** | **-** |
| **17 colcheias** | **53** | **1** |

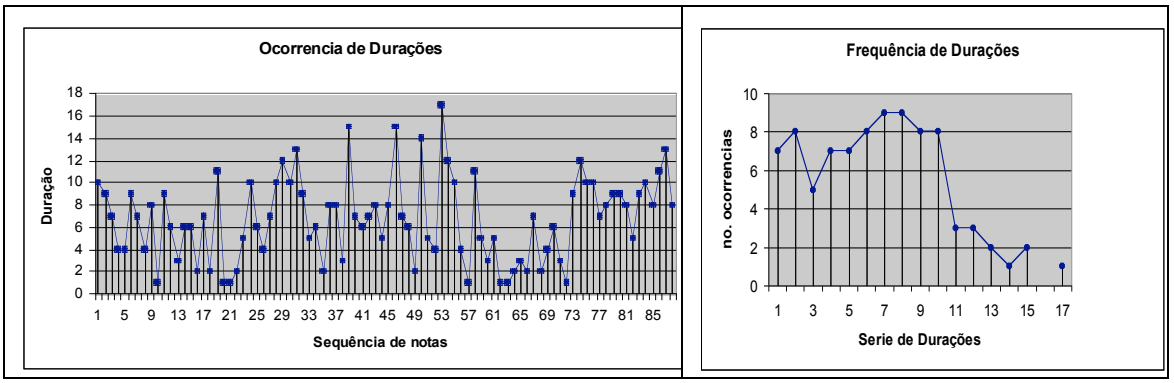



**Figura 4: Esquerda - Distribuição de Durações de acordo com o seu aparecimento em Klavibm II descrito na Tabela 1. Direita - Frequência Acumulada das Durações de acordo com o seu aparecimento em Klavibm II descrito na Tabela 2.**

## 3.2 Distribuição das Dinâmicas

Cada nota da partitura, salvo 4 excessões, possui uma dinâmica específica (bem ao gosto do pensamento derivado do serialismo integral). Desta forma, querendo observar, tanto uma possível gradação dinâmica, quanto algum padrão de suas ocorrências, há duas tabelas de dinâmicas, correlatas às tabelas de duração (tabela de dinâmica a cada nota e tabela de gradação das dinâmicas com as ocorrências.

**Tabela 3: dinâmicas na sequência da música**

| Notas (ordem) | Dinâmica | Notas (ordem) | Dinâmica | Notas (ordem) | Dinâmica | Notas (ordem) | Dinâmica |
|---|---|---|---|---|---|---|---|
| 1 | *sfff* | 23 | *pppp* | 45 | *p* | 67 | *fff* |
| 2 | *mf* | 24 | *?(pppp?)* | 46 | *pp* | 68 | *ff* |
| 3 | *ff* | 25 | *f* | 47 | *mf* | 69 | *f* |
| 4 | *pppp* | 26 | *pp* | 48 | *ff* | 70 | *ppp* |
| 5 | *sfff* | 27 | *p* | 49 | *sfff* | 71 | *mf* |
| 6 | *pp* | 28 | *fff* | 50 | *pppp* | 72 | *p* |
| 7 | *pppp* | 29 | *pp* | 51 | *pp* | 73 | *ff* |
| 8 | *fff* | 30 | *pppp* | 52 | *p* | 74 | *ppp* |
| 9 | *mp* | 31 | *sfff* | 53 | *pppp* | 75 | *mf* |
| 10 | *pppp* | 32 | *p* | 54 | *ppp* | 76 | *ff* |
| 11 | *f* | 33 | *p* | 55 | *p* | 77 | *mf* |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 | *mp* | 34 | *pppp* | 56 | *ff* | 78 | *?(mf?)* |
| 13 | *f* | 35 | *?(pppp?)* | 57 | *fff* | 79 | *pppp* |
| 14 | *pppp* | 36 | *ff* | 58 | *p* | 80 | *?(pppp?)* |
| 15 | *ff* | 37 | *pppp* | 59 | *ff* | 81 | *fff* |
| 16 | *fff* | 38 | *pp* | 60 | *sfff* | 82 | *ff* |
| 17 | *p* | 39 | *mp* | 61 | *ppp* | 83 | *fff* |
| 18 | *pp* | 40 | *sfff* | 62 | *ff* | 84 | *ff* |
| 19 | *pp* | 41 | *mp* | 63 | *f* | 85 | *fff* |
| 20 | *pppp* | 42 | *f* | 64 | *sfff* | 86 | *p* |
| 21 | *ff* | 43 | *p* | 65 | *ppp* | 87 | *ppp* |
| | *f* | 44 | *sfff* | 66 | *p* | 88 | *f* |

A obra está com uma gradação dinâmica em 10 graus, com repetições e ocorrências desiguais, porém sem que se note uma estratégia para elas. Faz-se por último uma pequena tabela de ocorrências dinâmicas em ordem crescente de ocorrências:

**Tabela 4: quantidade de dinâmicas por ocorrência, frequência de dinâmicas acumulada**

| Dinâmicas | Notas em que ocorrem (ordem na partitura) | Número de ocorrências |
|---|---|---|
| **pppp** | **4 - 7 - 10 - 14 - 20 - 23 - (24?) - 30 - 34 - (35?) - 37 - 50 - 53 - 79 - (80?)** | **12 ou 15** |
| **ppp** | **54 -61 - 65 - 70 - 74 - 87** | **6** |
| **pp** | **6 - 18 - 19 - 26 - 29 - 38 - 46 - 51** | **8** |
| **p** | **17 - 27 - 32 - 33 - 43 - 45 - 52 - 55 - 58 - 66 - 72 - 86** | **12** |
| **mp** | **9 - 12 - 39 - 41** | **4** |
| **mf** | **2 - 47 - 71 - 75 - 77 - (78?)** | **5 ou 6** |
| **f** | **11 - 13 - 22 - 25 - 42 - 63 - 69 - 88** | **8** |
| **ff** | **13 - 15 -21 - 36 - 48 - 56 - 59 - 62 - 68 - 73 - 76 - 82 - 84** | **13** |
| **fff** | **8 - 16 - 28 - 57 - 67 - 81 - 83 - 85** | **8** |
| **sfff** | **1 - 5 - 31 - 40 - 44 - 49 - 60 - 64** | **8** |

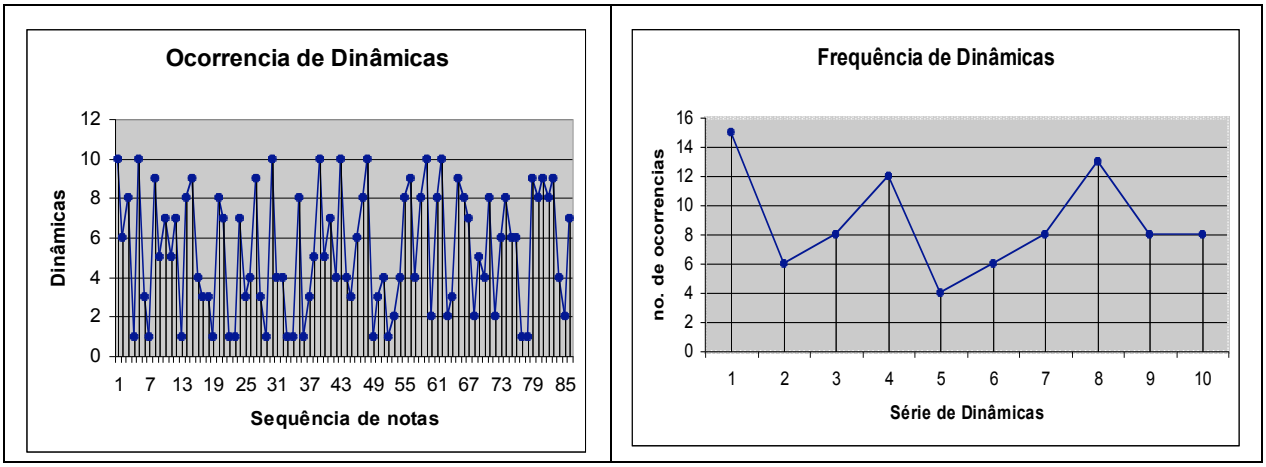



**Figura 5: Esquerda - Distribuição de Dinâmicas de acordo com o seu aparecimento em Klavibm II descrito na Tabela 5. Direita - Frequência Acumulada das Dinâmicas de acordo com o seu aparecimento em Klavibm II descrito na Tabela 6.**

## 4. Análise do Processo Algorítmico

Tecemos aqui algumas considerações sobre a obra *Klavibm II* partindo da nossa hipótese que os compositores Duprat e Cozzella utlizaram-se de um processo de composição assistida por computador. Pode-se, então, cogitar que os compositores tenham se utilizado de um algorítmo de carácter determinista ou aleatório/estocástico.

Por algorítmo determinista entendemos que os passos do processo composicional foram executados pelo computador através de procedimentos que levariam a uma única solução para a composição da obra. Ou seja, cada passo do processamento computacional teria sido realizado sem a utilização de nenhum gerador de números aleatórios e a composição foi organizada de forma unívoca. Se assumirmos a hipótese determinista, ao desvendarmos o processo algorítmo e ao implementá-lo novamente, deveríamos obter como resultado a mesma composição.

Todavia, pareceu-nos mais coerente, dada estrutura da obra e o contexto no qual a obra foi realizada que no processo algorítmico tenham sido utilizadas distribuições estocásticas. Como já destacamos, o trabalho pioneiro de Duprat e Cozzella ocorreu próximo de atividades como as do Experimental Music Studio (EMS) na Universidade de Illinois no campus Urbana-Champaign quando da composição da *ILLIAC Suite* em 1957 num computador IBM 7090, como descrito em Hiller (1959). Se olharmos para o momento histórico onde o trabalho de compositores como Xenakis (1971) e Koenign (1978) volta-se à utlização de distribuições de probabilidade e máscaras de tendências.

Um segundo ponto de reflexão é sobre a visão estruturalista que, possivelmente, Duprat e Cozzella utilizaram. Neste caso, coube-nos a hipótese que tenham utilizado um processo de serialização de *alturas, durações e dinâmicas*. Sob esta ótica os dois compositores utilizaram um computador IBM para *compor a obra* no sentido de gerar sequências destes três parâmetros musicais utilizando-se de distribuições de probabilidade. Sob este ponto de vista, as técnicas de serialismo integral desenvolvidas por Messiaen entre outros, poderiam estar relacionadas com a forma de organizar o material musical que serviu de dados de entrada no computador. Supomos que a distribuição de probabilidade utilizada no algorítmo composicional foi a *Uniforme*.

Todavia, ressaltamos que seria necessário fazer uma análise estatística mais detalhada para extrapolar a distribuição utilizada na obra.

Todavia, parece-nos que Duprat e Cozzela realizaram sua serialização peculiar, pois não se utilizaram de uma série de alturas, muito menos de uma série dodecafônica. O ponto de partida foi o próprio teclado do piano que foi tomado na sua totalidade como um espaço amostral ou um donimio sonoro 88 notas. Para os outros dois parâmetros musicais foi utilizada uma serialização com 17 durações e 10 intensidades como apresentado nas tabelas acima.

### 4.1 Possível Programa

Fazemos aqui uma hipótese de como seria o programa utilizado para compor esta obra, fazemos a apresentação no mesmo em *pseudo-code*. Buscamos um algorítmo de complexidade mínima que pudesse gerar a sequência que analisados e tentamos otiminizar o processo de avaliação de repetição de notas com um *array de flags*. Supomos então um processo de controle das notas geradas, através do array TECLAS, ou seja, cada nota diferente terá seu valor mudado de *false* para *true*. O programa entra num laço onde geram-se aleatoriamente os três parametros e verifica-se "*se o parâmetro altura da nota gerada é igual aos anteriores*". Se a altura gerada for diferente de todas as anteriores, os três parámetros são enviados para saída onde são guardados para, posteriormente, serem escritos na partitura.

```
Programa Klavibm2
    // Entram o vetor de flag false
     For i=1 to 88 TECLAS(i)=false;
     // Contador de notas geradas
     N= 0;
    // Laço para gerar as notas
    While N< 89
      // Gera valores aleatórios
      J = RND(17); K=RND(10); M=RND(88);
      // verifica repetição de nota
      IF not(TECLAS (M)) then
        Escreve J, K e M na partitura;
        TECLAS(M) = true
        N= N+1;
```

## 6. Conclusão

Apresentamos uma obra de referência histórica e fizemos uma análise paramétrica da mesma. Partimos apenas da partitura, pois não tinhamos nenhuma outra referência a respeito do processo composicional. O nosso objetivo foi mostrar que seria possível desenvolver uma estratégia voltada ao conceito de "Composição Assistida por Computador". Neste caso, evidenciamos que os próprios autores, se posicionaram a denotar que seria o computador o compositor da obra. O que mostramos foi que a hipótese de composição assistida é perfeitamente viável frente às evindências

encontradas na partitura. Parece-nos que em relação ao processo de análise a nossa pesquisa estaria completa, falta-nos uma próxima etapa que seria o processo de síntese. Ou seja, utilizando o algorítmo apresentado no texto, poderíamos desenvolver um simples programa para gerar uma família que composições relacionadas com o processo de Duprat e Cozzella. É importante ressaltar que a nossa análise vinculada ao uso de um algorítmo probabilístico, nos leva a possibilidade de gerar **170x88!** versões diferentes de *Klavibm II*. Um número finito, mas extremamente grande de variações que na sua riqueza e variedade estão diretamente vinculadas ao uso que Duprat e Cozzella fizeram das 88 teclas do piano como espaço sonoro.

## 7. Agradecimentos

Denise Garcia teve o apoio da FAPESP através de projeto "Faces da Música Eletroacústica Brasileira: o Grupo Música Nova e seu pioneirismo na utilização de recursos tecnológicos". Jônatas Manzolli tem o apoio do CNPq através de bolsa de produtividade em pesquisa.

## 8. Referências

Babbitt, Milton (1965). "The Structure and Function of Musical Theory", *College Music Symposium 5*.

Béhague, Gerard. (1979) Music in Latin America: an introduction. New Jersey: Prentice- Hall.

Cozzella, Damiano et al. (1963) Música Nova. *Invenção*. São Paulo, número 3, ano 2, p.5-6, junho.

Gaúna, Regiane. (2001) *Rogério Duprat: sonoridades múltiplas*. São Paulo: Editora da Unesp.

Hiller, L., Issacson, L. M. (1959) Experimental Music: Composition with an Electronic Computer. Westport, CT: Greenwood Press.

Jones, K. (1981) Compositional Applications of Stochastic Processes, Computer Music Journal Vol 5 No 2, MIT Press Cambridge Mass.

Koenig, Gottfried Michael. (1978) “Composition Processes http://www.earlabs.org/text/text.asp?textID=9, página mantida por EarLabs.

Leite, Vânia Dantas. (2000) Musicians and movements that initiated electroacoustics in Brazil. *Anais do XX Congresso Nacional da Sociedade Brasileira de Computação*, Vol. 1 Curitiba, ISBN 85-7292 - 050 - 1. Versão completa em CDROM.

Linz Maués, Igor. (1989) *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.

Neves, José Maria. (1981) *Música contemporânea brasileira*. São Paulo: Ricordi Brasileira.

Xenakis, I. (1971) Formalized Music: thought and Mathematics in Composition. Indiana University Press, ISBN: 0-253-32378-9.

# Pandora: uma caixa-clara tocada à distância

**Sérgio Freire**

Escola de Música - Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627 – 31270-010 – Belo Horizonte – MG – Brasil

sfreire@musica.ufmg.br

***Abstract***. *This paper describes the creation/assembling of a musical instrument built from a snare-drum, a Midi controller (*Lightning II*), a programming enviroment (*Max-Msp*), an audio amplifier and a loudspeaker (placed inside the drum), and discusses strategies of movement capture, data mapping and performance specifically developed for this setup.*

***Resumo***. *O artigo descreve a criação/montagem de um instrumento musical construído a partir de uma caixa-clara, um controlador Midi (*Lightning II*), uma plataforma de programação (*Max-Msp*), um amplificador de áudio e um alto-falante (que deve ser embutido na caixa-clara), e discute estratégias de captação de movimentos, mapeamento de dados e performance desenvolvidos especialmente para essa configuração.*

## 1.Introdução

O presente artigo descreve as principais etapas da montagem e da utilização de um instrumento de percussão tocado à distância, sem contato físico direto, com "baquetas" eletrônicas. Foram aqui utilizados uma caixa-clara, um controlador Midi (*Lightning II*), uma plataforma de programação (Max-Msp), um amplificador de áudio e um alto-falante (que deve ser embutido na caixa-clara). A lembrança de um instrumento de percussão que tocava sozinho, com um alto-falante embutido – trazida de um estágio no estúdio eletrônico da Academia de Música da Basiléia, Suíça – aliada à disponibilidade em minha escola de uma caixa-clara já sem serventia para a classe de percussão foram fatores decisivos para tal iniciativa.

Desde o início, tornou-se evidente que a busca por uma simulação teleológica (relações claras entre causa gestual e efeito sonoro) deveria nortear o trabalho e que, consequentemente, ele deveria não apenas ser ouvido, mas também visto. Nessa simulação, o trabalho de programação tanto do controlador (para o envio das informações desejadas advindas da performance) quanto do *software* (para a análise dessas informações e geração de respostas sonoras adequadas) é uma etapa fundamental. Também foram importantes para o desenvolvimento do projeto as características que Wessel e Wright (2002) consideram desejáveis em novos instrumentos digitais: (1) implementação do paradigma instrumental tradicional: "um gesto para um resultado acústico"; (2) controle do volume geral e da densidade dos sons produzidos; (3) correspondência entre o tamanho do gesto de controle e o resultado acústico.

## 2.O controlador *Lightning II*

Em 1996, Buchla lançou o controlador Midi sem fio *Lightning II*, baseado em tecnologia de sensores infra-vermelhos:

> *Lightning II* é um controlador Midi especializado que detecta a posição e o movimento de baquetas seguras com as mãos e transforma essa informação em sinais Midi, para o controle expressivo de instrumentos musicais eletrônicos. Além de funcionar como um poderoso controlador Midi, *Lightning II*, com seu sintetizador embutido de 32 vozes, configura-se como um instrumento completo, pronto para ser tocado.(...) Baseado em princípios da triangulação ótica, *Lightning II* coleta suas informações de pequenos transmissores infra-vermelhos que estão embutidos nessas baquetas. Sem restrições de fios, esses bastões oferecem completa liberdade de movimento em um espaço de performance que pode chegar a 3,65 metros de altura por 6,1 m de largura.[1]

O controlador é composto das seguintes partes: duas baquetas alimentadas por pilhas, um receptor (que pode ser fixado a um pedestal de microfone convencional) e um módulo de processamento digital, cuja caixa também contém o sintetizador Midi, conexões para dois pedais do tipo *switch*, e portas Midi (*in*, *out*, *aux*). Cada baqueta conta também com um *switch*, cujo acionamento também é transformado em um sinal Midi (figura 1). A movimentação das baquetas é realizada frente a um retângulo imaginário (cuja proporção entre altura e largura, segundo o manual, é de aproximadamente de 7:10), dividido em 8 zonas para cada mão (4 superiores e quatro inferiores). A programação básica consiste em associar um tipo de estímulo (que pode valer para todo o espaço de performance ou para zonas específicas) com uma resposta sonora.(Buchla, 1996)


**Figura 1: (a) módulo de processamento digital e baquetas; (b) receptor montado em um pedestal.**

O processamento digital está a cargo de um microprocessador de 8 bits (TMS370, da *Texas Instruments*), o sintetizador segue o padrão *General Midi*, e o controlador, além da capacidade de armazenar programações próprias do usuário, oferece uma série de *presets* que acoplam determinados tipos de gestos a resultados sonoros específicos. Dentre os vários *presets* permanentes, encontram-se *Rock Drums*, *Timpani*, *Stereo Marimba*, *BluGuitAccord*, *Dual Theramin*, *Jazz Traps* etc.(Buchla, 1996). Há também um *Conduct Program,* capaz de seguir gestos de um regente em seqüências de estímulos (compassos) pré-definidos.

Dentre as iniciativas antecessoras desse controlador sem fio podem ser apontadas o *theremin* (Ruschkowski, 1998), o sistema de espacialização quadrifônica desenvolvido por Schaefffer, Henry e Poullin (Bayle, 1993), o *light baton* (Bertini et al., 1992), o *radio baton* (Mathews, 1991). Embora nem todos esses instrumentos e sistemas possam ser caracterizados como realmente sem fio, a movimentação baseada em gestos que lembram os de um regente é uma característica comum.

Um breve exame da programação de concertos e congressos relacionados à computação e música mostra grande interesse e uma utilização intensa do *Lightning II,* desde seu lançamento. Na série de simpósios brasileiros de computação e música[2], encontram-se, p. ex., um workshop de Mark Goldstein sobre esse controlador em Recife, em 1996; uma composição de Bob Willey em Brasília, em 1997; uma comunicação científica no Rio de Janeiro, em 1999, publicada no ano seguinte (Cerana, 2000). Também na programação dos congressos dedicados a *New Interfaces for Musical Expression* (NIME), esse controlador se faz presente[3].

Mesmo já passados onze anos de seu lançamento, o controlador *Lightning II* ainda apresenta, ao lado das inevitáveis defasagens tecnológicas, grande versatilidade no palco, devido à tecnologia sem fio e à amplitude de movimentos que é capaz de captar. Sua *bandwidth* limitada (há um intervalo médio de 10 ms entre cada valor de um controlador "contínuo" enviado pela porta Midi) não chega a interferir na performance humana, pois é suficiente para captar com fidelidade os gestos sem inserir saltos ou latência audíveis/visíveis no processo de mapeamento. A principal deficiência detectada foi a baixa sensibilidade a ataques (o valor máximo de 127 na detecção de ataques é rapidamente atingido, mesmo com gestos suaves), o que nos levou a realizar uma nova programação da captação de movimentos na plataforma Max.-Msp. A divisão *a priori* do espaço de performance em oito zonas também não nos pareceu totalmente compatível com o potencial do controlador.

## 3.Captação de movimentos

Uma vez definida a necessidade de se trabalhar numa nova programação de captação de movimentos, o controlador *Lightining II* passou a ser tratado não mais como um instrumento *stand-alone*, e sim como uma interface capaz de gerar quatro coordenadas independentes de forma contínua - valores entre 0 e 127 para os eixos horizontal e vertical de cada baqueta -, e informações sobre acionamento de *switches* e pedais. A primeira conseqüência é que o retângulo de performance, dividido em 8 zonas para cada baqueta, passou a ser projetado em um quadrado virtual, sem subdivisões *a priori*. Deve-se notar que, na utilização desse controlador, movimentos no eixo vertical devem ter uma amplitude um pouco menor do que no horizontal para obtenção do mesmo efeito quantitativo.

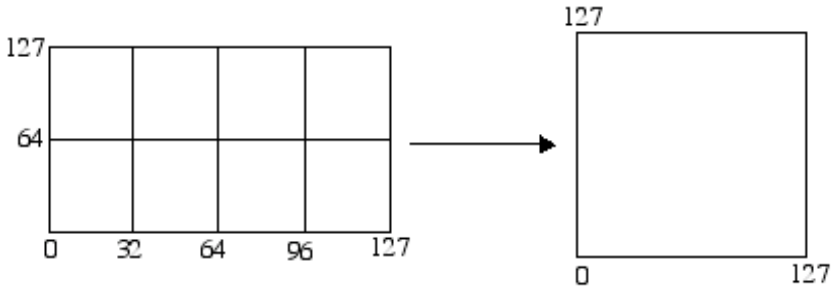



**Figura 2: O retângulo à esquerda representa as 8 zonas de performance pré-programadas no *Lightning II*. Os números nos dois eixos indicam os valores de transição entre essas zonas. À direita, uma representação da projeção desse retângulo em um quadrado virtual, cuja exploração passa a depender de programação específica.**

A primeira rotina a ser programada foi a detecção de ataques, que, como já exposto acima, foi considerada a mais deficiente nesse controlador. Para isso, não basta

apenas detectar o momento de mudança de direção de movimento, pois é também preciso quantificar o gesto preparatório desse ataque. Essa quantificação foi realizada segundo duas variáveis: - amplitude do movimento e duração do gesto – e requer necessariamente a determinação de um valor/momento inicial.

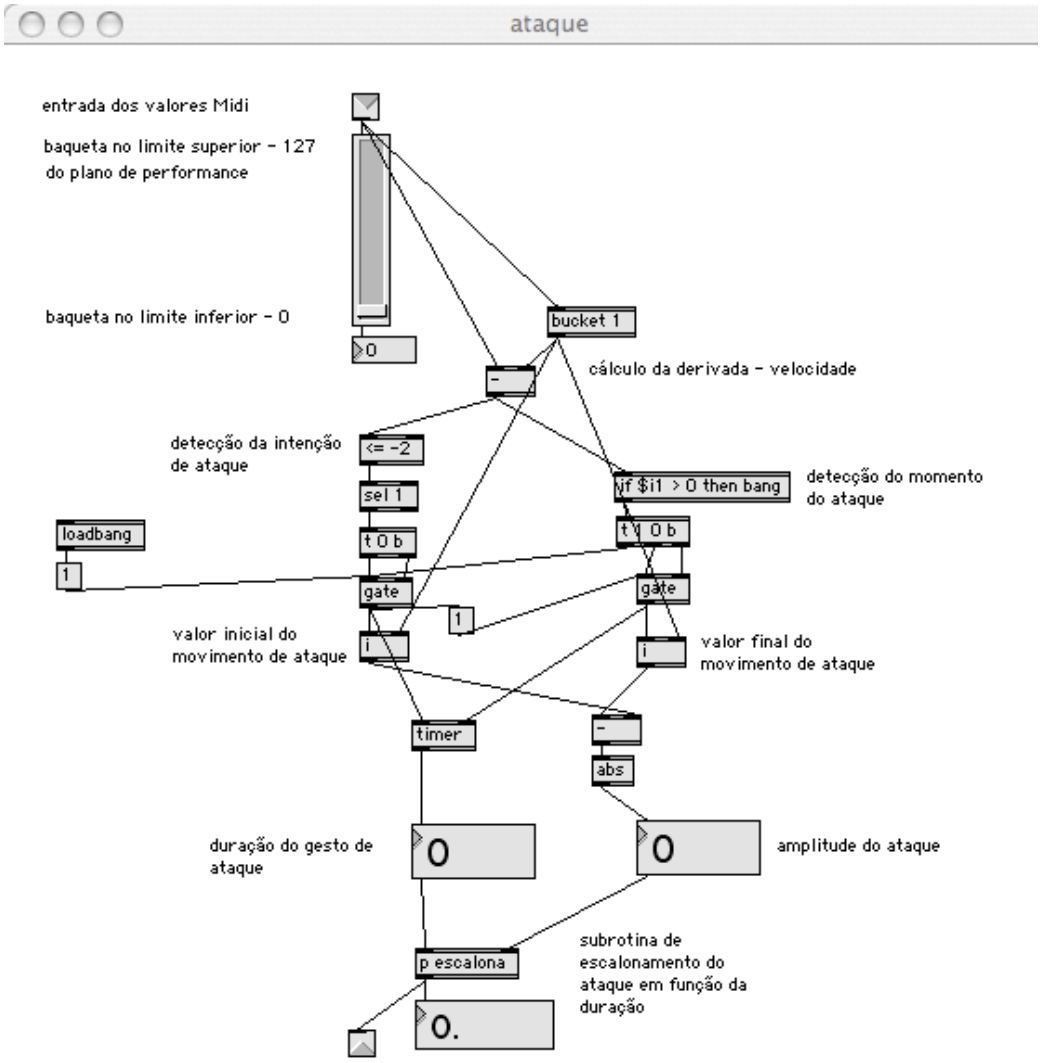



**Figura 3: *patch* responsável pela detecção e quantificação de um gesto de ataque.**

A velocidade é a derivada do movimento e pode ser digitalmente calculada como a diferença entre o valor de posição atual e o imediatamente anterior; o início de uma intenção de um ataque pode ser determinado por uma variação na velocidade que exceda certo limite. Assim, no momento em que a derivada excede esse valor limite (2, p. ex.), um cronômetro é ligado e o valor da posição imediatamente anterior é armazenado. No momento da mudança de direção do movimento (inversão de sinal da derivada, significando o momento do ataque), o cronômetro é desligado e o valor da posição desse instante é subtraído do valor inicial. Calcula-se, assim, a duração e a amplitude do movimento. É aconselhável que se faça um escalonamento – não linear, de preferência – dessa amplitude em função da duração do gesto, gerando, assim, o valor final da intensidade do ataque. Na presente implementação, durações muito curtas

amplificam essa intensidade, durações médias não a modificam, e durações maiores do que 500 ms atenuam fortemente a intensidade. Esse escalonamento pode ser exagerado, uma vez que a limitação de resolução de 7 bits (do controlador Midi) não mais se aplica. Obviamente, essa estratégia de detecção pode ser aplicada em qualquer uma das quatro direções cardinais. (figura 3)

Uma rotina útil e facilmente implementável é a divisão do plano de performance em zonas arbitrariamente definidas. No presente caso, explora-se uma divisão desse plano em 9 zonas, que será detalhada mais adiante. Uma outra fonte de informações de performance vem do uso de *switches* e pedais: além de seu simples acionamento, podem ser calculados o intervalo de espaçamento entre seu acionamento, a duração de cada acionamento, o reconhecimento de seqüências e combinações específicas entre eles etc.

As demais questões relativas ao processamento de movimentos captados por esse controlador estão centradas no mapeamento dessas informações a diferentes rotinas de controle e de processamento/síntese sonoras e serão tratadas a seguir.

## 4.Mapeamento e respostas sonoras

### 4.1 Pesquisa de sonoridades

Após uma divertida fase inicial de uso desse controlador Midi, surgiram dois incômodos: a dependência de sons sintetizados (no padrão *General Midi*, fortemente influenciados pela imitação de instrumentos existentes) e a localização dos sons resultantes dos gestos realizados com as baquetas, dependente das características do sistema de amplificação utilizado. Assim, à grande mobilidade de performance oferecida pelo *Lighting II* se contrapunham a pouca flexibilidade desses sons pré-fabricados e a forte direcionalidade do resultado sonoro.

A instalação de um alto-falante dentro da caixa-clara abriu uma nova frente de experimentações. Descobriu-se, primeiramente, que é muito importante que o alto-falante seja posicionado bem próximo à pele inferior da caixa-clara, porém sem contato direto. E também que tenha uma potência compatível com o volume sonoro normalmente esperado desse instrumento em condições normais de execução (figura 4).

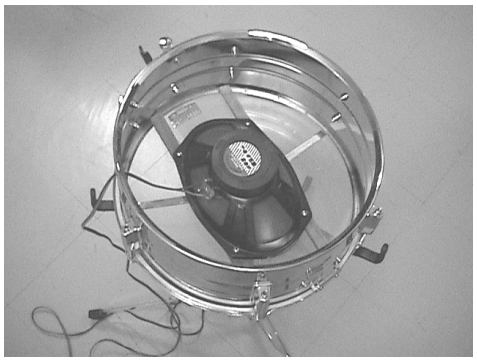


**Figura 4: caixa-clara (sem a pele superior) com alto-falante colocado rente à pele inferior.**

#### 4.1.1 Simulação de ataques

O próximo passo foi a pesquisa das sonoridades resultantes da interação entre o alto-falante e a caixa-clara. O primeiro desafio foi a simulação de um ataque realista, e nessa

tentativa ficou claro que o uso da esteira seria essencial para a maioria das respostas sonoras que poderiam advir dessa interação. Um impulso com duração entre 10 e 70 ms, com um envelope dinâmico triangular, foi a escolha para o modo de ataque simples. A forma de onda escolhida para esse impulso foi a senóide, após testes com ondas dente-de-serra, quadrada, faixas de ruído e ruído branco. A escolha da freqüência dessa senóide está diretamente ligada à tensão das peles da caixa (tanto a inferior quanto a superior), e nesse caso específico foi a faixa entre 100 e 200 Hz a que deu melhores resultados. A amplitude máxima do impulso, bem como sua duração, são diretamente proporcionais à intensidade do ataque detectado; a variação dessa intensidade também provoca pequenas alterações da afinação do impulso. Uma simulação da posição do ataque entre o centro e a borda da pele do tambor é feita com adição de harmônicos da fundamental escolhida, cujas amplitudes dependem da localização da baqueta no eixo horizontal no momento do ataque.

### 4.1.2 Ataques com ricochete

Outra resposta sonora implementada foi a simulação de um ataque com ricochete; nesse caso, um gesto de ataque provoca uma série de impulsos (cujos parâmetros são mapeados como no ataque simples), cuja velocidade, intensidade inicial e duração são também função da intensidade do ataque. Sucessivos ataques desse tipo são capazes de gerar um rulo (figura 5).

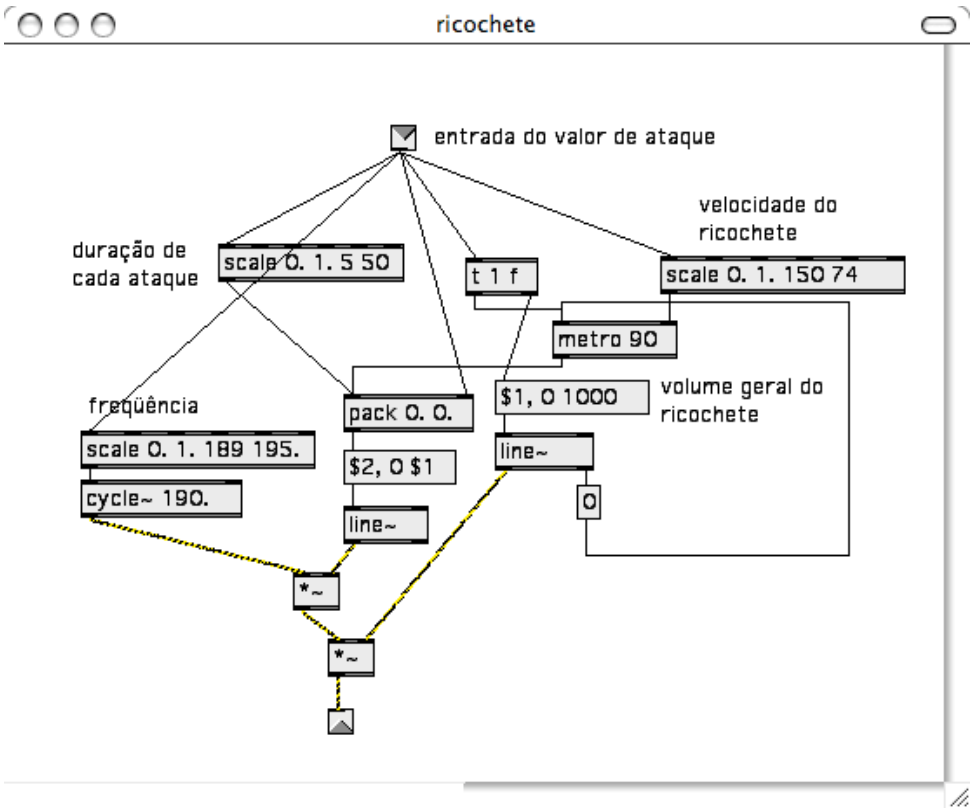



**Figura 5: mapeamento do valor de ataque ao controle de cinco variáveis de síntese: intensidade, duração e freqüência de cada ataque; velocidade e volume geral do ricochete.**

### 4.1.3 *Talking drum*

Outra resposta sonora implementada foi uma espécie de *talking-drum*, onde são exploradas as quatro coordenadas independentes do sensor. Uma das mãos executa ataques no eixo vertical (como descrito acima), ao mesmo tempo em que altera a afinação de uma onda dente-de-serra no eixo horizontal (entre 60 e 130 Hz). A outra mão atua em um plano cujos eixos controlam a freqüência central de filtros passa-banda que simulam os primeiro e segundo formantes de uma voz humana: o primeiro filtro atua entre 270 e 730 Hz, o segundo entre 840 e 2290 Hz (apud Winckel, 1960). Embora nem sempre a exploração desse plano formântico resulte em vogais conhecidas, uma qualidade vocal está sempre presente nessa sonoridade; com um pouco de prática é possível articular entonações e seqüências específicas de vogais (figura 6).

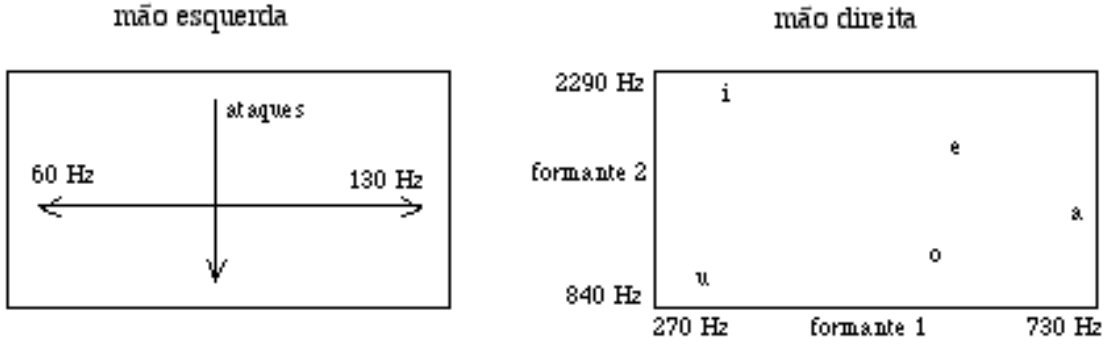



**Figura 6: o diagrama representa o espaço de performance na simulação de um *talking drum*. A mão esquerda controla os ataques no eixo vertical e a freqüência fundamental no eixo horizontal. A mão direita explora o plano dos formantes, no qual estão indicadas as posições aproximadas das principais vogais.**

### 4.1.4 Sons contínuos e variantes

A exploração de sons contínuos construídos com superposição de senóides em relação harmônica também trouxe resultados bastante satisfatórios. Foi programada uma divisão do plano de performance em nove zonas (3 x 3), onde cada zona aciona um harmônico específico de um som fundamental, com durações (entre 2,5 e 8 segundos) e envelopes dinâmicos pré-definidos. Na implementação atual, uma fundamental de cerca de 80 Hz (levemente variada no tempo) pode ser misturada aos harmônicos de ordem 3, 5, 7, 8, 9, 11, 13 e 15 . A figura 7 mostra a programação do espaço de performance para cada uma das mãos (a inversão de funções entre as mãos pode ser facilmente comutada), e a figura 8 mostra o patch responsável pela síntese dos parciais.

### 4.1.5 Controle de amostras sonoras

Arquivos de som também podem ser acionados e controlados por gestos de performance: certos tipos de música percussiva tornam-se quase irreconhecíveis, instrumentações diversas sofrem diferentes tipos de distorção, sons iterativos (já presentes em arquivos pré-gravados ou realizados por impulsos comandados pelo movimento da baqueta, como em um reco-reco) sem altura definida e de baixa amplitude podem ser muito expressivos. O uso de voz humana falada ou cantada em um volume mais alto provoca um efeito semelhante a uma caixa acústica de baixa qualidade ou com os cones de seus alto-falantes rasgados.

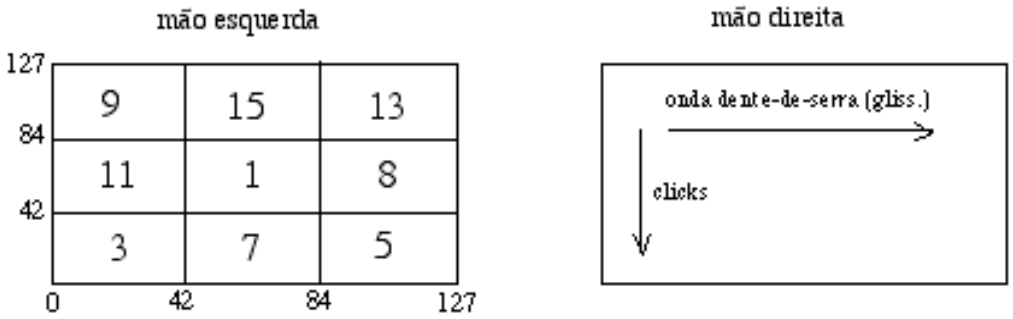



**Figura 7: o diagrama mostra o plano de performance para a rotina de geração de sons contínuos com espectro variável. A mão esquerda se move entre 9 divisões do espaço, acionando o harmônico correspondente ao número de cada zona. A mão direita toca sons iterativos nos dois eixos.**

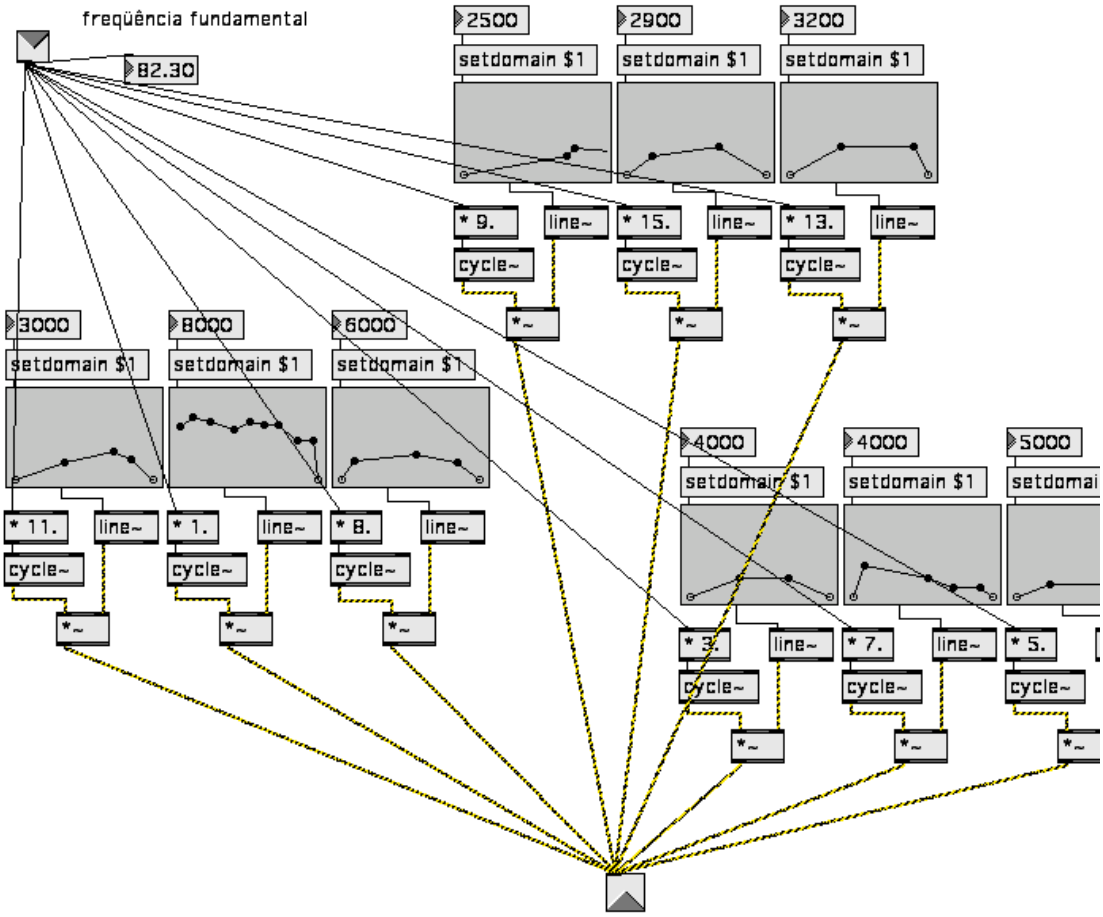



**Figura 8: detalhe do *patch* responsável pela síntese de parciais de uma freqüência fundamental comandada pelo movimento das baquetas. Os envelopes de cada harmônico têm formas e durações diferentes.**

### 4.2 Performance

A partir dos mapeamentos e respostas sonoras acima descritos, foi desenvolvida uma estratégia de performance com quatro seções principais: (1) exploração de ataques e rulos; (2) geração de sons contínuos com variação espectral; (3) ataques com caráter vocálico; (4) reprodução e processamento de trechos muscais com caráter percussivo. Essas seções são selecionados com um pedal, e variações dentro de cada seção são acionados pelo *switch* de uma das baquetas.

O nome Pandora foi dado a esse conjunto formado pelo "instrumento" e suas estratégias de execução, sendo que a inspiração mitológica tem mais a ver com a variedade de "entrada" do que com a de "saída" dessa caixa. Ela foi estreiada em um concerto eletroacústico no 37° Festival de Inverno da UFMG, em julho de 2005, em Diamantina (MG), e tem sido regularmente apresentada com pequenas modificações dessa programação[4].

## 5.Considerações finais e perspectivas

Nessa experiência, repetiu-se uma situação já relatada em um artigo anterior: "o trabalho com sistemas interativos nos coloca diante de situações dificilmente abarcadas por conceitos musicais tradicionais: as noções de composição e interpretação, instrumento e escritura, abstração e concretude perdem seus claros limites no processo criativo." (Freire, 2003)

O trabalho com Pandora deixou claro que a defasagem tecnológica do controlador *Lightning II* se encontra mais no processamento gestual e nas respostas sonoras oferecidas do que em sua originalidade como sensor. Outras aplicações (já em fase de implementação) com esse sensor incluem o controle de algoritmos de síntese percussiva por modelamento físico, e controle interativo de imagens e animações, tais como deformações de contornos e cor, arrastos, choques etc. Uma *bandwidth* de 10 ms é razoavelmente compatível com a captação de gestos humanos e com a taxa de geração de imagens (30 *frames* por segundo). Tópicos futuros incluem a exploração de um um espaço de até quatro dimensões e o reconhecimento de formas geométricas "desenhadas" no ar.

Acredito que as soluções para os desafios atuais da interatividade sonora e musical não estão mais na criação de instrumentos *stand-alone*, com hardware e programação específicas, e sim nas possibilidades de interconexão entre (1) diferentes tipos de interface (controladores e sensores), (2) softwares de processamento, análise e mapeamento gestual e (3) modos de geração sonora. O trabalho aqui relatado mostra que esses diferentes elementos de lutheria/programação/composição/interpretação podem ir sendo integrados à medida em que são encontrados/programados, segundo necessidades e contextos específicos.

## Bibliografia

Bayle, François (1993). "la musique acousmatique, ou l'art des sons projetés". In: Bayle, F. *musique acousmatique: propositions... ...positions*. Paris: INA; Buchet/Castel, pp. 47-67. Texto de 1984, revisado em 1993.

Bertini,Graziano e Carosi, Paolo (1992). "Light Baton: a System for Conducting Computer Music Performance." International Computer Music Conference. Proceedings, pp. 73-76. San Jose, EUA.

Buchla and Associates (1996). *Lighting II User's Guide*. (Manual do controlador Lighting II) Revisão de 01/12/1996. Copyright de Buchla and Associates.

Cerana, Carlos (2000). "Gesture control of musical processes: a MAX environment for Buchla's 'Lightning'". *Organised Sound*, vol. 5, pp. 3-7.

Freire, Sérgio (2003). "cvq: entre o meta-instrumento e a pseudo-obra". Anais do IX Simpósio Brasileiro de Computação e Música (vol 9 dos anais do XXIII Congresso da Sociedade Brasileira de Computação), pp. 271-276. Campinas: SBC.

Mathews, Max V (1991). "The Radio Baton and Conductor Program, or: Pitch, the Most Important and Least Expressive Part of Music." *Computer Music Journal*, vol. 15, no.4 , pp. 37-46.

Ruschkowski, André (1998). *Elektronische Klänge und musikalische Entdeckungen*. Sttutgart: Philipp Reklam.

Wessel, David e Wright, Matthew (2002). "Problems and Prospects for Intimate Musical Control of Computers". *Computer Music Journal*, vol. 26, no. 3, pp. 11-22.

Winckel, Fritz (1960). *Vues Nouvelles sur le Monde des Sons*. Paris: Dunod. Tradução de A. Moles e J. Lequeux. Publicado originalmente em alemão em 1952.

## Notas

[1] http://www.buchla.com/lightning/descript.html, visitado em 26/07/2007. As informações aqui presentes sobre as dimensões máximas do espaço de performance não concordam inteiramente com a do manual do equipamento. Em um caso, as proporções são de 6:10, e no outro de 7:10.

[2] http://gsd.ime.usp.br/sbcm, visitado em 26/07/2007.

[3] http://www.nime.org/2003/nime03_home.html, e http://hct.ece.ubc.ca/nime/2005, visitadas em 26/07/2007.

[4] Exemplos sonoros podem ser encontrados em http://www.musica.ufmg.br/~sfreire.

# Coherence and spontaneity in *Interferências (2003)*, for cello and computer[1]

**Daniel L. Barreiro**

Rua Rosalino Belline, 98 - Santa Paula - 13564-050 - São Carlos - SP - Brazil

dlbarreiro@gmail.com

***Abstract.*** *This paper presents some musical and technological aspects involved in the composition of Interferências (2003), for cello and computer (running a Max/MSP patch). It mentions the main characteristics of the work and how the Max/MSP patch contributes to generate a musical discourse that seeks both coherence and moments of spontaneity (as the result of a process of mutual influences between computer and performer).*

## 1. Brief outline of the composition

*Interferências* ('interference', in Portuguese) is a composition for cello and computer (running a Max/MSP patch) which is based on the interplay of three sonic layers based on sound materials of the following kind:

a) sounds produced by the cello live;

b) pre-recorded cello sounds that are triggered and processed in real-time by the computer according to the analysis of the audio signal produced by the cellist;

c) pre-processed electroacoustic sounds triggered by the person who controls the computer.

The sounds processed in real-time work as an intermediate level (a link, in fact) between the two other layers (the sounds of the instrument and the more heavily processed sounds of the pre-processed electroacoustic part). For this reason, they are not heavily transformed - only submitted to subtle changes regarding pitch, duration and amplitude. A 'counterpoint' is established between these three layers throughout the piece.

To a certain extent, the structure of *Interferências* can be regarded as a continuous process that incorporates some contrasting relations into its flow. The contrasts are established in such a way that does not justify the segmentation of the structure into different sections. There are, however, different 'segments', understood as moments in which the musical flow presents some fluctuations or variations. These variations are mainly related to differences in articulation and density of events (either in the instrumental or in the electroacoustic part, or even both), and to some fluctuations

1 This paper is based on some excerpts of the written component of my PhD in Musical Composition [Barreiro 2006], which was carried out between 2003 and 2007 at *The University of Birmingham* (UK) under the supervision of Professor Jonty Harrison and with a scholarship from the Brazilian Government through Capes Foundation. The Max/MSP patches discussed here were designed under the guidance of Dr Erik Oña.

in the flow of musical time. The phrases were organised to speed up the unfolding of time at certain moments and to slow it down at others, without signalling overt ‘sections’ to the listener.

Musical time was, therefore, a criterion taken into consideration for defining issues concerning the structure of the composition. Fluctuations in the overall time flow were explored through the use of different strategies derived from a research carried out during my Master’s degree [Barreiro 2000; Barreiro and Zampronha 2000a and 2000b]. These strategies include the use of variations in terms of speed and density of events, the level of contrast or similarity between sound materials and the use of unexpected events as 'impacts' that cause fluctuations in the time flow.

The work is conceived for stereo. Although one pair of loudspeakers might be considered enough for its performance, it is advisable to use at least two pairs instead of just one (especially when the work is performed in a large concert room). In this case, the placement of loudspeakers should follow the diagram presented in Figure 1, with two 'frontal' loudspeakers on the stage and two 'wide' ones on the floor – forming an arc. All the three sonic layers mentioned above are sent to the loudspeakers. The Max/MSP patch is designed in such a way that it allows independent volume control of each pair of loudspeakers. It is even possible to control the volume of the cello sounds and the volume of the two electroacoustic sonic layers independently in each pair. This allows, for example, that the cello signal be sent to the 'frontal' loudspeakers with a volume slightly higher than the 'wide' ones, consequently keeping the cello sounds mostly in the centre of the stereo image (which matches the position of the performer in the room).

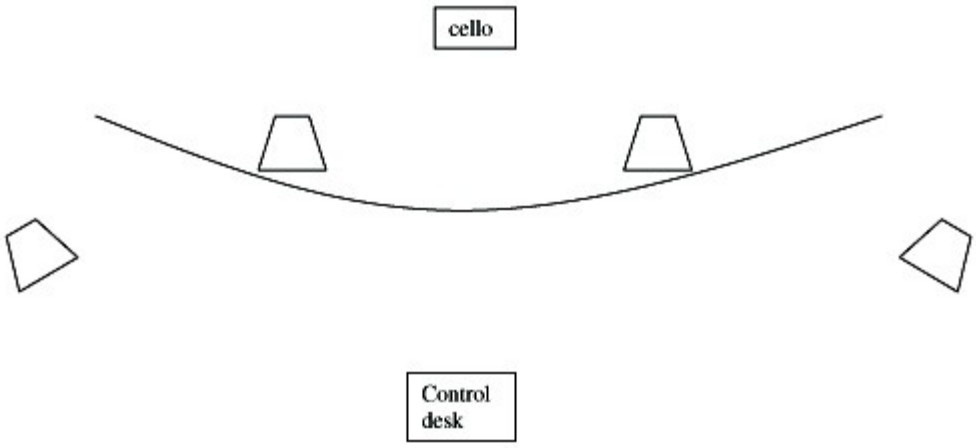



**Figure 1**

## 2. Sound material and compositional strategy

The electroacoustic part of *Interferências* (both the sonic layer generated in real-time and the layer of pre-processed sounds) is completely based on cello sounds captured in a recording session I had with the German cellist and composer Caspar Johannes Walter

in 2003.

As guidance for the recording session I elaborated some charts displaying general information (in words) about the sound-types I would like to have as source material for the piece. The intention was to get a set of isolated sounds that should be as varied as possible. The charts asked for glissandos, pizzicatos, trills, tremolos, ricochets (jeté), harmonics, sounds of tapping on the body of the instrument and on the fingerboard, sounds with and without vibrato, sul ponticello, sul tasto, normal playing, notes preceded by single and double appogiatura, etc. There were indications that each kind of articulation should be played with different durations (short or long), dynamics, speeds and in different registers of the instrument. There were no indications of pitches - they should be chosen by the player.

The sounds obtained with this strategy constituted the source material that, after being processed, was used for the composition of electroacoustic segments of various lengths that are triggered at specific moments in the piece by the person who controls the computer (making up the pre-processed sonic layer mentioned above).

Some of these samples from the recording session were also stored as unprocessed source sounds and constitute the material that is triggered and processed in real-time by the computer during the performance according to the analysis of the audio signal produced by the cellist.

For the recording session mentioned above, I also wrote some short phrases on single line staves. They displayed precise information about the durations, articulations and dynamics to be used for each event, plus some vague information about register (low, medium and high). Again, I deliberately omitted details about specific pitches. The intention was to make the performer concentrate on the gestures indicated by the other parameters only. He should not worry about playing the 'right' notes - his concentration should essentially be focused on the gestures. I wanted him to improvise with the pitches within the framework set by the other parameters, so I asked him to play each phrase several times with different notes and some slight variations of tempo in each occasion.

The composition of these phrases was inspired by the concept of *Unités Semiotiques Temporelles* ('Temporal Semiotic Units'), or simply UST, developed by a group of researchers at the *Laboratoire Musique et Informatique de Marseille* - M.I.M. [Mandelbrojt 1996]. In general terms this research can be defined as a classification of musical gestures according to their temporal signification, i.e. the effects they potentially cause on listeners in terms of temporal perception. To some extent, this concept is the opposite of that of the 'sound object', since it does not target the sound for itself "in its material, its inherent texture, its own qualities and perceptual dimensions", "independently of its origins or its meanings"[2]. On the contrary, in the case of the USTs the meaning or, more precisely the 'signification' (understood not in the sense of verbal or conventional meaning, but as a temporal sensation) is the target. They are called 'units' because they are defined as coherent entities that preserve their 'signification' regardless the musical context in which they occur. Although this last concept can be polemic, I found the classification of gestures in the theory of UST quite interesting and

2 From the definition of ‘sound object’ presented in Chion (1983).

decided to use some of them as compositional tools for organising my musical thinking while composing *Interferências*.

Here are some of the categories that were chosen from the theory of the USTs:

1) Activity followed by glissando, with acceleration (as if something has been set in motion and suddenly fallen down);
2) Activity followed by a steady and extended sound (forces in contradiction: movement versus stability);
3) Relative stability suddenly disturbed by a gesture performing an acceleration;
4) Irregular repetition;
5) Sudden deceleration (as if a sudden resistance forbids a movement to continue);
6) Obsessive repetition;
7) Progression;
8) Presentation of a musical idea whose progression is not clear.

The musical phrases recorded by Caspar Johannes Walter as a result of this process were not used for the composition of the electroacoustic part of the work whatsoever. They were just used as a guide during the composition of the instrumental part. During the compositional process I had other recording sessions (this time with the British cellist Ellen Fallowfield, who premiered *Interferências* in November 2003) in which I could try new phrases, revise previous ones, try new combinations of phrases, etc. until I finished the composition of the instrumental part.

The phrases from the recording sessions were therefore treated in an 'acousmatic' way. The compositional work consisted in identifying their salient features, cutting, selecting and re-arranging their segments into larger phrases, and combining them with new segments in order to generate a continuous musical argument for the instrumental part. This empirical process was guided by the aural judgement of the results obtained and had the aim to explore fluctuations in the flow of musical time, based on the USTs mentioned above.

## 3. The score and the Max/MSP patch

The result of this compositional process is a score that indicates duration, articulation and dynamics of the musical events to be played by the cellist. The choice of pitches is left for the performer - the score only gives general indications of register (see Figure 2). The pitch contour (indicated in terms of register) is more important than the specific pitches themselves. As a consequence, the performer is encouraged to concentrate on the gestures, i.e. on the articulations, the contour and the amount of energy he or she delivers while playing the piece, rather than the choice of pitches.

The computer, running a Max/MSP patch, analyses the following three parameters:

a) the time elapsed between the onsets of subsequent events played by the cellist;

b) the duration of each event;

c) the amplitude of each event.

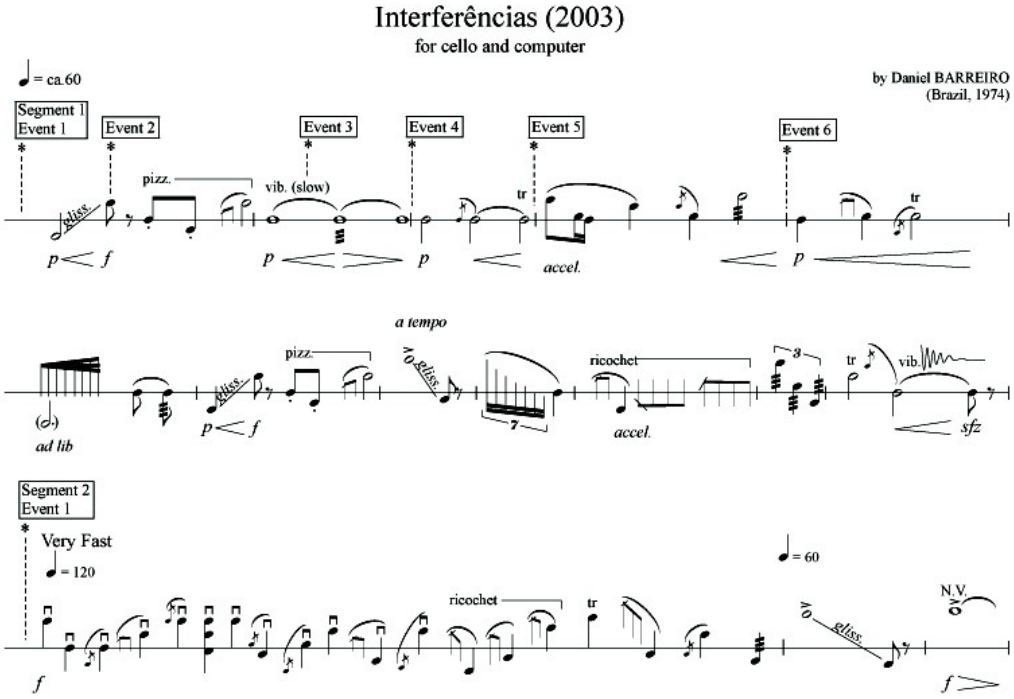



**Figure 2**

Based on the data obtained, it triggers and processes the pre-recorded cello sounds in real-time. Another person (not the cellist) follows the cues displayed on the score in order to trigger the playback of the pre-processed electroacoustic sounds and to change the settings that are used by the Max/MSP patch at specific moments in the piece.

The pre-recorded cello sounds (which are processed in real-time by the computer) are chosen according to a two-stage process. First, a category of sounds (based on the articulation they present) is chosen according to a list of probabilities that defines which categories are more likely to happen in each segment of the piece. Then, one specific sound file within that category is chosen randomly. Since the list of probabilities is changed from one segment of the piece to another, there are some subtle differences between the segments with regard to the characteristics presented by the sonic layer generated in real-time.

The definition of the density of events to be used in this sonic layer and the way the sounds are processed derive from data collected through the analysis of the cello signal. The analysis parameters mentioned above (interval between subsequent onsets, duration of the events and amplitude) are used to control three processing parameters (density of events to be triggered, transposition and amplitude). The way the analysis and the processing parameters relate to each other changes from one segment of the

piece to another. Therefore, at a given moment it may happen that the amplitude of the cello sounds determines the amplitude of the pre-recorded sounds, for example. At another moment, the duration of the cello sounds (and no longer the amplitude) may determine the amplitude of the pre-recorded sounds. These variations avoid the interaction between cello and the sonic layer generated in real-time becoming predictable and derive from the compositional intention of avoiding direct and constant mappings between parameters of analysis and parameters of processing.

The decision for generating this sonic layer in real-time and for using probabilistic and random processes was guided by the intention to incorporate the spontaneity and the surprises created by such a process in the composition.

Since the sounds that are triggered and processed in real-time are pre-recorded cello sounds, there are some connections between this layer and the cello sounds played live, especially when there are coincidences in terms of articulation. This kind of connection sometimes also happens between the cello and the pre-processed electroacoustic sounds, creating imitative relationships between the two.

The recapitulation of materials, especially of some kinds of articulations, happens in different moments in the piece, linking the musical events and consequently contributing to generate the sense of coherence throughout the work.

## 4. A closer look at the Max/MSP patch

In *Interferências* - as usually happens with Max/MSP programming - the sound analysis and processing operations are carried out by several modules (Max/MSP subpatches) embedded in the main patch.

One of the main controlling tools displayed on the main patch itself (not in a subpatch) is a pair of two-dimensional volume controllers (see Figure 3) that enable the user to set and change the volumes of the four different audio signals independently: the signal of the cello live, the signal that comes out of the reverb unit (embedded in the patch), the sonic layer generated in real-time and the layer of pre-processed electroacoustic sounds.

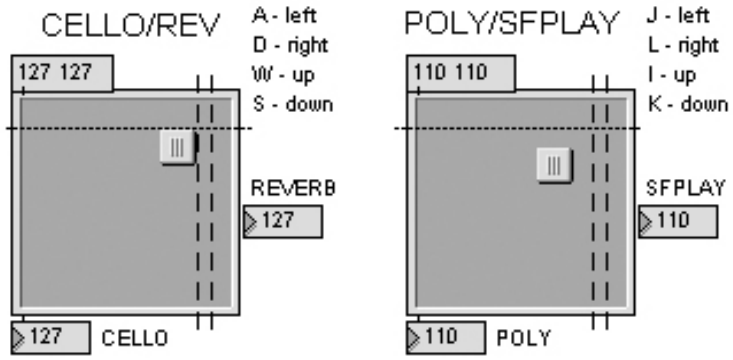



**Figure 3**

The volumes are controlled by pressing keys on the computer keyboard (A, D, W, S, J, L, I and K) which were programmed to guide the movements of the faders inside the squares (see Figure 3). Some other faders positioned after these controllers in the signal chain allow the control of the general volume that is sent to the loudspeakers. As a consequence, the patch works as a mixing desk.

The analysis of the time elapsed between subsequent onsets presented in the cello signal (which relates to tempo in broad terms) is carried out by the "TempoEstimation" subpatch shown in Figure 4.

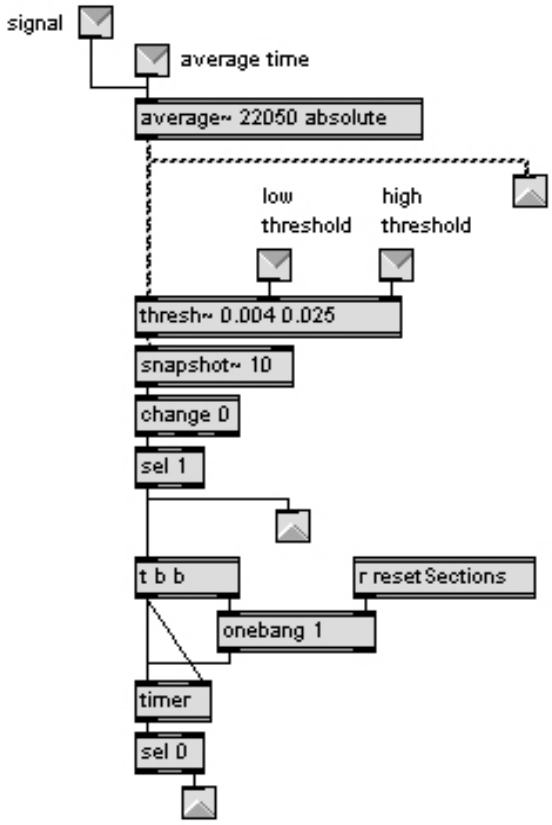



**Figure 4**

Whenever the amplitude of the signal goes above the high threshold (set in the 'thresh~' object), the 'timer' object receives a 'bang' that ends the previous count (reporting the time elapsed since the previous 'bang' in miliseconds) and starts a new one (which will be reported when the next onset is identified). A new onset is only identified by the 'thresh~' object after the amplitude of the signal goes below the low threshold and up again. A similar (but slightly different) subpatch, called "DurEstimation", is used for identying the duration of events played by the cellist (and no longer the time spans between subsequent onsets). The amplitude of the cello signal is also continuosly analysed by another module.

Data from these three analysis parameters (amplitude of the cello signal, duration of the events and time spans between subsequent onsets) are associated with the processing parameters (density of events, amplitude and transposition) in three subpatches that run in parallel throughout the performance of the piece. Each one of them is responsible for associating the three analysis parameters with one specific processing parameter. Therefore, based on data from each one of the three analysis parameters, the "PolifControlMachines" subpatch generates data to control the number of pre-recorded soundfiles that are triggered at a time. The "TransControlMachines" subpatch associates data from the three analysis parameters with data that control the speed of playback (and therefore the transposition) of the pre-recorded soundfiles. The "AmpControlMachines" subpatch performs the same kind of job in regard to the amplitude of the pre-recorded soundfiles that are triggered in real-time. Figure 5 and Figure 6 show two different modules that illustrate some of the approaches adopted to generate processing parameters from the analysis parameters.

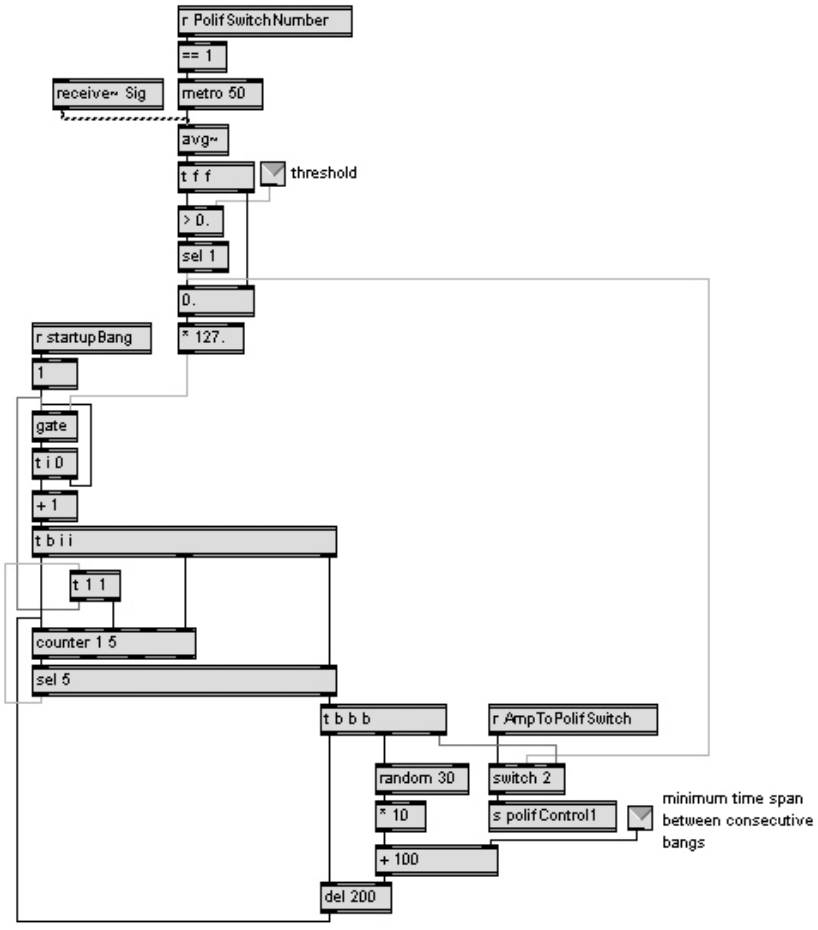



**Figure 5**

Figure 5 shows one of the modules that run inside the “PolifControlMachines” subpatch. This module, called 'AmpToPolifControl', transforms values of amplitude of the cello signal into data that control the density of events that are triggered in real-time. Density of events is understood here as the number of pre-recorded sound files that are triggered within a certain time span. The amplitude of the cello signal is averaged every 50 miliseconds and whenever it goes beyond a certain threshold, the subpatch sends either one single 'bang' or several 'bangs' to trigger the correspondent number of pre-recorded sound files. The choice between one or several 'bangs' is set beforehand. When the choice is for several 'bangs', the number of 'bangs' is proportional to the mean amplitude that has just been calculated by the avg~ object. The time span between consecutive 'bangs' is defined randomly within certain limits that can be changed during the performance.

Figure 6 shows the 'DurToTrans' module that runs inside the 'TransControlMachines' subpatch. This module transforms values of duration of the cello sounds into values for controlling the speed of playback (and therefore the transposition) of the sound files that are triggered in real-time.

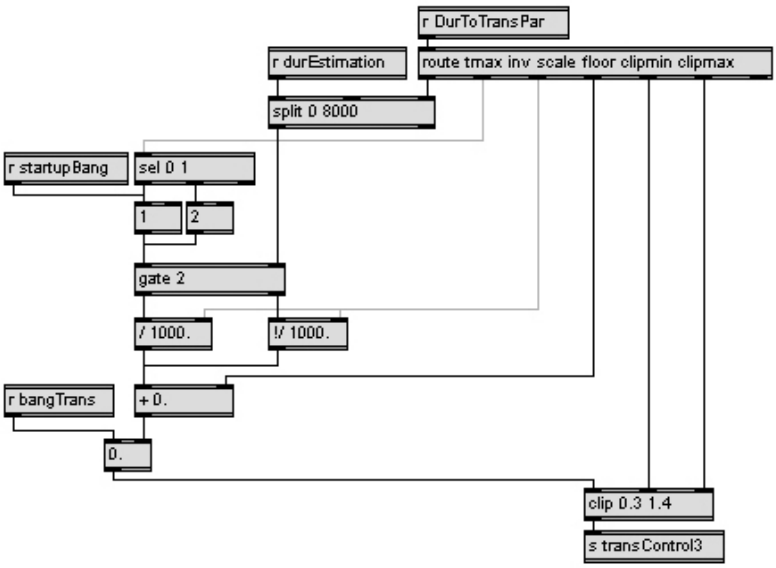



**Figure 6**

The values generated for speed of playback can be either proportional or inversely proportional to the duration of the cello sounds. The module also allows the user to define the range of possible values that it can output.

In total there are nine modules like the ones illustrate in Figure 5 and Figure 6 running inside the three subpatches responsible for generating processing parameters from data provided by the analysis parameters. The mappings between these two kinds of parameters are decided beforehand and set at specific moments during the performance of the piece through lists of instructions managed with the 'qlist' object. The mappings are then performed and sent to the 'poly~' object, which contains the main engine of the patch (see Figure 7) – the one responsible for playing back the

soundfiles using the processing parameters that were chosen.

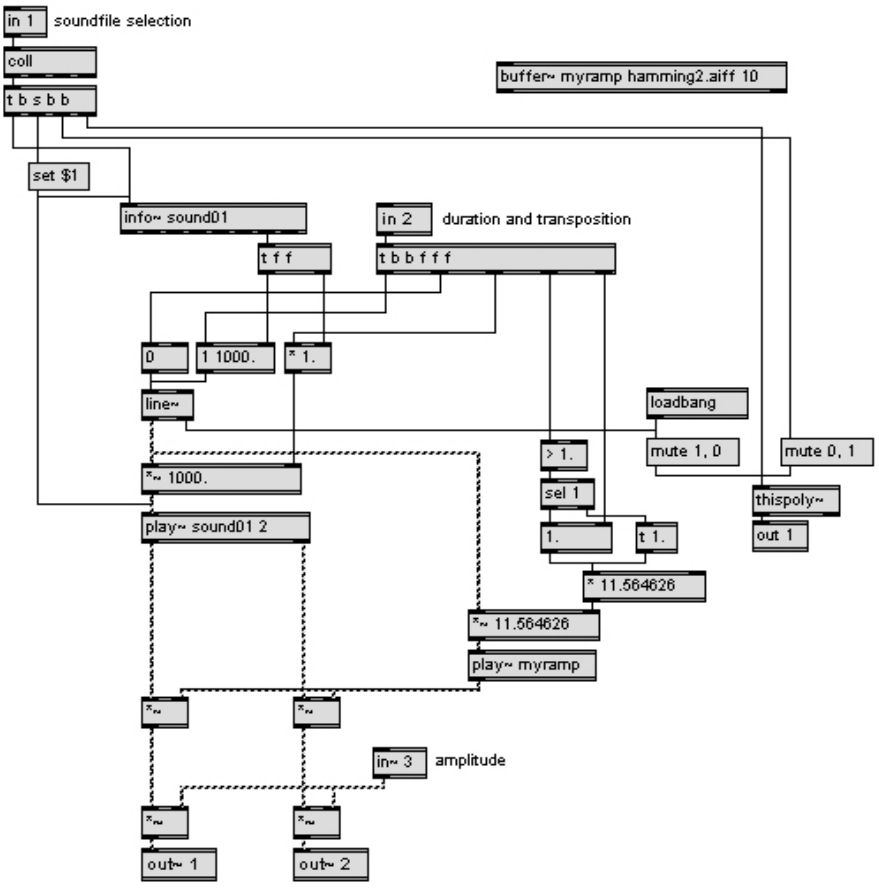



**Figure 7**

## 5. Considerations on the issue of interaction

The explanation presented so far encourages us to make some brief considerations on the issue of interaction in music. *Interferências* has a live-interactive component between instrument and computer, as the sounds of the instrument determine the production of another sonic layer by the computer. Data collected from the analysis of the instrumental signal determines when the computer has to trigger sound files. It also influences the transposition and amplitude applied to each sound file, and also the number (density) of events played within certain time spans. The precise choice of which sound files will be played back at each moment and their amplitude and transposition, however, is left for the computer to decide by means of operations that include probabilities and random procedures as explained in items 3 and 4 above. In other words, a certain degree of autonomy in the decision-making process is left for the computer.

Works composed for instrument and electroacoustic sounds on a fixed medium

do not play with that kind of interaction. The instrumental sounds do not change the audio signal of the electroacoustic part in any way. Nevertheless, there is interaction of another kind involved in such works - an interaction that does not happen in terms of audio signal, but on the musical level, as perceived relationships between sounds in the instrumental and electroacoustic parts. This kind of interaction is also meant to play an important role in *Interferências*.

These two different understandings of interaction can be illustrated by the considerations presented by Cort Lippe (2002) and Christopher Dobrian (2004), on one hand, and Flo Menezes (2002), on the other hand.

Cort Lippe (2002) mentions the existence of different kinds of interaction and highlights that interaction is not an exclusivity of live-electronics – the performance of a work for string quartet, for example, imposes the need for a constant interaction between performers. The different kinds of interaction range from the simple control of synchronism between events to the case of more complex operations, such as the realisation of decision-making processes both by computer and performer.

Likewise, Christopher Dobrian (2004) states that the control of synchronism between events by means of real-time applications does not necessarily configure interaction, but reaction. Interaction only happens, according to him, when both computer and performer have a certain level of autonomy to make decisions in real-time based on their reciprocal influences.

Flo Menezes (2002), on the other hand, believes that the most important kind of interaction between instruments and electroacoustic resources happens by means of the musical relationships established between them. For this reason, he states that, in order to be effective, the musical relationships need to be planned beforehand by the composer in the electroacoustic studio, which attests to his preference for working on compositions for instrument and electroacoustic sounds on fixed medium (although he also includes live-interaction components in some of his works).

The divergence between Lippe and Dobrian, on one hand, and Menezes, on the other hand, reveal that interaction can be understood both as the influences occurring between computer and performer in real-time and in terms of the musical relationships as (potentially) perceived by the listeners. Indeed, it is not difficult to imagine a situation in which the listener perceives the occurrence of musical interaction (relationships) when it does not actually happen from the composers' point of view.

## 6. Final considerations

This paper presented some considerations regarding sound material, compositional approach and the role of the live-interactive component used in *Interferências* (2003), for cello and computer.

One of the consequences of the approach adopted is that both computer and performer have a certain degree of freedom regarding the musical events they generate. For the performer, freedom is manifested in the process of choosing the pitches that she/he plays. The computer, in its turn, carries out some probabilistic and random processes for the selection of samples to be triggered and processed in real-time – which

results in a sonic layer that cannot be completely predicted in detail beforehand.

The investigation carried out in *Interferências* can benefit from the use of other processes, such as Markov chains, genetic algorithms and other AI tools. Also, some other kinds of interaction between instrument and computer can be investigated. It is in this direction that I intend to pursue my next live-electronics compositional explorations.

## 6. References

Barreiro, Daniel L. (2000). *Abordagens sobre o tempo na música contemporânea*. Master's Dissertation, Pontifícia Universidade Católica de São Paulo (Brazil).

_____. (2006). *Portfolio of Compositions*. PhD Thesis, The University of Birmingham (United Kingdom).

Barreiro, Daniel L. and Zampronha, Edson S. (2000a). "Um estudo de quantidades temporais a partir de uma análise qualitativa". *Revista Eletrônica de Musicologia* [online], 5(2). Available from: http://www.humanas.ufpr.br/rem/remv5.2/vol5.2/barreirozampronha/barreirozampronha.htm [Accessed November 2006].

_____. (2000b). "A seqüencialidade e as dilatações e compressões na percepção do tempo musical". *IV Fórum do Centro de Linguagem Musical da PUC-SP* [online]. Available from: http://www.pucsp.br/pos/cos/clm/forum/sumario.htm# [Accessed November 2006].

Chion, Michel (1983). *Guide des Objets Sonores*. Paris: Buchet/Chastel [excerpts of the translation by John Dack and Christine North found online on the EARS website: http://www.ears.dmu.ac.uk [Accessed November 2006].

Dobrian, Christopher (2004). "Strategies for Continuous Pitch and Amplitude Tracking in Realtime Interactive Improvisation Software". *Proceedings of the 2004 Sound and Music Computing conference (SMC04)*. France: Paris. Available from: http://music.arts.uci.edu/dobrian/PAPER_051.pdf [Accessed August 2006].

Lippe, Cort (2002). "Real-Time Interaction Among Composers, Performers, and Computer Systems". *Information Processing Society of Japan SIG Notes*, Vol. 2002(123), pp. 1-6. Available from: http://www.music.buffalo.edu/faculty/lippe/pdfs/Japan-2002.pdf [Accessed August 2006].

Mandelbrojt, Jacques (ed.) (1996). *Les unites sémiotiques temporelles*. Marseille: Laboratoire Musique et Informatique de Marseille.

Menezes, Flo (2002). "For a morphology of interaction". *Organised Sound 7(3)*. Cambridge: Cambridge University Press, pp. 305–311.

# Polyphony and Technology in Interdisciplinary Composition

**Paulo C. Chagas**

Department of Music – University of California, Riverside
900 University Ave. ARTS 138, Riverside, CA 92507, USA

paulo.chagas@ucr.edu

***Abstract***. *This paper discusses the relation between polyphony and technology from the perspective of two of my interdisciplinary compositions: "Circular Roots" (2004) – interactive performance for violin, electroacoustic sounds, video and image processing – and "Canções dos Olhos (Augenlieder)" (2005) – intermedia sound cycle for soprano, dance, video and electroacoustic sounds. It provides theoretic insights on musical gesture, interactivity, and the visibility and invisibility of the body in composition. The concept of interdisciplinary composition is also explored in relation to polyphonic subjectivity and technology. There is a need to develop a phenomenology of gestures and a systemic approach of interactivity in order to clarify the changing contexts of artistic creation involving digital technology.*

***Resumo***. *Este artigo discute a relação entre polifonia e tecnologia do ponto de vista de duas de minhas composições interdisciplinares: "Circular Roots" (2004) – performance interativa para violino, sons eletroacústicos, vídeo e processamento de imagem – e "Canções dos Olhos (Augenlieder)" (2005) – ciclo de canções intermídia para soprano, dança, vídeo e sons eletroacústicos. O artigo desenvolve noções teóricas sobre o gesto musical, interatividade e visibilidade e invisibilidade do corpo na composição. Aborda também o conceito de composição interdisciplinar em relação à subjetividade polifônica. O autor defende a necessidade de desenvolver uma fenomenologia de gestos e uma abordagem sistêmica de interatividade a fim de elucidar as transformações dos contextos de criação artística envolvendo tecnologia digital.*

**Keywords**: polyphony, technology, gesture, interactivity, visibility and invisibility, digital arts, autopoiesis.

## 1. Polyphony, polyphonic subjectivity, and technology

The notion of polyphony implies the perception of multiple and simultaneous sound events organized in a system of relationships producing a temporal processing of *meaning*. As I pointed in a former essay, polyphony is a "specific mode of operation of auditory perception, which distinguishes multiple and independent events and creates a musical difference between sound an environment" (Chagas 2005). This point of view

emphasizes polyphony as an experience of embodiment, which is triggered by sound phenomena. However, experiences of embodiment involving perception of simultaneous events are not restrained to the realm of sound. Rather they involve the structural coupling of different domains of experience that contribute to create the notion of temporality. The neurobiologist Francisco Varela describes time-consciousness as the articulation of different levels of temporality. According to Varela's enactive point of view, "any mental act is characterized by the concurrent participation of several functionally distinct and topographically distributed regions of the brain and their sensorimotor embodiment" (1999: 272).

The constructivist and psychoanalytic philosophy of Deleuze and Guattari closely relates polyphony to the ideas of *plurality* and *heterogeneity*. In his last book, *Chaosmose* (1992), Guattari develops the notion of "polyphonic subjectivity", which transverses the material and virtual universes of capitalistic society and operates by means of "technological machines of information and communication" (1992: 15). Guattari proposes a polyphonic analysis of subjectivity based on the "ethic-aesthetic paradigm" that is conceived as an alternative to scientific and philosophical models. He defines the polyphonic and heterogenic complexes of subjectivity as "machinic assemblages". His notion of "machinic" embraces both technological and abstract machines, such as social bodies, scientific discourses, cultural formations, desires, collective behaviors, and so forth. This "machinic" also extends the cybernetic concept of autopoiesis proposed by Maturana and Varela (1980) to the social domain, but through a different perspective than Luhmann's theory of social systems (Luhmann 1984; 1997). Instead of focusing on operationally closed systems, Guattari's "machinic" emphasizes the relations of alterity between collective entities. As subjectivity is not restricted to human consciousness but is also embodied in technology, the machinic assemblages of technical machines and human beings become an autopoietic character. The "machines of subjectivity" shape the autopoiesis of the post-human society in which human beings interact with technology.

Polyphonic composition conveys the idea of music consisting on multiple parts played by instruments and/or voices. *Interdisciplinary composition* extends musical polyphony to the dimension of the subjectivity that is related to and shaped by our artistic experience with technology. It embraces both the multiplicity of elements, layers and contexts, and the multiple kinds of interactions that are generated in the relationship between human and not-human bodies. In this sense, the idea of interdisciplinary composition emerges in the convergence of two tendencies: (1) the *process of differentiation* of the art system in which forms become media for the development of new forms and domains of experience (Luhmann 2000), and (2) the *autopoietic dimension of the machinic*, i.e. the experimental field of the arts dealing with the technological apparatus.

As pointed by Luhmann, art has the capability of integrating "the most heterogeneous modes of operation into an autopoietic functional nexus" (2000: 178). Using a variety of material as a basis (visual, textual and sound elements), the art system has therefore the capacity of integrating new elements produced by the technology. Electroacoustic music, for example, combines virtually all kinds of sounds, including electronic and concrete sounds, noise, sound of language, and vocal and instrumental music. Technology recycles the previous sound material and creates new meanings. The aesthetics of the "machinic" explores the parameters and functions of

machines, such as automation, interface, feedback and control of processes that can be connected with or independent from the human body. The creative process involving the interaction between human and technological processes brings awareness of the changing "relation of human subjectivity to its environment" (Hayles 1999: 290).

In the following sections, I will describe some aspects of the works *Circular Roots* and *Canções dos Olhos (Augenlieder)*. Both pieces reflect on the transformation of human existence in a heterogeneous environment and the interaction between human and technology in the artistic process. They represent different approaches of interdisciplinary composition concerning the connections between sound, image and body and the strategies for exploring the relationship between polyphony and technology.

## 2. *Circular Roots*: visible and invisible gesture

*Circular Roots* (2004) is an interdisciplinary sound and image composition for violin, electroacoustic sounds, video, and real time image processing. It uses gestural control by the violin player to process digital images projected on a screen. The violinist synchronizes his/her performance with the pre-produced electroacoustic sounds that are projected in the room as surround 5.1; there are no live-electronics and no live-processing of the violin sound itself. The piece was commissioned by the Centre de Recherches et Formation Musicales de Wallonie (CRFMW) and premiered at the Festival Ars Musica in Brussels (March 16, 2004). The work was developed in close collaboration with Luiz Carlos Joels, a Brazilian environment engineer and video artist, the violinist Izumi Okubo, and computer scientist Patrick Delges, who created the interactive design and imagine processing.

A variety of literary, visual, musical and sound materials formed the basis of my composition, including:

(1) The short story *Circular Ruins* by Jorge Luis Borges (1970). It tells the story of a man who came to the ruins of a temple, which had been devoured by ancient fires. He is possessed by the idea of dreaming a man, who should exist as a real person. The Fire helps him to accomplish his task and his son is born as a person that all creatures, except the Fire itself and the dreamer, would believe to be a man of flesh and blood. But later the man understood that he as well was an illusion, that someone else was dreaming him.

(2) Video images by Luiz Carlos Joels documenting the destruction of the Brazilian Amazon rain forest and the transformation of the natural environment. The video shows footage of the life in the Amazon region, such as the manufacture of manioc flour by "caboclos" (habitants of the countryside), a purification ritual held by a Brazilian Indian, and boats anchored in the harbor of Manaus (capital of the Amazon state).

(3) Recording of the virtuosic passage of the violin 1 part of Beethoven's string quartet op. 74, first movement, measures 221-246.

(4) Samples of my voice (whispers, laughs, screams, etc.) and concrete sounds recorded on the site of the former coal mine of Göttelborn, Germany.

*Circular Roots* generates polyphony by exploring the relation between instrumental gestures – produced by the human body manipulating the gestural

controller – and technological gestures – generated by the computer interface and the mapping strategies adopted by the composition. Sensors are attached to the violin bow and the body of the violinist capturing instrumental gestures, which are used to control image processing. The sensor date is interpreted as dynamic gestures that express the changing of some quality over time. The systematic use of feedback and recursivity for sound and image shaping emphasizes the *gesture of circularity* as the main idea of the composition.

*Circular Roots* shows how the use of technology contributes to the emergence of new interdisciplinary artistic situations. The starting point of my investigation is the role of gesture in musical understanding. According to Wittgenstein (1958; 1980; 2001) and Cavell (2000), gestures can be defined as expressions of instances and directions of projection of understanding (including musical and artistic understanding). Wittgenstein argues that when somebody tries to understand and explain a musical phrase, "sometimes the simplest explanation is a gesture" (1980: 69e). Gesture makes visible the interiority of musical understanding. But he also doubts that the understanding of a gesture can function as an explanation of musical understanding, for that gestures – as musical elements – can be re-interpreted in different contexts. [1]

In his phenomenology of gestures, Flusser (1994) gives an original insight on the gesture of listening, which he considers as a matrix of musical gestures. The gesture of listening is not a movement but a particular position of the body. The representation of this posture can be traced on the medieval iconography of Mary's conception. Mary conceives by listening. According to Flusser, listening to music "is the gesture of fertilization through the word (logos)" (1994: 151). This point of view is not new. As pointed by Eco (1997), there is no medieval author that does not refers to the transcendental property of music to convey the meaning of a invisible (divine) order, making visible things that are beyond our comprehension. This idea is also related to the theme of a "polyphonic order of the universe", which is the paradigm of medieval beauty (1997: 39). The transformation from invisible to visible accomplished by music is a process of embodiment. Therefore, the gesture of listening to music expresses the transformation of both "body in music and music in body" (Flusser 1994: 155). It is a gesture of a musical embodiment that puts the body in vibration and makes it function as a receptor and transmitter of information. Further, Flusser affirms the gesture of listening expresses the possibility of overcoming the "separation between man and world" (1994: 158). In this sense, music acts as interface in both existential and biological domains.

But the question here is how to approach musical gestures in the context of electronic and digital media and interfaces. Although the study of gestures became a focus of computer music research, performance and composition (see for example Wanderley and Depalle 2004; Miranda and Wanderley 2006), there is still a need to develop a more substantial and less mechanical theory of gestures and digital music instruments. I claim that is necessary to overcome the limitation of the structuralistic conception of gestures, which establishes a separation between information and meaning, and move towards cognitive and phenomenological approaches of gestures as a process involving "multiple levels of interconnected, sensorimotor activity" (Varela, Thompson and Rosh 1991: 206).

Such a phenomenological approach has to take in account the interplay between different kinds of gestures in musical performance. Traditional musical sounds are produced by analog gestures – vocal and instrumental. Vocal gestures are the closest related to the body, although this relationship can create all sorts of transgressions of the "natural" characteristics of the body. Instrumental sounds require a synchronized action between bodies and objects: the sound symbolizes the projection of a physical, muscular and intellectual effort of the vibrating object. Instrumental gestures make this projection visible.

Gestures in electroacoustic music are shaped both by traditional sonic model – such as vocal, instrumental and concrete sounds – and technological apparatuses. However, the meaning of the electroacoustic gesture is not conveyed by an external reference – for example analogies to known sounds – but is closed related to the apparatus that produces it. In these sense, electroacoustic music create gestures that are decoded as expressions of *programs*. Programs, according to Flusser (2000) are abstractions of concepts (scientific knowledge), which are ritualized in the post-historical society as "models." They replace the "myths" of the historical society based on alphanumeric codes. Programs stand for the communication that is made available by the technical apparatuses (computers, cameras, synthesizers, iPods, etc). Apparatuses are intelligent tools that change the meaning of the world. Musicians use them to enhance the musical possibilities through automation, simulation, sound synthesis, sound processing, information processing, etc.

In *Circular Roots*, the research of the musical gesture is anchored in the broader research on interactivity that explores sensor technology on musical performance. The violinist uses two sensors in the performance: an accelerometer attached to her bow and a flexometer attached to her skin in the inside angle of the elbow articulation. The accelerometer gives two values measuring the changing of the speed of the bow in both the X axis (horizontal) and the Y axis (vertical). As the bow is drawn across the string of a violin, the string vibrates back and forth smoothly in a saw-tooth-like motion. The displacement of the bow has to be as continuous as possible in order to produce a "sonorous", high quality vibration. The changes of velocity (speed) occur basically when the bow moves in the opposite direction in the X axis or when it moves toward a different string in the Y axis. But acceleration and retardation of bow velocity can also be related to the changing of specific sound qualities, such as loudness. The transition between loud and soft sounds requires both a changing of bow pressure and motion. The flexometer measures the gestures of expansion and contraction of the right arm, which holds the bow, at the elbow articulation. In opposition to the accelerometer, this sensor gives a much smoother curve of values over time because the violinist tries to keep the arm movement as continuous as possible in both directions.

The sensors capture all gestures produced by the violinist during the performance, including those that are not immediately translated into sound. An example of one gesture would be moving the arm to put the bow at the right position before starting a strong attack. This gesture and many others are a significant aspect of the performance and play a crucial role in musical understanding. Sensor data acquired by the gestures are mapped to the visual effects, which transform the video. MAX/Jitter (cycling74.com) software is used for image processing. The audiovisual interactive composition explores the patterns emerging from the combination of different layers of gestures relating the body to sound, music and visual information.

Thus, *Circular Roots* develops a process of coding and decoding gestures in the three levels of the composition:

(1) In the music for solo violin by applying fractals transformations to the Beethoven excerpt. I used the library Chaos by Mikhail Malt (1994) on the Patchwork programming environment. The fractals expand the harmonic, periodic progression of Beethoven's music.

(2) In the electronic music by creating an environment of concrete sounds without any reference to the musical and visual elements.

(3) In video composition by processing the images through the mapping of the gestures of the performance to the visual effects. Image processing algorithms include positive and negative feedback, delay, zoom, superposition of two layers of images, etc.

## 3. *Canções dos Olhos (Augenlieder)*: visible and invisible body, interactivity

The visibility and invisibility of the body and the interactivity on artistic creation are the central issues of *Canções dos Olhos (Augenlieder),* a composition for soprano, electronic music (surround 5.1), dance and digital image created in collaboration with the choreographer and media artist Johannes Birringer and the dancer Veronica Endo. The work exists in two versions: (1) the live performance version for soprano, dancer, electronic music and video projection; (2) the DVD version featuring the electronic music and the video dance. [2]

*Canções dos Olhos (Augenlieder)* was developed in a period of only two weeks (July 18-30, 2005) in the "Interaktionslabor", an international workshop founded and directed by Birringer in 2003 on the site of the former coal mine of Göttelborn, Saarland, Germany. The mining activity was suspended in the late 1990s and threw the region into a crisis of chronic unemployment. Several traces of the past remained more or less intact in the mine landscape. You can still see the imposing wind tower, which measures 74,2 meter and is the highest in Europe, the huge machinery spreading among several facilities, the large network of belt systems that served to transport the coal though the several processing unities, the heavy and dirty machinery rusting inside and outside, the workrooms, labs, hallways and other spaces still filled with equipment belonging to the miners – tools, reports, newspapers, calendars, pinup girls, photos, etc. Traces and inscriptions of this history appear in the architecture and physical environment. There are signs of human presence, as if the workers had left the place in a hurry fleeing from a sudden catastrophe. The abandoned and jeopardized infrastructure of the mine outlasts as a memory of the vanishing industrial society.

After the collapse of the mining economy, the local government set up an initiative for recycling the industrial landscape. The project called "future cité", aims to create a post-industrial living environment by attracting high-tech companies to the site of the former mine. Organized as a network of individuals and machines producing and exchanging information, this new environment represents the dream of a telematic society as expressed by Flusser (1985). The Interaktionslabor emerged inside this dream as a self-organizing "laboratory for interactive media, sound, design, digital video, telecommunications and performance" (Birringer 2005, 2005a). Since 2003, Birringer has promoted an annual summer workshop inviting groups of artists, scientists, and engineers from different parts of the world. Participants live for two weeks in the region (since 2005 at the site of the mine) and work on individual and collaborative projects in

which digital media and interactive performance are coupled with specific qualities of the physical environment. Originally, the events and works produced in the Interaktionslabor hoped to provide the structures – the medium/form relations in the sense of Luhmann (2000) – for observing the conversion of the post-industrial society into the utopia of the information society. As this process started to take place, as seen in a new photovoltaic plant which occupies a large space of the former mine, the dynamic of the Interaktionslabor has shifted its focal point. The sounds, images and other digital objects collected in the environment become less visible and sustainability emerges as a focus of the research and artistic projects.

*Canções dos Olhos (Augenlieder)* is inspired by the novel *Blindness* (1995; original title: *Ensaio sobre a Cegueira*) by the Portuguese author and Novel Prize winner José Saramago. In this story, Saramago has created a contemporary city where everyone loses their sight. Blindness spreads rapidly like an epidemic and provokes social collapse. Life is reduced to the basic instinct of survival; despair prevails. The author's literary metaphor of blindness points to the vulnerability of a society on the edge of chaos with no guarantees of stability. I had previously explored Saramago's novel in a former project for the Interaktionslabor 2004 – *Blind City* – a model for an interactive opera-installation that "focused on the haptic and the auditory, seeking to displace proprioception from vision, [to] make us 'see' without seeing" (Birringer 2005a).

When I conceived *Canções dos Olhos (Augenlieder)* for the Interaktionslabor 2005, I revised Saramago's narrative, focusing on the operational distinction between visibility and invisibility. Inspired by Schubert's *Winterreise*, I imagined a cycle of "intermedia songs" that explored the relations between sound, image and dance in the unique environment of the mine. [3] The song cycle focuses on the *Blindness'* main character, the doctor's wife – performed by the dancer – the only person that apparently can see in the virtual city where everyone else has gone blind. Her story is not told as a linear narration, but as an invisible layer of fiction that "actively probes the spaces between the different media" (Higgins 2002: 91).

The music of *Canções dos Olhos (Augenlieder)* was composed in the Max/MSP programming environment using basically two different kinds of samples: (1) recording of a solo soprano voice singing four songs using excerpted passages from Saramago's *Blindness*; (2) recording of my voice reading a cycle of five poems that I wrote for the chamber music composition *Canções dos Olhos* (2004, for mezzo-soprano, cello and piano). The sample material is processed through granular synthesis, using the Max/MSP external objects and abstractions developed by Nathan Wolek (2002). They offer a large variety of control of grain parameters and can be easily integrated in the patch structure of Max/MSP. Granular synthesis plays a significant role in the composition and shapes the overall texture of the electronic music.

The piece has five sections organized as sub-patches of Max/MSP. Each sub-patch contains a certain number of controls (between 8-16) for manipulating in real time the parameters of granular synthesis and also other secondary effects. I used a hardware MIDI-controller (Behringer BDF2000) and assigned the controls to faders and knobs. In the live-performance version of the *Canções dos Olhos (Augenlieder)*, the electronic music is produced in real time and the voice of the soprano can be also processed with Max/MSP, although this possibility was not explored when the work premiered on July

30, 2005 in Göttelborn. The DVD presents a version of the electronic music mixed with the recorded voice of the soprano.

The aesthetics of *Canções dos Olhos (Augenlieder)* reflects on *interactivity* and *collaboration* as distinguishing features of the interdisciplinary artwork involving technology. As I said, the idea of "intermedia song", which undergoes the conception of the piece, proposes an interdisciplinary form characterized by the connection between different domains of perception and experience – electronic music, digital image, body. The questions here are: (1) how it is possible to observe the unity of the interdisciplinary form; (2) which role interactivity plays in this process? As pointed by Luhmann (2000: 54-101), the observing operation that realizes the unity of the artwork is a distinction and indication that generates differences in both levels of first- and second-order observations. First-order observation distinguishes and indicates the arrangements of materials and elements assembled by the artwork itself; in other words, it observers *what* is the artwork. Second-order observation refers to further distinctions and observations; in other words, it observers *how* other observes the artwork. The concept of "intermedia song" embraces both kinds of observations: the arrangements of heterogeneous media that can be assembled by and observed in the artwork, and also the dynamic of observations – personal relationships, decisions, etc. – occurring in the creation process.

From the point of view of technology, the intermedia approach of *Canções dos Olhos* challenges the current understanding of "interactivity." In opposition to the discourse of interactivity that glorifies the connection between bodies and digital interfaces, I define interactivity as the embodiment of the collaborative experience that materializes the creation process in the *form* of the work itself. Interactivity is a being-in-the-world, not an ensemble of devices and instructions. Following the theory of autopoietic systems, interactivity cannot exist between human beings and machines. They operate in different living domains, which are operationally closed to each other. The artwork accomplishes the structural coupling between those domains. By observing technology, art deliberately chooses a negative approach in order to observe what is impossible to be observed. As Luhmann claims, the negative version of which is habitually performed turns into a figure of reflection. "Blindness – not seeing – becomes the condition of possibility for seeing" (2000:83).

In the beginning of the 21st century, interactivity emerges as a model of dialogue communication in a broader sense. The paradigm of interactivity, according to Flusser is chamber music (1985: 173-181), which anticipates the model of dialogue of the telematic society. [4] The dialogue characterizes the exchange of information between man and machine in a network structure. Contemporary art deals with new perceptions, situations, and experiences resulting of the hybridization of physical and virtual spaces. This is actually not new. For example, the adoption of perspective in the early Renaissance permitted the experimentation with the "difference between reality and appearance" (Luhmann 2000: 85). However, in the post-human society, interactivity becomes a "politic" dimension because of its new role in technology. Are the network structures devoted to improve the dialogue or to reinforce hierarchical structures? This question becomes pertinent when we observe the existing uses of technology by society, particularly the use of digital objects in art. As personal computers, digital recorders, sensors, and all kinds of digital interfaces become available to musicians, dancers, visual, digital and media artists, etc., the art system assimilates these elements in its

structures as recursive operations. We observe people making sound and images with computer, dancing for cameras, tracking data with sensors and playing with interfaces. However, what we usually see is either the machine dominating the human being or the human being using the machine as a slave for her/his purpose. In fact, we tend to reproduce in our relationship with technology the same patterns of oppression and exploitation that inherently drive capitalist and imperialist systems.

The discourse of interactivity reveals the paradox of art: the system produces its own references by distinguishing between perception and communication and excludes the physical world. It doesn't matter how that particular sound in that particular music is generated, if by simple clapping of hands or by complex sound synthesis algorithms. The "materiality" of computers and other digital objects belong to the environment and can never become a "component of the system's operational sequences" (Luhmann 2000:99). Interactivity occurs only if it is embedded in the *forms* of the art system itself, distinguished either as first- or second-order observations. Solely as such it becomes an operation of the system and can be reproduced as communication. This point of view may be interpreted as a criticism of technology in the artistic domain. But it is not. In fact, such a conception of interactivity "resolutely embraces a perspective interested in 'how' things emerge, rather in 'what' they are" (Luhmann 2000:100). And this is quite optimistic.

## 4. Conclusion

In this article, I showed how the idea of interdisciplinarity, conceived as an expansion of polyphony to the domain of intermedia composition, is developed in the pieces *Circular Roots* (2004) and *Canções dos Olhos (Augenlieder)* (2005). Both pieces explore topics of literary texts (Borges and Saramago) and were created in collaboration with other artists and engineers.

In the first section, I introduce the notion of "polyphonic subjectivity" and the "aesthetics of the machinic" as categories for understanding the changing relation of human subjectivity to the environment.

*Circular Roots* focuses on the visibility and invisibility of the musical gesture in an interactive live-performance composition. Physical gestures of the violinist are captured with different sensors and mapped to parameters of digital image processing algorithms. The relationship between physical gestures and mapping strategies generate different meanings for the intermedia composition. However, I argue that is necessary to adopt a more phenomenological approach of gestures in order to develop an extensive theory of the emergence and generalization of musical gestures (c.f. Hatten 2004). In other words, new musical instruments and interfaces need a new theory of musical gestures.

*Canções dos Olhos (Augenlieder)* focus on the visibility and invisibility of the body in a composition for electronic music, digital image and dance. It explores the topic of invisibility as a metaphor of the blindness generated by the technical apparatuses in our society. Related to the body, the question of interactivity occupies a central place in the aesthetics of this work. I claim that "interactivity" should not be restrained to the human-machine paradigm – a problem that can be observed in the way technology is used by many artists today – but rather should focus on the strategies necessary to make visible the multiple relationships that shape the creative process.

## Notes

[1] Wittgenstein compares musical chords with facial gestures as an example of how the context determines the meaning. He says, for example: "The reinterpretation of a facial expression can be compared to the reinterpretation of a chord in music, we hear it as a modulation first into this, then into that key" (1958: 144).

[2] A detailed description of *Canções dos Olhos (Augenlieder)* is found in Chagas (2006c). The present section elaborates and expands some topics of the former article.

[3] The term "intermedia" was first used by Dick Higgins in 1966, in the context of Fluxus. Higgins allegedly borrowed the idea from Samuel Coleridge (1812). See Higgins (2002: 91-93).

[4] For Flusser, the telematic society is one in which people are devoted to the creative exchange of information in a non-hierarchical network (1985: 177). According to Flusser, the universe of music is made of calculations and computations, which is the essence of digital thinking (1985: 179).

## References

Ascott, Roy. (2003). Telematic Embrace: Visionary Theories of Art, Technology, and Consciousness. Berkeley, CA: University of California Press.

Birringer, Johannes. (2005). Interaktionslabor. Retrieved June 30 2007, from http://www.interaktionslabor.de/

Birringer, Johannes. (2005a). FutureHouse, Blind Mine. Retrieved June 30 2007, from http://people.brunel.ac.uk/dap/futurehouse.html

Birringer, Johannes. (2005b). Performing Art Performing Science. Retrieved June 30 2007, from http://people.brunel.ac.uk/dap/paps.html

Borges, Jorge Luis. (1970). Fictions. London: Penguin.

Cavell, Stanley. (2000). Excursus on Wittgenstein's Vision of Language. In: The New Wittgenstein. Alice Crary and Rupert Read (eds.). 21–37. London; New York: Routledge.

Chagas, Paulo C. (2003). Polyphony, Form, Art Sonore. Ph.D. Dissertation, Université de Liège [not published].

Chagas, Paulo C. (2005). "Music and embodiment: a critical approach of the theory of autopoiesis." Trans – Revista Transcultural de Musica: 9. URL: http://www.sibetrans.com/trans/trans9/chagas.htm

Chagas, Paulo C. (2006a). "Virtuality and Metadesign: Sound Art in the Age of Connectivity". In: Erkki Pekkilä et al (Ed.) Music, Meaning, and Media. Acta Semiotica Fennica XXV (Approaches to Musical Semiotics 11) (pp. 137-53). Helsinki: International Semiotics Institute.

Chagas, Paulo C. (2006b). "Game and dialogue: Composing with machinery". In: Eero Tarasti (Ed.) Music and the Arts. Acta Semiotica Fannica XXIII (Approaches to Musical Semiotics 10) (pp. 157-70). Helsinki: International Semiotics Institute.

Chagas, Paulo C. (2006c). "The Blindness Paradigm: the Invisibility and Visibility of the Body." Contemporary Music Review 25.1/2: 119-30.

Deleuze, Gilles & Guattari, Félix. (1997). A Thousand Plateaus: Capitalism and Schizophrenia. Translated by Brian Massumi. Minneapolis: University of Minnesota Press.

Eco, Umberto. (1997). Art et Beauté dans l'Esthétique Médiévale. Paris: Bernard Grasset.

Flusser, Vilém. (1985). Ins Universum der technischen Bilder. Göttingen: European Photography.

Flusser, Vilém. (1994). Geste: Versuch einer Phänomenologie. Frankfurt am Main: Fischer.

Flusser, Vilém. (2000). Towards a Philosophy of Photography. Translated by Anthony Mathews. London: Reaktion Books.

Guattari, Félix. (1992). Chaosmose. Paris: Galilée.

Hansen, Mark B. N. (2004). New Philosophy for New Media. Cambridge, MA: The MIT Press.

Hatten, Robert S. (2004) Interpreting Musical Gestures, Topics, and Tropes. Mozart, Beethoven, Schubert. Bloomington, IN: Indiana University Press.

Hayles, N. Katherine. (1999). How We Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics. Chicago: The University of Chicago Press.

Higgins, Hannah. (2002). Fluxus Experience. Berkeley, CA: University of California Press.

Luhmann, Niklas. (1984). Soziale Systeme. Grundriß einer allgemeinen Theorie. Frankfurt am Main: Suhrkamp.

Luhmann, Niklas. (1990). The Autopoiesis of Social Systems. In N. Luhmann. Essays on Self-Reference (pp. 1-20). New York: Columbia University Press.

Luhmann, Niklas. (1995). Social Systems. Translated by John Bednarz, Jr., with Dirk Baeker. Stanford, CA: University of Stanford Press.

Luhmann, Niklas. (1997). Die Gesellschaft der Gesellschaft. Frankfurt am Main: Suhrkamp.

Luhmann, Niklas. (2000). Art as Social System. Translated by Eva M. Knodt. Stanford, CA: University of Stanford Press.

Malt, Mikhail. (1994). Patchwork Librairie Chaos. Référence. Paris: Ircam.

Maturana, Humberto R. & Varela, Francisco J. (1980). Autopoiesis: The Realization of Living. Boston Studies in the Philosophy of Science, vol. 42. Boston: D. Reidel.

Miranda, Eduardo R. and Wanderley, Marcelo M. (2006). New Digital Musical Instruments: Control and Interaction Beyond the Keyboard. Middleton, WI: A-R Editions.

Saramago, José. (1997). Blindness. Translated by Giovanni Pontiero. San Diego: Harcourt.

Varela, F. J. & Thompson, E., Rosch, E. (1991). The Embodied Mind. Cambridge, MA: The MIT Press.

Varela, Franciso J. (1999). The Specious Present: A Neurophenomenology of Time Consciousness. In J. Petitot & F. J. Varela, G. Pachoud, Jean-Michel Roy (Eds.) Naturalizing Phenomenology: Issues in Contemporary Phenomenology and Cognitive Science (pp. 265-314). Stanford, CA: Stanford University Press.

Wanderley, M. and Depalle, P. (2004). Gestural Control of Sound Synthesis. Proceedings of the IEEE, vol. 92, no. 4.

Wittgenstein, Ludwig. (1958). Philosophical Investigations. G. E. M. Anscombe, R. Rhees (eds.). Oxford: Basil Blackwell.

Wittgenstein, Ludwig. (1980). Culture and Value. Translated by Peter Winch. Chicago: The University of Chicago Press.

Wittgenstein, Ludwig. (2001). Tractatus Logico-Philosophicus [English translation by D. F. Pears and B. F. McGuinness]. London; New York: Routledge.

Wolek, Nathan. (2000). A Granular Toolkit for Cycling74's Max/MSP. Retrieved June 30 2007, from: http://www.nathanwolek.com/nathanwolek/papers/gtk_SEAMUS2002.pdf

# Technical Papers

11º Simpósio Brasileiro de Computação Musical

SBCM

11th Brazilian Symposium on Computer Music

# Sistema de Marca d'Água Digital no Domínio do Tempo para Sinais de Áudio*

**Karoline M. O. Nunes, Marcelo S. Pinho**

Divisão de Engenharia Eletrônica
Instituto Tecnológico de Aeronáutica
Praça Marechal Eduardo Gomes, 50 – 12.228-900 São José dos Campos, SP

karolm@ita.br, mpinho@ieee.org

***Abstract.*** *In this work, it is presented a variant of the algorithm introduced by Lie and Chang to watermark an audio signal. The original algorithm is based on amplitude modification of two frames of a group of samples (GOS) which is divided in three frames. Based on the absolute mean of samples of the three frames, it is established the rules to embed one bit in this GOS. The sample amplitudes of two frames are modified in accordance to the bit which should be embedded. This paper shows an alternative procedure, working in the three GOS frames, to modify the sample amplitudes. The new procedure reduces the distortion produced by the watermark. The two schemes are tested in a group of 5 songs and the results show an improvement around* $1.5dB$ *in the mean square error.*

***Resumo.*** *Nesse trabalho, é apresentada uma variação do algoritmo proposto por Lie e Chang para introduzir uma marca d'água em um sinal de áudio. O algoritmo original é baseado na modificação das amplitudes de dois quadros de um grupo de amostras (GOS - Group Of Samples) que contém três quadros. Baseado na média dos valores absolutos das amostras, são estabelecidas regras par embutir um bit em um GOS. As amplitudes das amostras de dois quadros são alteradas de acordo com o bit que deve ser embutido. Esse artigo propõe um método alternativo, operando nos três quadros do GOS, para modificar as amplitudes das amostras. Esta nova proposta reduz a distorção produzida pela marca d'água. Os dois métodos são testados em um conjunto com 5 músicas e os resultados mostram uma melhora de aproximadamente* $1.5dB$ *no erro médio quadrático.*

## 1. Introdução

Com o crescimento do número de usuários das redes públicas de telecomunicações e com o avanço da tecnologia digital, surgiram diversos sistemas de distribuição de sinais multimídia através destas redes. Estes sistemas de distribuição estão gerando uma verdadeira revolução no mercado. Devido a facilidade de se copiar e alterar um sinal digital, junto com os novos sistemas de distribuição, surge uma série de questões envolvendo a segurança da informação. A técnica de inserir informação adicional em sinais digitais é uma ferramenta útil para combater os problemas relativos à segurança da informação. De

*Trabalho desenvolvido com o apoio da CAPES

fato, a inserção de uma assinatura digital, de uma marca d'água frágil ou de uma marca d'água robusta pode auxiliar na solução de questões sobre autenticidade, integridade e propriedade dos sinais. Por esta razão, nos últimos anos, muita atenção foi dada a esta área [Petitcolas et. al. 1999].

No caso em particular da distribuição de sinais de áudio, a questão mais discutida é a preservação dos direitos autorais. De fato, estas discussões motivaram a implementação de vários mecanismos contra a pirataria, tais como o sistema que impede a cópia de uma mídia e o sistema que impede a reprodução de um sinal em dispositivos não autorizados. Ainda por conta do problema de cópias não autorizadas de sinais de áudio, recentemente, muito esforço foi realizado em estudos de técnicas de marca d'água robusta para áudio digital [Swanson et. al. 1998, Kirovski and Malvar 2003, Lie and Chang 2006].

Em [Lie and Chang 2006] foi apresentado um esquema robusto para embutir uma marca d'água em sinais de áudio, através da alteração das amplitudes das amostras. Na proposta em questão, o sinal de áudio é dividido em grupos de amostras e cada grupo ($GOS$ - *Group Of Samples*) é subdividido em três quadros. Cada bit da marca d'água é embutido em um $GOS$, através da alteração da amplitude de dois quadros.

Este artigo propõe uma variação do esquema proposto em [Lie and Chang 2006], em que os três quadros de um $GOS$ são alterados de acordo com o bit da marca d'água. Esta nova variante permite atingir a mesma robustez do esquema original, distribuindo melhor a distorção provocada pela introdução da marca d'água. Este trabalho está dividido da seguinte forma. A segunda seção apresenta e analisa o algoritmo proposto em [Lie and Chang 2006]. A descrição da nova versão é apresentada na seção 3. Os resultados obtidos através da aplicação dos dois esquemas em sinais de áudio são mostrados na quarta seção. Fechando o artigo, a conclusão está na seção 5.

## 2. Marca d'Água Baseada na Modificação das Amplitudes das Amostras

Em [Lie and Chang 2006], foi apresentado um sistema de marca d'água digital para sinais de áudio, baseado na modificação das amplitudes das amostras. A idéia central do algoritmo de Lie e Chang ($LCA$ - *Lee Chang Algorithm*) é modificar as amplitudes de um grupo de amostras ($GOS$), de acordo com um dos bits do sinal da marca d'água. Se o bit que se deseja adicionar é igual a 0, as amplitudes são ajustadas para satisfazer uma determinada condição, que representa a introdução do bit 0. Caso contrário, as amplitudes são ajustadas para satisfazer outra condição, que representa a introdução do bit 1.

Seja $x[n]$ o sinal de áudio que será marcado e seja $b[k]$ a seqüência de bits da marca d'água. O $LCA$ divide a seqüência $x[n]$ em $GOS$'s com $\ell$ amostras cada e subdivide cada $GOS$ em três quadros, com $\ell_1, \ell_2$ e $\ell_3$ amostras respectivamente. Sendo assim, o $k^{\text{ésimo}}$ $GOS$ é composto pelas amostras $x[(k-1)\ell+1], \ldots, x[k\ell]$, e os quadros podem ser representados pelos vetores

$$\mathbf{q}_1^k = (x[(k-1)\ell+1], \ldots, x[(k-1)\ell+\ell_1]), \quad (1)$$
$$\mathbf{q}_2^k = (x[(k-1)\ell+\ell_1+1], \ldots, x[(k-1)\ell+\ell_1+\ell_2]), \quad (2)$$
$$\mathbf{q}_3^k = (x[(k-1)\ell+\ell_1+\ell_2+1], \ldots, x[k\ell]), \quad (3)$$

onde $\mathbf{q}_i^k$ representa o quadro $i$ do $k^{\text{ésimo}}$ $GOS$. Para os três quadros, as médias dos

valores absolutos são calculadas através das seguintes expressões.

$$a_1^k = \frac{1}{\ell_1} \sum_{j=(k-1)\ell+1}^{(k-1)\ell+\ell_1} |x[j]|, \quad (4)$$

$$a_2^k = \frac{1}{\ell_2} \sum_{j=(k-1)\ell+\ell_1+1}^{(k-1)\ell+\ell_1+\ell_2} |x[j]|, \quad (5)$$

$$a_3^k = \frac{1}{\ell_3} \sum_{j=(k-1)\ell+\ell_1+\ell_2+1}^{k\ell} |x[j]|. \quad (6)$$

Os valores obtidos são ordenados, dando origem às variáveis

$$a_{min}^k = \min_{i=1,2,3} \left\{a_i^k\right\}, \quad (7)$$

$$a_{med}^k = \operatorname{med}_{i=1,2,3} \left\{a_i^k\right\}, \quad (8)$$

$$a_{max}^k = \max_{i=1,2,3} \left\{a_i^k\right\}, \quad (9)$$

onde $\mathrm{med}$ representa o operador mediana. A idéia básica utilizada pelo LCA para embutir o bit $b[k]$ no $k^{\text{ésimo}}$ $GOS$, é ajustar as amplitudes de tal forma que $a_{max}^k - a_{med}^k < a_{med}^k - a_{min}^k$, se $b[k] = 0$ e $a_{max}^k - a_{med}^k > a_{med}^k - a_{min}^k$, caso contrário. No entanto, para que a marca d'água seja robusta a ataques (intencionais ou não), o algoritmo utiliza um limiar de proteção, $t_p$. Sendo assim, os valores das amplitudes são modificados para satisfazer as seguintes condições.

$$a_{max}^k - a_{med}^k + t_p < a_{med}^k - a_{min}^k, \text{ se } b[k] = 0, \quad (10)$$

$$a_{max}^k - a_{med}^k > a_{med}^k - a_{min}^k + t_p, \text{ se } b[k] = 1, \quad (11)$$

Após modificar as amplitudes, o $LCA$ verifica se a distorção introduzida pela adição da marca d'água não compromete a qualidade do áudio. Esta verificação é feita através do modelo psicoacústico utilizado no padrão MPEG [Noll 1997], sem considerar qualquer efeito de mascaramento. Sendo assim, para cada $GOS$, o $LCA$ realiza dois passos: (a) alteração das amplitudes e (b) verificação da qualidade do áudio marcado. Estas duas etapas são detalhadas nas próximas subseções.

### 2.1. Ajuste das Amplitudes

Com o objetivo de atender as condições apresentadas nas equações (10) e (11), o LCA altera as amplitudes de dois quadros do $GOS$, seguindo a seguinte regra.

- $b[k] = 0$,
  Se a condição em (10) já é satisfeita pelo $GOS$, nada é feito. Caso contrário, aumenta-se $a_{med}^k$ e diminui-se $a_{min}^k$.
- $b[k] = 1$,
  Se a condição em (11) já é satisfeita pelo $GOS$, nada é feito. Caso contrário, aumenta-se $a_{max}^k$ e diminui-se $a_{med}^k$.

É interessante observar que se $b[k] = 0$ e se a condição em (10) não for satisfeita, o acréscimo da mediana, $\delta_{med}$, e o decréscimo do mínimo, $\delta_{min}$ devem ser tais que

$$a^k_{max} - (a^k_{med} + \delta_{med}) + t_p < a^k_{med} + \delta_{med} - (a^k_{min} - \delta_{min}).$$

Portanto,

$$2\delta_{med} + \delta_{min} > t_p + a^k_{max} + a^k_{min} - 2a^k_{med} \tag{12}$$

Analogamente, no caso em que $b[k] = 1$,

$$2\delta_{med} + \delta_{max} > t_p - a^k_{max} - a^k_{min} + 2a^k_{med} \tag{13}$$

No $LCA$, os ajustes são realizados de tal forma que o acréscimo de um dos blocos é idêntico ao decréscimo aplicado no outro bloco. Sendo assim, se $b[k] = 0$, $\delta_{med} = \delta_{min}$ e se $b[k] = 1$, $\delta_{max} = \delta_{med}$. Portanto, as equações (12) e (13) se transformam em

$$\delta_0 = \frac{t_p + a^k_{max} + a^k_{min} - 2a^k_{med}}{3}, \tag{14}$$

$$\delta_1 = \frac{t_p - a^k_{max} - a^k_{min} + 2a^k_{med}}{3}, \tag{15}$$

onde $\delta_s$ representa o acréscimo (ou decréscimo) no caso em que $b[k] = s$, $s = 0, 1$. Para produzir um acréscimo (ou decréscimo) de $\delta_s$ em um quadro, o $LCA$ multiplica cada uma das amostras do quadro $i$ por um fator $w_s$. Sendo assim, a alteração da média dos valores absolutos é dada por

$$\begin{aligned} \delta_s &= |a^k_i - w_s a^k_i| \\ &= |1 - w_s| a^k_i, \end{aligned}$$

No caso em que a média dos valores absolutos deve ser acrescida de $\delta_s$, $w_s > 1$ e conseqüentemente,

$$w_s = 1 + \frac{\delta_s}{a^k_i}. \tag{16}$$

No caso inverso, $w_s < 1$ e portanto

$$w_s = 1 - \frac{\delta_s}{a^k_i}. \tag{17}$$

A multiplicação de dois quadros adjacentes por fatores diferentes pode produzir uma mudança abrupta no sinal de áudio, gerando um efeito de borda que pode ser notado pelo usuário. De fato, se este efeito de borda não for retirado, o usuário irá escutar *clicks* na transição de alguns quadros. Com o objetivo de retirar este efeito, o $LCA$ utiliza uma função que altera o fator ao longo do quadro, progressivamente. A Figura 1 ilustra a diferença dos fatores utilizados para ajustar os valores das amostras ao longo de um $GOS$ com 270 amostras, onde o primeiro quadro terá suas amplitudes ampliadas de 10% e o terceiro quadro terá suas amplitudes reduzidas de 15%.

Com a introdução da função que torna progressiva a alteração da amplitude de um quadro, o fator $w_s$ deixa de produzir uma modificação de $\delta_s$ na média do valor absoluto. Para resolver este problema, o $LCA$ utiliza um procedimento iterativo, aumentando (ou reduzindo) o valor de $w_s$ gradativamente até atingir a modificação, $\delta_s$ desejada.

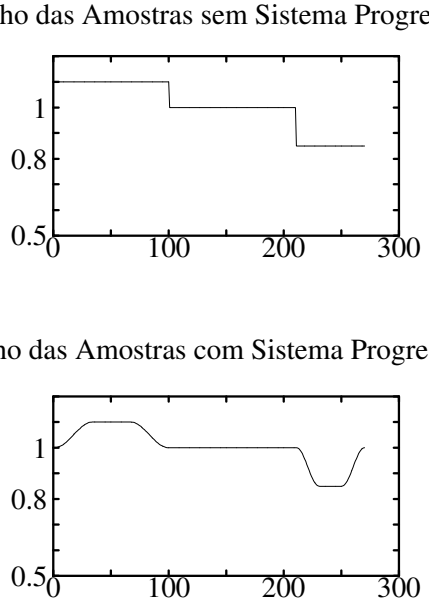



**Figura 1. Sistema Progressivo do Ajuste das Amplitudes**

### 2.2. Verificação da Qualidade

Após o ajuste das amplitudes, o algoritmo de Lie e Chang verifica se a distorção provocada por este ajuste compromete a qualidade do áudio. Esta verificação é realizada através do modelo psicoacústico utilizado no padrão MPEG, sem considerar o efeito de mascaramento. Seja $\hat{x}[(k-1)\ell+1], \ldots, \hat{x}[k\ell]$ o sinal do $k^{\text{ésimo}}$ $GOS$ após o ajuste das amplitudes. Sendo assim, o sinal erro é definido por

$$e[j] = \hat{x}[j] - x[j]. \tag{18}$$

O $LCA$ considera que a distorção introduzida pelo ajuste das amplitudes é aceitável se o sinal erro possui um espectro tal que mais de $85\%$ das magnitudes do espectro estão abaixo do modelo apresentado em [Terhardt et. al.], que representa o nível mínimo de audição, quando não existe qualquer outro som no ambiente (*threshold in quiet*). A Figura 2 ilustra a curva do limiar apresentado em [Terhardt et. al.], que pode ser obtida através da expressão

$$\text{limiar}(f) = 3.64f^{-0.8} - 6.5e^{-0.6(f-3.3)^2} + 0.001f^4 \tag{19}$$

onde $f$ representa a freqüência em $KHz$ e o limiar é dado em $dB$.

Caso mais do que $15\%$ das componentes do espectro do sinal erro estejam acima do limiar de audição estabelecido em (19), o $LCA$ reduz o limiar para a proteção contra ataques, introduzido na etapa de ajuste das amplitudes. Utilizando este limiar reduzido, o $LCA$ recalcula os ganhos da primeira etapa. Em seu trabalho original, Lie e Chang não informam qual a estratégia adotada quando não existe um limiar para proteção contra ataques tal que a condição imposta para a qualidade do áudio seja atendida.

## 3. Reduzindo a Distorção Produzida pelo LCA

O método denominado *Quantization Index Modulation* (QIM), proposto em [Chen and Wornell 2001], é um dos métodos mais populares para a inserção de

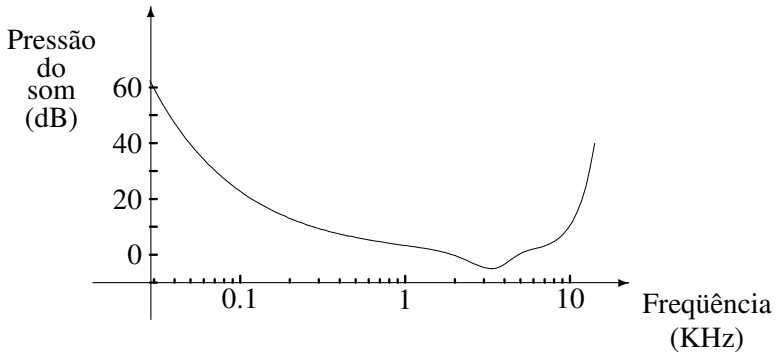



**Figura 2. Limiar de Audição no Silêncio**

informação adicional em sinais digitais. Sabendo que um sinal digital com $\ell$ amostras pertence a uma grade de pontos em $\mathbb{R}^\ell$, uma solução para inserir um bit neste sinal, é a utilização de um subconjunto de pontos da grade, $\mathcal{A}_0$, para indicar a inserção do bit 0 e a utilização de um outro subconjunto, $\mathcal{A}_1$, para indicar a inserção do bit 1. Para que o sistema consiga recuperar o bit da marca d'água é necessário que os conjuntos sejam disjuntos, i.e., $\mathcal{A}_0 \cap \mathcal{A}_1 = \emptyset$. No algoritmo de inserção da marca d'água, o sinal original é mapeado (ou quantizado) no ponto mais próximo, pertencente ao conjunto $\mathcal{A}_0$ ou $\mathcal{A}_1$, dependendo do bit a ser inserido. A Figura 3 ilustra a idéia do central do QIM, para um sinal com $\ell = 2$ amostras, onde cada amostra é representada com 3 bits. Neste caso, a grade possui 64 pontos, $2^3 \times 2^3$. Na Figura 3, o símbolo $\circ$ indica os pontos do conjunto $\mathcal{A}_0$, o símbolo $\star$ indica os pontos do conjunto $\mathcal{A}_1$, o símbolo $\times$ indica que o ponto não pertence a nenhum dos dois conjuntos e o símbolo $\odot$ representa um possível sinal em que se deseja inserir um bit da marca d'água.

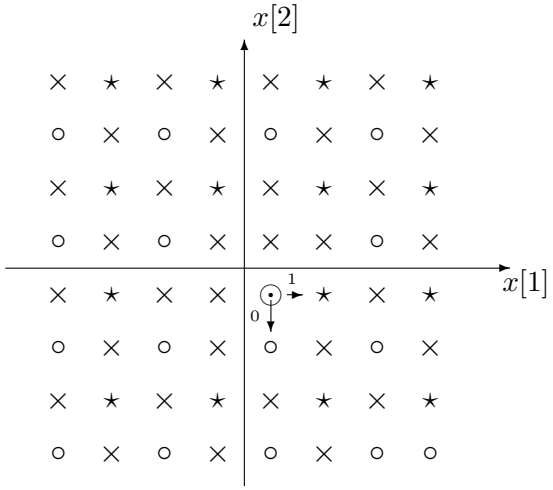



**Figura 3. QIM operando em $\mathbb{R}^2$**

De uma forma geral, é possível observar que o algoritmo de Lie e Chang também cria dois conjuntos em $\mathbb{R}^\ell$ que representam o bit 0 ou o bit 1. De fato, no $LCA$, o conjunto $\mathcal{A}_0$ é a coleção de todos os sinais que satisfazem a condição estabelecida em (10).

Da mesma forma, o conjunto $\mathcal{A}_1$ é composto por todos os sinais que satisfazem a desigualdade apresentada em (11). Na verdade, a diferença fundamental entre a idéia básica do QIM e o $LCA$ está na forma de mapear o sinal original no sinal marcado. Devido a complexidade de se buscar o ponto mais próximo quando $\ell$ cresce significativamente, o mapa utilizado no QIM é inviável de ser utilizado no $LCA$. Por esta razão, este último utiliza uma solução heurística para encontrar um ponto em $\mathcal{A}_0$ (ou em $\mathcal{A}_1$) que seja próximo do sinal original. Esta solução heurística é baseada no vetor $(a^k_{min}, a^k_{med}, a^k_{max})$. No entanto, é interessante notar que o $LCA$ não encontra nem mesmo um ponto em $\mathcal{A}_0$ (ou em $\mathcal{A}_1$) que produz a mínima distância para o vetor $(a^k_{min}, a^k_{med}, a^k_{max})$. De fato, isto é uma conseqüência de se optar por alterar apenas dois quadros do $GOS$.

Com o objetivo de reduzir a distorção provocada pela introdução da marca d'água, sem alterar o limiar de proteção a ataques, este artigo propõe uma variante do $LCA$. Esta variante, que será denominada, $vLCA$, utiliza uma heurística para mapear o sinal original no sinal em $\mathcal{A}_s$, $s = 0, 1$, tal que a distância do vetor $(a^k_{min}, a^k_{med}, a^k_{max})$ para o vetor correspondente do sinal em $\mathcal{A}_s$ é mínima. As equações (10) e (11) podem ser reescritas da seguinte forma

$$a^k_{med} - \frac{a^k_{max} + a^k_{min}}{2} > \frac{t_p}{2}, \text{ se } b[k] = 0, \tag{20}$$

$$\frac{a^k_{max} + a^k_{min}}{2} - a^k_{med} > \frac{t_p}{2}, \text{ se } b[k] = 1, \tag{21}$$

As equações acima mostram claramente que o bit a ser adicionado estabelece um relacionamento de ordem entre $a^k_{med}$ e o ponto médio entre $a^k_{min}$ e $a^k_{max}$. Sendo assim, a solução que minimiza a distância (considerando a norma $L_2$) entre o vetor $(a^k_{min}, a^k_{med}, a^k_{max})$ e o seu correspondente no sinal marcado, é deslocar (para cima ou para baixo) cada uma das componentes dos vetores de um valor constante.

É interessante observar que em alguns casos particulares, não é possível deslocar as três componentes de um mesmo valor. Isto ocorre ou porque o $a^k_{min}$ não pode ser negativo, ou porque $a^k_{max}$ não pode ultrapassar um valor máximo, ou porque um deslocamento desta forma iria inverter as posições do mínimo com a mediana ou da mediana com o máximo. Em todos estes casos, a alteração de uma ou duas componentes está limitada. A solução nestes casos é produzir a máxima alteração possível nas componentes com restrições e alterar a(s) componente(s) restante(s) de forma a atingir a condição apresentada em (20) ou em (21).

Ainda é importante observar que seguindo a heurística do $vLCA$, caso seja necessário alterar as amplidutes de um $GOS$, os três quadros do $GOS$ serão modificados.

## 4. Resultados

O $LCA$ e o $vLCA$ foram implementados e o desempenho foi medido através da aplicação dos dois métodos em um conjunto de sinais de áudio para testes. Este conjunto de teste foi composto por 5 trechos de 30 segundos de diferentes tipos de música, conforme indicado na Tabela 1.

O tamanho dos blocos utilizados foram $\ell_1 = 341$, $\ell_2 = 341$ e $\ell_3 = 342$, gerando assim grupos com 1024 amostras.

**Tabela 1. Sinais de Áudio para Teste**

| | Título | Artista | Álbum | Início do Trecho |
|---|---|---|---|---|
| audio1 | O mundo é um moinho | Cartola | Cartola | 2 min. |
| audio2 | s'Wonderful | Ray Conniff | 16 Most requested songs | 30 seg. |
| audio3 | School | Supertramp | Paris | 4 min. |
| audio4 | Think of me | Rosemary Ashe Sarah Brightman Steve Barton | The phantom of the opera | 2 min. |
| audio5 | Passe em casa | Tribalistas | Tribalistas | 30 seg. |

O limiar utilizado para combater ataques contra a marca d'água foi calculado da mesma forma como em [Lie and Chang 2006]. Inicialmente, este limiar é projetado para que a marca d'água seja recuperada corretamente sempre que o ataque provocar um desvio no vetor $(a^k_{min}, a^k_{med}, a^k_{max})$ menor ou igual a

(a) $(\epsilon a^k_{min}, -\epsilon a^k_{med}, +\epsilon a^k_{max})$, caso $b[k] = 0$,
(b) $(-\epsilon a^k_{min}, \epsilon a^k_{med}, -\epsilon a^k_{max})$, caso $b[k] = 1$,

onde $\epsilon > 0$. É interessante observar que os desvios apresentados no item (a) representam o pior caso para $b[k] = 0$, quando se limita a um desvio máximo de $\pm\epsilon(a^k_{min}, a^k_{med}, a^k_{max})$. Também é possível observar que o item (b) representa o pior caso para $b[k] = 1$. Para que os desvios apresentados não provoquem erros na marca d'água, é necessário que o limiar $t_p$ seja tal que

$$t_p = (a^k_{min} + 2a^k_{med} + a^k_{max})\epsilon \tag{22}$$

Nos testes realizados nesse trabalho, foi adotado o valor de $\epsilon = 0.01$.

Para retirar o efeito de borda que seria introduzido caso dois quadros consecutivos fossem multiplicados por ganhos diferentes, os algoritmos utilizaram a função, $f[j]$, definida a seguir, para tornar progressivo o ganho do bloco $i$ do $k^{\text{ésimo}}$ GOS.

$$f[j] = \begin{cases} 1, & \text{se } \left|j - \frac{\ell_i}{2}\right| \in \left(0, \frac{(1-\alpha)\ell_i}{2}\right); \\ \frac{1}{2}\left\{1 + cos\left[\left(j - \frac{(2-\alpha)\ell_i}{2}\right)\frac{\pi}{\alpha\ell_i}\right]\right\} & \text{se } \left|j - \frac{\ell_i}{2}\right| \in \left(\frac{(1-\alpha)\ell_i}{2}, \frac{(1+\alpha)\ell_i}{2}\right); \\ 0, & \text{caso contrário,} \end{cases} \tag{23}$$

onde $\alpha$ é uma constante dentro do intervalo $[0, 1]$. Na verdade, a função definida em (23) é muito popular na área de telecomunicações e é conhecida como função cosseno levantado. A constante $\alpha$ é usualmente chamada de fator de *roll-off* e tem como objetivo suavizar a descontinuidade de um pulso retangular. Se $\alpha = 0$, $f[j]$ é um pulso retangular. Se $\alpha = 1$, $f[j]$ é um período de uma função senoidal deslocada. Nesse trabalho, foi utilizada a função cosseno levantado, com fator de *roll-off* igual a 0.1.

Nas duas variantes, foi adotado o critério de qualidade baseado no limiar de audição em silêncio. Sendo assim, quando a solução encontrada pelo ajuste das amplitudes não satisfaz o critério de qualidade, o limiar $t_p$ é reduzido e o algoritmo recalcula os ajustes das amplitudes.

Como nas duas variantes, o limiar de proteção foi o mesmo, os dois algoritmos produzem sinais marcados com a mesma robustez. Sendo assim, a comparação entre os dois sistemas deve focar a distorção produzida por cada um deles. A Tabela 2 apresenta o erro médio quadrático entre o sinal original e o sinal marcado para o $LCA$ e para o $vLCA$. Através da Tabela 2, é possível observar que na média, o $vLCA$ produz um ganho de 1.5 dB ($10\log_{10}\left(\frac{7.7e-5}{5.4e-5}\right)$)sobre o $LCA$, para o grupo de teste utilizado.

**Tabela 2. Erro Médio Quadrático entre o Sinal Original e o Sinal Marcado**

| Sinal | $LCA$ | $vLCA$ |
|---|---|---|
| audio1 | 1.7e-5 | 1.2e-5 |
| audio2 | 1.5e-5 | 9.2e-6 |
| audio3 | 4.8e-5 | 3.3e-5 |
| audio4 | 5.4e-6 | 4.4e-6 |
| audio5 | 3.0e-4 | 2.1e-4 |
| média | 7.7e-5 | 5.4e-5 |

Um fato interessante observado nos testes, foi a existência de $GOS$'s que mesmo sem a utilização do limiar, $t_p$, o sinal erro gerado pelo algoritmo possuía um espectro que não satisfazia o critério de qualidade de [Lie and Chang 2006]. Este fenômeno ocorre nos dois algoritmos. Sendo assim, foi necessário estabelecer um valor mínimo para o limiar $t_p$. Nestes casos, quando o algoritmo atinge este valor mínimo, ele adiciona a marca d'água, mesmo que esta provoque uma distorção acima do desejado. Para avaliar o efeito deste método, foi calculado o espectro médio ao longo de todos os $GOS$'s de um sinal de áudio. Este espectro médio foi comparado com o limiar de audição em silêncio. A Tabela 3 apresenta esta comparação. Na $2^a$ coluna e na $4^a$ coluna são apresentadas as porcentagens das componentes do espectro média que estão acima do limiar de audição, para o $LCA$ e para o $vLCA$ respectivamente. Nas colunas $3$ e $5$ são apresentadas o valor máximo da diferença entre o espectro médio e o limiar de audição. Através dos resultados da Tabela 3, é possível notar que o sinal erro produzido pelos dois algoritmos está significativamente acima do limiar de audição em silêncio. No entanto, é importante ressaltar que o sinal erro está embutido no sinal marcado. Sendo assim, existe o efeito do mascaramento do sinal erro. De fato, o critério original proposto em [Lie and Chang 2006] é muito rigoroso, pois o sinal erro será sempre mascarado pelo sinal marcado. Uma extensão interessante para este trabalho seria analisar a qualidade subjetiva dos sinais marcados e qual seria o efeito de se considerar critérios menos rígidos, como por exemplo, os critérios adotados na quantização das amostras no padrão MPEG [Noll 1997].

**Tabela 3. Comparação do Espectro Médio do Sinal Erro com o Limiar de Audição em Silêncio**

| Sinal | $LCA$ | | $vLCA$ | |
|---|---|---|---|---|
| | $nf$ | $d_{max}$ | $nf$ | $d_{max}$ |
| audio1 | 18.6% | 15.5 dB | 18.0% | 14.2 dB |
| audio2 | 20.1% | 14.1 dB | 19.7% | 12.2 dB |
| audio3 | 27.9% | 16.4 dB | 24.4% | 14.5 dB |
| audio4 | 13.1% | 12.0 dB | 9.6% | 10.8 dB |
| audio5 | 41.2% | 23.4 dB | 39.1% | 21.1 dB |

## 5. Conclusão

Este trabalho apresentou uma variante do algoritmo proposto por Lie e Chang [Lie and Chang 2006], onde o procedimento para modificar as amostras do sinal de áudio é melhorado. O novo algoritmo para introdução de marca d'água digital, que é a principal contribuição deste trabalho, produz uma distorção menor que o algoritmo original, sem perder em robustez. Os dois esquemas foram implementados e testados em um conjunto composto por 5 trechos de músicas. Os resultados obtidos mostram um ganho de aproximadamente $1.5dB$ no erro médio quadrático.

## Referências

Petitcolas, F.A.P. and Anderson, R.J. and Kuhn, M.G. (1999). Information hiding - a survey *Proceedings of the IEEE*, 87(7):1062-1078.

Swanson, M.D. and Zhu, B. and Tewfik, A.H. and Boney, L. (1998). Robust audio watermarking using perceptual masking *Signal Processing*, 66:337-355.

Kirovski, D. and Malvar, H.S. (2003). Spread spectrum watermarking of audio signals *IEEE Transactions on Signal Processing*, 51(4):1020-1033.

Lie, W. and Chang, L. (2006). Robust and high-quality time-domain audio watermarking based on low-frequency amplitude modification *IEEE Transactions on Multimedia*, 8(1):46-59.

Noll, P. (1997). MPEG digital audio coding *IEEE Signal Processing Magazine*, 14(5):59-81.

Terhardt, E. and Stoll, G. and Seewann, M. (1982) Algorithm for extraction of pitch and picth salience from complex tonal signals *Journal of the Acoustical Society of America* 71:679-688.

Chen, B. and Wornell, G.W. (2001) Quantization index modulation: a class of provably good methods for digital watermarking and information embedding *IEEE Transactions on Information Theory* 47(4): 1423-1443.

# Composição de Paisagens Sonoras Digitais Através da Síntese Evolutiva

**José Fornari[1], Adolfo Maia Jr.[1,2], Jônatas Manzolli[1,3]**

[1]Núcleo Interdisciplinar de Comunicação Sonora – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6166 – 13.091-970 – Campinas – SP – Brazil

[2]Departamento de Música, Instituto de Artes (DM/IA)

[3]Departamento de Matemática Aplicada (IMECC)

{fornari,adolfo,jonatas}@nics.unicamp.br

***Abstract.*** *We present here the late development of the digital sonic structure system implemented in Pd (Pure Data) that uses Evolutionary Computation as a systemic and organizational strategy. The method here developed is called Evolutionary Sound Synthesis and is here described in its latest version where psychoacoustic descriptors are used to determine and manipulate its sonic typology. The synthesis results are here linked with the concept of soundscape design, as previously described by others and mentioned below. Finally, the late implementation is described together with its graphic interface.*

***Resumo.*** *Apresentamos neste artigo o desenvolvimento final de um sistema de síntese de estruturas sonoras digitais desenvolvido em Pd (Pure Data) que utiliza-se de Computação Evolutiva como estratégia sistêmica e organizacional. O método que desenvolvemos, denominado de Síntese Evolutiva (SE), é aqui descrito, bem como as últimas etapas da pesquisa onde são utilizados descritores psicoacústicos para determinar e manipular a tipologia sonora. O resultado sonoro deste processo é por nós relacionado com o conceito de paisagem sonora digital, no sentido já discutido por outros autores e apresentados a seguir. Finalmente, descrevemos a implementação final do sistema e a sua interface gráfica.*

## 1. Introdução

Computação Evolutiva, CE, é uma metodologia de otimização computacional que busca ótimos locais para um problema definido por uma população de possíveis soluções que se transformam dinamicamente [Koza,97]. Os métodos de computação evolutiva não possuem qualquer condição inicial ou campo fixo de definição, teoricamente tais métodos poderiam ser usados na resolução de qualquer tipo de problema [Bäck,96]. A utilização de CE em Computação Musical e Computação Gráfica tem sido ampliada nos últimos anos e na literatura há vários estudos sobre tais aplicações. [Miranda,03] investigou o potencial de algoritmos denominados de "vida artificial" (Alife) no contexto da criatividade musical. [Soddu,02] usou algoritmos generativos aplicados à multiplicidade de possíveis soluções descritas como estruturas arquitetônicas e representadas por computação gráfica. [Garcia,00] usou algoritmos genéticos na

automação de projetos de técnicas de síntese sonora. [Gibbs,96] desenvolveu animações gráficas com figuras articuladas usando programação genética. [Biles,94] usou algoritmos genéticos para criar o software GemJam, um programa que articula improvisações de jazz em tempo-real. [Horner,93] desenvolveu um método evolutivo para a geração automática de algoritmos aplicados à síntese FM.

No trabalho aqui apresentado, a nossa motivação para utilizar CE como método de construção de paisagens sonoras baseia-se em três premissas:

1. CE permite que a geração de formas de onda, seu sequenciamento temporal e espacialização sejam dinamicamente transformados num contexto adaptativo.

2. A utilização de um conjunto alvo e formas de onda pré-escolhidas pelo usuário/compositor pode tornar-se um controle eficiente pelo qual guia-se a evolução do processo e seu consequente resultado sonoro.

3. O resultado sonoro combina variedade e similaridade adaptativa.

O estudo aqui reportado vem se desenvolvendo desde 1999 no NICS. O primeiro projeto foi denominado "*VoxPopuli*", um ambiente evolutivo para gerar seqüênciais musicais através do protocolo MIDI (*Musical Instrument Digital Interface*) [Moroni et all,00], posteriormente, baseado nos resultados do primeiro sistema foi implementado o JaVOX, um sistema que ampliou as funcionalidades do primeiro e foi desenvolvido em JAVA (http:/www.geocities.com/Artemis_Moroni/JaVox/). O uso de CE aplicada à síntese sonora, denominado de Síntese Evolutiva de Segmentos Sonoros, iniciou-se com um método baseado em curvas psicoacústicas [Fornari,01]. Dentro do contexto, criou-se uma metodologia baseada em um sistema imunológico artificial apresentada em [Caetano,05].

Recentemente, estudamos a utilização de funções de localização sonora ITD (*inter-aural time difference*) como um parâmetro de controle do genótipo sonoro [Fornari,07]. A utilização de localização espacial junto com a capacidade da síntese evolutiva de gerar sons com similaridade variante mostrou-se uma abordagem para a criação e controle dinâmico de paisagens sonoras digitais.

Nas próximas seções apresentamos a idéia geral do método, destacando as conexões entre a Síntese Evolutiva e descritores sonoros. Apresentamos também o modelo matemático desenvolvido, seguido da implementação em Pd.

## 2. Método: Síntese Evolutiva

Iniciamos com um breve resumo do processo de síntese que desenvolvemos. A seguir apresentamos um modelo que se utiliza da localização espacial através do método ITD (*inter-aural time difference*) como fator de adequação do processo ou critério de "*fitness*".

### 2.1 Síntese Evolutiva de Segmentos Sonoros (SE)

No princípio da Síntese Evolutiva (SE), descrito em [Fornari,01], segmentos sonoros digitais (waveforms) são utilizados como indivíduos de uma população que sofre a ação evolutiva. O grau de adequação de cada segmento sonoro é medido pela distância em relação a um *conjunto alvo* de outros segmentos sonoros o qual representa o fator condicionante do meio.

O incremento temporal desta evolução artificial é dado pela *geração*, equivalente à um ciclo computacional do processo onde a população evolui, gerando novos segmentos sonoros que são condicionados pelo conjunto alvo. Os processos de *seleção* e *reprodução* agem na população a cada geração. A *seleção* compara os indivíduos da população em relação à sua semelhança com os indivíduos do conjunto alvo. Esta semelhança é dada pela função de adequação (i.e. medida de distância) que, a cada geração da população, determina o individuo mais próximo ao alvo, o qual é chamado de *melhor indivíduo*. A reprodução gera novos indivíduos na população através de dois operadores genéticos: *crossover* que, a cada geração, substitue todos os indivíduos da população por descendentes que são obtidos por cruzamento com o *melhor indivíduo* da geração atual e mutação que produz variações randômicas nestes indivíduos.

### 2.2 SE e Localização Espacial

Com a inclusão de localização espacial no processo, constatamos que o método de síntese poderia ser expandido conceitualmente. Neste artigo, nós restringimos, por simplicidade, a localização espacial às duas dimensões do plano horizontal em relação aos ouvidos do usuário. Não consideramos a altura da fonte sonora em relação ao ouvinte. Claramente o modelo pode ser estendido com a inclusão de localização 3D. Desta forma, estudamos a utilização de funções de localização sonora como um novo parâmetro do genótipo sonoro. Isso nos levou a considerar a síntese evolutiva não apenas para a transformação dos fatores psicoacústicos, mas também para a manipulação da interpretação do som, descrita por alguns aspectos relacionados à cognição de estímulos sonoros (descritores apresentados em 4.1).

A localização espacial utilizada como função de adequação permite a criação de uma seqüência de amostras sonoras posicionados no espaço que apresentam uma similaridade convergente como característica do próprio processo de síntese. Estes sons com posicionamento espacial acabam, ao longo do tempo, compondo um ambiente sonoro que apesar de variar continuamente é reconhecido por sua similaridade. Deste modo, supomos que este comportamento sonoro da síntese evolutiva poderia ser utilizado para a criação do que chamamos de *paisagens sonoras digitais.*

### 2.3 SE e Paisagem Sonora Digital

Conceitualmente o termo paisagem sonora (i.e. *soundscape*) denota um ambiente complexo no qual a informação sonora nunca se repete e no entanto é cognitivamente auto-similar, uma vez que é facilmente reconhecido pela percepção auditiva humana. Paisagens sonoras podem ser naturais (ex: som de uma floresta) ou não (ex: som de um canteiro de obras em atividade). Segundo [Schafer,77], tais sistemas são caracterizadas por três tipos de referências sonoras: *keynotes* - correspondente ao centro de referència sonora, um ruído de fundo cujo equivalente, em obras tonais, seria a tônica, *signals* - são os sinais sonoros conscientemente percebidos e presentes no ambiente sonoro que alertam e chamam a atenção do ouvinte, e *soundmarks* - são os sinais sonoros únicos ao contexto de um soundscape que caracterizam por sua similaridade e aspectos culturais como enfatiza [Truax,78].

Dentre estes três aspectos fundamentais, percebemos que o resultado sonoro da síntese evolutiva se inseria no paradigma descrito acima especialmente no que tange à "*similaridade variante*" da população de formas de ondas que produzem *soundmarks* que são manipuladas pela informação espacial. A síntese evolutiva que desenvolvemos

passou assim a ter uma aplicação criativa: composição de paisagens sonoras digitais. A implementação em Pd permite que o usuário controle a geração das paisagens, não apenas pela manipulação dos parâmetros de síntese e do conjunto alvo, mas também através de um arquivo de texto, chamado de *score*, que organiza o processo de composição.

## 3. Modelagem

Inicialmente foi necessário o desenvolvimento de uma representação de genótipo sonoro, ou seja, do conjunto de parâmetros que caracterizam amostras sonoras num dado contexto musical, que não apenas permitisse o reconhecimento e discriminação de sua natureza psicoacústica, mas que também fosse computacionalmente viável. Iniciamos com o estudo de três modelos do genótipo sonoro descritos em [Fornari,03]: 1) *genótipo como forma de onda (waveform*), 2) *genótipo como um conjunto de curvas psicoacústicas*, e 3) *genótipo como espectrograma* sonoro e culminamos por descrever o genótipo através de descritores cognitivos não-contextuais, os quais são descritos a seguir na seção 4.

### 3.1 Função ITD

Conforme descrito em [Murray,04] a função ITD simula o mecanismo pelo qual a audição percebe a localização de uma fonte sonora pela diferença de tempo de recepção do som entre os ouvidos. Se não houver atraso (se ambos os ouvidos receberem a mesma informação sonora simultaneamente) a localização da fonte é percebida como estando em frente ao ouvinte, caso contrario, percebe-se a fonte sonora deslocada horizontalmente, num dado ângulo de azimute. A função ITD emula esse efeito através da inserção de atraso entre dois canais que contém a mesma informação sonora.

#### *3.1.1 SoundMarks no Campo de Localização Sonora*

Sob o ponto de vista matemático, este modelo é constituído de um espaço de triplas **G**={(W, I, L)}, chamado de Espaço Genotípico, onde $0 \leq I \leq 1$ é o fator de intensidade sonora, e $-1 \leq L \leq 1$ é o fator de localização espacial (2D) ITD, dado pelo ângulo de azimute $\theta$, onde $L = (90^o - \theta)/ 90^o$ e $0^o \leq \theta \leq 180^o$. Para maiores detalhes, veja [Fornari,07]. O conjunto de todos os possíveis valores do par (I,L) é chamado de "*Campo de Localização Sonora*" (SLF). Em nosso modelo, este é descrito como um semi-círculo e o par (1,0) é associado com o som de maior intensidade e localizado em frente ao ouvinte. A dispersão espacial no SFL é caracterizada pela distribuição dos conjuntos finitos de pares S = (I, L) conforme mostrado na Figura 1.

Definimos como População um sub-conjunto finito de elementos de **G**. Em nosso modelo iniciamos com uma população $P^{(0)}$ e conjunto alvo **T** e, interativamente, construímos uma seqüência de um número r de populações $G^{(1)}$, $G^{(2)}$,…, $G^{(r)}$, onde a k-ésima população é um sub-conjunto de **G** com N indivíduos (elementos) $\boldsymbol{G}^{(k)} = \{G_1^{(k)}, G_2^{(k)}, \ldots, G_N^{(k)}\}$ e os indivíduos são dados pelas triplas $G_i^{(k)} = (W_i^{(k)}, I_i^{(k)}, L_i^{(k)})$. O conjunto alvo tem M indivíduos $\mathbf{T} = \{t_1, t_2, \ldots, t_M\}$ cujo j-ésimo indivíduo é dado por $tj = (W_j^{(T)}, I_j^{(T)}, L_j^{(T)})$. A dispersão espacial na SLF é caracterizada pela distribuição do conjunto de pares $S_i = (I_i, L_i)$ conforme mostra a Figura 1. Estes pares, onde os Operadores Genéticos são aplicados são chamados de Genótipos de Espacialização Sonora (SSG). O conjunto alvo $\mathbf{T} = \{T_k = (I_k, L_k)$, onde $k=1,\ldots,M\}$ pode, em principio, ser gerado por uma série de controladores gestuais associados a posição e movimento

do usuário. Isto permitiria que impressões perceptuais guiem interativamente o processo evolutivo de distribuição espacial do som.

Uma vez que $\boldsymbol{G}^{(k)}$ e **T** são subconjuntos de **G** nós definimos a distância entre estes dois como se segue:

$$d_{ij}(G^{(k)},T) = \frac{1}{2}\frac{\left|I_i^{(k)} - I_j^{(k)}\right|}{A} + \frac{1}{4}\frac{\left|L_i^{(k)} - L_j^{(k)}\right|}{B} \qquad \textbf{(1)}$$

onde as constantes A e B são respectivamente dadas como os fatores máximos de intensidade e localização, e cuja distância é normalizada dentro do intervalo [0,1].

A distância $d_k$ entre $\mathbf{G}^{(k)}$ e **T** é definida por:

$$\mathbf{d_k} \equiv d(\boldsymbol{G}^{(k)},\mathbf{T}) = \min_{i,j} d_{ij}(G^{(k)},T) \qquad \textbf{(2)}$$

onde i=1,…,N e j=1,2,…, M. Observe que a função de distância usa apenas dois fatores de SSG.

O melhor indivíduo da k-ésima população $G^{(k)}$, $G_{i*}^{(k)} = ( W_{i*}^{(k)}, I_{i*}^{(k)} , L_{i*}^{(k)})$ é aquele que apresenta a distância $\mathbf{d_k} \equiv \mathbf{d}(\boldsymbol{G}^{(k)}, \mathbf{T})$. Este novo indivíduo (o melhor da população) pode ser usados pelo processo evolutivo interativo, conforme descrito em 4. Para controlar a saída sonora nós usamos a distância descrita acima para definir a *Similaridade Espacial* como segue:

Dado dois indivíduos da população $\mathbf{G}_i^{(k)}$ , $\mathbf{G}_j^{(k)}$ , eles são similares se $\mathbf{d}( \mathbf{G}_i^{(k)}, \mathbf{G}_j^{(k)}) \leq \varepsilon$ , onde ε é um número arbitrário e a distância **d** é definida pela Eq. (1).

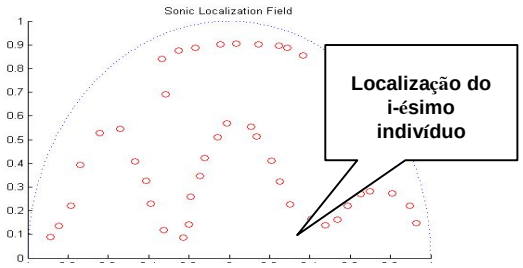



**Figure 1. Campo de localização sonoro** (SLF).

### 3.2. Operadores Genéticos na Espacialização Sonora

*3.2.1 Crossover*

Dado o melhor indivíduo da k-ésima geração $\mathbf{G}_{i*}^{(k)} = ( W_{i*}^{(k)}, I_{i*}^{(k)}, L_{i*}^{(k)})$ e a taxa de crossover α**,** onde $0 \leq \alpha \leq 1$, os parâmetros de localização dos indivíduos da População serão transformados como se segue:

$$I_i^{(k+1)} = \alpha\, I_{i*}^{(k)} + (1-\alpha).\, I_i^{(k)}, \text{ e} \qquad \textbf{(3)}$$

$$L_i^{(k+1)} = \alpha.\, L_{i*}^{(k)} + (1-\alpha).\, L_i^{(k)}$$

para $1 \leq i \leq N$, e k= 0,1,…, R.

onde R é o número de interações.

*3.2.2 Mutação*

Similarmente, dado o melhor indivíduo da k-ésima geração $G_{i*}^{(k)} = ( W_{i*}^{(k)} , I_{i*}^{(k)} , L_{i*}^{(k)})$ e a taxa de mutação ß, onde $0 \leq ß \leq 1$, a operação de mutação é definida como se segue:

$$I_i^{(k+1)} = \beta_1 . (rand) + (1-\beta_1). I_i^{(k)} \text{ e} \quad \textbf{(4)}$$

$$L_i^{(k+1)} = \beta_2 . (rand) + (1-\beta_2). L_i^{(k)}$$

$$\text{para } 1 \leq i \leq N, \text{ e } k=0,1,\ldots, r.$$

onde "rand" é o parâmetro randômico no intervalo C e as taxas $\beta_1$ e $\beta_2$ controlam o grau de aleatoriedade da operação de mutação. Nesta implementação nós consideramos $\beta_1 = \beta_2$ por simplicidade.

## 4. Implementação

A primeira implementação do SE utilizando a linguagem de programação Pd serviu para salientar os pontos que deviam ser abordados pelo método inicial [Fornari, 01]. Dos diversos aspectos percebidos, viu-se a necessidade de se incorporar uma série de extensões ao método de modo a torná-lo mais semelhante à evolução biológica. Entre eles, destacam-se as seguintes extensões.

- *Conceito de variação demográfica, dado pela implementação de uma população de tamanho variável onde os indivíduos apresentam um determinado tempo de vida.*
- *Conceito de reprodução sexuada, dado pela criação de indivíduos com gênero sexual.*
- *Conceito de localização e mobilidade geográfica de indivíduos através da implementação de genótipos com funções de localização espacial sonora.*

Estas três extensões serviram para re-orientar não só o desenvolvimento do SE, mas também a sua aplicação. Dada a sua crescente proximidade com modelos de evolução biológica, a SE passou a se aproximar da geração de estruturas mais complexas, ou seja, ao invés da síntese de sons, a produção de paisagens sonoras. Percebemos que a grande diversidade de formas de onda e a similaridade entre elas, era uma característica importante do método.

### 4.1 Descritores Percepção Sonora

No desenvolvimento da SE, a primeira e a terceira extensão descritas acima estão diretamente relacionadas à geração de paisagens sonoras. Neste tipo de aplicação o número de fontes sonoras pode variar (população de tamanho variável) bem como as fontes sonoras podem se deslocar no espaço (localização espacial). Apesar de não relacionada diretamente a paisagens sonoras, achamos interessante implementar também a segunda extensão descrita acima: *a reprodução sexuada*. Para isso, criamos o conceito de gênero no indivíduo da síntese evolutiva. Partindo do conceito de reprodução sexuada uma nova extensão do método da síntese evolutiva foi acrescida do conceito de genes diplóides. Assim, em cada genótipo tem-se que cada cromossomo é formado por genes dominantes e/ou recessivos, tal e qual na biologia.

Os genes são dados por descritores cognitivos não-conceituais, tal como apresentados em [Leman,03]. Existem diversas representações de descritores, tal como ataque,

harmonicidade, inharmonicidade, rugosidade, entropia, flutuação, etc. Para essa implementação da SE, utilizamos quatro descritores: *onsetness*, *loudness*, *pitchness* e *location*. Estes descritores foram definidos neste trabalho com base nos estudos de [Leman,04]. Cada um dos descritores são normalizados em escalas entre zero (nulidade ou ausência do descritor em questão) e um (presença máxima do descritor).

Assim definimos o descritor *onsetness* como aquele representando a quantidade de ataque (*onset*) presente no som, sendo que onsetness é próximo de zero para sons que não possuem ataque perceptual, tal como é o caso de sons da família das cordas friccionadas (não *pizzicato*) quando este efeito é desejado e pode ser efetivamente conseguido com músicos com bom treinamento. Do mesmo modo, *onsetness* seria próximo de um para sons que sejam praticamente puro ataque, tal como o som de um estalo (sem reverberação), ou um pulso unitário.

O descritor *pitchness* representa o grau de tonalidade de um som. Para sons ruidosos, que não apresentam qualquer definição perceptual de *pitch*, tem-se pitchness=0. Para sons com espectro harmônico, como o som de uma nota musical de um instrumento melódico, sem qualquer presença de ruído (como o som de um diapasão) tem-se pitchness=1. O descritor *loudness* determina o grau de percepção de intensidade sonora, lembrando-se que este não está unicamente relacionado com a intensidade do sinal sonoro, mas também com o ataque, a freqüência do parcial fundamental e a composição espectral do som em questão.

No processo de seleção existem quatro parâmetros correspondentes aos três descritores (definidos acima) e à localização sonora. Estes definem a máxima distância permitida para cada descritor do genótipo do indivíduo em relação ao alvo. Por exemplo, se num dado processo de seleção o parâmetro de distância de um dado descritor for *dd=0,5* então será eliminado da população qualquer indivíduo cujo mesmo tipo de descritor d, comparado aos seus correspondentes no conjunto alvo satisfizer $|d - da| >= dd$ a todos os genótipos do conjunto alvo (i.e. *da* é um descritor do conjunto alvo).

O processo de reprodução é aqui determinado por quatro fatores. Os dois primeiros são descritos no genótipo do indivíduo: *localização espacial e gênero*. Apesar de não ser uma característica genotípica, por motivos de conveniência computacional a localização do indivíduo é descrita no arquivo texto que representa o genótipo do indivíduo. Este é dado por duas variáveis normalizadas, I (intensidade) e L (angulo de azimute), dada por I=[0,1] e L=[1,1] (respectivamente correspondente aos ângulos de 0° e 180°). Nesta implementação definimos quatro tipos de gênero dos indivíduos: *m (macho), f (fêmea), s (estéril) e b (ambos)*. As únicas reproduções possíveis são as que obedecem às seguintes regras:

$m \,\&\, f \Rightarrow m \mid f \mid b$

$[\, m \mid f \mid b] \,\&\, b = s$ *(reprodução com b sempre gera indivíduos estéreis).*

***onde:*** *& = AND, | = OR e x & y é a reprodução entre x e y.*

### 4.2 Implementação em *Pure Data*

A saída de som da SE também foi expandida. Ao invés de ser apenas formada pelo melhor indivíduo de cada geração, esta passou a ser definida pelo usuário através de dois parâmetros: m (melhores, ou mais próximos do alvo) e p (piores, ou mais distantes do alvo) seguidos por uma variável [0,1] referente a porcentagem de indivíduos na

população. Exemplificando, *m=0.3* corresponde à saída dada pela mistura de 30% dos indivíduos da população ordenados do melhor ao pior. A implementação inicial foi feita na forma do arquivo Pd, se00.pd (versão 0.0). A melhoria desta implementação inicial resultou no se01.pd (versão 0.1), cuja figura abaixo mostra sua interface gráfica (*front-end*):

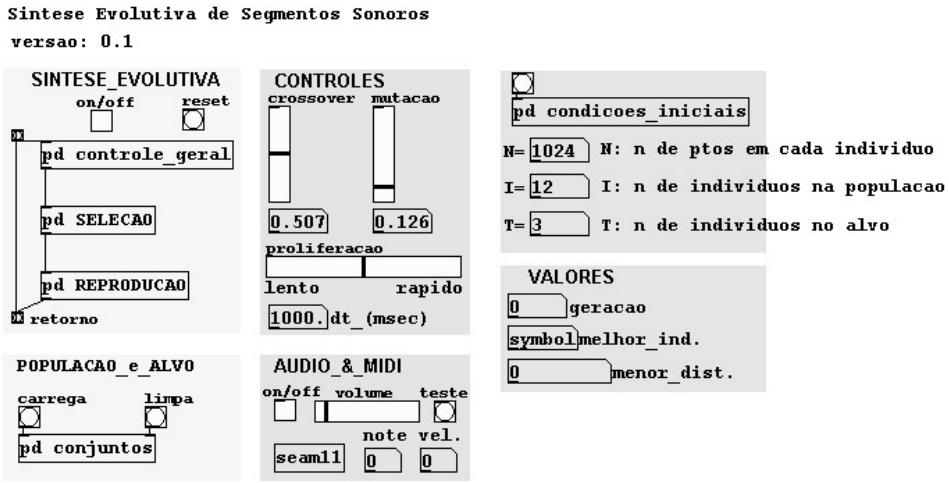



**Figura 2. Primeira implementação do SE, o se01.pd**

Ao iniciar a síntese, o *se01.pd* vai selecionar o individuo da população que é mais próximo dos indivíduos do conjunto alvo. Este será o primeiro melhor indivíduo e em seguida o processo de reprodução modifica todos os 12 elementos do conjunto população através das operações genéticas entre cada indivíduo da população e o melhor indivíduo escolhido anteriormente.

### *4.2.1 Arquivo de Controle Paramétrico*

Na fase final da implementação em Pd, o método da síntese evolutiva foi remodelado com base nos novos critérios apresentados na seção anterior e o sistema foi assim expandido para a geração de paisagens sonoras digitais. Obteve-se um sistema mais complexo, que inclusive utiliza-se de um arquivo texto, que chamamos de *score*. Este arquivo contém instruções ordenadas no tempo que organizam o processo de geração de paisagens sonoras. Além do arquivo score, a SE é também controlada pelos parâmetros contínuos das taxas dos operadores genéticos (crossover e mutação), a taxa de proliferação populacional, bem como pela modificação dinâmica dos indivíduos do conjunto alvo pelo usuário em tempo real. No modelo de *score*, tem-se toda a informação inicial e as linhas de comando são organizadas de forma sequencial. Na figura abaixo tem-se um exemplo típico do arquivo *score*:

**Table 1 – *Score* utilizado para controle da Síntese Evolutiva**

| | |
|---|---|
| Title<br>ParteTeste;<br><br>instructions<br>Esta eh a sintese evolutiva de \\<br>soundscapes com score ParteTeste\\ | *Title*: o nome do *score*.<br>*instructions*: instruções da organização deste score que irão aparecer ao usuário, sob a forma e comentário, na janela do programa.<br>*num-pop*: número inicial de indivíduos na população, que deve estar em conformidade com o número de WAV *files* no arquivo *pop*.<br>*num-alvo*: número inicial de indivíduos no conjunto alvo que, igualmente, deve estar em conformidade com o número de WAV *files* no arquivo *alvo*. |

| | |
|---|---|
| \\ | *proli*: valor inicial de proliferação da reprodução de indivíduos. |
| Clique SPACE para prosseguir\\ | *cros*: valor inicial da taxa de crossover. |
| \\ | *mut*: valor inicial da taxa de mutação. |
| modos do score: \\ | |
| time selec onset loud pitch location \\ | Logo em seguida, tem-se as linhas de comando, ou *command-lines*, dada em uma única linha encerrada por ";". Estas são ordenadas com as seguintes instruções: |
| time repro gender proximity\\ | |
| output proxim x distant y all\\ | |
| \\ | *Time*: tempo de atraso para a execução da linha, dado em segundos. |
| global: selecao\\ | *Mode*: determina o modo de operação: *sel* processo de seleção, *rep* reprodução e *out* saída sonora. |
| step: reproducao\\ | |
| state: output\\ | |
| ; | |
| num-pop 7; | |
| num-alvo 3; | |
| proli .5; | |
| cros .5; | |
| mut .1; | |
| 00 sel .5 .5 .5 .5; | |
| 00 rep .5; | |
| 00 out p 1; | |
| 01 out p 5; | |
| 05 out l 1; | |

O genótipo de cada segmento sonoro da população é também dado por um arquivo de texto. Seguindo o padrão do objeto "*textfile*" do Pd, onde cada linha é encerrada pelo símbolo ";". A primeira linha do arquivo de texto do genótipo especifica a localização espacial do indivíduo (*location*). Esta é dada através de dois parâmetros, Intensidade, que varia entre I = [0,1], e ângulo de azimute que varia entre A = [-1,1] respectivamente correspondendo à uma variação de ângulo entre [-180°, 0°]. A segunda linha arquivo de texto do genótipo determina o "tempo de vida" (*lifespam*) deste indivíduo, ao longo do processo evolutivo. Este tempo é dado em segundos. Após terminado este tempo pré-estabelecido, o indivíduo é automaticamente eliminado da população. Para representar indivíduos "imortais" usa-se "zero" como parâmetro (conforme é mostrado no exemplo da figura 3).

A terceira linha determina o gênero do indivíduo (*gender*), podendo este ser: *m (masculino), f (feminino), b (ambos), s (estéril)*. Conforme foi mostrado, existem duas regras que determinam a reprodução anteriormente, sendo que se o indivíduo for *s* não pode reproduzir e permanecerá na população até terminar seu tempo de vida ou ser eliminado por distanciamento do conjunto alvo.

As próximas linhas do genótipo determinam os três descritores cognitivos escolhidos para representar o genótipo em si. Estes formam o cromossomo diplóide do indivíduo onde existe o conceito de dominância e recessividade. Então, em cada linha tem-se o valor para o gene dominante dado pelo parâmetro d, seguido de seu valor numérico (entre [0,1]) para o gene recessivo, dado por r, seguido de seu valor normalizado. O conceito de hereditariedade e caracterização fenotípica seguem as simples regras dispostas pela genética de *Gregor Mendel*. Um exemplo típico é dado abaixo:

```
location 1 -1;
lifespam 0;
gender m;
onsetness d .5 r .5;
loudness d .1 r .9;
pitchness d .9 r .1;
```

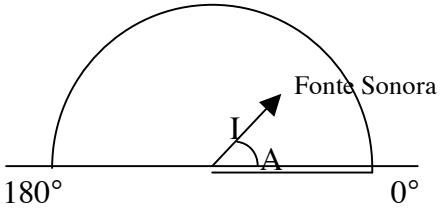



**Figura 3. Esquerda: Exemplo de um genótipo da SE, representado por um arquivo de texto. Direita: Localização do indivíduo da SE.**

### *4.2.2 Interface Gráfica*

O *front-end* da implementação da SE é visto na figura 4. Nesta janela inicial o usuário é convidado a digitar o nome de um *score* que guiará o processo evolutivo. Ao digitar o nome do score uma nova janela aparece com este novo nome no titulo e com outros parâmetros evolutivos. Na figura 4, centro, é mostrada um exemplo desta janela. Caso não haja um *score* definido, a próxima janela é similar àquela localizada à esquerda da figura 4.

Como se percebe, não é necessária a utilização de um *score* na síntese evolutiva. Este funciona como uma partitura musical que descreve as passagens de uma composição musical. Analogamente, a não utilização de um *score* assemelha-se à improvisação musical, pois deixa a cargo do usuário o controle em tempo real de todos os parâmetros do processo de síntese.

## 5. Discussão

Do ponto de vista qualitativo sonoro, estamos agora desenvolvendo projetos composicionais onde poderemos verificar o potencial musical desta nova aplicação. Todavia, a nova implementação da SE está disponível em: http://www.nics.unicamp.br/~fornari/se. Esperamos com isso que uma grande quantidade de pesquisadores da área avaliem esta implementação qualitativamente, gerem amostras sonoras e emitam seus pareceres . É nosso intento que tenhamos conseguido chegar a uma ferramenta musical interessante para o compositor de música eletroacústica. Coerente com o método de implementação, buscamos agora uma estratégia populacional para testar a variedade e aplicabilidade do método de síntese que desenvolvemos.

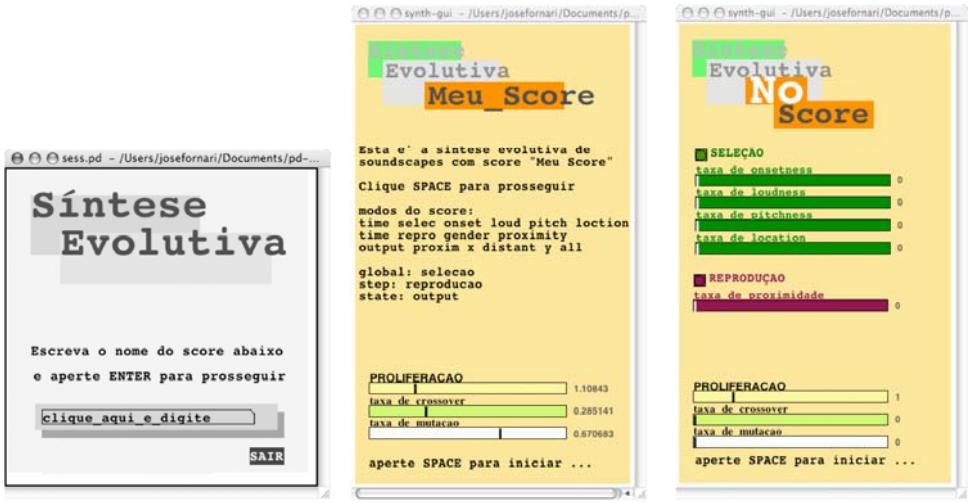



**Figura 4. (esquerda) Janela front end da interface do programa. (centro) Janela com o nome do *score* e os respectivos controles para a síntese evolutiva. (direita) Janela da síntese evolutiva feita sem a presença de um *score*.**

## 6. Conclusão

Apresentamos a implementação do sintetizador evolutivo em Pd, com processamento e controle em tempo-real. Esta implementação abre as portas para um novo conceito de processamento não-determinístico de ambientes sonoros, ou paisagens sonoras digitais, uma vez que o método gera sons que se modificam adaptativamente, convergindo com similaridade variante.

A interface gráfica (vide figura 4) da SE também apresentou grande melhoria em relação à primeira versão implementada [Fornari,01]. No entanto, o Pd não oferece grandes recursos gráficos em sua plataforma atual. Teoricamente, é possível utilizar apenas o *engine* do Pd processando em *background* controlado por uma plataforma gráfica mais desenvolvida, por exemplo, em JAVA. Isto tornaria a visualização do programa mais aprazível e protegeria o código do algorítmo.

Do ponto de vista de aplicações musicais, estamos interessados em utilizar o método com dispositivos de localização espacial para obtermos um processo interativo em tempo real onde o conjunto alvo poderia ser substituido por uma *buffer* dinâmico. A segunda aplicação, é justamente voltada à composição de paisagens sonoras utilizando uma grande variedade de segmentos sonoros, onde poderíamos testar, então, a capacidade do sistema como ferramenta composicional.

### Agradecimentos

José Fornari teve o apoio da FAPESP através de bolsa de pós-doutoramento desenvolvida no NICS, processo 04/00499-6R. Jônatas Manzolli tem o apoio do CNPq através de bolsa de produtividade em pesquisa.

## Referências

Koza, John R., “Genetic Programming”, Encyclopedia of Computer Science and Technology, 1997.

Bäck, T. “Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms”, Oxford Univ. Press. 1996.

Miranda, E. R.. "On the evolution of music in a society of self-taught digital creatures." Digital Creativity, Vol. 14, No. 1, 29-42. 2003.

Soddu C. O. "A Generative Approach to Art and Design" Leonardo, 1 June 2002, vol. 35, no. 3, pp. 291-294. MIT Press. 2002.

Garcia, Ricardo A. "Automatic Generation of Sound Synthesis Techniques". Proposal for degree of Master of Science. MIT - Fall. 2000.

Gibbs, J. "Easy Inverse Kinematics using Genetic Programming". Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press. http://www.red3d.com/cwr/evolve.html. 1996.

Biles, J. A., "Gen Jam: A Genetic Algorithm for Generating Jazz Solos", Proceedings of the 1994 International Computer Music Conference, (ICMC'94), 131-137. http://www.it.rit.edu/~jab/GenJam.html. 1994.

Horner, A., Beauchamp, J., and Haken, L., "Machine Tongues.16. Genetic Algorithms and Their Application to FM Matching Synthesis," Computer Music Journal, vol. 17, pp. 17-29. 1993.

Moroni, A., Manzolli, J., Von Zuben, F., Gudwin, R., "Vox Populi: An Interactive Evolutionary System for Algorithmic Music Composition", Leonardo Music Journal, San Francisco, USA, MIT Press, Vol. 10. 2000.

Fornari, J., Manzolli, J., Maia Jr., A., Damiani F., "The Evolutionary Sound Synthesis Method". SCI conference. Orlando, USA. 2001.

Caetano, M. F., Manzolli, J., Von Zuben, F. J. "Application of an Artificial Immune System in a Compositional Timbre Design Technique". In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.. (Org.). Artificial Immune Systems, Lecture Notes in Computer Science. Berlin: Springer-Verlag, v. 3627, p. 389-403. 2005.

Fornari, J., Manzolli, J., Maia Jr., A., Damiani F., "The Evolutionary Sound Synthesis Method". SCI conference. Orlando, USA. 2001.

Schafer, R. M. (1977). “The Soundscape”. ISBN 0-89281-455-1.

Truax, B.. (1978) “Handbook for Acoustic Ecology”. ISBN 0-88985-011-9.

Fornari, José Eduardo. “Síntese Evolutiva de Segmentos Sonoros”. Dissertação de Doutoramento. DSIF/FEEC/UNICAMP. 2003.

Leman, M., Vermeulen, V., De Voogdt, L., Moelants, D., & Lesaffre, M. (2004). Correlation of Gestural Musical Audio Cues. *Gesture-Based Communication in Human-Computer Interaction:5th International Gesture Workshop, GW 2003*, 40-54.

Murray, J. C., Erwin, H. R., and S. Wermter, "Robotic sound source localization using interaural time difference and cross-correlation", presented at KI-2004, September 2004.

Fornari, J.; Maia Jr. A.; Manzolli, J.. “Generating Soundscapes Using Evolutionary Spatial Control”. In Proceedings of EvoWorkshops, EvoMusarts, Spring-Verlag: Berlin. pg 517-526, Valencia. 2007.

# Um modelo computacional das teorias de Edmond Costère e da Teoria de Conjuntos implementado em uma ferramenta analítica em PHP

**Prof. Dr. Marcus Alessi Bittencourt**

Universidade Estadual de Maringá,
Centro de Ciências Humanas Letras e Artes,
Curso de Graduação em Música,
Av. Colombo, 5790 - Bloco G-34, sala 101,
87020-900 - Maringá, PR - Brasil.

mabittencourt@uem.br

***Abstract.*** *This paper presents a computational model in PHP of the theories by Edmond Costère (Costère, 1954) combined to Set-Theory (Forte, 1973 and Rahn, 1980) and implemented in the form of an analytical calculator available to the community in the internet as an HTML page. Such a tool allows quick access to the diagnostics generated by the computational model by means of an HTML page containing a printable form with a concise and clear data visualization. This HTML interface is presented and its operation is explained step by step together with the implementation of the computational model used. The applications of such a tool are discussed in detail and tetrachords from the piece Op. 33a by Arnold Schoenberg are analyzed by the calculator as an exemplification of the process.*

*Keywords: Music Analysis, Set-Theory, Edmond Costère.*

***Resumo.*** *Este artigo apresenta uma modelagem computacional em PHP das teorias musicais analíticas de Edmond Costère (Costère, 1954) combinadas à Teoria Musical dos Conjuntos (Forte, 1973 e Rahn, 1980) e implementada na forma de uma calculadora analítica disponível à comunidade pela internet como uma página em HTML. Tal ferramenta permite o rápido acesso aos diagnósticos analíticos gerados pelo modelo computacional por meio de uma página de HTML contendo um formulário conciso e de fácil visualização e impressão. Esta interface em HTML é apresentada e seu funcionamento explicado passo a passo conjuntamente com a implementação do modelo computacional utilizado. As utilidades de tal ferramenta são discutidas em detalhe e tetracordes tirados da peça Op. 33a de Arnold Schoenberg são analisados pela calculadora a título de exemplo.*

*Palavras-chave: Análise Musical, Teoria de Conjuntos, Edmond Costère.*

## 1. Introdução.

### 1.1. Sobre as teorias de Costère.

As teorias musicais do musicólogo francês Edmond Costère, pseudônimo do eminente magistrado da Suprema Corte francesa Edouard Coester (1905-2001), apesar de serem em grande parte desconhecidas do meio acadêmico mundial, nunca deixaram de fascinar e intrigar, tanto positivamente como negativamente, os poucos que a conhecem. O objetivo geral da obra de Costère consiste na tentativa de formular uma teoria geral da música que, embasada em pressupostos físicos sonoros e supostamente universais e irrefutáveis, serviria para explicar o

fazer musical de todas as épocas e culturas, demonstrando assim a inexistência de uma real ruptura técnica entre a música do passado e a do presente. Apesar da multiplicidade de estilos musicais possíveis, Costère observava a existência de algo objetivo, físico-acústico e não cultural, subjetivo, capaz de gerar relações de polarização, de atração entre alturas, um princípio sempre ativo nas sonoridades. Costère formalizou e quantificou esta idéia na forma de sua **Lei da Atração Universal** que estabelece que as menores distâncias entre dois pontos representam as passagens de escoamento principais das forças de atração sonora. Desta maneira, cada altura-classe dada teria cinco atratores chamados de **notas cardinais**, a saber: ela própria, as quintas justas ascendente e descendente (o menor caminho no sentido da série harmônica) e as duas alturas adjacentes (o menor caminho no sentido da escala), que podem variar de acordo com o temperamento usado e que traduzem-se nos semitonos ascendente e descendente no nosso sistema ocidental de temperamento igual com doze notas por oitava. Utilizando este raciocínio, dada uma coleção hipotética de alturas-classe X, a **densidade de atração** (de polarização) de cada altura-classe do universo cromático, quer esta pertença ou não a esta coleção X dada, é calculada como sendo o número das notas cardinais desta altura-classe que existir na coleção dada (ver fig. 3). Da mesma forma, dada uma coleção hipotética de alturas-classe X, calcula-se a densidade de atração de uma outra coleção de alturas-classe Y somando-se as densidades individuais de cada altura-classe constituinte desta coleção Y.

A partir deste princípio, Costère desenvolve um extenso sistema de classificações e diagnósticos analíticos de todas as combinações harmônicas possíveis no temperamento igual ocidental, reduzindo o número destas a 351 tipos por meio de relações de equivalência transposicional. Refutáveis ou não, os livros de Costère promovem uma reflexão aguda sobre o fazer musical, em especial o do século vinte, e são apresentados com uma desenvoltura e lógica fascinantes em uma prosa apaixonada e de deliciosa leitura. No Brasil, o interesse nas teorias de Costère tem principal origem na figura do compositor Willy Corrêa de Oliveira e no núcleo de seus alunos e ex-alunos, dos quais eu mesmo faço parte confirmando a regra. Foge ao escopo deste artigo defender, refutar ou explicar em detalhe as teorias de Costère, o que seria uma tarefa demasiadamente extensa. Para isso, o leitor deve referir-se aos próprios textos de Costère (Costère, 1954 e 1962) e aos trabalhos de Marisa Ramires (Ramires, 2001) e Brian Ellard (Ellard, 1973). Sobre a polêmica em torno da aceitação das teorias de Costère, é possível encontrar um pequeno material curioso em edições da década de 60 do periódico Music & Letters (W., J.A., 1963 e Vale, 1964) comentando o então recém-publicado *Mort ou Transfigurations de l'Harmonie* (Costère, 1962). Ainda assim, minhas explicações que seguirão sobre o funcionamento da calculadora analítica hão de fornecer uma razoável introdução ao pensamento de Costère e permitir um vislumbre de suas possibilidades.

### 1.2. Sobre a Teoria de Conjuntos e sua associação à teoria de Costère.

Sobre a Teoria Musical dos Conjuntos, esta dispensa aqui uma maior apresentação por sua extensa divulgação e utilização no meio acadêmico musical (Forte, 1973) (Rahn, 1980) (Straus, 1990) (Oliveira, 1998) (Perle, 1968). Talvez surpreenda aqui a sua utilização ao lado dos princípios de Costère, mas o fato é que é uma preocupação comum às duas teorias a idéia de enxergar semelhanças e equivalências entre sonoridades com a finalidade de criar tipologias destas, o que é a base para um processo de redução do número de combinações de alturas ao menor número possível de tipos básicos. Brian Ellard (Ellard, 1973) já havia comparado em sua tese de doutorado teorias contemporâneas as mais diversas, como as de Costère, Forte, Perle, Messiaen (Messiaen, 1944), Hindemith (Hindemith, 1945), Slonimsky (Slonimsky, 1947) e Hanson (Hanson, 1960), apontando suas semelhanças e diferenças de uma maneira extremamente lúcida e construtiva.

## 2. O projeto de uma calculadora analítica.

O estudo das propriedades combinatórias e de equivalência entre diversas coleções de alturas, grafando seus fluxos de atração e seus potenciais de invariância sob processos de transposição e inversão tornou-se uma tarefa importantíssima tanto para meu trabalho como analista e professor como para meu trabalho como compositor. Apesar de não vir ao caso os teores de uso ou benefício que extraio destes estudos, é suficiente dizer aqui como justificativa para este projeto que senti uma necessidade premente de construir uma ferramenta computacional capaz de realizar todos os cálculos analíticos de que eu necessitava, que apesar de não serem tão complexos do ponto de vista dos matemáticos são certamente entediantes e dispendiosos do ponto de vista dos músicos. Também era imperativo o acesso, de maneira rápida e clara, às interpretações e diagnósticos feitos a partir da análise dos dados levantados. A simples consulta às "Table Synoptique des 351 Échelonnements" e "Tableaux Analytiques des Échelonnements" do "*Lois et Styles*" de Costère (Costère, 1954) ou às tabelas dos livros de Straus (Straus, 1990) ou Forte (Forte, 1973), não providencia um acesso muito conveniente ao estudo de coleções de altura-classe.

Para implementar tal projeto, escolhi o PHP e o HTML para que a calculadora fosse absolutamente multi-plataforma e terminasse acessível a toda a comunidade musical pela internet sem a necessidade de instalação de um software especial, bastando apenas o software navegador normalmente pré-existente em qualquer computador. Para uma boa introdução ao HTML e ao PHP, ver (Musciano & Kennedy, 2000) e (Lerdorf & Tatroe, 2002), respectivamente.

A tradução para o português das terminologias inglesas da Teoria de Conjuntos e das francesas das teorias de Costère também impuseram algumas dificuldades, principalmente por meu desacordo com algumas das traduções no excelente livro de Marisa Ramires (Ramires, 2001). Mesmo assim, produzi duas versões do programa em PHP, uma em inglês e a outra em português, ambas disponíveis no endereço eletrônico "http://www.dtp.uem.br/musica/Costere".

A seguir descreverei a interface da calculadora analítica, ao mesmo tempo explicitando os processos e análises realizados na construção da página-interface em HTML pelo programa em PHP. Como é sabido, quando o arquivo PHP (que fica escondido do usuário no lado do servidor) é requisitado via internet, o servidor executa as instruções deste arquivo, assim manufaturando uma página em HTML que é depois repassada ao usuário. A página-interface em HTML contém formulários e controles para a entrada e envio de dados referentes à coleção de alturas-classe e desta maneira o usuário prossegue controlando os processos analíticos.

## 3. Descrição da implementação da calculadora.

### 3.1. Entrada de dados.

O topo da interface (ver fig. 2) apresenta dois formulários HTML para entrada dos dados definidores da coleção analisada. Do lado esquerdo temos a entrada de dados no formato tradicional da Teoria de Conjuntos. No campo "*Centro*" entra-se a altura-classe a ser definida como zero, utilizando-se a nomenclatura inglesa (C, C#, Db, E, etc...). Sobre o quadrado (checkbox) sob o título de "*contra alvo*" falaremos mais tarde (no ítem 3.6). O campo "*Coleção-Fonte*" é utilizado para entrar a coleção de alturas-classe a ser analisada, com as alturas-classe separadas por espaços. Se entrados números maiores do que 11, estes serão revertidos ao número equivalente no cálculo em módulo-doze. Aqui também pode-se utilizar como substitutos de "10" e "11" as letras "t" (ten) e "e" (eleven), respectivamente. Também

pode-se aqui entrar notas repetidas, medida conveniente para calcular o efeito de **dobramentos** e **reforços** nas forças de atração gravitacional do sistema. Do lado direito, temos a entrada de dados no formato usado por Costère (como em "*Lois et Styles*", de 1952). No campo "*Código Costère*" entra-se o "**Número Representativo dos Sons Constitutivos**" (Costère, 1952). No campo "*Desvio*" entra-se a transposição em semitonos a partir de Dó do modelo representado pelo código Costère. Para enviar a coleção para análise clica-se no botão "*marcar*" do formulário correspondente.

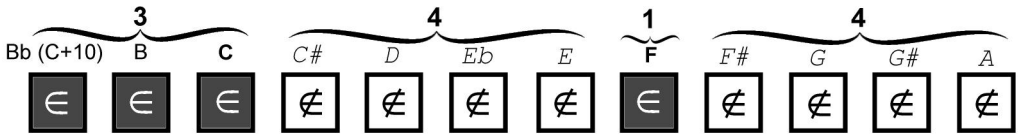



Figura 1 — Exemplificação do "Número Representativo dos Sons Constitutivos", 34 14 (com desvio = 10), do tetracorde inicial, {0 5 10 11} (considerando-se Dó = 0), da peça para piano Op. 33a de Arnold Schoenberg.

A partir dos dados introduzidos será feita a visualização e análise da coleção entrada. A visualização é feita por um mosaico de imagens em tipo PNG montado em formato de tabela HTML e mostrando a notação musical da coleção entrada. Como convenção, as notas acidentadas são enharmonizadas sempre como C#, Eb, F#, G# e Bb, apesar de ser possível a entrada de dados do campo "*Centro*" em outras enharmonizações. A entrada de dados em formato da Teoria de Conjuntos implicará na tradução automática da coleção em formato Costère e seu auto-preenchimento no formulário correspondente. O inverso acontece se a entrada de dados for em formato Costère. Tal tradução automática promove a primeira conveniência fornecida pela calculadora.

Após a entrada dos dados, o modelo computacional calculará internamente as tabelas de densidades e construirá uma database de matrizes contendo valores numéricos relativos a presença e quantidade de alturas-classe, densidades individuais de alturas-classe, densidades de tríades perfeitas, entre outras. Estas matrizes representam e quantificam o estado de todas as forças atrativas (gravitacionais) ativas sobre a coleção entrada.

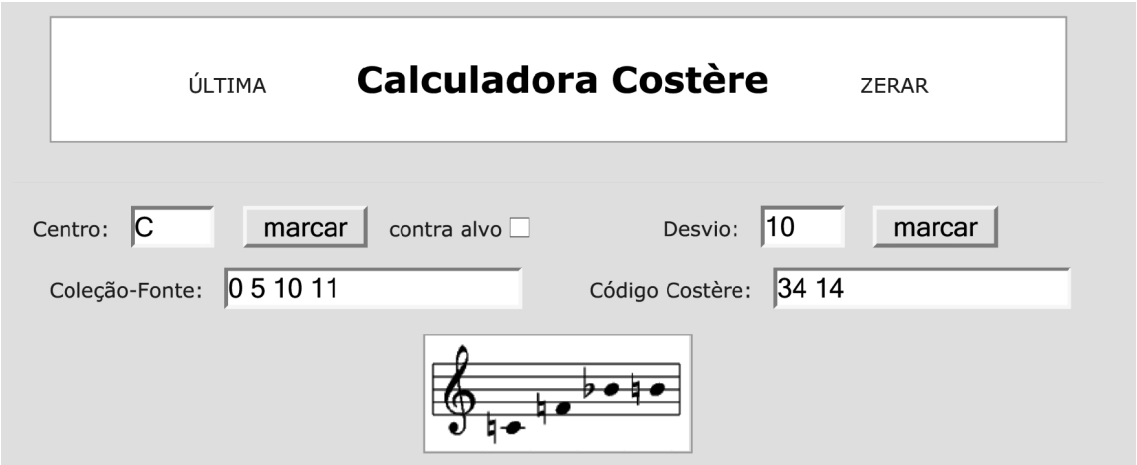



Figura 2 — Área para entrada de dados da interface em HTML com o tetracorde inicial da peça para piano Op. 33a de Arnold Schoenberg como coleção-fonte.

### 3.2. Visualização das análises.

Abaixo da área de entrada de dados e o mosaico contendo a visualização da coleção (ver figs. 3 e 4), temos uma seção da página-interface em HTML mostrando a análise, os diagnósticos analíticos e outras visualizações dos cálculos executados. Inicialmente apresenta-se, nos mesmos moldes utilizados por Costère, as tabelas de densidades cardinais, tonais maiores e menores-inversas, acondicionadas em uma tabela HTML. Por minha própria conta acrescentei a este rol as densidades tonais menores, traduzidas das densidades tonais menores-inversas como auxílio ao usuário que se sentir confuso com a noção de tríade menor-inversa, ítem controverso da teoria de Costère. Em outras duas tabelas HTML são mostradas as tabelas das densidades transposicionais e das inversionais, esta última tabela também acrescentada por minha própria conta por ser perfeitamente extrapolável a partir da idéia de densidade transposicional.

Na primeira tabela de densidades, a linha com as letras estabelece a altura-classe ou fundamental da tríade a que se referem os números das colunas. A primeira linha de cima mostra a quantidade presente na coleção analisada da altura-classe referida pela coluna. A coluna da extrema esquerda estabelece o tipo de densidade mostrada. Por exemplo, a coluna sob a letra C mostra, de cima para baixo, a densidade cardinal da altura-classe Dó, a densidade da tríade maior construída sobre Dó (Dó-Mi-Sol), a densidade da tríade menor construída sobre Dó (Dó-Mib-Sol) e a densidade da tríade menor-inversa construída sob Dó (Dó-Láb-Fá).

Nas duas tabelas imediatamente abaixo (densidades transposicionais e inversionais) são indicadas as densidades de cada transposição e inversão da coleção dada. As transposições são indicadas com as marcações $T_0$, $T_1$, $T_3$, etc... e as inversões com $T_0I$, $T_1I$, $T_2I$, etc... segundo a nomenclatura preferida por Rahn (Rahn, 1980) para a Teoria de Conjuntos. Em todas estas tabelas, os números entre parênteses referem-se a densidades de alturas-classe ou agrupamentos destas que não pertencem (ou seja, são extrínsecas) à coleção analisada. Como um auxílio à leitura das tabelas, as células contendo os valores de densidade máximos em cada quesito (densidades cardinal, tonal, transposicional e inversional) são reforçadas em cinza-escuro.

**Tabela de Densidades:**

| | *1* | *0* | *0* | *0* | *0* | *1* | *0* | *0* | *0* | *0* | *1* | *1* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **C** | **C#** | **D** | **Eb** | **E** | **F** | **F#** | **G** | **G#** | **A** | **Bb** | **B** |
| **Cardinal:** | **3** | **(**1 | 0 | 1 | 2**)** | **3** | **(**2 | 1 | 0 | 1**)** | **3** | **3** |
| **Tonal maior:** | **(**6 | 4 | 3 | 5 | 5 | **7** | 6 | 4 | 4 | 4 | 6 | 6**)** |
| **Tonal menor:** | **(**5 | 3 | 4 | 6 | 6 | 6 | 4 | 4 | 4 | 6 | **7** | 5**)** |
| **Tonal menor-inversa:** | **(**6 | 4 | 4 | 4 | 6 | **7** | 5 | 5 | 3 | 4 | 6 | 6**)** |
| | **T0** | **T1** | **T2** | **T3** | **T4** | **T5** | **T6** | **T7** | **T8** | **T9** | **T10** | **T11** |
| **Transposicional:** | **12** | **(**9 | 5 | 2 | 4 | 9 | 10 | 9 | 4 | 2 | 5 | 9**)** |
| | **T0I** | **T1I** | **T2I** | **T3I** | **T4I** | **T5I** | **T6I** | **T7I** | **T8I** | **T9I** | **T10I** | **T11I** |
| **Inversional:** | 5 | 2 | 4 | 9 | 10 | 9 | 4 | 2 | 5 | 9**)** | **12** | **(**9 |

Figura 3 —Análises do tetracorde inicial da peça para piano Op. 33a de Arnold Schoenberg.

### 3.3. Visualização das análises segundo a Teoria de Conjuntos.

Abaixo das tabelas (ver fig. 4) temos duas colunas onde são apresentados primeiramente e à esquerda, os resultados de análises de características pertinentes à Teoria de Conjuntos, como a **Forma Normal** (a rotação mais compactada da coleção, do grave para o agudo), tanto a da coleção dada como a de sua inversão, os **tipos $T_n$ e $T_nI$** a que a coleção pertence (os moldes básicos que representam, respectivamente, o conjunto de todas as transposições da coleção e o conjunto de todas as transposições e inversões da coleção), seu **Vetor Intervalar** (que mostra a quantidade encontrada na coleção de cada intervalo-classe não-direcionado) e a **invariância** da coleção (número de alturas-classe mantidas inalteradas) após operações de transposição e inversão. Cabe aqui marcar que relações de invariância após transposição também eram uma preocupação de Costère, atendida em suas análises pela "**tabela das transposições vicinais**". O cálculo da Forma Normal segue o algoritmo descrito por Allen Forte (Forte, 1973). No entanto, a nomenclatura usada por Forte em sua tabela de coleções é descartada devido à sua incômoda leitura. Aqui preferiu-se a convenção numérica geralmente aceita (Rahn, 1980), marcando-se as coleções com os números das alturas-classe entre chaves para indicar a Forma Normal, entre parênteses para indicar o tipo $T_n$ e entre colchetes para indicar o tipo $T_nI$. Na tabela de invariâncias por transposição e inversão, as células indicando invariância total são reforçadas em cinza-escuro, como auxílio à leitura.

A seguir são realizados alguns diagnósticos pertinentes tanto à Teoria de Conjuntos como à de Costère, analisando-se a invariância e o conteúdo intervalar da coleção. A coleção será considerada "**de conteúdo intervalar limitado**" se em seu Vetor Intervalar não estiverem presentes todos os tipos de intervalos-classe não-direcionados e será "**de conteúdo intervalar múltiplo**" se ocorrer o contrário. A coleção será "**de transposições limitadas**" se em alguma de suas transposições (excetuando-se a $T_0$, logicamente) ocorrer invariância total e será "**de transposições ilimitadas**" se não ocorrer invariância total em nenhuma transposição (exceto a $T_0$). Da mesma maneira, interpreta-se a ocorrência ou não-ocorrência de invariância total por operações de inversão classificando-se a coleção como sendo "**de inversões limitadas**" ou "**de inversões ilimitadas**", respectivamente.

### 3.4. Visualização de diagnósticos segundo as teorias de Costère.

Nesta mesma área central da interface HTML, mas na coluna da direita, aparecerão diagnósticos sobre a coleção analisada, segundo as teorias de Costère, que são os resultados da interpretação de diversos testes realizados pelo modelo computacional sobre os dados das tabelas de densidades da coleção analisada. A detalhada descrição dos testes realizados servirá aqui certamente como resumo da engenhosa tipologia de características criada por Costère.

#### 3.4.1. Investigação das relações de simetria e reversibilidade.

Se houver ocorrência de invariância total por inversão, a coleção será considerada "**simétrica a ela mesma**", ou seja, a coleção possui a mesma sequência de intervalos acima e abaixo de um eixo imaginário. Se este eixo passar por cima de uma altura-classe este será um "**eixo mediano**". Caso o eixo não passe por cima de nenhuma altura-classe mas sim no meio de duas alturas-classe adjacentes este eixo será um "**eixo intercalar**". Os eixos serão indicados na análise pelas alturas-classe por onde passam, referidas pelas letras-nome das notas e pelos números destas em formato da Teoria de Conjuntos, segundo a altura-classe escolhida como centro (zero). Cabe frisar aqui que a cada eixo corresponde um outro à distância de um trítono.

Se os eixos forem medianos e pelo menos um destes pertencer à coleção analisada, a coleção é marcada como "**com nota mediana**", sendo que esta nota tem a possibilidade de funcionar como um pivô de simetrias. No caso das coleções simétricas a elas mesmas, a transposição da inversão que gerou a invariância total é indicada como uma coleção "**relativa**". Caso a coleção não seja simétrica a ela mesma, ela é considerada "**assimétrica**" e o tipo $T_n$ da coleção inversa (e também chamada por Costère como relativa) é indicado, junto com seu código Costère. A indicação da coleção inversa possui um link para a análise dela mesma.

Em seguida, calcula-se e indica-se a **coleção complementar** à analisada, ou seja, aquela coleção cuja soma com a coleção analisada equivale à totalidade dos doze sons cromáticos. A indicação da coleção complementar também possui um link para a análise dela mesma. O teste que se segue verifica a natureza desta coleção complementar. Se a coleção complementar for equivalente somente por transposição à coleção analisada, esta coleção é considerada "**reversível a ela mesma**". Se a coleção complementar for equivalente somente por inversão à coleção analisada, esta coleção é considerada "**reversível à sua inversão**". Se a coleção complementar for equivalente tanto por transposição como por inversão à coleção analisada, esta coleção é considerada "**reversível a ela mesma e à sua inversão**".

### 3.4.2. Investigação das relações de densidade.

Em seguida acontecem os testes avaliando as características de densidade da coleção analisada, o que envolverá agenciar praticamente toda a teoria contida no livro de 1954 de Costère.

#### 3.4.2.1. Relações cardinais de densidade.

Primeiramente, avalia-se se a coleção é "**Cardinalmente Densa**" ou "**Cardinalmente Transitiva**". Caso o número de elementos da coleção seja menor ou igual ao número de elementos da sua coleção complementar, compara-se a soma das densidades dos elementos intrínsecos à coleção com a soma das densidades do mesmo número dos elementos mais densos extrínsecos a esta. Caso o número de elementos da coleção seja maior do que o da sua coleção complementar, compara-se a soma das densidades dos elementos extrínsecos com a soma do mesmo número dos elementos intrínsecos menos densos. Se a soma dos elementos intrínsecos à coleção analisada considerados for maior ou igual à soma dos elementos extrínsecos considerados, esta coleção é considerada "Densa". Caso contrário, ela é "Transitiva".

O próximo teste verifica a relação entre as densidades individuais dos sons constitutivos da coleção e de seus sons extrínsecos. As notas mais densas da tabela cardinal são consideradas pólos de atração cardinal e, se existir apenas um pólo e este for uma nota intrínseca (constitutiva), esta coleção é considerada "**com pólo cardinal tônico**". Se houver mais de uma nota constitutiva com este valor mais denso, esta coleção é considerada "com pólos cardinais tônicos" caso estes pólos sejam em número de dois e a coleção seja simétrica a ela mesma (pois nesse caso os pólos são sempre duplos). Caso estes pólos sejam em número de dois e a coleção for assimétrica ela é considerada "**com pólos cardinais politonais**". Se os pólos forem em número maior do que dois mas sem totalizar o número de notas constitutivas, a coleção também é considerada "com pólos cardinais politonais".

Se os pólos forem notas extrínsecas à coleção, esta é considerada "**com pólo cardinal extrínseco**", caso o número de pólos seja um, ou considerada "com pólos cardinais extrínsecos", se o número de pólos for igual a dois e a coleção for simétrica a ela mesma. Se houverem tanto pólos constituintes como extrínsecos à coleção, esta é considerada "**com pólo cardinal equilibrado**", caso o número de pólos constituintes seja um, ou considerada "com

pólos cardinais equilibrados", caso o número de pólos constituintes seja dois e a coleção for simétrica a ela mesma.

Se a coleção possuir pólos extrínsecos e existir uma nota (ou duas no caso de uma coleção simétrica a ela mesma) dentre as constituintes cuja densidade se destaca das outras notas intrínsecas mas sem no entanto se igualar à densidade dos pólos extrínsecos, esta coleção é considerada "**com pólo cardinal cadencial**" ou "com pólos cardinais cadenciais" conforme a ocorrência de assimetria ou simetria, respectivamente. Para terminar, se todas as notas constituintes tiverem a mesma densidade, a coleção é considerada "**em equilíbrio cardinal**".

Em todos os casos, se existirem pólos estes são indicados tanto pela letra-nome de sua nota como pelo valor de sua altura-classe, segundo o zero escolhido.

**3.4.2.2. Relações transposicionais de densidade.**

A seguir, são realizados testes quanto à estabilidade transposicional da coleção analisada. Se a densidade da $T_0$ da coleção analisada for igual à maior densidade da tabela de suas densidades transposicionais, esta coleção é diagnosticada como "**Não-Transposicional**", ou seja, ela não tem tendência para gravitar em direção a uma transposição de si mesma. Se a maior densidade dentre as transposições não for a da $T_0$, esta coleção é considerada "**Transposicional**" e as transposições mais densas são indicadas.

**3.4.2.3. Relações de conteúdo tonal e de densidade tonal.**

Seguem-se testes sobre as características Tonais da coleção. Se dentre os sons intrínsecos à coleção não se encontrem tríades perfeitas ou ainda mesmo quintas justas, esta coleção é considerada "**Atonal**". Havendo possibilidades de formação de tríades perfeitas dentro da coleção analisada, esta é considerada "**Tonal**". Se não houverem tríades perfeitas dentro da coleção analisada mas existirem quintas justas, esta é considerada "**Neutra**".

Se a coleção for Tonal, possuir **componentes binários** (ou seja, quintas justas possíveis de serem mediadas tanto com terças maiores como menores) e a tríade intrínseca mais densa for um dos componentes de um par binário, a coleção será marcada "**Binária-M**", caso a tríade maior do par predomine, "**Binária-m**", se a menor predominar, ou "**Binária-Mm**", se a densidade das tríades do par binário for igual.

Após este teste e se a coleção for Tonal ou Neutra, avalia-se ainda a densidade das tríades constitutivas, no caso de uma coleção tonal, ou a densidade das tríades extrínsecas que incluirem as quintas justas dos componentes neutros, no caso de uma coleção Neutra. Se a tríade avaliada mais densa for maior, a coleção é marcada "**Maior**", se esta tríade for menor, a coleção é marcada "**menor**", e se não houver predomínio entre as tríades avaliadas, a coleção é marcada "**Maior-menor**".

Em seguida avalia-se a estabilidade Tonal da coleção. Aqui, as tríades perfeitas que possuirem o maior valor de densidade da tabela de densidades serão consideradas pólos tonais. Para os testes seguintes, consideraremos como pertencentes à coleção tanto as tríades perfeitas efetivamente intrínsecas, no caso de coleções Tonais, como as tríades perfeitas extrínsecas que incluirem as quintas justas dos componentes neutros intrínsecos, no caso de uma coleção Neutra. Se houverem apenas pólos tonais pertencentes à coleção analisada, esta será marcada "**Tonalmente Estável**". Se os pólos tonais forem somente não-pertencentes, a coleção é considerada "**Tonalmente Explosiva**" (versão minha do termo francês "*détonnante*" utilizado por Costère). Caso a coleção seja Explosiva e existir uma tríade não-pertencente (ou duas no

caso de uma coleção simétrica a ela mesma) cuja densidade ultrapassa a de todas as outras, esta coleção será "**Tonalmente Explosiva Cadencial**" e esta tríade pólo configurar-se-á como o foco tonal das forças atrativas do sistema. No caso de existirem tanto pólos pertencentes como não-pertencentes, a coleção é diagnosticada como sendo "**Tonalmente Equilibrada**".

O próximo teste verifica, avaliando apenas as tríades perfeitas intrínsecas à coleção analisada (para uma coleção Tonal) ou as tríades perfeitas extrínsecas que contêm as quintas justas dos componentes neutros intrínsecos à coleção (para uma coleção Neutra), se existe uma (no caso de coleções assimétricas) ou duas (no caso de coleções simétricas a elas mesmas) dentre estas tríades cujas densidades se sobrepõem às demais avaliadas. Se tal for o caso, a coleção é chamada "**Tônica**" e estas tríades são indicadas pelo valor da altura-classe de sua fundamental, seguida de sua letra-nome e do indicativo da modalidade: "M" para maior, "m" para menor. Se existir mais de uma (no caso de coleções assimétricas) ou mais de duas (no caso de coleções simétricas a elas mesmas) dentre estas tríades cujas densidades ultrapassem as demais avaliadas, a coleção é considerada "**Não-Tônica**" e ela será ainda marcada "**neutra**", "**politonal com componente binário**" ou simplesmente "**politonal**", caso a coleção analisada seja Neutra, seja Tonal com algum componente binário, ou seja Tonal sem componentes binários, respectivamente.

### 3.4.3. Investigação das relações funcionais imitativas.

Os próximos testes verificam as **características modulantes** da coleção analisada, no sentido muito particular empregado por Costère. Explicando melhor este conceito, Costère coloca que caso existam várias tríades perfeitas (ou quintas justas) intrínsecas a uma coleção e caso forem iguais os padrões numéricos de notas intrínsecas existentes nos espaços entre a fundamental, terça, quinta e fundamental oitavada destas tríades (ou entre fundamental, quinta e fundamental oitavada de um componente neutro), um perfil melódico poderia ser imitado sobre cada uma destas sonoridades tonais (triádicas ou neutras), independentemente de suas densidades, assim efetivamente passando o tal perfil melódico por uma mudança de modo, enfim, por uma "**modulação**". Costère não usa aqui exatamente o termo "**imitação modal**" mas é basicamente esta a idéia que ele sugere para este tipo de propriedade. Os testes aqui verificam a configuração de notas intrínsecas entre cada nota das tríades ou quintas justas de uma coleção e, caso todas as configurações sejam iguais, a coleção é considerada "**integralmente imitativa-modal**". Se algumas destas configurações forem iguais mas outras não, a coleção é considerada "**parcialmente imitativa-modal**" e as entidades tonais ou neutras passíveis deste tipo de modulação entre si são indicadas entre colchetes pelo valor das alturas-classe de suas fundamentais, seguidas de suas letras-nome e dos indicativos de suas modalidades: "M" para maior, "m" para menor, "n" para neutra.

A última sessão de testes envolvendo as teorias de Costère verifica se existe um mesmo padrão numérico de notas intrínsecas nos espaços entre as notas de uma mesma tríade (ou quinta justa) intrínseca à coleção analisada. Isto possibilitaria o que Costère chama de "**imitação tonal**", ou seja, a imitação de um perfil melódico em outra região da coleção, mas sobre a mesma sonoridade tônica, desta maneira sem ocorrer mudança de modo, como na "imitação modal". Se todas as sonoridades triádicas ou neutras apresentarem tal propriedade, a coleção é marcada "**integralmente imitativa-tonal**". Caso apenas algumas sonoridades triádicas ou neutras apresentem esta propriedade, a coleção é "**parcialmente imitativa-tonal**". Após este diagnóstico (e se imitações tonais forem possíveis), as entidades tonais passíveis de imitação tonal são indicadas pelo valor das alturas-classe de suas fundamentais, seguidas de suas letras-nome e dos indicativos de suas modalidades: "M" para maior, "m" para menor, "n"

para neutra. Continuando este processo, ainda é indicado o tipo de imitação tonal possível sobre cada uma das sonoridades localizadas, dentre cinco tipos: a) existência de um número igual de graus entre todas as notas da sonoridade tonal; b) existência de uma configuração igual de graus entre o par [fundamental-5ª-8ª] e [3ª-8ª-10ª]; c) existência de uma configuração igual de graus entre o par [fundamental-5ª-8ª] e [5ª-10ª-12ª]; d) existência de uma configuração igual de graus entre o par [fundamental-5ª] e [5ª-8ª]; e) existência de um número igual de graus entre o par [fundamental-3ª] e [3ª-8ª].

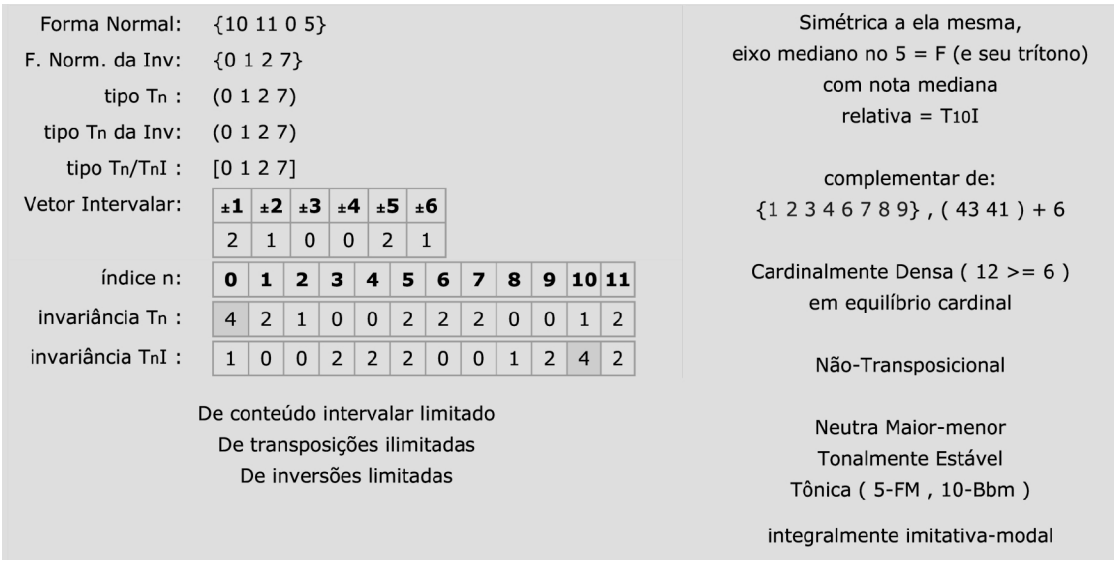



Figura 4 —Análises e diagnósticos do tetracorde inicial da peça Op. 33a de Schoenberg.

### 3.4.4. Conclusão dos diagnósticos segundo as teorias de Costère.

Após todos estes testes a coleção termina avaliada segundo todos os quesitos propostos por Costère em seu livro de 1954, com resultados praticamente idênticos aos grafados nas "Tableaux Analytiques des Échelonnements" presentes no final da mesma obra. Algumas esparsas divergências ocorrem devido a algumas ambigüidades de interpretação encontradas no próprio texto de Costère (envolvendo por exemplo o escopo da transferência de propriedades tonais às coleções neutras) e também devido ao fato de que foram intencionalmente eliminados testes visando localizar propriedades de imitação modal ou tonal envolvendo sonoridades como a tétrade de sétima de dominante e sua inversão, a tétrade meio-diminuta. Apesar do interesse de Costère nessas sonoridades, elas foram desconsideradas como entidades tonais (no sentido empregado por Costère) devido a seu caráter controverso, pois a sétima menor no nosso temperamento igual é muito distante do sétimo harmônico da série harmônica para que uma fusão tímbrica total ocorra, a meu ver.

### 3.5. As tabelas de resumo.

A página de HTML continua com um resumo das propriedades transposicionais da coleção analisada (ver fig. 5), mostrando a invariância e densidade de cada transposição e indicando, tanto em notação padrão da Teoria de Conjuntos como em letras-nome, as alturas-classe de

cada transposição, com as notas comuns (invariadas) marcadas em negrito. Cada coleção transposta contém um link remetendo à sua própria análise. A isto segue um resumo das propriedades inversionais, nos mesmos moldes.

A interface HTML termina com a montagem, no mesmo sistema de mosaico de imagens PNG descrito anteriormente, de todas as notações musicais das transposições e inversões da coleção analisada, com as notas comuns (invariadas) marcadas com cabeças brancas e as demais, com cabeças pretas.

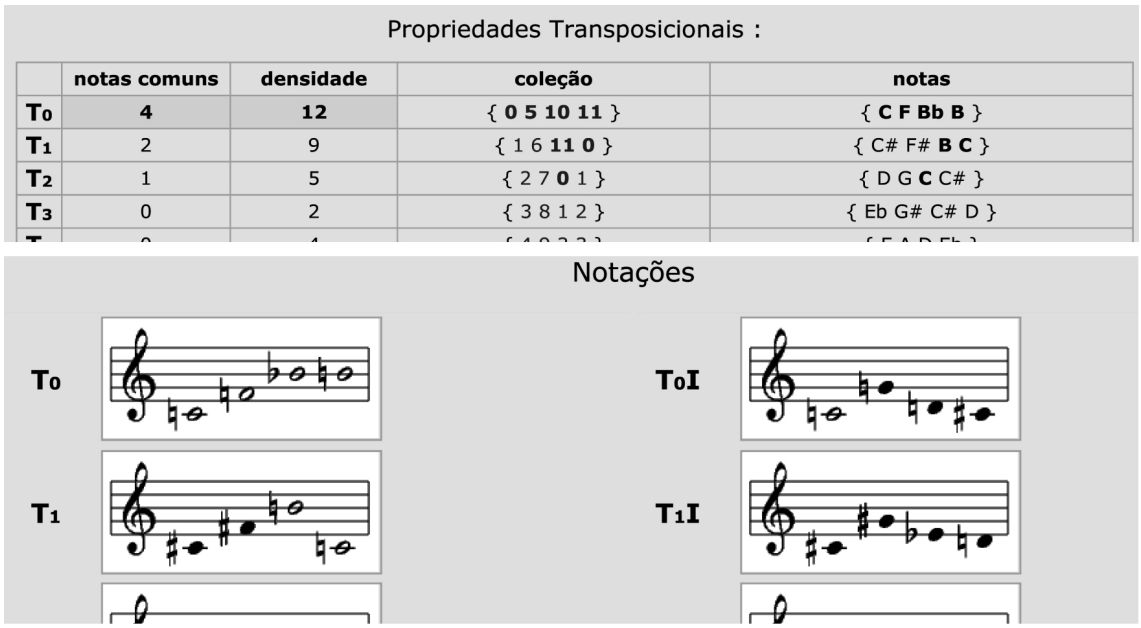



Figura 5 —Fragmentos das seções de notações e de resumos das propriedades transposicionais da calculadora, analisando o tetracorde inicial da peça Op. 33a de Schoenberg.

### 3.6. Sobre o checkbox "*contra alvo*".

Voltando à questão do quadrado (checkbox) marcado "*contra alvo*" no formulário de entrada de dados (ver fig. 6), aquele controle possibilita a entrada de uma segunda coleção. Nesta variante, todas as relações transposicionais, inversionais, de invariância e de densidades transposicional e inversional são feitas em relação a uma segunda coleção, dita "alvo". Todas as demais informações e diagnósticos continuam a se referir à primeira coleção, dita "fonte". A serventia deste processo consiste principalmente em evidenciar as relações de invariância e de fluxos atrativos entre a $T_0$ da coleção fonte e todas as transposições e inversões da coleção alvo. Os resumos e notações também mostram a coleção "alvo", ainda marcada com negritos e notas de cabeça branca segundo a presença de relações de invariância com a $T_0$ da coleção "fonte".

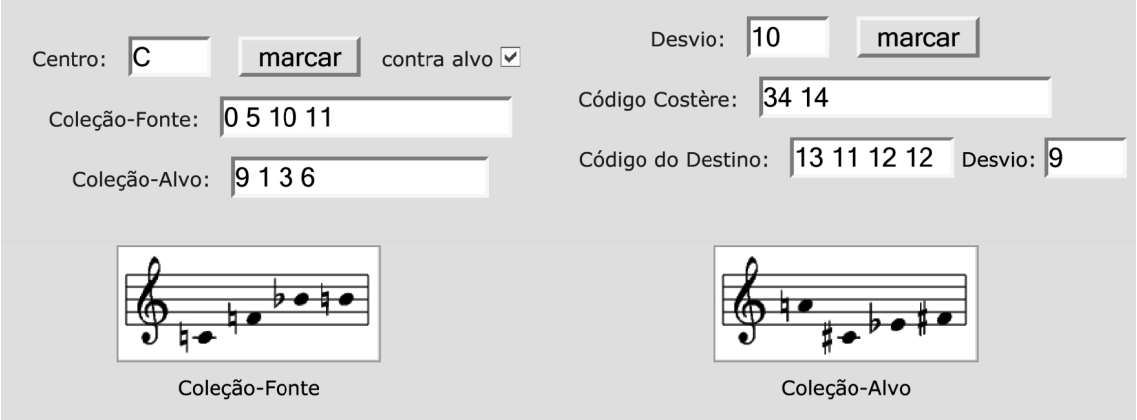



Figura 6 —Visão da área de entrada de dados, marcada " *contra alvo*" e tendo como fonte e alvo os dois tetracordes iniciais da peça Op. 33a de Schoenberg, respectivamente.

## 4. Considerações finais e prognósticos futuros.

Esta calculadora tem sido utilizada satisfatoriamente por mim tanto para fins composicionais, facilitando o estudo das propriedades combinatórias de coleções de alturas-classe, como para fins de auxílio à análise de repertório do século vinte nas disciplinas de análise musical que ministro. Além da velocidade de colheita dos dados analíticos, a razoável qualidade gráfica da calculadora permite a rápida fatura de material instrucional e de material de apoio à composição, tais como tabelas de propriedades, listagem em notação musical de todas as formas, transposições e inversões de séries (inclusive dodecafônicas) e tabelas de relações de invariância entre coleções por transposição e inversão.

Sobre ramificações futuras deste projeto, o código básico da calculadora pode ser facilmente adaptado para criar um software capaz de gerar, em qualidade excelente de impressão, um catálogo ou tesauro de todas as coleções de alturas-classe possíveis, em formato expandido ou semelhante ao da interface HTML aqui apresentada. Uma outra possibilidade interessante que vislumbrei para um aplicativo "irmão" a este apresentado seria a criação de uma espécie de "lista telefônica reversa", ou seja, uma ferramenta em que seria possível entrar as características analíticas desejadas e recobrar uma listagem das coleções capazes de atender àquelas propriedades.

## 5. Referências.

COSTÈRE, Edmond. *Lois et Styles des Harmonies Musicales*. Paris: Presses Universitaires de France, 1954.

COSTÈRE, Edmond. *Mort ou Transfigurations de l'Harmonie*. Paris: Presses Universitaires de France, 1962.

ELLARD, Brian. *Edmond Costère's Lois et Styles des Harmonies Musicales, an English translation and commentary*. Rochester: University of Rochester PHD thesis, Eastman School of Music, 1973.

FORTE, Allen. *The Structure of Atonal Music*. New Haven: Yale University Press, 1973.

HANSON, Howard. *Harmonic Materials of Modern Music*. New York: Appleton, Century and Crofts, 1960.

HINDEMITH, Paul. *The Craft of Musical Composition*. New York: Associated Music Publishers, 1945.

LERDORF, Rasmus & TATROE, Kevin. *Programming PHP*. Califórnia: O'Reilly, 2002.

MESSIAEN, Olivier. *Technique de mon language musical*. Paris: Leduc, 1944.

MUSCIANO, Chuck & KENNEDY, Bill. *HTML and XHTML: the Definitive Guide*. Califórnia: O'Reilly, 2000.

OLIVEIRA, João Pedro Paiva. *Teoria Analítica da Música do Século XX*. Lisboa: Gulbenkian, 1998.

PERLE, George. *Serial Composition and Atonality*. Califórnia: University of California Press, 1968.

RAHN, John. *Basic Atonal Theory*. New York: Schirmer Books, 1980.

RAMIRES, Marisa. *A Teoria de Costère, uma perspectiva em análise musical*. São Paulo: Embraform, 2001.

SCHOENBERG, Arnold. *Klavierstück op. 33a*. Viena: Universal Edition, 1929.

SLONIMSKY, Nicolas. Thesaurus of Scales and Melodic Patterns. New York: Coleman-Ross Co., Inc., 1947.

STRAUS, Joseph Nathan. *Introduction to Post-Tonal Theory*. New Jersey; Prentice Hall, 1990.

VALE, Sydney. *A Defence of Edmond Costere*. Music & Letters, Oxford University Press, Oxford, Vol. 45, No. 3, pg. 311-312, Julho de 1964.

W., J. A. *Mort ou transfigurations de l'harmonie by Edmond Costere*. Music & Letters, Oxford University Press, Oxford, Vol. 44, No. 4, pg. 380-381, Outubro de 1963.

## 6. Link para a calculadora.

http://www.dtp.uem.br/musica/Costere

# Sound Synthesis based on Semantic Descriptors

**Cesar Costa[1,3], Fábio Furlanete[1,2], Jônatas Manzolli[1], Fernando Von Zuben[3]**

[1]Núcleo Interdisciplinar de Comunicação Sonora (NICS) - Universidade Estadual de Campinas (UNICAMP) Rua da Reitoria, 165. CP 6166 – Campinas, 13091-970, BRAZIL

[2]MUT – Universidade Estadual de Londrina (UEL)
Rod. Celso G. Cid, Km 375 Londrina, BRAZIL

[3]LBIC / DCA / FEEC / Universidade Estadual de Campinas (UNICAMP), Bloco G2 - Sala LE 14G, CP 6101 Campinas, 13083-970, BRAZIL

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

***Abstract.*** *One of the main issues on sound design processes is how to describe the intended sound result, and how to use such description in the synthesis procedure. With the augmented interest on multimedia description notation, with standards like MPEG-7, we suggest the use of such scheme that aims at characterizing audio content to establish control reference in a synthesis process. In this paper we present an interactive sound design methodology with a user-defined semantic description paradigm. We briefly discuss audio notation and present our approach, implementation issues, and results.*

***Resumo.*** *Um dos principais pontos do processo de design sonoro é como descrever o resultado sonoro desejado, e como usar esta descrição no processo de síntese. Com a ampliação do interesse em notação descritiva para multimídia, como o protocolo MPEG-7, sugere-se o uso deste tipo de mecanismo que tem como objetivo caracterizar o conteúdo do áudio como uma forma de estabelecer controle no processo de síntese. Neste artigo apresenta-se um método de design sonoro vinculado a um paradigma de descrição semântica fornecido pelo usuário. É discutida a notação de áudio e são apresentados a nossa abordagem, aspectos da implementação, e resultados obtidos.*

## 1. Introduction

Recently, multimedia content description has become an effervescent research area. The main worry is to establish how multimedia material could be described promoting general understanding and accessibility. Industry standard such as MPEG-7 has already emerged as a promising candidate, as emphasized by studies on multimedia metadata on the Web (van Ossenbruggen el all, 2005), the *Cuidado* music browser (Pachet et all, 2003), and the development of a music content semantic annotator like MUCOSA (Herrera et all, 2005).

Under the possibility of establishing a descriptive standard for audio, associating qualitative parameters for indexing and network retrieval, the same parameters can be used to describe the intended result in a sound synthesis process.

An ideal interface for sound synthesis would effectively adapt to the requests of the user, so that the obtained and the intended sound materials would express noticeable resemblance. However, for such accomplishment, sound description and expectations of the designer should be somehow absorbed by the software. This is actually done with acoustical or psychoacoustical feature analysis. But it is not only a translation or mapping problem here: the system should be able to deal with subjective sound evaluations. We have already proposed the use of existent sound material as reference (Costa et all, 2006), and this paper presents further development on this direction.

The idea of associating qualitative descriptions to sounds is not new. If we rewind at least to the 1950s we will find the efforts made by the Group d'Essay (Schaeffer, 1966) to build sound typologies and morphologies based on qualitative parameters (Smalley, 1986). Some works like (Nicol et all, 2006), which mapped FM synthesis into a timbre space defined by linguistic qualities, and (Johnson & Gounaropoulos, 2006), who also used linguistic expressions but now implementing a direct control of a computational audio processing engine, have dealt with this possibility. Nevertheless, we go further by not establishing a fixed semantic set but, as in early time of electroacoustic composition, we let it open to be adapted to designer's sonic experience and to be associated with a creative project without using a direct parametrical control. Thus, we present a synthesis process named *Semantic Synthesis* where high-level qualitative parameters are used to control a generative process. Using artificial intelligence algorithms, including machine learning and adaptive systems, we implemented a synthesizer that adapts itself to user's qualitative expressions toward a more intuitive interaction in a sound design process.

In this paper, the state-of-the-art on audio notation is firstly discussed. Next, we present a methodology to achieve user-orientation on a sound design process with a briefly exposition on how artificial intelligence techniques can be helpful. Further, the *Semantic Synthesis* architecture is described. Preliminary results and considerations on future directions and potential applications are included in the concluding remarks.

## 2. AUDIO NOTATION REVIEW

Recently, there is an increasing interest on developing a mature formalism on description and analysis of sound contains. It varies from physical characteristics retrieved by sound analysis methods such as (Loureiro & Serra, 1997), to human meaningful relations like: what's being said – sound retrieval by speech recognition (Kurihara et all, 2006) - or who's saying – audio notation techniques (Turnbull et all, 2006).

Audio notation techniques is based on the use of keywords related to sound parametrical analysis. It's a tool for storing descriptive data originated from any kind of method, including both traditional low-level features and high-level contextual meaning.

Martínez (2004) has described the MPEG-7 standard as: formally named "Multimedia Content Description Interface", it is a standard for describing the multimedia content data that supports some degree of interpretation of the information meaning, which can be passed onto, or accessed by, a device or a computer code. It uses meta-data structures

named Multimedia Description Scheme (MDS) defined with a Description Definition Language (DDL) based on XML and permits both text visualization humanly readable and compressed binary coded. MPEG-7 has a specific framework and basic semantics for commonly regarded low-level audio features such as spectrum envelope, attack time or harmonicity level. CLAM Library for Audio and Music (Amatriain et all, 2002) [http://mtg.upf.edu/clam/] includes data structure and tools for semantic analysis but let opened which features to be described.

## 3. USER ORIENTED SOUND DESIGN

Historically, synthesis processes have benefited from low-level analysis method. In fact, audio coding like MP3 is a direct appliance of analysis/synthesis practices. The association between a parameter and the audio signal is ruled by physical and mathematical relations, and therefore tend to generate universal models.

Such association earns complexity with high level features such as qualitative parameters. Hence, there are few models or tools for automatic analysis and even less regarding synthesis. Comparatively to other senses, auditory qualification is the only one that does not have a standardized qualitative set. In fact, other senses expressions like bright (visual) or rough (tact) are commonly borrowed. Not enough, the way different people link such expressions with real sound attributes seems to vary, being their culture references influenced by past experiences.

In this context, the Semantic Synthesis we introduced here aims to digitally generate sounds related and controlled by a model which entails a qualitative description. This description model consists in a set of fuzzy values generated by the user according his/her way of perceiving and qualifying the sound. Beyond a global psychoacoustic model, we have local sets of parameters individually shaped by the listeners experience and context. In order to active our goal and take advantage of artificial intelligence algorithms, the computational models used in our synthesis engine must be designed to (re)adapt themselves to each specific description set. The research area that deals with algorithms capable of improving themselves automatically through experience is known as *Machine Learning* (Mitchell 1997).

## 4. SYSTEM ARCHITECTURE

### 4.1 Semantic Sound Trajectories

The main concept of our system is named as “sound trajectories”. The user starts by defining a set of parameters (terms) he/she wants to work with. It is a kind of personal dictionary. Them he/she will listen to sounds and notate a collection of samples according to the dictionary. It is done by scoring pre-defined sound windows with values in the interval [0..1] according to a personal evaluation which is based on the dictionary.

Let us consider a collection of sound $\mathbf{S} = \{s_1, s_2, s_3, \ldots, s_N\}$ with **N** samples and a set of keywords $\mathbf{W} = \{w_1, w_2, w_3, \ldots, w_M\}$ with **M** words pre-defined by the user. For each sound in **S** we define a *Semantic Sound Trajectory (SST)* $\mathbf{T^k}$ with k=1…**N** as follows:

$$\mathbf{T^k} = \{(\Delta t_1, G_1(n_1)), (\Delta t_2, G_2(n_2)), \ldots, (\Delta t_P, G_P(n_P))\} \quad \textbf{Eq.1}$$

where each pair ($\Delta t_j$, $G_j(n_j)$) for j = 1..**P** is provided by the user and they are named as "moments". $\Delta t_j$ is a pre-defined time window and $G_j(n_j)$) is a score mark.

As previously said, it contains the user personal evaluation on how a time window ($\Delta t_j$ ) is related to a keyword from the dictionary **W**. Thus, from the user point of view, the design process consists on three steps:

- **Step 1:** start with a set of samples, define the keyword dictionary, create the SST for each sound. The SSTs will be the model to the system's data base associating keyword scoring to sound fragments, defining a space with **K** semantic trajectories $\mathbf{T} = \{ \mathbf{T^1}, \mathbf{T^2}, \ldots, \mathbf{T^K} \}$.
- **Step 2:** feed the systems data base with $\mathbf{T} = \{ \mathbf{T^1}, \mathbf{T^2}, \ldots, \mathbf{T^K} \}$. The trajectory correspondent to each notated sound file is automatically stored in the data base.
- **Step 3:** use the system to create new sound moments ($\Delta t_j$, $G_j(n_j)$). This synthesis process of new sound fragments as well as its interpolations in a new trajectory is handled by the system.

Using the SSTs our goal is to be able to describe sound qualities at several levels, from granular scope (microsound) to a whole sonic event (macrosound). Quality evolution in certain scale will influence the emergence of other qualities in a wider scale. For example, a fast evolution on grains energy determines a specific type of attack. Further, a fast attack followed by a slow decay and a little sustain defines the sonority of an individual sonic event. This is the very basis of sound morphology and typology.

The time evolution of a quality description can be defined as a trajectory in the quality space which is a sub-set of **T**. Trajectories characteristics may define other qualities in another sub-set, which could have its own trajectories as well. Figure 1 shows a graphical illustration of this multi-level interplay. Each screen represents a parametric quality space in a different level. The circles represent instantaneous quality points while the curves represent a trajectory in the space.

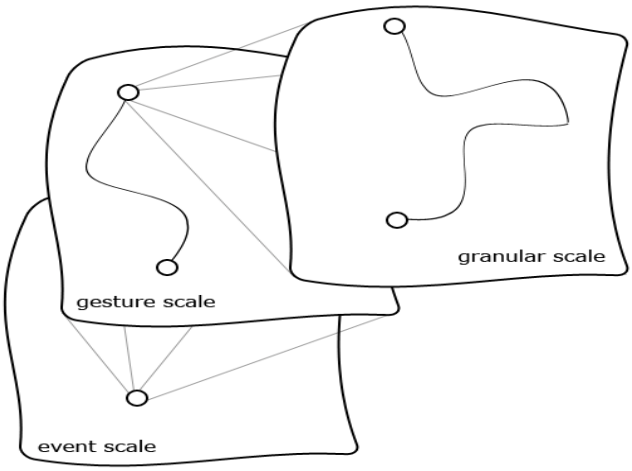



**Figure 1. Graphical illustration of the interaction of multi-level quality spaces**

### 4.2 Synthesis Process

The synthesizer has three main structures:

- *Description schema*: data structure that operates as an interface between semantic description and software;
- *Synthesis Engine*: general purpose audio synthesizer;
- *Translation Unit*: adaptive computational structure that for a given description returns parametric values for the synthesis motor.

The way these structures interact to synthesize audio is presented in Figure 2. The synthesizer receives a semantic description formatted into a description schema as input. The translation unit converts this description into control parameters of the synthesis engine in order to generate sound outputs.

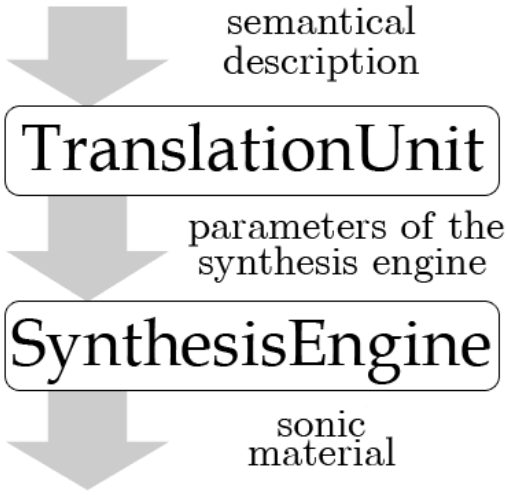



**Figure 2. Synthesis Process**

Both synthesis engine and description schema are pre-defined structures with well known behavior. The translation unit is an adaptive structure modified by learning machine algorithms, which performs statically during the synthesis process. It maps the description space into the synthesis motor parametric space. Due to mapping variation from user to user, the translation unit is always different.

To learn space mapping, the translation unit is trained with mapping samples, i.e., semantic descriptions and the sound material they represent. The learning procedure is presented in Figure 3.

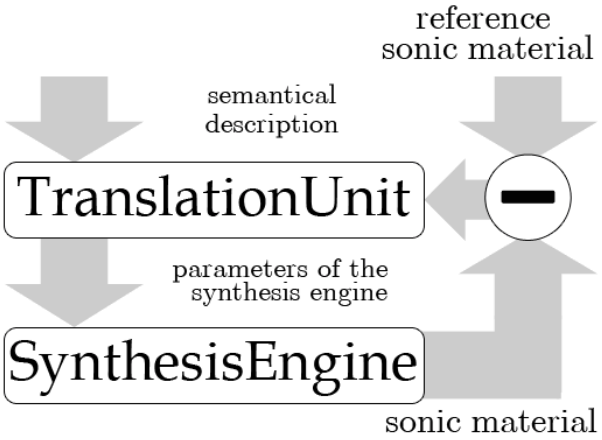



**Figure 3. Learning Process**

### 4.3 Implementation Considerations

A semantic synthesizer working with single-level granular scale trajectory has been fully implemented as a C++ shared library. It uses an abstract structure as synthesis engine for modular implementation of any general solution and an artificial neural network as the translation unit.

*4.3.1 Description Schema*

In a single-level quality space schema it could be structured parametrically. The schema is composed of a set of adjectives. A quality point is defined as a list of numerical values associated with the adjectives. For example, one may select the adjectives *heat, roughness, brightness* as the description schema and a quality point could be defined as (0.7, 0.5, 0.1), respectively for heatness, roughness and brightness.

*4.3.2 Translation Unit*

A multi-layer perceptron (MLP) artificial neural network with back-propagation to implement the learning algorithm is used as the translation unit. It is based on the fast artificial neural network (FANN) library (Nissen 2003). MLP works as a universal function interpolator, i.e., it maps a set of sampled points into a functional space. Here, the task is to approximate the function that relates a point from the qualitative parametric space to a point in the audio sample space.

*4.3.3 Synthesis Engine*

MLP is used to generate sound material directly from semantic description. However, MLP performance and capability to generalize any function is affected by space dimensionality and the quality of the available dataset. That is why we have adopted a Discrete Fourier Transform to implement the synthesis engine.

*4.3.4 Trajectories Interpolation*

User is expected to define the semantic trajectory by means of defining quality points. A quality point is composed of a sequence of qualitative parameters and the equivalent

time stamp. A complete curve of qualitative values is generated by linear interpolation of such points in the parametric space.

*4.3.5 Trajectories Translation*

The output of the system is an audio stream that refers to the parametric trajectory. The translation is done in a granular size window scale, and the final stream is obtained using the overlap-and-add technique. The translation is done by presenting the instantaneous parameter to the MLP and converting the resultant floor and residue into a sound window.

## 5. Results

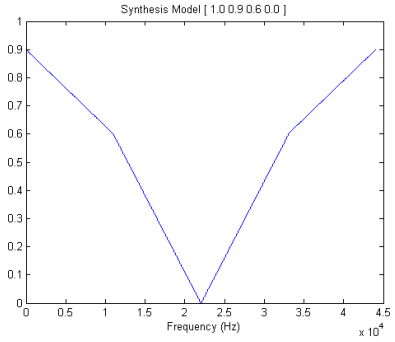
The system is expected to be capable of generalizing any given quality space to sound material mapping. This could be tested by two manners: with parametric synthesizers with well-known behavior; with a real human mapping and hearing sections procedures.

Using parametric synthesizers, it is possible to verify the system capability to learn a specific behavior. Their mapping must be sampled creating a training data set. This kind of experiment is not enough to estimate the capability to generalize human quality relations to sound material. However, it shows system capability to associate low-level audio behavior with some sort of parametrical control.

With human mapping there would be necessary that a designer specifies his/her own training set. After training, parametric specific data have to be synthesized and presented to designer for parametric classification. Then, human classification can be compared with the original parameter set that resulted on the sound material.

In the current implementation, with high dimensional synthesis data presented to the MLP, the experiments were limited to simple parametric synthesis. A four parameters synthesizer based on subtractive synthesis was programmed in MATLAB. Sound material could be controlled regarding white noise energy, fixed low-pass filter gain, fixed band-pass filter gain and fixed high-pass filter gain. A comprehensive training set was created to map the whole parametric space, with 625 samples presented to the network. Figure 4 shows some experimental results.

a)




b)

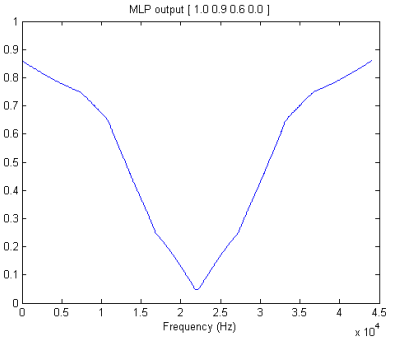
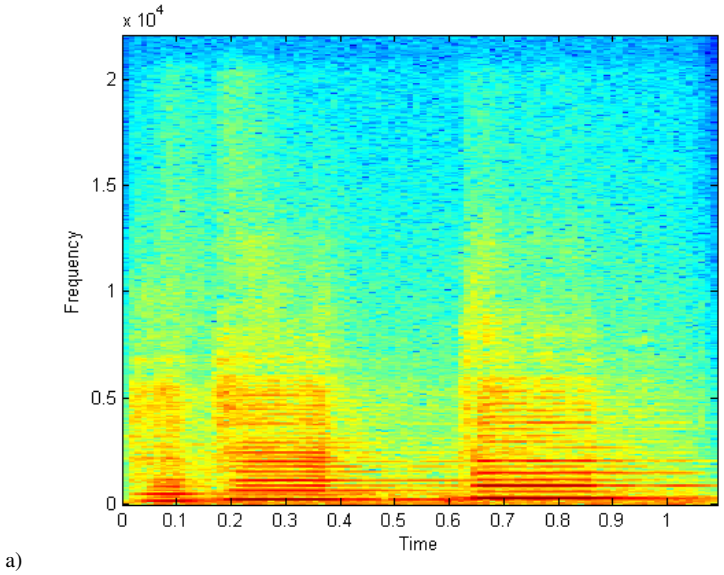
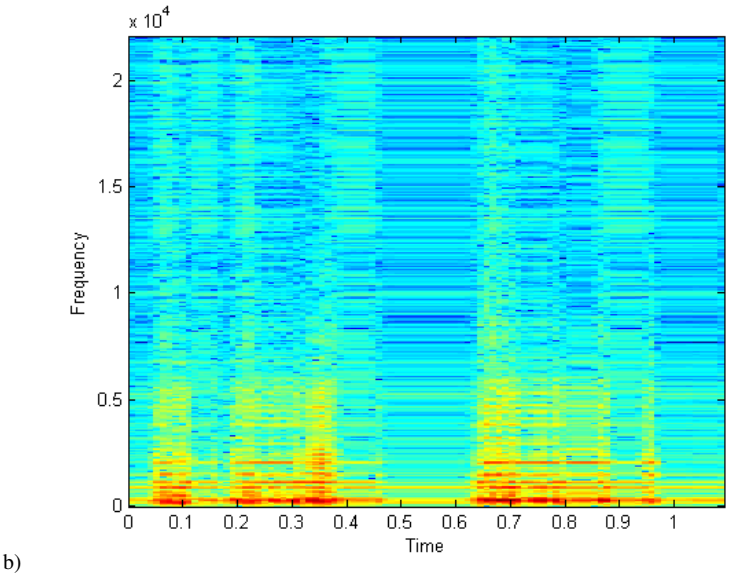



**Figure 4. (a) Synthesis model output and (b); MLP output for the same parametrical stimulus**

In this particular experiment, the MLP was capable of establishing a coherent approximated map. However, it has its limitations depending on the parametric behavior. For example a synthesizer with some sort of pitch parameters (such as FM), this kind of behavior would require an extensive training set since it needs an extremely connected behavior on the artificial network. On high dimensional data, it would be impossible to cover the whole synthesis space.




a)




b)

**Figure 5. (a) Clarinet sample used for training and (b) system output**

As a second experiment we covered the pitch problem. As input, a clarinet record (Figure 5a) was presented with parameters regarding to pitch and time position. Two different pitches have been presented. The output (Figure 5b) shows good response for

the time positioning and energy mapping but, as expected, it did not generalized well the pitch parameter and spectral shape.

This problem extends to real world sounds and any given semantic description which is very likely to be dependent on complex signal behavior. Evolution solved this but molding specialized systems that process the raw signal into a lower dimensional and friendlier processed stimulus for the neural system.

Our approach is to use more specialized synthesis models, supported by psychoacoustic knowledge. An audio codec, as Vorbis (Moffitt 2001) can be used as the synthesizer motor. Vorbis is an open source audio codec. It works by spectral analysis, granular segregation, codification and further overlap-and-add reconstruction. It codifies the original signal window into spectral envelop (floor) and residual spectral density (residue). It also uses mapping system for multi-channel. Vorbis work as a dimensionality reducer. In the training module, audio window is first segregated into a floor and a residue, filtered by the Vorbis psychoacoustic model, and then presented to the MLP. Hence, in the synthesis module, the MLP provides as outputs floor and residue. Further they are joined in a regular audio window.

## 6. Conclusion and Further Works

The possibility of synthesizing sound material directly from semantic description of any ordinary mapping makes it a universal synthesizer. It can be used to override complex synthesis systems, to add new control paradigm to traditional methods or to generalize sound material mappings into a computational process.

It is expected that its generalization performance get enhanced with more specialized synthesis engines. The current implementation architecture has been projected for modular use, and other modules are being developed.

Regarding future directions, we first expect to have a functional implementation with other synthesis engines in hand for testing and validation. Further we aim to obtain better results with a more comprehensive description scheme. The capability of interaction with audio description standards like MPEG-7, or notation structures on audio frameworks like CLAM, would allow other programs to easily control our library.

Also, to allow the system to interact with more complex descriptions, including multi-level representation, it would be necessary to develop a translation unit based on highly structured computational models. Pieces of software like IQR (Bernardet et all, 2002) could be used to conceive such models. We are also going to consider the use of Support Vector Machines as an alternative (Cortes et all, 1995).

## 7. Acknowledgments

Authors would like to thank Brazilian Agencies CNPq and FAPESP for the financial support. Costa is supported by FAPESP, Furlanete by Capes, Manzolli and Von Zuben are supported by CNPq.

## 8. References

J. van Ossenbruggen, F. Nack, and L. Hardman. (2005) That Obscure Object of Desire: Multimedia Metadata on the Web (Part II). IEEE Multimedia, 12(1), January-March 2005.

Pachet, F., La Burthe, A. and Aucouturier, J-J, (2003) «The Cuidado Music Browser: An end-to-end EMD system» in Proc. 3rd International Workshop on Content-Based Multimedia Indexing, September 2003, Rennes (France)

Herrera, P., Celma, O., Massaguer, J., Cano, P., Gómez, E., Gouyon, F. & ,Koppenberger, F. (2005): MUCOSA: Proceedings of 6th International Conference on Music Information Retrieval; London, UK (pp. 77-83).

Costa, C. R., Manzolli, J., Von Zuben, F.J., (2006) "Population-Based Generative Synthesis: A Real-Time Texture Synthesizer based on Real-World Sound Streams" In Proceeding of 4th AES Brazil Conference, São Paulo, Brazil.

Schaeffer, P. (1966) Traité des objets musicaux: Essai interdiscipline. Le Seuil, Paris (France).

Smalley, D. (1986) Spectromorphology and Structuring Processes. In The Language of Electroacoustic Music, Org. Emmerson S., Harwood Academic Publishers, New York (USA).

Nicol, C., Brewster, S., Gray, P., (2006) "Designing Sound: Towards A System For Designing Audio Interfaces Using Timbre Spaces" Proceedings of ICAD 04-Tenth Meeting of the International Conference on Auditory Display, Sydney, Australia.

Johnson, C. Gounaropoulos, A. (2006) Timbre interfaces using adjectives and adverbs. In Proceedings of the 2006 Internacional Conference on New Interfaces for Musical Expression (NIME06), Paris, France.

Loureiro, R. Serra, X. (1997) A web interface for a sound database and processing system. In Proceedings of the Internacional Computer Music Conference, ICMC 1997.

Kurihara, K. Goto, M. Ogata, J. Igarashi, (2006). T."Speech pen: predictive handwriting based on ambient multimodal recognition". In Proceedings of the SIGCHI conference on Human Factors in computing systems: 851-860.

Turnbull, T. Barrington, L. Lanckriet, G., (2006) "Modelling Music and Words using a multi-class naïve Bayes approach". In proceedings of the 7th Internacional Conference on Musical Information Retrieval, Vitoria, Canada.

Martínez, J.M, (2004) MPEG-7 Overview (Version 10) ISO/IEC JTC1/SC29/WG11 Palma de Mallorca, October 2004.

Amatriain, X. , Arum, P. , and Ramírez, M. (2002) CLAM, Yet Another Library for Audio and Music Processing? In Proceedings of the 2002 Conference on Object Oriented Programming, Systems and Application (OOPSLA 2002), Seattle, USA, 2002. ACM.

Pierce, J.R., (1992) "The Science of Musical Sound" rev. Ed. W.H. Freeman and Company, New York.

Nissen, S., (2003) “Implementing Implementation of a Fast Artificial Neural Network Library (fann)”, Intern Report, Department of Computer Science, University of Copenhagen (DIKU).

Moffitt, J., (2001) “Ogg Vorbis—Open, Free Audio—Set Your Media Free” Linux Journal, Specialized Systems Consultants, Inc. Seattle, WA, USA, Vol. 2001, Issue 81es, Article No. 9.

Bernardet, U., Blanchard, M. Verschure, P. F. M. J.: (2002) “IQR: a distributed system for real-time real-world neuronal simulation.” Neurocomputing, 44-46: 1043-1048.

Cortes, C. and Vapnik, V., (1995) "Support-Vector Networks”, Machine Learning, 20.

# Applications of Group Theory on Granular Synthesis

**Renato Fabbri, Adolfo Maia Jr.**

Núcleo Interdisciplinar de Comunicação Sonora – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6166 – 13 091-970 – Campinas – SP – Brazil
{renato,adolfo}@nics.unicamp.br

***Abstract.*** *This paper presents an application of the theory of finite groups on granular synthesis. We show how to apply finite groups to an ordered set of grains in order to get sound streams with cyclical or permutational characteristics. We point out that internal content of the grains can also be transformed creating linked structures. Using different layers of time sequencing we get a rich "polyphony" of granular sound streams. In addition we presents a computer implementation of our model named FIGGS (Finite Groups in Granular Synthesis), a rich and flexible interface developed in the Python programming language and SAGE, a software for algebra and geometry experimentation.*

## 1. Introduction

It is well known that complex symmetry principles have been used in musical composition since ancient times. To point just a few examples, there are a number of studies of the music of J. S. Bach, the Mozart´s dice game, or the recurrent and recursive use of the golden proportion by B. Bartók in the past century. In present days, the study of symmetries in an arbitrary number of dimensions makes use of the modern Theory of Finite Groups (see [Budden 1972] for an overview and also a musical application). The composer I. Xenakis [Xenakis 1991] dedicated a chapter, in his now, classical book *Formalized Music*, to applications of groups in algorithmic composition. On the other hand, due greatly to the exponential growth of digital industry in the second half of the twentieth century, electroacoustic music had its major developments, and now early tedious procedures can rely on fast computer processing. Roughly speaking, composition in electroacoustic music incorporates sound construction and temporal organization methods of these sounds. In this way, it is natural to argue whether group theory, that is a power tool to explore and represent cyclic and symmetric structures, can be applied in electroacoustic music for creating symmetric and cyclic organization of sound materials. This work is our contribution in this direction. In order to have focus on this approach we are most interested in the well known *Granular synthesis* technique (see [Roads 1996, 2001] for an overview). In this way we propose an exploration of symmetries in granular synthesis through applications of finite groups actions on sonic granular structures.

In the next section, we present some theoretical preliminaries and some simple examples of finite groups and also some comments on Granular Synthesis. In Section 3, we show a model, that is, an application of finite groups in granular synthesis. In Section 4 we describe shortly some of the computer implementation based on the

Python programming language using external packages for numeric manipulations and SAGE, a software for algebraic manipulation. A number of models are possible and this work is a demonstration of our method´s potential. Nevertheless, as far as we know, this is the first time one makes use of Finite Groups in Granular Synthesis. Some sound results are available on the Internet and are briefly discussed in Section 5. In Section 6 we make a conclusion and list some perspectives of the method we named FIGGS (Finite Groups in Granular Synthesis). In last section we present some bibliography.

## 2. Theoretical Preliminaries

### 2.1. Group Theory

Formally, a group G is a set with a binary rule (which we will denote by ' • ' in this work) that satisfy these four fundamental properties:

1) if $g_1, g_2 \in G$ than $g_1 \bullet g_2 \, G$ (Closure)

2) $g_1 \bullet (g_2 \bullet g_3) = (g_1 \bullet g_2) \bullet g_3$ (Associativity)

3) $\exists\ e \in G : g \bullet e = e \bullet g = g$ (Identity)

4) $\forall\ g \in G,\ \exists\ g^{-1}\ :\ g \bullet g^{-1} = g^{-1} \bullet g = e$ (Inverse Element)

We denote as (G, •) a group with an operation •. A group is finite if it has a finite number of elements, else it is called infinite. A group is called *commutative* or *abelian* if the commutative property is satisfied for all its elements, that is:

$\forall\ g_1, g_2 \in G, g_1 \bullet g_2 = g_2 \bullet g_1$

Some common simple examples:

a) (Z, +) where Z is the set of the integers.

The set of the integers form a group with the sum operation. The identity element is the number zero and for each integer $a \in Z$, its inverse is $-a$. (Z,+) is abelian.

b) $(Q^+, x)$, where $Q^+$ is the set of the positive rationals.

The identity element is the number 1, and for any $q \in Q^+$, $q^{-1} = 1/q$, and $1/q \in Q^+$. $Q^+$ is abelian. This is the simplest example of a multiplicative group.

c) $(Z_p, \oplus)$, where $Z_p$ is the set of the integers modulo $p$, with $p$ a natural number.

The identity element is zero, and for any $a \in Z_p$ its inverse $a^{-1}$ is given by

$(p - a) \bmod(p)$. When $p$ is a prime number $Z_p$ becomes a multiplicative group.

d) $(M_n^+, .)$, where $M_n^+$ is the set of invertible quadratic matrices $n \times n$, and . is the usual matrix product. The identity element is the *Identity Matrix* $I_n$ and the inverse element of an element $A$ is the inverse matrix $A^{-1}$.

We are most interested in permutation groups of a number $n$ of objects. Not all groups have permutations as elements, but each finite group is isomorphic to a

permutation group. So, we can restrict ourselves to permutation froups without loss of gnerality. Besides that, permutation groups are a mathematical tool for describing symmetries, which are strong factors of artistic thinking. For more on Group Theory see, for example, [Budden 1972].

### 2.2. Granular Synthesis

Granular synthesis is commonly known as a technique that works by generating a rapid succession of tiny sounds, metaphorically referred to as sound grains or yet as microsounds. Granular synthesis is mostly used by musicians to compose electronic or computer music as it can produce a wide range of different sounds, but it has also been used in speech synthesis. Granular synthesis is largely based upon D. Gabor's idea of representing a sound using hundreds or thousands of elementary sound particles [Gabor 1947]. His approach to "elementarity" was inspired by the *Uncertainty Principle* of Quantum Mechanics. He proposed the basis for representing sounds combining frequency and time domains information. D. Gabor's point of departure was to acknowledge the fact that the ear has a time threshold for discerning sound properties. Below this threshold, different sounds are heard as clicks, no matter how different their spectra might be. The length and shape of a wave cycle defines frequency and spectrum properties, but the ear needs several cycles to discern these properties. D. Gabor referred to this minimum sound quantity as an acoustic quantum and estimated that it usually falls between 10 and 30 milliseconds.

The first formal musical thoughts using D. Gabor's sound representation (*time x frequency* space) in music were, probably, initiated by composer I. Xenakis. A computer-based granular synthesis system did not appear until works of C. Roads [Roads 1988] and B. Truax [Truax 1988]. As far as the idea of sonic grains is concerned, any synthesizer capable of producing rapid sequences of short sounds may be considered as a granular synthesizer. However, it is an important fact that the very concept of grain differs in many cases, as for D. Gabor and I. Xenakis. The latter describes a grain as the instantaneous measured of a Fourier Partial associated with a particular frequency and amplitude, which is, for him, the most elementary characteristic of a sound. In this case, a cloud of grains can be thought as a cluster of points relatively close to each other in the *frequency x amplitude* space. And indeed, I. Xenakis proposed a density parameter to measure the compactness of a cloud and the duration of a grain as an external parameter to control the sound as a whole. For D. Gabor, however, duration is an internal parameter of the grain itself, which can have a complex content in terms of Fourier partials. D. Gabor borrows the concept of quantum of action from Quantum Mechanics to define a quantum of sound. The quantum action of sound A is in the order of unity, that is

$$A = \Delta\omega.\Delta T = 1 \qquad (1)$$

In early development, C. Roads suggested the following definition: *"A grain is a signal with an amplitude envelope in the shape of a quasi-Gaussian bell curve"* [Roads 1988]. This is close to D. Gabor's original definition in the sense that it accepts grains with complex spectral content. However, the concept of "elementary" (or quantum) is not strictly taken into account in C. Roads' definition. The interpretation of "quasi-Gaussian bell curve" can be very general. In addition, his concept of grain density is actually a measure of the number of grains occurring within a given time interval.

C. Roads also suggests that granular synthesis can be classified as a form of additive synthesis. We therefore prefer to consider this definition as being for a specific type of granular synthesis, which we refer to as Short-Time Additive Synthesis (STAS). A key problem in granular synthesis is controlling the evolution of sound grains in time. Most granular synthesis systems have used stochastic methods to control evolution of hundreds or even thousands of grains. A handful of different methods have been proposed. For instance, E. Miranda devised Chaosynth, a granular synthesizer of the STAS type that uses cellular automata to manage the spectrum of the sound grains [Miranda 2002]. Chaosynth explores the emergent behavior of cellular automata to produce coherent grain sequences with highly dynamic spectra. The states of the cellular automata defines frequency and amplitude values for an additive synthesis engine that produces the granular streams. A critical review and present status of Granular synthesis can be found in [Thomson 2004].

Once the sonic grains are discrete entities, it is possible to think them as objects that in which we can impose internal and external symmetries, the last one related to temporal organization of grains.

## 3. A Theoretical Model

In this section we presents one of our models. The idea is to show a way to construct an application from groups to granular synthesis.

Here we are going to consider four permutation groups.

a) *The Symmetric Group of degree n*. the group of all the permutations on an ordered set of *n* elements.

b) *The Alternating Group of degree n*: the group of even permutations on a set of *n* elements, that is the set of permutations obtainable from an even number of two-element swaps.

c) *Dihedral Group*: the symmetry group of a n-sided regular polygon for n > 1.

d) *Cyclic Group*: a group which can be generated by a single element and the group operation. In our case we use Permutations Cyclic Groups.

In this first model we take one of the groups mentioned above. We are, currently, taking random elements of these groups, i.e. permutations belonging to specific classes, and applying it to the state of our granular parametric set. The above model can be applied to any set of grains in order to define a sound stream whose one of the main psychoacoustic characteristic is the cyclical sound structure in time. In addition, it is possible to apply group transformations inside the grain itself. This can be done, for example, using a set of transformations on granular parameters (frequencies, duration, amplitude, etc) and imposing on this set of transformations a group structure. This approach leads to construction of granular structures with strong internal correlation.

# 4. Algorithm and Computational Implementation

## 4.1. Development Framework

We have based our implementation on *Python*, an object-oriented and high-level programming language that uses automatic memory management and dynamic typing. The language has an open, community-based development. It have been used for a wide range of tasks: GUI design, web development, stand-alone programs, among others. It has been extensively used by Google and NASA, for example.

Python's standard library comes with a bundle of modules that provides basic functionalities, but the language is truly extensible for audio manipulation by accessing some of the innumerable third party modules such as PySndObj, NumPy and Matplotlib. For computational algebra, we can point the excellent SAGE (Software for Algebra and Geometry Experimentation), which is written in Python and a modified version of Pyrex (SageX) unifying specialized open-source math software and filling in functionality gaps. SAGE is a stand-alone software that uses an individual python interpreter to run scripts, so we can install python modules to it and extend SAGE's functionalities[1]. For our present implementation, besides standard functionalities in SAGE, we used:

- wxPython: For GUI design.
- NumPy: For fast numerical array processing.
- PyAudioLab: for writing NumPy arrays to sound files.
- Matplotlib: for plotting graphs like waveforms and spectrograms.

## 4.2. Algorithms

For the sake of complementarity, in this section, we show how we can combine Python facilities with the calculational power of SAGE for algebraic structures such as, in our case, groups. Within this framework we can create a lookup table for a sinusoidal waveform like this:

```
import numpy

cycle = numpy.linspace( 0, 2*numpy.pi, 2**11, endpoint=False )

table = numpy.sin( cycle )
```

Here, we imported the NumPy module and used it's linspace function. That function takes as arguments a starting number, an ending number, and a number of steps, respectively, and returns an array with “number of steps” elements that specifies the number of evenly spaced samples from start to stop. Writing x**y in python is evoking the number x raised to power y. The end number is not included in the constructed array if endpoint is declared “False”, which is our case since sin(2*pi) is sin(0). Our lookup table is the sin function called upon the defined cycle array. We can use this table by calculating an increment SI = f * N / SR (where f is the frequency of the signal we

1 There are plans for future versions of SAGE supporting it's use as a normal python module. See discussion thread “Is it possible to use SAGE from regular python?” at: http://groups.google.com/group/sage-support/

want to use, N is the length of our lookup table, and SR is our sample rate), and iterate over the table:

```
for i in range( int(duration*SR) ):
        sig[i] = table[ int(ap) ]
        ap = (SI + ap)%N
```

The function range calls a list of integers from zero to the input value. If we have a desired duration, the number of samples of the signal equals the integer part of the duration (seconds) times our sample rate.

Invoking groups in SAGE is easy. Here goes some examples for permutation groups:

```
S = SimmetricGroup(5)
```

This creates a symmetric group of degree 5.

```
A = AlternatingGroup(4)
```

This creates an alternating group of degree 4.

```
C = CyclicPermutationGroup(7)
```

As mentioned above, a cyclic group can be generated by a single element g ($g^2$, $g^3$...). Its representation as a permutation group is called by the CyclicPermutationGroup(x) function (*x* is the order and the degree of the group).

Invoking elements of a group can be done in various ways. We illustrate here two of them on the SAGE command line interpreter:

```
sage: S = SymmetricGroup(3)
sage: list(S)
[(), (2,3), (1,2), (1,2,3), (1,3,2), (1,3)]
sage: list(S)[1]; list(S)[3]
(2,3)
(1,2,3)
sage: S.random_element(); S.random_element()
(1,3,2)
(2,3)
```

These tuples are elements of the group $S_5$, they are permutations. In this permutation representation we read (2,3) as element in place 2 goes to the place 3 and element 3 goes to place 2. We read (1,3,2) as element in place 1 goes to place 3, element in place 3 goes to place 2 and element in place 2 goes to place 1. We can apply this permutation to an integer:

```
sage: g = list(S)[4]
sage: g
(1,3,2)
sage: g(1)
3
```

If we have an ordered set (a list) X of parameters and a permutation (a group element) g, we can apply that permutation on that set with these lines:

```
for i in range( len(X) ):
        perm[ g( i +1 ) -1 ] = X[i]
```

This +1 and -1 occur because *Python* lists have indexes from zero to the length of the list minus one, and permutations in SAGE ranges from one to the length of the set in which it acts. This takes us to the last procedure: writing the sound file.

We are using NumPy arrays in order to add functionalities to array manipulations, to simplify its use and to improve processing speed. There is a module named Pyaudiolab that is dedicated to writing NumPy arrays to sound files, and opening sound files as a NumPy arrays. It supports all sound file formats supported by *libsndfile*, including WAV, AIFF, Ircam SF, FLAC and RAW.

We can write a 16 bit WAV sound file from a NumPy array (here instanced as the signal object) using Pyaudiolab like this:

```
import pyaudiolab
fmt = pyaudiolab.formatinfo( 'wav', 'pcm16' )                (1)
sndfile = pyaudiolab.sndfile( name , 'write', fmt, nc, sr)   (2)
sndfile.write_frames( signal, len(signal) )                  (3)
```

(1) Specifying the file format

(2) Creating the object sndfile for file writing, specifying its name, its format, its number of channels and its sample rate.

(3) Calling the write_frames method of the created sndfile object to write samples in the signal array for its entire length.

As a way of experimenting with this implementation, the user chooses the number of grains and inputs parameters for individual grains. Than chooses the parameters to be permuted, the groups that acts on each of these parameters and commands for the sound to be written.

GUI programming issues will not be discussed since they are beyond this work's subject. Even so, we will present our interface and use some screenshots to illustrate a way to approach the features presented.

### 4.3 FIGGS Interface

We have developed a GUI (Graphical User Interface) in Python for FIGGS (Finite Groups in Granular Synthesis). With this we aim to furnish facilities in order to represent structural ideas as settings of values allowing the user to concentrate on musical creation's symmetric aspects. We intend this GUI interface for the FIGGS model to be friendly and flexible enough in order to allow the user to construct his own random distribution. Random distributions for granular synthesis are the quickest way to input parameters, but, of course, the user can also input the parameters values in a deterministic way as he/she wishs.

This interface has two tabs: one for granular parametric specifications, and another for sound parameters on which chosen groups acts, each with its proper controls.

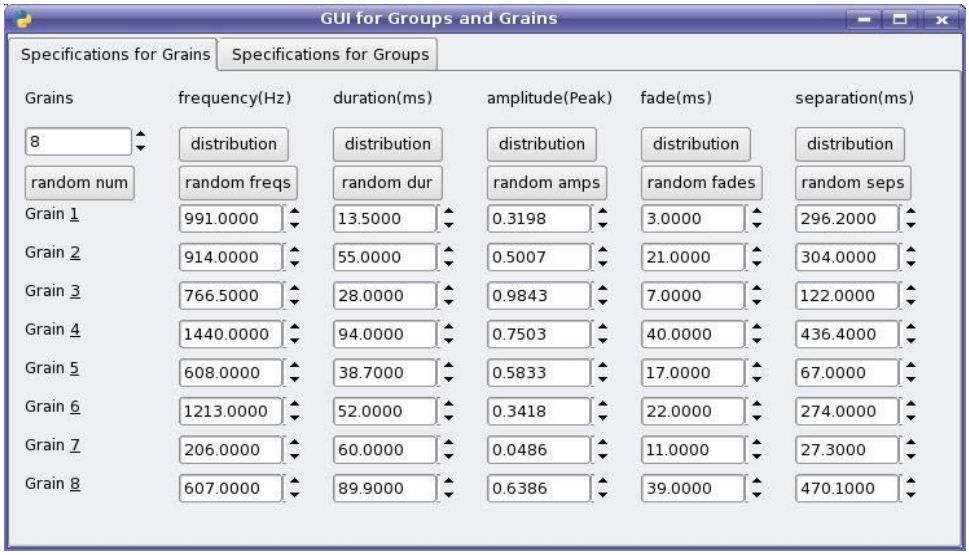



*Figure 1. FIGGS Panel for Grain Specifications*

Each parameter of each grain can be inputed independently or by ease of a random distribution. Different distributions are useful here but only the homogeneous one is currently implemented in the GUI. Negative separations are understood as superposition. This creates an ordered set of grains, which is going to be manipulated by group structures. Once this table is filled in, it can be manipulated by classes of permutations. Mathematically speaking, these manipulations are specific permutation groups acting on columns of this matrix. A musician can use this freedom of experimentation in order to explore proportions among the physical aspects of the grains and their evolution in time. These proportions are a significant factor of musical creation (see, for example, [Xenakis 2001]) and analysis [Shmulevich et al. 2001]. For a recent musical example using this approach and FIGGS, please see Section 5.

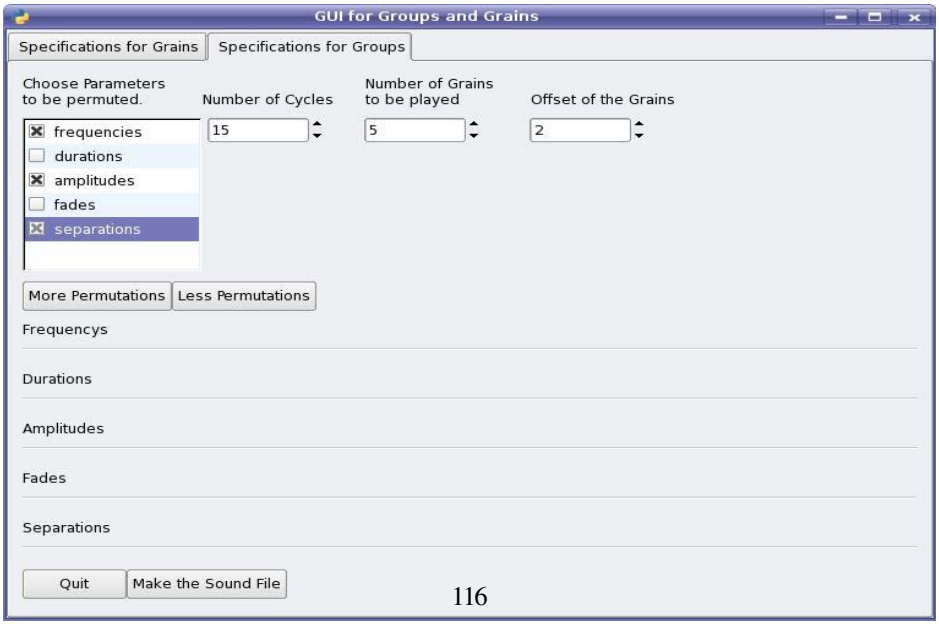





*Figure 2. FIGGS Panel for Groups - Choosing Grain Parameters*

Figure 2 shows a panel where we can choose independently those parameters that we want to permute. That is done by stating the number of times the specified grains are going to be played (the number of cycles), the number of grains of each cycle (a joint subset of the grains set), and an offset of this subset in the grain set. These are the necessary inputs before the sound file can be done, so the user can choose to play the grains in the exact sequence defined in the grain panel for an arbitrary number of times.

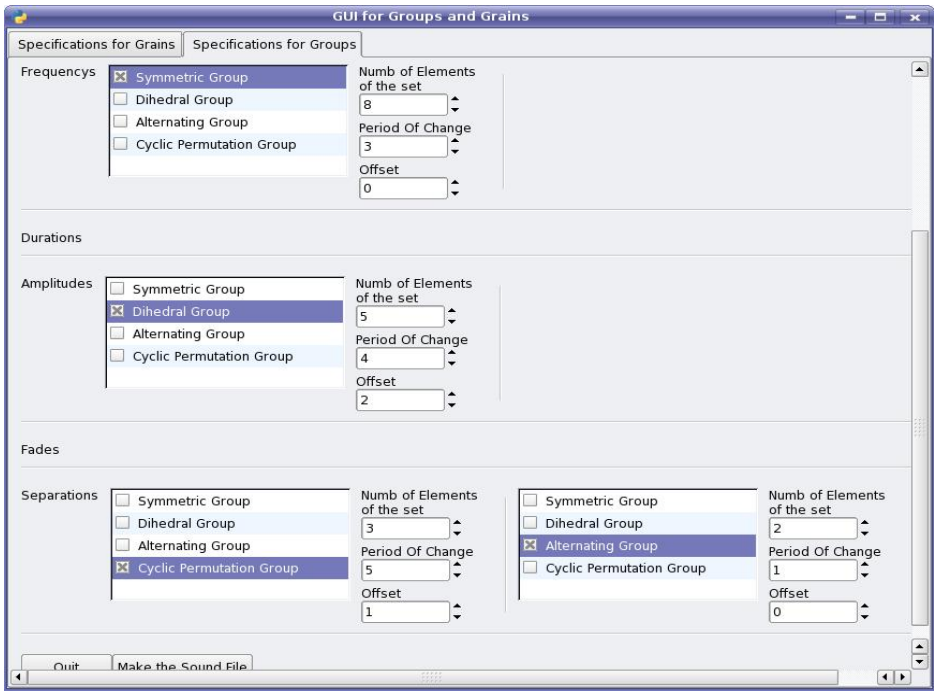



*Figure 3. FIGGS Panel for Groups - Specifying Groups*

Figure 3 shows some group usage. The number boxes besides each group list are the number of elements in which the group acts (top), the number of cycles between each group action, and an offset of the action set in the grains set. There are numerous ways in which a group can act on a parameter set, and there are other groups to be explored as well, and more options are going to be available in future developments. In this screen shot's version, the action is performed by successive actions of random elements of a group (i.e. permutations) on the set. Notice that the number of the permuted elements need not to be equal to the number of grains played or the number of grains in the grains set. In fact, that is one of the main features of this FIGGS implementation. The user can freely move back and forth from the grain panel and the group panel, and command the sound file to be written any number of times. This allows making any number of sounds with related behavior and related sounds.

Finally, it should be mentioned that is possible to have group actions on spectral contents of a set of grains. This will be implemented in a near future.

## 5. Resulting Sounds

One of the main tools used in computer based audio recognition is the representation of sounds as a long term statistical distribution of its local spectral features. Such an analysis of structures created within FIGGS methods would be about the same if there were no group actions involved, for statistical occurrences of the events remains the same. Even so, there are strong indicatives that musical cognition often rely on other symbolic and analytical level than a physical holistic description can reveal [Aucouturier, Defreville, Pachet 2007]. Musical patterns can be thought as being concurrent relations of frequencies and relations of grain onset timings in the metric pattern. Melodies are an ordered set of these relations [Shmulevich et al. 2001]. In general, FIGGS structures, if build with densities in the range about 0.1 to 10 grains per second, resembles melodies that, due to the set of frequencies choosed, has fixed melodic scales. With this respect, the users can find a sound bank with some recent patterns examples, at: http://cortex.lems.brown.edu/~renato/sonic-art/nics/FIGGS-patterns-ex.zip

Different granular synthesis techniques can be used in the current FIGGS interface. One can, clearly, make synchronous and asynchronous granular streams, as well as quasi-synchronous granular synthesis**.** Irregular onsets of the grains leads to a controllable thickening of the sound spectrum [Truax 1988]. Here is a link for downloading a sound bank with according examples:

http://cortex.lems.brown.edu/~renato/sonic-art/nics/FIGGS-clouds-ex.zip

It should be pointed that the only amplitude envelope currently implemented in the GUI is trapezoidal.

A musical experimental piece (~10 min) was created using FIGGS capabilities. It can dowloaded at:

http://cortex.lems.brown.edu/~renato/sonic-art/nics/reflexoes-paradoxais.mp3

This piece is about two prose manuscripts by Fernando Pessoa (1888-1935), namely *Solidão* (1947, 1915?) and *Reflexões Paradoxais* (1916?), and was composed in 2007 under artistic supervision of J. A. Mannis (Music Department, UNICAMP). Passages that makes use of FIGGS are mainly before 02'32” and after 06'52”.

The current SAGE/Python code is already available at: http://cortex.lems.brown.edu/~renato/sonic-art/nics/FIGGS.zip. A brief README.txt instructs how the framework must be set to run the script.

An executable version of FIGGS, as well as the examples above mentioned, will be available, by August 2007, at:

http://www.nics.unicamp.br/atual/pessoal_renato.htm.

## 6. Conclusion and Perspectives

As mentioned in the introduction section, group theory can be a valuable tool for algorithmic composition. Nevertheless, as far as we know, its application in sound synthesis was not pursued seriously until now. Our model is a preliminary study in this direction. We have used, for the sake of simplicity, the well known finite permutation groups which can demonstrate the potential of the model for more complex

applications. In addition we can point out some directions and perspectives for future work:

A) In order to construct musical structures we can use superposition of sequences of grains. In other words we can construct layers which can be controlled independently (or interdependently) in terms of content and duration of the grains as well as their time sequencing. This conception can be envisaged as a tool for granular synthesis composition. In addition, new rhythmic aspects can emerge from this kind of sound design.

B) The use of other groups whose elements are not permutations, such as Fundamental Groups, which can be used to describe and impose orbits related to topological spaces.

C) In order to generate an arbitrary quantity of grains we can make use of probabilistic distributions such as Gaussian, Binomial, Bernoulli, among others. Also from the parametric point of view, different granular parameters can have linked values.

D) Sounds can be generated not only by lookup tables, but by other synthesis methods, such as FM synthesis, and have its synthesis parameters controlled by groups in order to have streams of related and coherent sound particles. Sound particles can be generated as well by using loaded sounds, or slicing it at group controlled intervals.

## References

Xenakis, I., *Formalized Music*, Bloomington: Indiana University Press (1971); also, *Formalized Music*, 2d ed., New York: Pendragon Press (1991).

Budden, F.J., *The Fascination of groups*, CUP,(1972).

Gabor, D., *Acoustical Quanta and the Theory of Hearing*, Nature **159** (4044), pp. 591-594,(1947).

Roads, C., *Introduction to Granular Synthesis*, Comp. Mus. Jour., **12** (2), pp. 11-13 (1988).

Roads, C., *Microsound*, MIT Press, Cambridge,MA, (2001).

Roads, C., *Computer Music Tutorial*, MIT Press, Cambridge, MA (1996).

Thomson, P., *Atoms and errors: towards a history and aesthetics of microsound*, Organized Sound, **9** (2), pp. 207-218, (2004).

Miranda, E. R., *Computer Sound Design: Synthesis Techniques and Programming*, Oxford: Focal Press (2002).

Aucouturier, J.-J, Defreville, B., Pachet, F., *The bag-of-frame approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music*, Journal of the Acoustical Society of America, 2007.

Shmulevich, I., Yli-Harja, O., Coyle, E. J., Povel, D., Lemström, K., *Perceptual Issues in Music Pattern Recognition – Complexity of Rhythm and Key Finding*", Computers and the Humanities, Vol. 35, No. 1, pp. 23-35, February 2001.

Python Software Foundation,, "*Python Documentation*", http://www.python.org/doc/, as acessed in 02/06/2007.

Stein, W., *SAGE – "System for Algebra and Geometry Experimentation"*, at http://www.sagemath.org/sage/documentation.html, a acessed in 02/05/2007.

*Scipy and Numpy documentation* at http://www.scipy.org/Documentation, as acessed in 02/06/2007.

Cournapeau, D., *"Pyaudiolab homepage"*, at http://www.ar.media.kyoto-u.ac.jp/members/david/softwares/pyaudiolab/, as acessed in 02/06/2007.

# A Real Time and Interactive Music Spatialization System

**Leandro Ferrari Thomaz[1], Regis Rossi Alves Faria[1]**

[1]Laboratório de Sistemas Integráveis (LSI) – Escola Politécnica – Universidade de São Paulo (USP)
Av. Prof. Luciano Gualberto, travessa 3 nº 380 – 05508-900 – São Paulo – SP – Brazil
{lfthomaz, regis}@lsi.usp.br

***Abstract.*** *This work involves the investigation on techniques for musical spatialization, and the development of a real time spatialization system, based on a multichannel auralization engine (AUDIENCE). The constructed system uses the Ambisonics technique, which allows the recreation of a three-dimensional sound field. The system reproduces spatial music through an arrangement of loudspeakers around the listener where sound sources can assume some free positions, and also be put into motion. Results of experiments with a musical piece show that the system can be used by composers, conductors, and producers for the musical spatialization task.*

## 1. Introduction

Spatial localization relates to the human ability of judging the direction and the distance of a sound source. Thus, we can understand sound spatialization as the act of positioning a sound source with the intention of stimulating this capacity in a listener.

Having this property to call the attention for certain events, the natural course would be using sound spatialization in music. But this would be delayed by some centuries until it was consciously realized by the composers. By the 20th century, some composers would consider space as a new parameter to explore in music and, more than fifty years later, systematically use it in compositions called space or spatial pieces (TROCHIMCZYK, 2001).

The motivation of this research is to create a computational tool to assist composers, conductors and producers in the task of spatialization of sound sources. With a system assisted by computational tools, the user can create experiments with virtual space positioning (a positioning example can be seen in Figure 1).

The system is capable of reproducing spatial pieces in the form suggested by the composer, allowing the user to manipulate and create new forms of spatial presentation of the same musical piece. It can also be used for live presentations. This work results from an application of the first implementation of the AUDIENCE spatial sound production, transmission and reproduction architecture (FARIA, 2005; FARIA; ZUFFO, 2006).

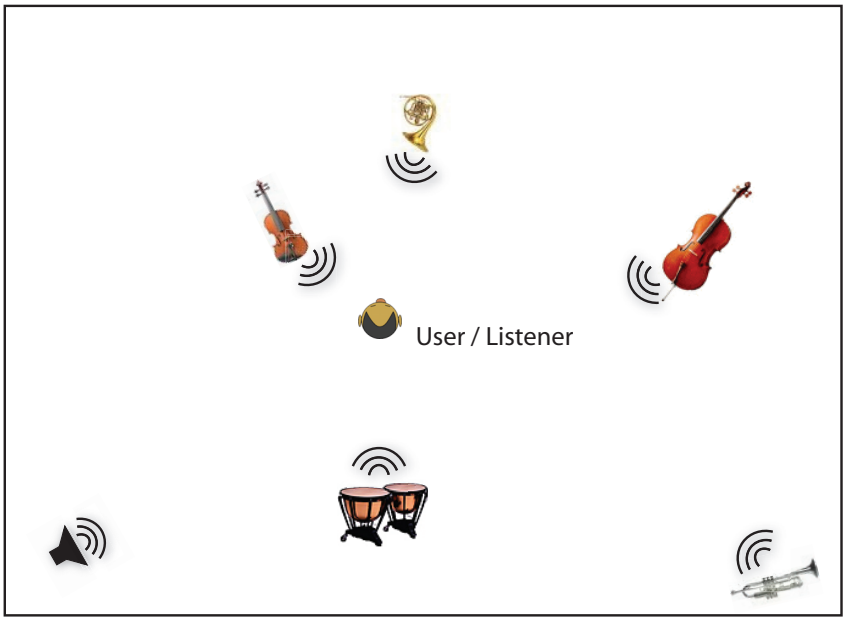



**Figure 1. Positioning of virtual sound sources surrounding the user**

In this paper, we describe the implementation of the sound spatialization system and the experiments made with it for the production and execution of a spatial musical piece. Further details about the historical use of spatialization, other sound spatialization systems and this implementation can be obtained in THOMAZ (2007).

## 2. Sound Spatialization System

A sound spatialization system is understood to be a complete chain of processes used to spatialize sound sources, whether a musical piece or a movie with surround sound. The spatial reproduction, by means of a configuration of some loudspeakers or headphones, is just one of the tasks.

In the AUDIENCE system the input of a sound spatialization system is comprehended by the description of the acoustic scene, i.e., which sounds will be spatialized and how they will behave spatially throughout the execution, and of the description of the temporal behavior of sounds in the scene, that can be a score, recorded sounds or other instruction that produces sound.

From the acoustic scene description, the next block is responsible for positioning the sound sources, either through the positioning of the proper musicians, the use of a mixer to position a sound in a recording channel or spatialization software.

Next, the sound can be encoded by means of some auralization technique, so as to organize temporal and spatial sound information. The decoding block does the inverse work and reconstructs the original information. Within these blocks, algorithms of sound signal compression can be used.

A transmission block is responsible for taking the sound information of the place where it was produced and bring it to the place where it will be reproduced. This can be done using analog or digital medias (e.g. magnetic tapes, DVDs, CDs) or digital transmission through networks and protocols of communication (e.g. computer networks).

Finally, the reproduction block materializes in sound waves the spatialized sound information, as it was described in the input of the system. It can be made, for example, by means of a musician performance or using loudspeakers.

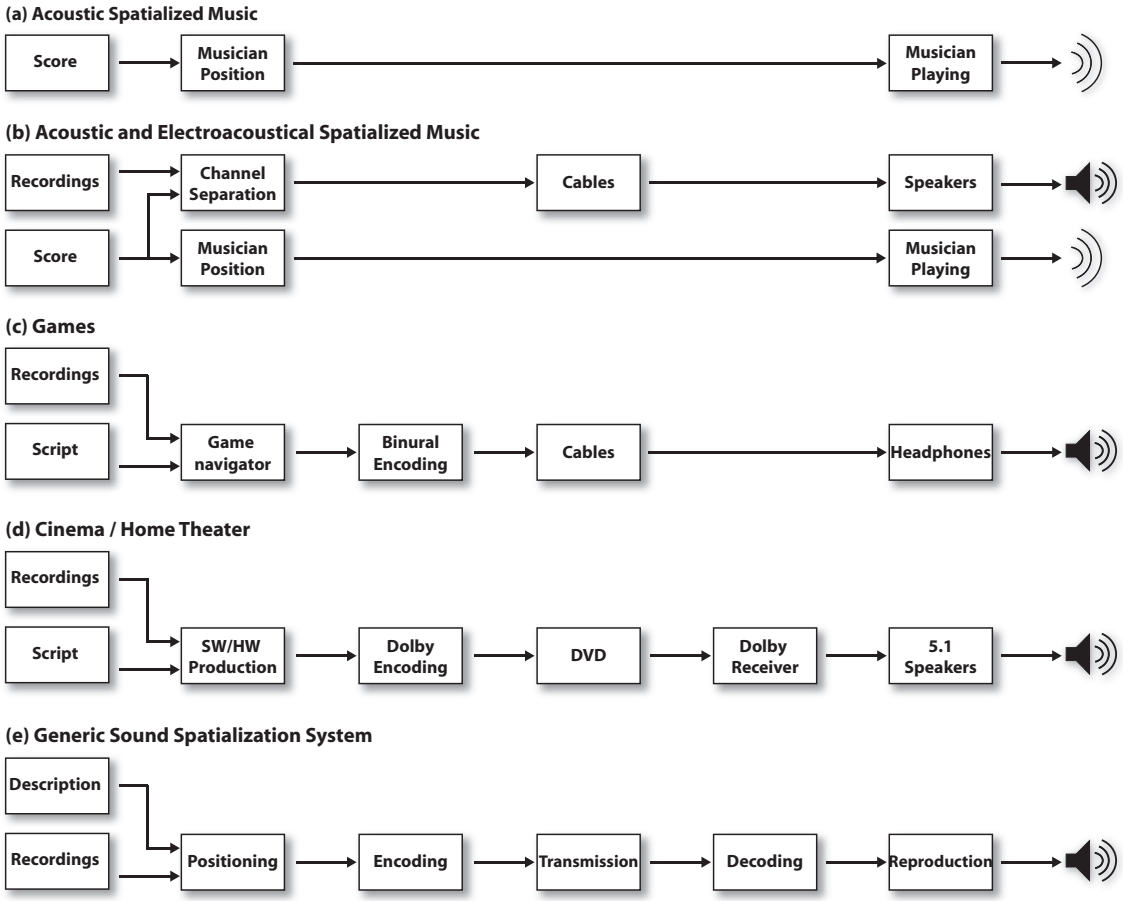



**Figure 2. Generic Sound Spatialization System and examples**

Figure 1 shows some examples of sound spatialization systems and a generalization. As it can be seen, it is not necessary for a system to present all blocks. Frequently, a sound spatialization system is implemented by a subgroup of these blocks, and their use depends on the necessities of the spatial sound scene described by the composer/conductor/producer.

## 3. Ambisonics Auralization Technique

Auralization techniques have the goal of recreating three-dimensional sound fields by means of physical and mathematical modeling (FARIA, 2005). The Ambisonics auralization technique was chosen for this work due to its easy implementation, low computational cost, and its *sweet spot* capability to serve a number of users. This technique also satisfies the need for a realistic three-dimensional sound field recreation, and allows the use of flexible configurations of loudspeakers, which is a requirement for the free spatialization of musical scenes. Next, we give a brief description of the Ambisonics technique. Further details can be found in Gerzon (1973), Bamford (1995)

and Daniel (2001). Gerzon (1973) developed a system capable of recording and reproducing sound fields. The system was based in physical/mathematical approach and it was capable of reproducing sound fields with height information (periphony).

Ambisonics coding can also be made by means of synthesis, where a monophonic signal can be (virtually) positioned in space through manipulations with gains in each channel. The virtual sources, through encoding equations that receive the rotational and elevation angles, are located on a unitary reference sphere. Perception of distance must be applied to the signal through attenuations and delays. The storage and transmission of Ambisonics signals depend mainly on the order of the system, which defines the number of channels. An interesting property of the encoding equations is the ability to manipulate the sound field through simple trigonometric operations, with low computational cost, as described by Malham and Myatt (1995).

Decoding of Ambisonics channels is made through matrices of gains calculated for a particular speaker's configuration. As the calculation of these gains can turn extremely complicated depending on the regularity of loudspeakers positioning, they are previously calculated for the most common configurations (FURSE, 2006). Moreover, shelf filters are used to make psychoacoustics adjustments.

## 4. Using the AUDIENCE Spatial Sound Engine

The AUDIENCE spatial sound engine, proposed by Faria (2005) is the framework of the present real time interactive spatialization system implementation. It is based on a modular approach with four functional layers.

This division allows the use of different techniques and tools in the implementation of the functions executed in each layer (e.g. programs, plug-ins, and external equipment), keeping the communication between them through predefined interfaces. With this approach, it is not necessary to remake the entire system when a substitution of tools and techniques used in one layer is needed. This guarantees independence and interoperability between layers, and flexibility in the choice of algorithms for third part tools. Next, we briefly present the present implemention abstraction of the four layers following the AUDIENCE architecture.

The first layer is responsible for the description of the acoustic scene, informing to the next layer sound sources attributes such as the position of objects, position of the listener and parameters of the environment (e.g. acoustic parameters). These data can be supplied through user interaction directly on the software or by another system that uses the AUDIENCE engine, such as a virtual reality navigator.

Acoustic simulation has a very important role in the production of sound realism; therefore the spatial perception and its quality is directly associated to this. This simulation is done in the second layer and calculates the sound propagation from the sources to the listener, considering the environment (surrounding) where they are immersed. Several techniques such as geometric models (e.g. ray tracing, radiosity, source-image), ondulatory models (e.g. finite elements) or statistical models can be used and implemented.

In the third layer Ambisonics or other formats can be used for both temporal and spatial sound encoding, and to generate coded items for transmission. Here compression

or streaming can occur, depending on the way of transmission or on the used media. In the present implementation, the transmission is accomplished using digital audio patching within the same engine instantiation.

The last layer of the engine decodes the signal generated by the encoder and reproduces the final sound field through loudspeakers or headphones. It is sensitive to the equipment available in the system, number and disposal of loudspeakers. Depending on the choice of codification format and the availability of loudspeakers, a great variety of reproduction configurations can be considered. The mixing from the various sound sources is also made in this layer.

Further information on the AUDIENCE engine can be found in FARIA (2005) and FARIA (2006).

## 5. Sound Spatialization System Implementation

Implementation of the sound spatialization system was made over the Pure Data (Pd) programming platform. Pd, developed by Miller Puckette (2006), is a graphical environment for programming musical and audio applications, similar to MAX/MSP.

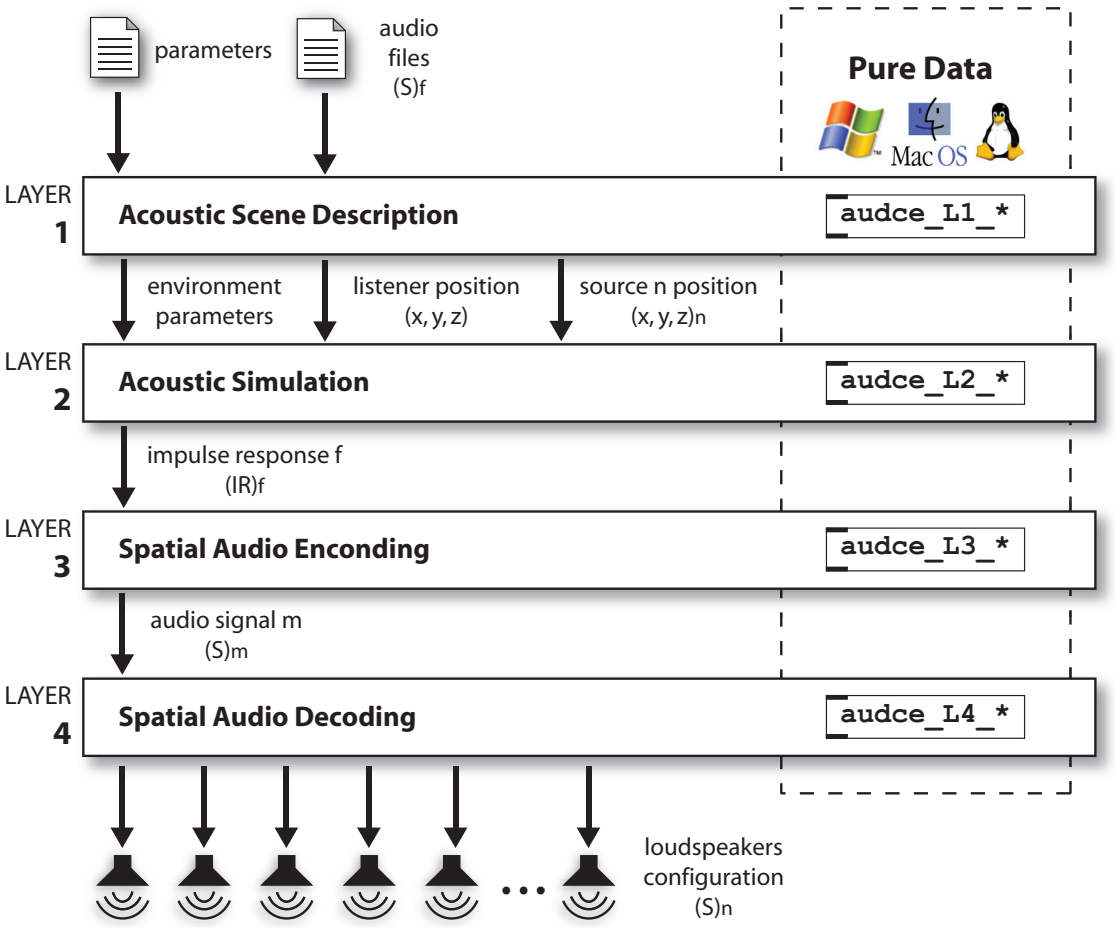



**Figure 6. AUDIENCE engine implementation**

Pd is an open and flexible tool, with low latency processing of audio. Source code is open, and it can be compiled in diverse platforms such as Windows, MacOS and Linux. Pd allows new blocks to be created by the user, called externals, which were a valuable tool to prototype AUDIENCE layers functionalities. Programming of these blocks is made with C language, and must follow some rules so that they can be used in a Pd patch.

Figure 6 illustrates an adaptation, specific for this project, of AUDIENCE layers. The index *f* represents the number of sound sources, *m* the number of signals of audio codified and *n* the number of loudspeakers. Strict aspects of synchronization between the layers were relaxed, since Pd itself is responsible for the synchronization of the audio flow.

A set of Pd blocks was developed to satisfy the description of each layer functionality, the AUDIENCE system conventions, the desired interface between layers, and the application specific requirements,(see Table 1).

Blocks were divided in six groups, identified by its nomenclature, in which `Ln` indicates the layer (`n` being this layer) that the block implements, `gui` indicates blocks related with the graphical interface, and `aux` indicates the auxiliary blocks.

**Table 1. Pd blocks developed, under AUDIENCE framework**

| Group | | Blocks | Pd Type | Description |
|---|---|---|---|---|
| Layer 1 | `audce_L1_*` | `audce_L1_pd_gui` | *patch* | Control of scene input parameters based on a graphical interface |
| Layer 2 | `audce_L2_*` | `audce_L2_allen` | *external* | Based Acoustic simulator in in the method of virtual sources of Allen (1979) |
| Layer 3 | `audce_L3_*` | `audce_L3_amb_3rd~` | *external* | 3rd order Ambisonics Encoder based on the source position (using Ambisonics equations in a free-field environment) |
| | | `audce_L3_amb_ir~` | *patch* | Ambisonics Encoder based in impulse responses generated by acoustic simulation |
| Layer 4 | `audce_L4_*` | `audce_L4_amb_3rd~` | *external* | 3rd order Ambisonics Decoder |
| Graphical Interface | `audce_gui_*` | `audce_gui_grid` | *patch* | Sound sources and listener positioning |
| | | `audce_gui_srcfile` | *patch* | Audio source file gui tool |
| | | `audce_gui_volume` | *patch* | Volume controller |
| Auxiliary | `audce_aux_*` | `audce_aux_convolver~` | *external* | FFTW library based convolution block |
| | | `audce_aux_mixer` | *patch* | Signals mixer |
| | | `audce_aux_move` | *patch* | Automatically moves a sound source |
| | | `audce_aux_srcfile` | *patch* | Control de audio source file reproduction |

## 6. Experiment and Results

### 6.1. Experiment Methodology

This experiment included the recording of the sound sources in studio, the construction of a Pd patch using the blocks described in the previously section, and based in the musical piece's requirements, and the reproduction of the musical piece. From this reproduction, validation tests have been carried through to evaluate the quality of the spatialization.

### 6.2. Musical Piece Selection

The selected piece was *The Unanswered Question*, by American composer Charles Ives, due to its wealth of timbres (with one string quartet, two flutes, oboe, clarinet and trumpet), and for the division in three groups of the instruments, facilitating the rehearsal and recording. This piece was composed in 1906, and presented unusual elements, a mark of the compositions of Charles Ives. The piece lasts about five minutes, and was called by Ives a "cosmic drama" (IVES, 1953).

The piece is constituted of *three layers*, each one with its own style, texture, instrumentation and music. The string quartet executes a choral music in the first layer. A trumpet periodically enunciates a question and, after each one, the woodwind quartet tries to formulate a reply. Ives suggests, in the text that precedes the score, that one must separate physically the instruments in three groups: *string quartet*, which must be outside the stage or far from the other instruments; *trumpet*, in the middle of the audience; and *woodwind quartet*, at the stage. Thus, we have three distinct sound source groups.

Based on Ives's suggestions, innumerable possibilities of spatial arrangements can apply. One can experiment beyond the positions suggested by Ives, considering each instrument a sound source, and not as a group. Thus, nine sound sources can be arranged in space, creating new effects, different from the ones imagined by the composer.

### 6.3. Musical Piece Revision and Production

Edition and revision of the score had been made based in two editions. Both presented small errors and differences, needing wide revision and confection of a new edition.

Although, for the execution of the piece, the instruments that compose each group could have been recorded in a joint form, we preferred to capture each instrument separately, so that it was possible for each one of them to be a sound source in the system, and not a group. Due to the necessity of having each group of musicians executing the piece in a joint form (mainly for synchronism tuning) they were isolated inside the studio through bays of absorbent material (fibreglass), leaving the musicians only with the vision of the conductor.

For each instrument, recording was made with two microphones, a dynamic one and a condenser one, to promote isolation features in mixing the capture of each sound source. Thus, for the mixing and final mastering of the sources both signals were used, taking into account the advantages of good isolation of the dynamic microphone and the wider frequency response of the condenser microphone.

### 6.4. Experiment Pd Patch

Each application has a specific patch that is constructed in accordance with the following requirements: *number of sound sources, movement of the sources* (if some source needs automatic movement) and *level of sound immersion* (influences the Ambisonics order and the configuration of loudspeakers). Figure 7 shows the patch used in tests with nine sound sources (shown around the listener in the central rectangle).

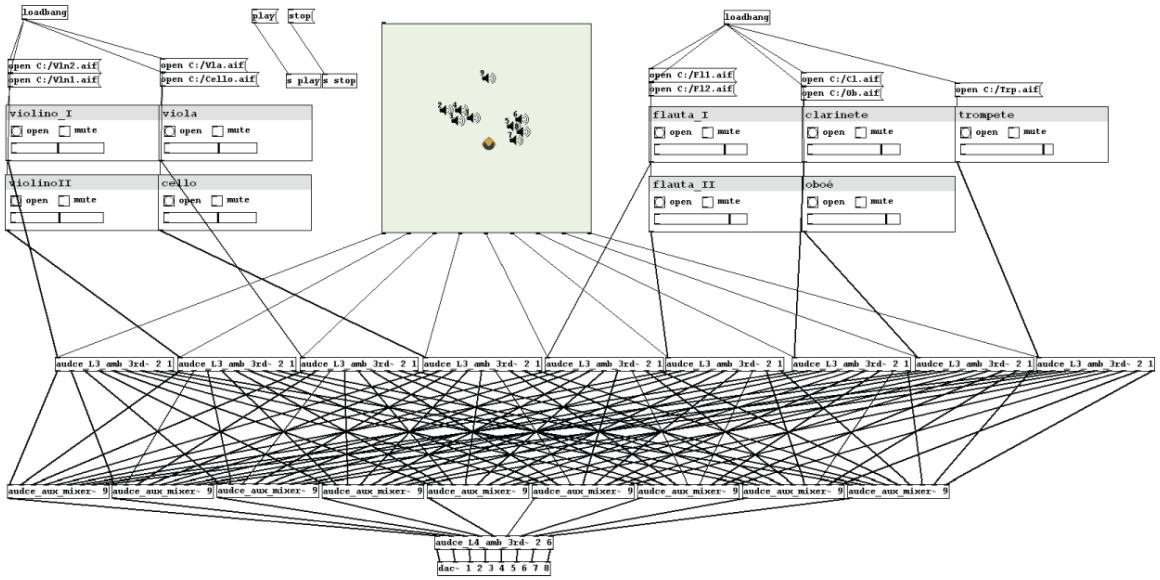



**Figure 7. Patch *for The Unanswered Question* spatialization with nine sound sources**

### 6.5. Tests with Users

The goal of the validation tests is to estimate a level of sound immersion achieved by the system through simple usage and subjective tests.

**Table 2. Validation tests**

| # | Tests | Evaluated parameters | Description |
|---|---|---|---|
| 1 | Blind source localization | Presence<br>Directionality<br>Source distance<br>Room size | Identification of sound sources only by hearing, without access to the interface, to evaluate the quality of the sound field recreation. Executed two times, with three and nine sound sources with predetermined positions. |
| 2 | Interactive source localization | Usability<br>Applicability | Interactive use of the system by the user, to evaluate the applicability of the system, in which the user could modify the positions of sound sources. |

Experiment was made with musicians and non-musicians. Each person was directed sit to a chair located in the center of the arrangement of loudspeakers and submitted individually to three executions of the musical piece, first in a blind hearing

session, then in an interactive hearing one, where one could manipulate the position(s) of sources in real time. Later they answered a questionnaire to evaluate the subjective item of the experience. Table 2 contains a characterization of the carried tests.

The questionnaire had questions based on attributes we intend to evaluate, and each one of these questions has an explanation of the parameter for better evaluation. Answers are based in a scale from one to five, as a quality indication of the measured parameter. In accordance with Berg (2003), this type of evaluation, where numerical scales are associated with verbal anchors, are commonly used in tests of this nature with good results.

### 6.6. Results

The following results refers to the system without the use of the second layer acoustic simulator, and with a second order Ambisonics system. Audition was accomplished using a quadraphonic square shaped configuration.

In the tests, answers of four users of the system had been evaluated (three musicians and one not). Table 3 summarizes the answers of the questionnaires filled by the four users (the correct source identification column indicates how many sources had been correctly pointed by the user in the two test cases of table 2 ).

**Table 3. Results of validation tests**

| User | Musician | Correct source indentification (3 and 9 sources) | Source Localization | Source Distance | System Usability |
|---|---|---|---|---|---|
| 1 | Yes | 3 / 6 | 4 | 4 | 5 |
| 2 | Yes | 3 / 6 | 4 | 3 | 4 |
| 3 | Yes | 3 / 7 | 4 | 4 | 5 |
| 4 | No | 3 / 5 | 3 | 4 | 5 |

During the experiment, users tended to fix the vision in a loudspeaker, as an aid to perceive the localization of the sound. Therefore, some of the answers had pointed the position of the loudspeaker next to the virtual sound source. To reduce this visual guidance and to improve the immersive experience, the users were instructed to close their eyes during the execution of test 1.

Due to the polyphonic nature of the piece texture (the instruments do not present easily independent melodic lines amongst them) the individual localization of the nine instruments were degraded during test 1. One revealed difficult to identify the strings position, maybe due to the choral style of its composition In the same way, the relatively fast interventions of the woodwind quartet (around five seconds), increased difficulty in locating the instruments. On the other hand, when the user could effectively move the instruments with the graphical interface, the identification was significantly facilitated (test 2).

The answers to the questionnaires, based on the two tests, scored an average of four points for source localization and distance. Maximum score was obtained for easiness of use.

## 7. Conclusions and Future Works

The system had a steady behavior on a commodity Pentium IV-based computer, and ran without performance problems, that is, without perceivable delays in the reproduction of sound, even when nine sounds sources were auralized simultaneously. However, we expect some decrease in performance when using both the acoustic simulator and the convolution of impulse responses. For this, the use of more powerful computers or a computer cluster may be necessary.

Results drawn from individual tests of the acoustic simulation block indicated the necessity of improvements, especially in the calibration of scales and values for the Ambisonic signals.

The lack of familiarization with the instruments and its timbral differences degraded the localization of the instruments, mainly between the strings, the two flutes and oboe with clarinete. Therefore, for source localization without access to the graphical interface (test 1), the acquired ability was necessary to distinguish timbres and to hear similar instruments executing different (melody) lines.

The results obtained during test 2, when users interacted with the system, showed that directionally and distance of the sources, based on second order Ambisonics encoding, were similar to a real sound field.

Through the experiments, mainly test 2, we conclude that the system is promising for using in musical applications and that it can be a low cost and valuable tool for composers, conductors and producers to use spatialization. The easiness in patch construction and use of the system corroborates this conclusion.

These results supply cues for further improvements. Some future works and possible extensions of the system can be enumerated:

1. advanced tests, with more users and based on standard subjective tests methodologies;
2. development of new blocks for the system of sound spatialization, such as the new acoustic scene descriptors (e.g. using X3D), user interfaces (e.g. joystick, head-tracking), acoustic simulators (e.g. finite ray tracing for arbitrary polyhedrons (BORISH, 1984)) and new encoders/decoders (e.g. WFS, binaural with HRTF, and MPEG Surround). Part of these are addressed in the scope of the OpenAUDIENCE[1] project;
3. experiments with three-dimensional configurations of loudspeakers, already implemented in the decoding block for cubic and dodecahedrical speakers configuration;
4. the use of high order Ambisonic encoders/decoders (DANIEL; MOREAU, 2003) to improve the sound field reconstruction;
5. Ambisonics decoding for any speaker configuration, not restricted to regular geometry dispositions (LEE; SUNG, 2003), allowing the adaptation of the

[1] Available at: http://openaudience.incubadora.fapesp.br/portal. Last access: 3 jun 2007.

system in places where the positioning of loudspeakers are restricted (e.g. CAVEs, where the projection cannot be blocked by a speaker).

6. automatic calibration of the system, to normalize the intensity and equalization of each speaker.

## Acknowledgements

The authors want to express their gratitude to LAMI laboratory of the School of Arts and Communication of the University of São Paulo (ECA-USP), for kindly providing the studio facility for sound recording experiments, and to prof. José Augusto Mannis, for his valuable technical and artistic assistance in recording sessions. We also thank the CAVERNA Digital laboratory of the Polytechnic School and all the music undergraduate students who have collaborated in the recording.

## References

Allen, J. B. and Berkley, D. A. (1979) "Image method for efficiently simulating small-room acoustics". Journal of the Acoustical Society of America, v.65, n.4, p. 943-950.

Bamford, J. S. (1995) "An Analysis of Ambisonic Sound System of First and Second Order". 270 p. Dissertação (Mestrado) – University of Waterloo. Waterloo, Canada.

Berg, J., Rumsey, F. (2003) "Systematic Evaluation Of Perceived Spatial Quality". In: AES International Conference On Multichannel Audio, 24., p.184-198.

Borish, J. (1984) "Extension of the image model to arbitrary polyhedral". Journal of the Acoustical Society of America, v.75, n.6, p. 1827-1836.

Daniel, J. (2001) "Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimedia". 319 p. PhD Thesis – l'Université Paris. Paris.

Daniel, J. and Moreau, S. (2004) "Further Study of Sound Field Coding with Higher Order Ambisonics". In: Proceedings of the 116th AES Convention, Berlin.

Faria, R. R. A.; Zuffo, J. A. (2006) "An Auralization Engine Adapting A 3-D Image Source Acoustic Model To Ambisonics Coder For Immersive Virtual Reality". In: Aes International Conference, 28., Piteå.

Faria, R. R. A. et al. (2005) "AUDIENCE - Audio Immersion Experiences in the CAVERNA Digital". In: Proceedings of the 10th Brazilian Symposium on Computer Music, Belo Horizonte. p. 106-117.

Furse, R. (2006) "First and Second Order Ambisonic Decoding Equations". Available at: http://www.muse.demon.co.uk/ref/speakers.html. Last access: 3 jun 2007.

Gerzon, M. (1973) "Periphony: With-Height Sound Reproduction", Journal of the Audio Engineering Society, v. 21, n. 1, p. 2-10.

Ives, C. E. (1953) "The Unanswered Question", New York: Southern Music Publishing Co. Inc.

Lee, S.; Sung, K. (2003) “Generalized Encoding and Decoding Functions for a Cylindrical Ambisonic Sound System”. IEEE Signal Processing Letters, v. 10, n. 1, p. 21-23.

Malham, D. and Myatt, A. (1995) “3-D Sound Spatialization using Ambisonic Techniques. Computer Music Journal, v. 19, n. 4, p. 58-70, inverno 1995.

Oppenheim, A.; Schafer, R. (1998) “Discrete-Time Signal Processing”. 2 Ed. Prentice Hall.

Puckette, M. (2007) “Pd Documentation”. Available at: http://crca.ucsd.edu/~msp/Pd_documentation/. Last access: 3 jun 2007.

Thomaz, L. F. (2007) “Aplicação à música de um sistema de espacialização sonora baseado em Ambisonics”, Dissertação (Mestrado), EPUSP, São Paulo.

Trochimczyk, M. (2001) “From Circles to Nets: On the Signification of Spatial Sound Imagery in New Music”. Computer Music Journal, v. 25, n. 4, p. 39-56.

# Can the Red Queen Help Catch the Snark? A Co-Evolutionary Waveform Transformation Approach

**Marcelo Caetano[1], Jônatas Manzolli[2], Fernando Von Zuben[2]**

[1]IRCAM – 1 place Igor Stravinsky – Paris, France 75004

[2]LBiC and NICS - University of Campinas - Brazil PO Box 6159

caetano@ircam.fr, vonzuben@dca.fee.unicamp.br, jonatas@nics.unicamp.br

***Abstract.*** *Many works, ranging from biology to optimization techniques, claim to have encountered the Red Queen face to face*[*]*. Nobody has ever been able to capture the Snark, though. In this paper, we describe a novel framework for waveform transformation entitled 'The Hunting of the Snark' based on the Red Queen Principle, which, in Lewis Carroll's words, states that "it takes all the running you can do, to keep in the same place". As an extension of our previous work that consisted of the application of evolutionary techniques to waveform synthesis, we generated waveform transformations that resulted from the exploration of the soundspace driven by a variation of the evolutionary 'arms race' paradigm applied to two populations of waveforms, denominated Predators and Prey. A mathematical distance between these populations gives fitness evaluation and the Predators chase the Prey, who try to escape. This spatial dynamics, which was dubbed 'Snark chase', gives rise to the parallel temporal evolution of two sets of waveforms that have the potential to be used in music composition, improvisation and possibly real-time performance. The genetic sound operators such as crossover, mutation and selection are explained in the light of the present implementation and preliminary results are shown.*

## 1. Introduction

Within the western musical framework, ever since the very dawn of electroacoustic music, the focus of a group of composers has been broadened from musical notes to musical sounds. Some compositional techniques sprouted from the manipulation of these sounds resulting, for example, in transformations of sonic features such as timbre or sound quality. Composers such as Trevor Wishart (2000) and Leigh Landy (1993) have produced works that can be said to gravitate around the utilization of sound transformations and its aesthetical implications. Early techniques included the use of analog equipment to store and manipulate the musical material (sounds themselves). An immediate consequence is that the results were restricted by the procedural limitations of the equipment itself. Enter the digital computer and its great musical potential and flexibility in sound manipulation. According to John Chowning (2000), digital waveforms can represent virtually any sound imaginable *given the correct sequence of numbers (digital sound samples)*. Max Mathews (1963) stated that "there are no theoretical limitations to the performance of the computer as a source of musical

[*] See, for example, Lythgoe 1998 and Paredis 1997.

sounds, in contrast to the performance of ordinary instruments". Therefore, with an appropriate representation of all the possible sounds, denominated the soundspace, and a proper manipulation of certain properties of the representation and/or exploration of certain regions of the soundspace, the process of sound transformation corresponds to the generation and manipulation of sounds that evolve in time and therefore can be potentially applied to music composition, improvisation and real-time performance.

Sound transformations are usually associated with timbral metamorphosis [Landy 1993]. Since the pioneering work of Helmholtz (1885), timbre has been closely associated with spectral contents. Consequently, most sound transformation techniques described in the literature make use of manipulations in the spectral domain by means of a sound model that attempts to describe exhaustively the temporal evolution of the partials over the course of the sound. These initiatives rely on the analysis/synthesis paradigm, which requires the previous analysis of a sound to permit the synthesis step. Among the most famous such models are the Phase Vocoder [Flanagan 1966] and spectral modeling synthesis (SMS) [Serra 1998], [Smith 1997]. Moorer (1978) has extensively described how the phase vocoder can be used in computer music applications and Wishart (2000) and Landy (1993) have outlined applications of the Phase Vocoder in sound transformations in the perspective of the Composers' Desktop Project (http://www.composersdesktop.com). There are approaches to both models attempting to calculate high-level descriptors [Serra 1998], [Amatriain 2002], [Tardieu 2004] that would allow us to manipulate salient timbral features of sounds independently, without affecting the other dimensions [Jensen 1999]. There is no consensus on what or how many these features are, however [Caetano 2005, 2007]. The approach most closely related to this work is that of Caetano (2007) based on the application of bio-inspired algorithms featuring characteristics of self-organization as tools to explore the soundspace of digital waveforms and generate sound transformations for music composition and improvisation. The method that will be presented in this paper is a variation of the evolutionary waveform synthesis (ESSynth) approach [Manzolli 2001a,b], which follows the principle that soundspace is virtually any digital waveform and that sounds can be generated by means that would promote the emergence of novel or even unexpected results. In other words, contrary to the analysis/synthesis approach, we propose the exploration of the soundspace of digital waveforms using evolutionary computation methods. Evolutionary computation is usually associated with creativity [Bentley 1999]. Dawkins (1986) stated that "as the search space gets larger, more and more sophisticated searching procedures become necessary. Effective searching procedures become, when the search space is *sufficiently* large, indistinguishable from true creativity". There are computer models that illustrate well this argument and constitute an instructive bridge between human creative processes and the evolutionary creativity of natural selection [Latham 1992]. In this framework, sound transformations are evolving digital waveforms. Caetano *et al.* (2005) have proposed the evolution of a population (set) of waveforms towards a static target set. This process ends when (or if) the waveforms reach the target. The focus of the current investigation is to co-evolve both sets, named Predators and Prey, allowing the evolutionary process to run ideally indefinitely, which is also known in the literature as an arms race [Dawkins 1986]. The goal of the Predator set is to chase the Prey (minimize their distance to the Prey set), while the Prey must escape the Predators (maximize their distance to the Predator set).

The next section briefly overviews the red queen principle (RQP) and its connection with co-evolution, attempting to present the motivation of the use of co-evolution as a paradigm for sound transformation. We then proceed to thoroughly explain 'The Hunting of the Snark'. The algorithm, the genetic parameters and operators are properly sketched. Next, we focus on the results, comprising the experiment and the discussion of its outcome. Finally, the conclusions and future work are presented.

## 2. The Red Queen Principle in Co-Evolution

Evolution can be defined as the accumulation of change in the individuals of a population over the generations due to natural selection. Holland (1975) devoted himself to the study of adaptive natural systems and, inspired by biological evolution, he proposed Genetic Algorithms (GAs) to indicate that adaptation mechanisms can be properly implemented in computers. GAs mimic nature in accordance with Darwin's survival of the fittest principle, exchanging information in a structured yet random way. GAs codify attributes that fully characterize the elements of a search space, using the language of computers. The resulting search space contains the candidate solutions, and the evolutionary operators will implement exploration and exploitation of the search space aiming at finding global optima. The GA iteratively manipulates populations of individuals at a given generation by means of simple genetic operations of selection, crossover and mutation. GAs tend to lose variability over the generations [Davis 1991]. Co-evolution helps preserve and even generate variability [Palazios 2004]. This can be understood as different strategies for catching the prey (or, alternatively, for escaping) being developed by individuals over the generations. Co-evolution refers to the simultaneous evolution of two or more interacting species. This framework for evolution may be conceived so that the species cooperate or compete. The application of competitive co-evolution to problem solving has been of interest in the GA community because *competition*, in its most general sense, encourages the generation of better *competitors* [Paredis 1995]. To the best of our knowledge, there have been no proposals for the use of co-evolution in waveform transformation.

### 2.1. Evolutionary Arms Races and the Red Queen Principle

There are ways in which mutation and natural selection together can lead, given enough time, to a building up of complexity, beauty and efficiency of design. Evolutionary 'arms races' is one of them, under the concept of co-evolution. There are arms races between predator and prey, parasites and hosts, even between males and females within one species [Dawkins 1986]. Arms races consist of the improvement in one lineage's (say prey animal's) equipment to *survive*, as a direct consequence of improvement in another (say predator's) lineage's evolving equipment [Dawkins 1986]. The principle of zero change in success *rate*, no matter how great the evolutionary progress in *equipment*, has been given the memorable name of the Red Queen Principle (RQP) by the biologist Leigh van Valen (1973). In *Through the Looking Glass* [Carroll 1872] the Red Queen seized Alice by the hand and dragged her, faster and faster, on a frenzied run through the countryside, but no matter how fast they ran they always stayed in the same place. Alice was understandably puzzled, saying, 'Well in our country you'd generally get to somewhere else-if you ran very fast for a long time as we've been doing.' 'A slow sort of country!' said the Queen. 'Now, here, you see, it takes all the

running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!' The RQP is directly related to positive feedback through symbiotic relations between individuals of the different populations competing, it raises the question "Can a natural evolutionary system support continual change?"

## 3. The Hunting of the Snark

The 'Hunting of the Snark' method was named after Lewis Carroll's homonymous nonsense poem that describes the voyage of a crew after an imaginary creature that cannot be caught. In this paper, 'The Hunting of the Snark' has two populations (sets) of sounds, Predators and Prey. Each population contains a number of individuals that are sounds represented by $N$ samples of a digital waveform at a sampling frequency of $FS$ samples per second. The waveform is the genetic code (genotype) that carries all the information regarding the sound and can be manipulated. The resultant sound (phenotype) is the characteristic that can be perceived. The 'Hunting of the Snark' algorithm is shown in Figure 1 and the input parameters are summarized in Table 1 and were adapted from Palacios-Durazo and Valenzuela-Rendón (2004). Fitness evaluation is given by a distance measure (Hausdorff metric) between Predators and Prey as follows: we calculate the Euclidean distance between two waveforms regarding them as vectors with $N$ dimensions. That is, each sound sample is considered as a vector component. As shown in the algorithm in Figure 1, the distance from each of the $K$ Predators to all the $M$ Prey is measured, resulting in a $K$ by $M$ matrix of distances. For each Predator, fitness is the minimum distance found from all the $M$ Prey. The same is done for each Prey. Now that each individual has a distance value attributed, the best Predator is the closest to the Prey, and the best Prey is the furthest away. Every individual "sees" all other individuals and diversity is not necessarily preserved in terms of speciation. This particular spatial dynamics described in our work was dubbed 'Snark chase'. The fitness measure produces a non-linear mapping of the genotype into the phenotype and the 'Snark chase' dynamics conducts the exploration of the soundspace. Thus, we defined the 'Snark chase' as "the improvement in the Prey's evolving equipment to *escape*, as a direct consequence of improvement in the Predator's evolving equipment to *chase* the Prey."

```
(*Initialize Populations*)
        Load predefined Predator population
        Load predefined Prey population
(*Main Cycle*) generations
    repeat
(*Competition Cycle*) Snark chase
    for each p1 ∈ Predators
    for each p2 ∈ Prey
        measure distance between p1 and p2
    end for
    end for
        Fitness of Predators and Prey calculated based on the minimum distance of each individual to all individuals
        in the competing population
            One generation of a GA is applied to Predators
                Crossover, mutation, selection
            One generation of a GA is applied to Prey
                Crossover, mutation, selection
    until termination criteria met
```

**Figure 1. 'The Hunting of the Snark' algorithm.**

### 3.1. Genetic Parameters

The user will adjust the search according to predefined requirements achieved by the manipulation of the parameters of the genetic algorithm. It follows closely the original ESSynth method [Manzolli 2001a,b], [Caetano 2005].

*3.1.1. Size of the Population*

The size of the population directly affects the efficiency of the GA [Davis 1991]. A small population supplies a small covering of the search space of the problem. A vast population generally prevents premature convergences to local solutions. However, greater computational resources are necessary [Davis 1991]. In 'The Hunting of the Snark', the size of the population does not change along the generations.

*3.1.2. Coefficient of Mutation*

It determines the probability of mutation [Holland 1975]. A properly defined coefficient of mutation prevents a given individual of the population from stagnating in a particular position and also promotes exploration and exploitation of the search space. A very high coefficient of mutation causes the search to become essentially random and increases the possibility of destroying a good solution [Davis 1991]. In 'The Hunting of the Snark', the coefficient of mutation ranges from 0 to 1. It is user defined (input argument to the algorithm) and does not change along the generations, being responsible for limiting the noisy distortion caused by the mutation operator, described in section 4.2.2.

### 3.2. Genetic Operators

It is the genetic operators that transform the population along successive generations, being responsible for the emergence of evolution in computers. A standard genetic algorithm evolves, in its successive generations, by means of three basic operators, crossover, mutation and selection, described as follows.

*3.2.1. Crossover*

It represents the mating between individuals [Holland 1975]. The central idea of crossover is the propagation of the characteristics of the individuals in the population by means of the exchange of information segments between them, which will give rise to new individuals. In 'The Hunting of the Snark', crossover operation exchanges chromossome segments, i.e. a certain number of samples, between individuals sharing the same 'gene pool' through a smooth transition. There is no crossbreeding between Predators and Prey. There are many different possibilities concerning the selection of which individuals will be crossed-over with each other. One must be aware, though, that

**Table 1. Input parameters of 'The Hunting of the Snark'**

| | |
|---|---|
| *N* | Number of samples per individual |
| *FS* | Sampling rate |
| *K* | Number of Predators |
| *M* | Number of Prey |
| *NumInt* | Number of generations |
| *coefMut* | Coefficient of mutation |

the selection operator for crossover has a great influence in the diversity of the genetic pool. We wish to preserve diversity to its maximum to guarantee a good exploration of the search space along the generations. However, diversity maintenance must not slow down convergence. Many strategies were tried beforehand: randomly selecting both individuals, crossing each individual with a randomly chosen individual, crossing the best individual with a randomly chosen individual. A suitable strategy was found to be crossing each individual with the best individual in each generation to pass along the best individual's gene pool while maintaining diversity.

The crossover operation is depicted in Figure 2 and consists in adding a sound segment from the best individual to each individual of a given population. The operation is somehow similar to granular synthesis and can be interpreted as adding a single sonic quantum from the best individual. The segment is obtained as follows: we randomly generate the width of the segment, called *slice*, between *N/10* and *N/5*. Next we generate the center of the segment, which must be a value between *1+slice/2* and *N-slice/2* to guarantee that the entire segment always falls between *1* and *N*. We then proceed to create the crossover functions, shown in Figure 3 parts a) and c). The crossover function in part c) of Figure 3 is a Hanning window centered at *center* with width equal to *slice* and zero-padded to size *N*. The crossover function in part a) of Figure 3 is one minus the previous function. Then the function in part a) of Figure 3 is applied to one individual of a given population, resulting in the segment seen in Figure 3 part b), and the crossover function shown in Figure 3 part c) is applied to the best individual of the same population, resulting in the segment seen in Figure 3 part d). The resultant segments are added producing another individual, denominated Breed, shown in part c) of Figure 2. It should be noted that the individual being crossed-over with the best individual in that generation is preserved almost entirely, receiving only a narrow segment from the best. The crossover functions are generated for each individual of both populations.

*3.2.2. Mutation*

It produces random modifications and is responsible for the introduction and maintenance of genetic diversity in the population [Holland 1975]. In 'The Hunting of the Snark', the mutation operator is as follows: *N* values randomly generated between 1−*coefMut* and 1 are used to multiply each of the corresponding elements of a given individual (waveform) of the population. This operator introduces a certain noisy distortion to the original waveform that is equivalent to a non-linear perturbation in the genotype that reflects in phenotype. Notice that the higher the value of *coefMut*, the stronger the perturbation.

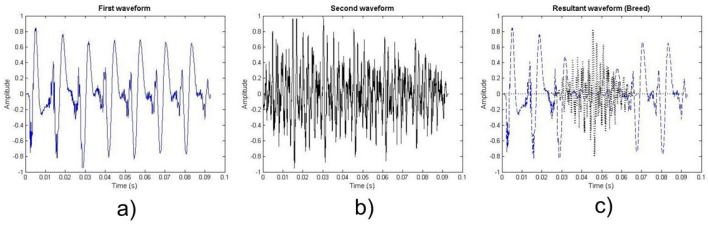



**Figure 2. Depiction of the crossover operation in time domain. Part a) and b) show the waveforms to be crossed-over and part c) shows the result of the operation.**

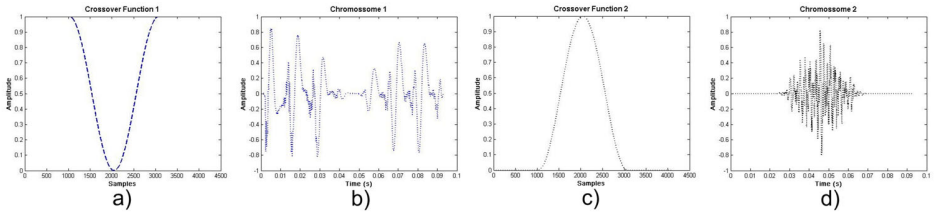



**Figure 3. Crossover functions and the sound segments that result after their application.**

*3.2.3. Selection*

After mutation all individuals are passed to the next generation. The absence of selection for the next generation is a consequence of practical experience. We found that all the selection strategies we tried (roulette wheel, random selection and rank) caused too much selective pressure and consequently the diversity dropped to zero very soon.

## 4. Results

The purpose of this section is not only to present some outcomes of co-evolution in terms of sonic results, but also to allow the reader to have a better understanding of how it works. For such, we need to highlight that the evolution of the waveforms corresponds to a sound transformation procedure, that the Snark chase dynamics is responsible for how this transformation is done and that co-evolving the sounds has the desired side effect of helping the preservation of diversity, apart from the obvious fact that co-evolution in this case means that both populations of waveforms are being transformed at the same time. The result of this experiment can be heard at `ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/SBCM2007/sound_samples.zip`.

The overall result of the application of the method can be easily inferred as a sound transformation procedure that adds partials (spectral complexity) to the sounds over the generations mainly because of the crossover operator. In order to try to "show" the transformation procedure, the output sound set resulting from a run of the program will be shown and discussed. The Predator waveforms were taken from a recording of the electric bass and the Prey from piano chord sounds. The resultant waveforms of one individual from each population will be presented in generations 1, 2, 5, 10, 50 and 100 to illustrate the transformations of the waveform along the generations. The generational waveform display attempts to draw attention to the variability each individual presents over the generations, despite the fact that it preserves some of its original information contents. A 3D plot of the short-time Fourier transform (STFT) – thereon referred to as dynamic spectrum - of one individual from each of the populations in the first and last generations will be compared to exemplify the spectral transformations induced by the method. The dynamic spectrum of the original and resultant individual highlights the changes to the temporal envelope evolution of the partials induced by the method. Finally, we show plots of the distances of the individuals along the generations. It might be surprising at first to see the distance value being used as a measure of diversity since different individuals can have the same distance. On the other hand, one individual can only have a single distance value. Therefore, different distance values necessarily mean different individuals. We can even go one step further and suggest that small distances imply small differences (and vice-

versa) because waveforms (individuals) that are only slightly different present a small distance (they are correlated vectors). But we must be careful because, although it is correct for the genotype (waveforms), we know it is far from necessarily true for the phenotypic differences (how they sound). All we need to remember is that differences in phase do not affect greatly the sound but alter considerably the shape of the waveform.

### 4.1. Experiment

Since the experiment basically consists of a run of the program, we are committed to show that the result is a different sound transformation procedure for each population. Due to the loss of variability over the generations, the individuals of the same population do tend to sound alike after one or two hundred generations. However, we know that the initial individuals were different, so we will always have variations of the transformation. Therefore, we consider the results satisfactory if we verify that one individual from each population actually corresponds to a sound transformation and that they are different. The genetic parameters adopted in the experiment were 10 individuals in both the Predator and Prey populations to supply a satisfactory coverage of the search space. The program was run for 200 generations. *FS* is 44,100 so the highest representable frequency is 22,050 Hz. *N* was set 4096 so each chromosome is a wave-format sound segment of approximately 0.0929s. The coefficient of mutation was set 0.05 and was obtained empirically because higher values have shown to distort so much the waveforms that the results were almost too noisy and masked the transformation. Next, we present the waveforms of both Predator 1 in Figure 4 and Prey 1 in Figure 5 in generations 1, 2, 5, 10, 50 and 100. The top row from left to right shows generations 1, 2 and 5 and the bottom row, also from left to right, show generations 10, 50 and 100. Figure 6 shows 3D plots of the STFT of the original Predator 1 and Prey 1 and their transformed versions after 200 generations. The top row shows Predator 1, from left to right in the first and in the last generation. The bottom row shows Prey 1, from left to right in the first and in the last generation. These figures are similar to spectrograms in that they show the temporal evolution of the envelope of each partial frequency along the course of the sound. Finally, Figure 7 shows plots of the fitness values of both populations in all 200 generations. Part a) shows fitness value of all 10 Predators in all 200 generations, part b) shows fitness value of all 10 Prey in all 200 generations and part c)

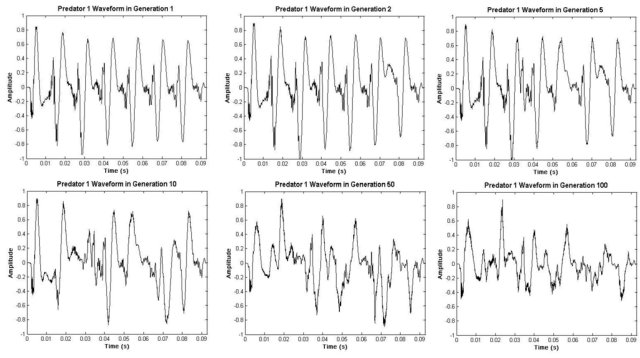



**Figure 4. Waveform evolution for Predator 1. From left to right, the top row shows the waveform Predator 1 in generations 1, 2 and 5 and the bottom row in generations 10, 50 and 100.**

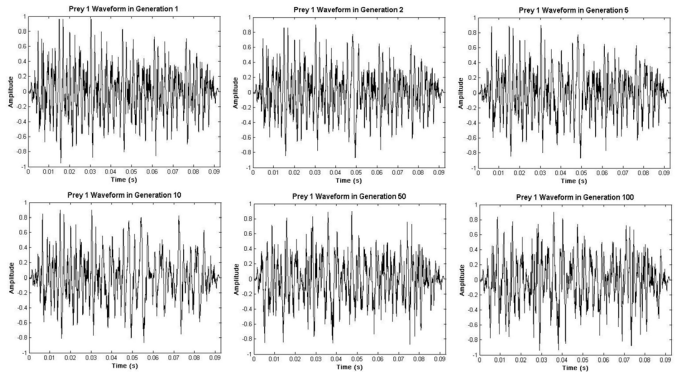



**Figure 5. Waveform evolution for Prey 1. From left to right, the top row shows the waveform Prey 1 in generations 1, 2 and 5 and the bottom row in generations 10, 50 and 100.**

on top fitness value of the best Predator in each generation and at the bottom fitness value of the best Prey for all 200 generations.

### 4.2. Discussion

Figures 4 and 5 show clearly the waveform transformation in the course of the generations. For example, from a close examination of Figure 4, it becomes clear how in the second generation Predator 1 already possesses one cycle that is different from the others. It comes from the best individual in that generation via the crossover operator. The Prey 1 waveform shown in Figure 5 is somewhat noisier than that of Predator 1, making it more difficult to visually identify the changes it suffers over the generations. However, it becomes more evident if we examine the dynamic spectrum from Prey 1 in the first and last generations in Figure 6, which illustrates the adding up of spectral complexity in the individuals along generations. The temporal envelopes of the partials have changed a great deal, which means that not only the waveform itself has changed, but also the spectral information. We need to bear in mind that while changes to the

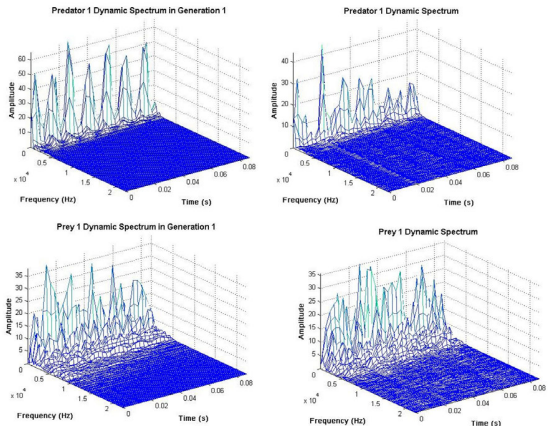
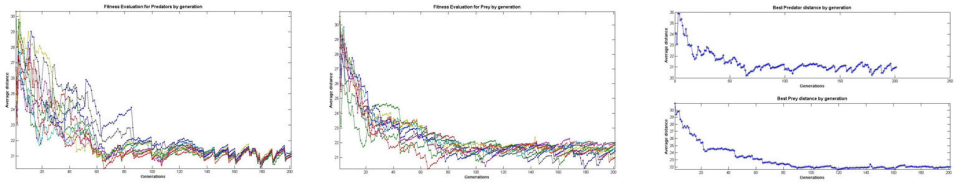



**Figure 6. Spectro-temporal transformation to Predators and Prey. The top row shows the dynamic spectrum for Predator 1 and the bottom row for Prey 1. The left column shows both individuals in the first generation and the right in the last generation.**



a) b) c)

**Figure 7. Distance evolution along generations. Part a) shows the distance value for all the Predators, part b) for all the Prey and part c) shows the distance value of the best Predator on top and best Prey at the bottom.**

phases of the partials modify the waveform but are not perceived in general, changes to the amplitude of the partials also affect the shape of the waveform and are certainly perceived as a different sound. Obviously, the same goes for Predator 1 also shown in Figure 6, which permits us as well to "see" the result of the distortion caused by the mutation operator as the noisy contents in high frequencies that the resultant waveforms present and that were not present in the dynamic spectra of the original waveforms.

Finally, to analyze Figure 7, it is important to remember that the waveforms are kept with the same label along all the generations. This means that the waveform that was loaded as Predator 1, for instance, is never relabeled during the process; it only suffers the changes made by the crossover and mutation operators. Furthermore, we expected the overall distance of Predators to decrease, indicating that they do chase the Prey. This can be confirmed by visual inspection of Figure 7 part a), which shows a decrease of distance values over the generations that means that the Predators do get closer to the Prey by means of the genetic operators alone. The evolution of the distance value for the best individual in each generation confirms this general tendency. Figure 7 also hints at the maintenance of diversity because of the different fitness values that are represented by the curves farther apart. For the Predators, from the beginning of the generations until at least halfway through to the end there is considerable variability, but then around the 150$^{th}$ generation the Predators seem to converge to a single oscillating genotype (waveform) that is supposedly the most adapted to chase those specific Prey. This oscillating behavior seems to confirm the co-evolutionary predator-prey dynamics. The Prey do not group together like this over the generations, keeping roughly the same diversity. This might reflect a strategy developed by the Prey to keep away. Maybe keeping a more diverse gene pool helps the Prey avoid be reached by the Predators. Although it is too soon to jump to conclusions like this, we would like to be able to find out whether a specific genotype for the Prey induces the Predators to converge to a certain specific region of the soundspace similarly to what the static target sounds in ESSynth cause (Caetano et al. 2005). This would mean we could "direct" the transformations of the Predators towards some desired general result with the right population of Prey.

## 5. Conclusions

We have described 'The Hunting of the Snark', a waveform transformation method that has co-evolution as paradigm. It can be seen as a variation of the evolutionary sound synthesis method (ESSynth), which applies a GA to a population of waveforms that are driven towards another static population as target. Here, the target waveforms move away motivated by a variation of a model of co-evolution. We denominated the two

populations of waveforms Predators and Prey and the sound transformation method consists of the Predators chasing the Prey and the Prey trying to run away from the Predators. Both populations are being transformed in parallel in what we have dubbed Snark chase. Ideally, the Snark chase dynamics means that, as long as we have variability, the process goes on. Put in a different way, the Predators will never reach the Prey if they keep successfully running away, but they will keep chasing them anyway. This supposedly means that the sound transformation process is virtually endless because the target is moving away. However, due to the nature of the crossover operator, its exhaustive application over hundreds of generations forces homogeneity in the genotype (waveforms), which necessarily means that there will be a loss of diversity along the way. This loss of diversity corresponds to the convergence of the algorithm. We expect, though, the Predators to try and catch the Prey and the Prey to run away as fast as they can. This fact should be reflected as a decrease in distance values for Predators. We have indeed found a decrease in distance for the Predators over the generations that can be interpreted as the Predator population approaching the Prey. Also, upon convergence, there is indication of oscillatory behavior, characteristic of co-evolution. The Prey are being transformed just like the Predators. The very same crossover, mutation and selection operators are applied to both populations and the waveforms of one individual from each population in some predefined generations we have shown confirm that both Predators and Prey are being transformed, endorsing the use of co-evolution as paradigm for evolutionary sound transformation.

Future work might include testing different mutation operators; comparing the results of evolutionary and co-evolutionary waveform transformation, verifying if a certain group of Prey drives the Predators towards a specific region of the soundspace and even experimenting with multi objectives, so that the Predators would not simply chase the Prey, corresponding to different kinds of competition.

## Acknowledgements

This work was fully developed under the supervision of Profs. Fernando Von Zuben and Jônatas Manzolli both at the University of Campinas and supported by FAPESP (process 03/11122-8) and CNPq. The main author is presently supported by CAPES (process 4082-05-2) and advised by Prof. Xavier Rodet at IRCAM.

## References

Amatriain, X., Bonada, J., Loscos, A., Serra, X. ``Spectral processing," in DAFX - Digital Audio Effects: 373-438, John Wiley and Sons, 2002.

Bentley, P. (1999). Evolutionary Design by Computers. S. F., Morgan Kaufmann.

Caetano, M., Manzolli, J. and von Zuben, F. Self-Organizing Bio-inspired Sound Transformation. EVOMusart workshop. Proc. of the EVOStar, 2007.

Caetano, M., Manzolli, J., von Zuben, F. Interactive Control of Evolution Applied to Sound Synthesis. Proc. of FLAIRS, EUA, 2005.

Carroll, L. Through the looking glass and what Alice found there. Macmillan, 1872.

Chowning, J. M. Digital Sound Synthesis, Acoustics and Perception: A Rich Intersection. Proc. of the DAFX-00, 2000.

Davis, L. “Hanbook of Genetic Algorithms”. New York:Van Nostraud Reinhold,1991.

Dawkins, R., “The Blind Watchmaker”, Penguin Books, 1986.

Flanagan, J., Golden, R. Phase Vocoder, Bell Syst. Tech. J. 45:1493-1509, Nov. 1966.

Helmholtz, H. von. On the Sensations of Tone. London, Longman, 1885.

Holland, J., Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, 1975.

Jensen, K. Timbre models of musical sounds. Ph.D. Dissertation, Dep. Computer Science, Univ. Copenhagen, 1999.

Landy, L. (1993) Sound Transformations in Electroacoustic Music. http://www.composersdesktop.com/landyeam.htm. Access in 06/2007.

Latham, W., Todd, S., “Evolutionary Art and Computers”, Academic Press, 1992.

Lythgoe, K. A., Read, A. F. Catching the Red Queen? The advice of the rose. Trends Ecol. Evol. 13: 473-474, 1998.

Manzolli, J, A. Maia Jr., Fornari J.E. and Damiani,F. 2001a. The Evolutionary Sound Synthesis Method. ACM Multimedia, Ottawa, Ontario, Canada.

Manzolli, J, A. Maia Jr., Fornari J.E. and Damiani,F. 2001b. Waveform Synthesis using Evolutionary Computation. Proc. of the VII SBCM.

Mathews, M. V. The Digital Computer as a Musical Instrument. Science, 142 (3592), pp. 553-557, 1963.

Moorer J. The use of the phase vocoder in computer music applications, Journ. of the Audio Eng. Soc., 26 (1/2), 1978.

Palacios-Durazo, R., Valenzuela-Rendón, M. Similarities between Co-Evolution and Learning Classifier Systems and their Applications, GECCO 1: 561-572, 2004.

Paredis, J. Coevolutionary computation. Art. Life. 2 (4): 355-375, 1995.

Paredis, J. Coevolving cellular automata: be aware of the red queen! Proc. of ICGA: 393-400, 1997.

Serra, X. J. Bonada. "Sound Transformations Based on the SMS High Level Attributes", Proc. of DAFX Workshop, 1998.

Smith, J. O. (2007) Spectral Audio Signal Processing. CCRMA, Stanford University. http://ccrma.stanford.edu/~jos/sm/Spectral_Modeling_Synthesis.html. Access in 06/2007.

Tardieu, D. Synthèse et transformation sonore par descripteurs de haut-niveau. Master thesis, Université Aix Marseille, 2004.

Van Valen, L. A new evolutionary law. Evolutionary Theory, 1: 1-30, 1973.

Wishart, T. (2000) Computer Sound Transformation: A Personal Perspective from the UK. http://www.trevorwishart.co.uk/transformation.html. Access in 06/2007.

# Context Sensitive Harmonic Processor

**Andre Luiz Luvizotto**[1] **, César Rennó Costa** [1 2] *

[1]Núcleo Interdisiplinar de Comunicação Sonora - NICS
Caixa Postal: 6166, Campinas, São Paulo, Brasil

[2]Laboratório de Bioinformática e Computação Bio-Inspirada - LBiC
Caixa Postal: 6166, Campinas, São Paulo, Brasil

andre@nics.unicamp.br, cesar@nics.unicamp.br

***Abstract.*** *We present here a new-fashioned audio processor directed to perform harmonic distribution manipulation. It's a tool that operates over timbre, being sensitive to pitch context. The process is based on a wavelet-based filter-bank that is dynamically tuned to the note's fundamental pitch. As a single entry process it acts as harmonic equalizer while with two entries it works as harmonic controller.*

## 1. INTRODUCTION

Helmholtz [Helmholtz, 1863] on his classic "On the Sensations of Tone as a Physiological Basis for the Theory of Music" defined tone as a combination of a fundamental and countless overtones. Followed by other studies, its consensus that harmonic distribuiton has a primary role over timbre perception of musical sounds. Thus, on music sounds, variations on partials implies on variations over timbre.

We present here a processor focused on the timbre manipulation of musical interesting sounds by means of partials equalization. It's based on dynamic wavelet filter bank proposed in [Beltran and Beltran, 2003] controlled by a pitch detection algorithm. A preview version limited to this application has been presented in [Luvizotto and Costa, 2007].

The processor can also work as a harmonic controller of inharmonic sounds when using a harmonic material to control pitch detection and applying the filter bank over another sound. In this sense it actually would analog to an automatic and dynamic subtractive synthesizer.

The Wavelets transform is a method that combines decomposition in elementary contributions and hearing like properties [Kroland-Martinet, 1988]. Its main feature for our application is the possibility of fast filter design with few parameters. Also, we consider the pitch detection which can be a very complex task due large bandwidth and inharmonic partials [Jehan, 1997].

The paper is outlined as follow: in the first section we treat the main aspects about wavelets analysis and the mathematical background, including the formulation of the used scale and shift parameters as well as the chosen mother wavelet. In the following section we handle the pitch detection algorithm with an overview of the Harmonic Product Spectrum method. In the last two sections there will be an explanation of the implementation and also a discussion about the results and performance of our model.

*Supported by Fapesp.

## 2. WAVELET ANALYSIS AND FILTER BANK

Usually the process of partial extraction is based on *Short Time Fourier Transform*, but as we know by the Heisenberg uncertainty principle an ideal time localization gives rise to a non-ideal frequency localization and vice versa[Chui, 1997][Mallat, 1998].

*Wavelet Analysis* can be seen as a powerful tool to fix this problem that provides a flexible time-frequency window and that is the main reason to choose wavelets to implement the filter-bank. In formulating a Continuous Wavelet Transform (CWT) a scale parameter is introduced to adjust the width of the sliding time window process. The mathematical basis of the Fourier Transform are sine waves whereas the basis of the CWT is a wavelet family generated by the mother wavelet $\psi$(t).

As proposed by Beltrán in [Beltran and Beltran, 2003] we are considering a complex generalization of the Morlet wavelet given by:

$$\psi(t) = C' e^{\frac{-t^2}{2}} \left( e^{j\omega_0 t} - e^{\frac{-\omega_0}{2}} \right) \tag{1}$$

for the mother wavelet.

This function as proposed by Kronland-Martinet et al [Kroland-Martinet, 1988] requires small corrections to ensure that the admissibility condition for an analyzing wavelet is satisfied. However, in practice taking $w_0 > 5$ is enough [Beltran and Beltran, 2003]. In this case the Fourier transform of the Complex Morlet's wavelet is:

$$\hat{\psi}(\omega) = C e^{\frac{-(\omega-\omega_0)^2}{2}} \tag{2}$$

C' e C are normalization constants in the time and frequency domain, respectively.

As we are interested on a filter bank we need to have how to control the frequency resolution of each band. This way we can tune all the bands of our processor to the sound's partials of a given note. We should included a parameter $k$ in the equation 2 to control the bandwidth of the filter. So we have in the frequency domain:

$$\hat{\psi}(\omega) = C e^{\frac{-(\omega-\omega_0)^2}{2k}} \tag{3}$$

The set of scale is employed to provide a logarithm-resolution on the frequency axis. We will also insert, in equation 3, a discrete factor $s_i$ to slide the central frequency of the first band, i.e, changing this factor we will be sliding the first band and generating all the others. To prevent aliasing problems the filter bank generates first the bands with highest frequencies. That is, it gets the fundamental pitch $f_f$ of the played note, calculates the last frequency band that will be used and then calculates the central frequencies $\omega_c$ of the remaining bands changing $s_i$ as follow:

$$\omega_{c_1} = f_f N \tag{4}$$

and

$$\omega_0 = s_{min} \omega_{c_1} \tag{5}$$

We will call $s_1$ of $s_{min}$ cause it can be considered a tuning factor related to the Nyquist criterion which fine tunes $\omega_0$ to the desire first (highest) band's central frequency $\omega_{c_1}$.

And for $2 \leq i \leq N$ we have:

$$s_i = \frac{N}{n_p} \tag{6}$$

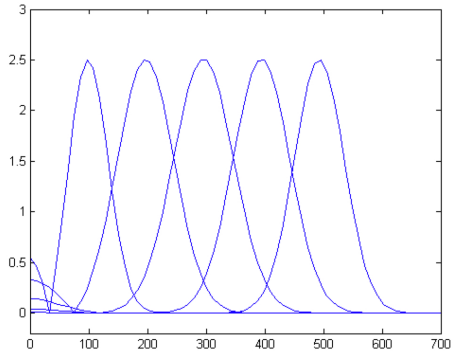



**Figure 1: Filter bank generated by the mother wavelet presented in equation 9 with $\omega_c = 100$Hz, $k = 22000$ and $s_{min}$ = 6.91 and $s_i$ given by equations 6.**

$$\omega_c = \frac{\omega_0}{s_i} \tag{7}$$

$N$ is the number of bands and $n_p$ is the band's number of the $i$-*th* frequency that is being calculated. We can see in the equation 7 the relation between the central frequency $\omega_c$ and the frequency $\omega_0$ for all bands. Then we have in the frequency domain the follow equation:

$$\hat{\psi}(\omega) = C_s e^{\frac{-(s_i\omega-\omega_0)^2}{2k}} \tag{8}$$

where $C_s$ is also a normalization constant. We have in equation 8 the main expression for the filter bank in the frequency domain. We need now to find its *time* representation to be able to use the convolution representation to filter the input signal. The follow expression was obtained from the equation 1, taking into account properties of Fourier transforms and by changing the continuous factor $a$ for the discrete one $s$:

$$\psi(t,s) = \frac{\sqrt{k}}{s} C' e^{\frac{-kt^2}{2s^2}} \left( e^{\frac{j\omega_0 t}{s}} - e^{\frac{-\omega_0}{2\sqrt{k}}} \right) \tag{9}$$

The equation 9 presents the mother wavelet that will be used with the frequency slide parameter $s$ and bandwidth parameter $k$.

Now we'll expose the pitch detector algorithm.

## 3. PITCH DETECTION

Pitch detection algorithms can be classified in two separate categories, time-domain based on period detection and spectral-domain. The first kind of detection seems to be the most straightforward idea, which consists in looking to the input signal as an amplitude fluctuations in the time domain and try to find repeating patterns from the waveform that could lead us to its periodicity. However due to its limitations we will use in this paper the second category of pitch detection algorithms. The chosen method is *Harmonic Product Spectrum*, or HPS.

If the input signal is a note, with a well defined pitch, then its spectrum should consist of a series of peaks, corresponding to harmonic components that are integer multiples of the fundamental frequency. The HPS algorithm measures the maximum coincidence for these harmonics [de la Cuadra et al., 2001].

If we compress the spectrum a number of times (downsampling), and compare it with the original spectrum, we can see that the strongest harmonic peaks line up. The first peak in the original spectrum coincides with the second peak in the spectrum compressed

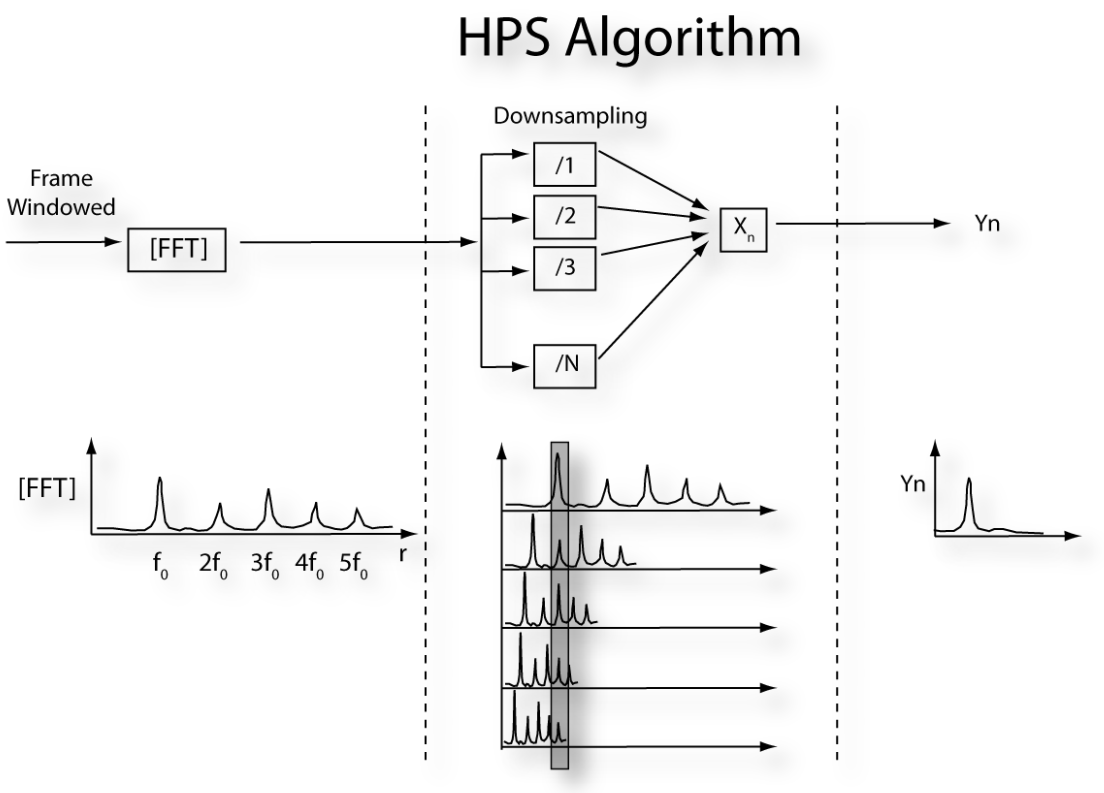



**Figure 2: Overview of the HPS algorithm.**

by a factor of two, which coincides with the third peak in the spectrum compressed by a factor of three. Hence, when the various spectrums are multiplied together, the result will form clear peak at the fundamental frequency. An comprehensive overview of this method can be found in [Galembo and Askenfelt, 1994]. Figure 2 demonstrates the HPS algorithm graphically.

## 4. IMPLEMENTATION

The algorithm was implemented with *Matlab* according to the diagram of the figure 3:

The input signal first reaches the pitch detector (that we call by Dr. Pitch) which outputs the fundamental frequency's value $f_f$ to the filtering stage. Then the filter parameters $\omega_0$ and $s_i$ are calculated and inserted into the equation 9. As we know each band of the filter is related to a wavelet with bandwidth factor $k$ and central frequency $\omega_c$ which is sampled with a sampling frequency $f_s$. As a results we have a matrix $M_{m\times n}$ where $m$ is the number of sampled points and $n$ the number of filter bands $N$.

In practice $M$ is left-side multiplied by a weight's matrix $P_{1\times n}$ that has the informations about the gain of each band, i.e, values between 0 and 1 came from the processor' slides. At last this resulted matrix is time convoluted with the input sound vector providing the filtered output. When the next signal comes Dr. Pitch sends the new $f_f$ to the filter stage. The parameters are actualized to generate the up-to-date wavelets and go on like before.

We used a Hanning window with 1024 and also 2048 samples with overlap of 50%. The window size depends on the lowest frequency (large period) of the input signal. We are looking for other methods of pitch detection that could be more efficient and computationally better.

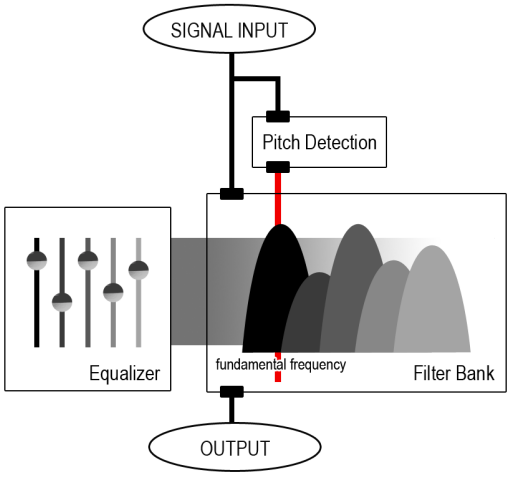



**Figure 3: Implementation Block Diagram**

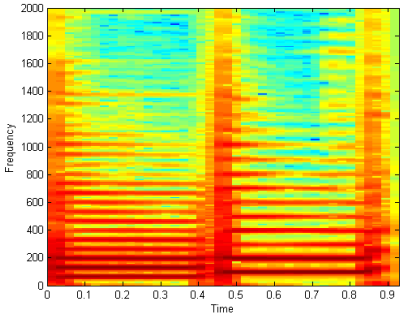



**Figure 4: Sonogram of the input signal.**

## 5. RESULTS

We applied the proposed processor with ten and twelve bands to many different sounds. Guitar, bass, percussion instruments, some different guitar tunings, for example, with the 6th string tuned to C1 since we were interested in evaluate how Dr. Pitch would work at extreme conditions. These examples can be find at www.nics.unicamp.br/∼andre/processor.

We can see on the figure 4 that Dr. Pitch worked very well, tracking both of notes correctly. We can also observe the difference between the spectrograms of figures 4 and 5 considering the number of partials presented. Only the first five partials were letting pass in figure 5 while all others five were filtered.

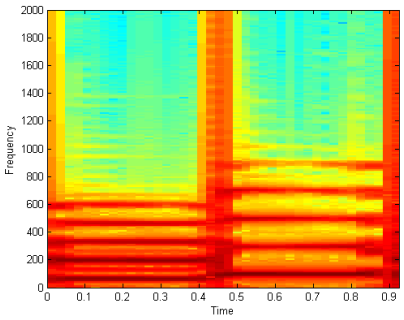



**Figure 5: Sonogram of the output signal.**

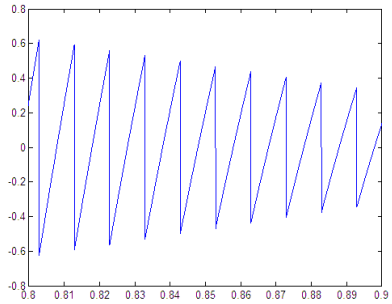



**Figure 6: Original sawtooth wave.**

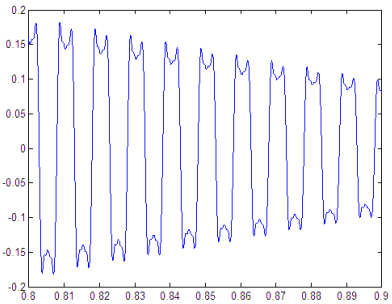



**Figure 7: Filtered sawtooth wave. Notice that it is closed to a square waveform.**

As a second example we have a sawtooth wave on figure 6 where its first four even harmonics were filtered. We can see on figure 7 that the resulted waveform is closed to a square wave, as expected. The more we increase the number of the filtered partials the more waveform matches to the square waveform.

Finally a square wave, figure 8, were filtered letting pass just the fundamental frequency. As we can observe on figure 9 the resulted waveform is closed to a simple sine.

## 6. CONCLUSION AND FURTHER IMPLEMENTATIONS

We presented the first results of a new-fashioned sound processor. A method of partials extraction was proposed as well as a pitch detector based on the autocorrelation function. The essential mathematical tools was revised in the first sections and a briefly discussion of the pitch detection in the section three.

This first prototype made with *Matlab* could give us a good idea of the sonic

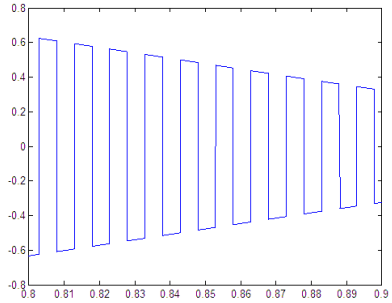



**Figure 8: Original square wave.**

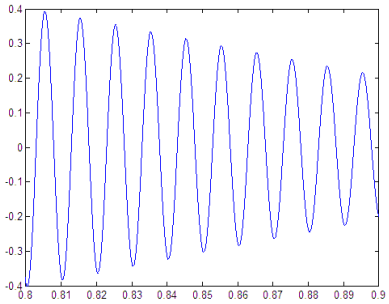



**Figure 9: Filtered square wave. Notice that it is closed to a simple sine.**

potential of the model. The best results were obtained in low frequencies sounds for example with bass drums, bass and lower guitar sounds.

We are currently developing a low latency C++ implementation to be used as a cross-platform plugin. We are also verifying the possibilities of working with other pitch detection methods, for example a wavelet based algorithm, and use it as another parameter on the graphical interface that could be selected by the user depending which kind of sound it wants to filter for example bass, brass, clarinet, piano etc.

## 7. ACKNOWLEDGMENT

We would like to thank Professor Adolfo Maia Jr. for helping us with the mathematical revision, formalization and kindly tips.

## References

Beltran, J. R. and Beltran, F. (2003). Additive synthesis based on the continuous wavelet transform: A sinusoidal plus transient model. In *Proc. of the 6th Int. Conference on Digital Audio Effects*.

Chui, C. K. (1997). *Wavelets: a mathematical tool for signal processing*. Society for Industrial and Applied Mathematics.

de la Cuadra, P., Master, A., and Sapp, C. (2001). Efficient pitch detection techniques for interactive music. *Proceedings of ICMC*.

Galembo, A. and Askenfelt, A. (1994). Measuring inharmonicity through pitch extraction. *STL-QPSR*, 35(1):135–144.

Helmholtz, H. V. (1863). *On the Sensations of Tone as a Physiological basis of the Theory of Music*. Dover Publications.

Jehan, T. (1997). *Musical Signal Parameter Estimation*. PhD thesis, Université de Rennes 1.

Kroland-Martinet, R. (1988). The wavelet transform for analysis, synthesis, and processing of speech and music sounds. *Computer Music Jornal*, 12(4):11–20.

Luvizotto, A. L. and Costa, C. R. (2007). Harmonic processor. In *Convenção Nacional da AES Brasil*.

Mallat, S. (1998). *A Wavelet Tour of Signal Processing*. Academic Press.

# Analytical Features to Extract Harmonic or Rhythmic Information

**Giordano Cabral[1], Jean-Pierre Briot[1], Sergio Krakowski[2], Luiz Velho[2], François Pachet[3], Pierre Roy[3]**

[1]Laboratoire d'Informatique de Paris 6 (LIP6), Université Paris 6 - CNRS
104 avenue du Président Kennedy, 75016 Paris, France

[2]Instituto Nacional de Matemática Pura e Aplicada (IMPA)
Estrada Dona Castorina, 110, 22460-320 Rio de Janeiro, RJ, Brazil

[3]SONY Computer Science Lab
6 rue Amyot, 75005 Paris, France

{Giordano.Cabral,Jean-Pierre.Briot}@lip6.fr,
{skrako,lvelho}@visgraf.impa.br, {pachet,roy}@csl.sony.fr

***Abstract.*** *This work aims to evaluate the effectiveness of EDS as a tool to automatically extract descriptors for real-world problems, such as melody extraction, chord recognition, and sound classification, comparing its performance and development time to traditional approaches. Each of these problems constitutes a case study, and along with the comparative results we present some remarks about the descriptor extraction procedure.*

## 1. Introduction

The last few years have witnessed an effort in rendering the descriptor extraction process automatic, in order to 1) improve (current) features efficiency; 2) create new features in an easier and faster way; and 3) allow non-signal processing experts to create sound descriptors. In this scenario, the Extractor Discovery System (EDS) [Pachet and Roy 2007] a very promising option, addressing the automation of the whole process. It started to be developed in 2003, and has been continuously improved ever since.

In a previous work [Cabral et al. 2005] we explored the power of EDS to find harmonic descriptors (chord recognizers) in a completely automated way. That work tried to simulate the use of the system by a non-expert user. The current work extends the previous one, by evaluating EDS capacity to find good descriptors for some well-known and usual problems, namely the f0 estimation, chord recognition, and percussive sound classification whereas EDS is operated by an expert.

The experiments presented in this paper intend to provide some examples of employing EDS in real-world situations. These examples should be indicative of the possibility of automatically extracting features, revealing its strengths and weaknesses. In a simple way, we are trying to answer the following question "supposing someone wants to develop an application such as a chord recognizer, a melody extractor, an accompaniment system, or whatever other tool that needs to classify sound fragments, would it be interesting to use an automatic descriptor extraction tool like EDS instead of traditional techniques"?

We try to answer this question by presenting three case studies, which actually make part of broader systems being developed by Sergio Krakowski at the VISGRAF/IMPA group in Rio de Janeiro, Brazil, and Giordano Cabral at LIP6 and Sony CSL in Paris, both advised by Francois Pachet. More particularly, we are interested in transcribing melody, harmony, and rhythm of songs, with the purpose of building interactive systems.

The transcription of musical information usually relies on a general approach: to perform a short-term analysis on a sliding window, and track the result over time using some kind of dynamic modeling such as HMM's or GMM's [Pachet and Briot 2004]

(Figure 1). This analysis typically means the computation of a feature which is pertinent to the problem (e.g. the autocorrelation of the signal, or the pitch class profile).

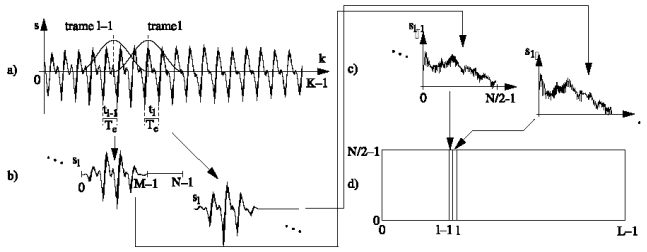


**Figure 1 – short-term analysis of a signal.**

Nevertheless, those features could hypothetically be automatically discovered by a descriptor extractor, such as EDS. That's precisely the goal of the present work, which compares the performance and development time of EDS to those of traditional approaches.

The next 3 sections respectively describe the 3 problems we are investigating, and the main features used for their analysis. Section 5 gives further detail on how EDS works. Sections 6, 7, and 8 relate the 3 case studies: f0 estimation, chord recognition, and percussive sound classification, presenting the methodology adopted in both approaches as well as the results that were found. Finally, section 9 makes some overall comments and remarks, and section 10 presents the conclusion and future work.

## 2. F0 Estimation

The problem of estimating the fundamental frequency of a sound is well known by the scientific community. Among the applications of this problem we can cite automatic melody transcription and real time accompaniment. There are many f0 estimation techniques (and even different taxonomies) available in the literature. However, one general classification is clearly observed: the distinction between approaches dealing with the information on the time domain from others based on the frequency domain.

Time-based techniques rely on the high correlation of a signal with this same signal shifted by particular values. These values are usually called *lags*, each one referring to a respective frequency. The correlation is computed for a set of candidate *lags*, and the highest correlation is taken as 1/f0 [Klapuri 2004]. One variation called AMDF [Pachet and Briot 2004] is claimed to be more robust once it redefines the correlation as the difference between the original signal and the shifted one, instead of their multiplication. Alternatively, the *cepstrum* can be used as a replacement for the correlation function [Klapuri 2004].

Frequency-based techniques analyses the signal transformed into the frequency-domain. Whenever a quasi-periodic signal have period T, it can be considered periodic with period nT. This behavior can be seen in the frequency domain by a regularity of the peaks found in the DFT. Thus, a probability function of the fundamental candidates is made by convolving the resultant DFT with comb functions aligned to the multiples of the candidate frequency. An interesting variation of this technique was suggested by [Kunieda 1996]. It is considered as of spectral-interval type and seeks for periods in the frequency domain by performing a sort of autocorrelation of the spectrum. Other variations exist, notably the ones based on the human auditory model [Cheveigné and Kawahara 1999].

In general lines, the frequency-based approach is usually more appropriate to higher register signals, since the DFT has better resolution around higher frequencies, while time-based estimators are more appropriate to lower ones, since more distant periods provide greater precision in the correlation [Klapuri 2004]. Ideally, different methods could be merged in order to achieve robustness, maintaining a good performance independently on the register.

## 3. Chord Recognition

The ability of recognizing chords is important for many applications, such as interactive musical systems, content-based musical information retrieval (finding particular examples or themes in large audio databases), and educational software. Chord recognition means the transcription of a sound into a chord, which by its turn can also be subdivided, for example in a root note and a type. Most part of the works involving harmonic content (chord recognition, chord segmentation, tonality estimation) [Sheh and Ellis 2003][ Yoshioka 2004] use the same core technique (even though slight variations may appear in the implementation): to compute an harmonic feature, such as the *Pitch Class Profile* (PCP) [Fujishima 1999], or the *chromagram* [Bartsch and Wakefield 2001], and a subsequent machine learning algorithm to find patterns for each chord class.

*PCPs* are vectors of low-level instantaneous features, representing the intensity of each pitch of the tonal scale mapped to a single octave. These vectors are calculated as follows: 1) a music recording is converted to a Fourier Transform representation (Figure 2a to Figure 2b); 2) the intensity of a pitch is calculated (Figure 2b to Figure 2d) by the magnitude of the spectral peaks, or by summing the magnitudes of all frequency bins that are located within the respective frequency band (Figure 2c); 3) The equivalent pitches from different octaves are summed, producing a vector of 12 values (eventually 24 to deal with differences in tuning and/or to gain in performance), consequentially unifying various dispositions of a single chord class (Figure 2e and Figure 2f). For example, one can expect that the intensities of the frequencies corresponding to the notes C, E and G in the spectrum of a Cmaj would be greater than the others, independently on the particular voicing of the chord. The *chromagrams* follow a different method to be computed, but are conceptually the same.

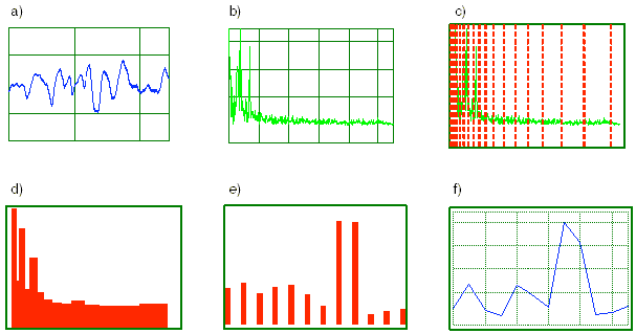



**Figure 2 - Steps to compute a PCP. The signal (a) is converted to Fast-Fourier representation (b); the FFT is divided into regions (c); the energy of each region is computed (d); these energies are folded into a 12-note vector (e); the final vector is normalized (f).**

The idea behind the use of *PCPs* for chord recognition is that the PCPs of a chord follow a pattern, and patterns can be learned from examples. Thus, machine learning techniques [Mitchell 1997] can be used to generalize a classification model from a given database of labeled examples, in order to automatically classify new ones. So, for the *PCP* of a chord, the system will respond the most probable (or closest) chord class, given the previously learned examples. The original *PCP* implementation from Fujishima used a KNN learner [Fujishima 1999], and more recent works [Gomez and Herrera 2004] successfully used other machine learning algorithms.

## 4. Percussive Sound Classification

Percussive sound classification is a vast field in which many techniques have been suggested but until now no one can be considered standard nor universal. The general problem is to automatically classify percussive sounds in order to subsequently retrieve the rhythmic structure or transcribe the rhythmic score. In that way, it can be seen as analogous to the melody extraction problem, but instead of notes, we are interested in

sounds with different timbres. The set of possible timbres can be incredibly high due to the diversity of instruments, and then solutions vary according to particular drum/percussion sets and the information one wishes to extract.

Thus, a number of different strategies have been applied to specific problems. In [Gouyon et al. 2000] the authors addressed the bass/snare drum discrimination, and eventually considered the automatic extraction of rhythmic structures. For that, they observed the zero-crossing rate estimation on the decay part of the sound. Another approach is the source separation as found in [FitzGerald 2004]. The idea is to consider each sound as made by one or more sources and, by subspace analysis, find the best sources that model the sounds. Finally we can find an extensive work done by Herrera, comparing many feature selection methods and classification techniques applied first to drum kit [Herrera et al. 2002] and afterwards to percussion instruments [Herrera et al. 2003].

## 5. EDS

The Extractor Discovery System, developed at Sony CSL, is a heuristic-based generic approach for automatically extracting high-level music descriptors from acoustic signals. EDS is based on Genetic Programming [Koza 1992], used to build extraction functions as compositions of basic mathematical and signal processing operators, such as Log, Variance, FFT, HanningWindow, etc. A specific composition of such operators is called feature (e.g. *Log (Variance (Min (FFT (Hanning (Signal)))))*), and a combination of features forms a descriptor.

Given a database of audio signals with their associated perceptive values, EDS is capable of generalizing a descriptor. Such descriptor is built by running a genetic search to find relevant signal processing features matching the description problem, and then machine learning algorithms to combine those features into a general descriptor model.

The genetic search performed by the system is intended to generate functions that may eventually be relevant to the problem. The best functions in a population are selected and iteratively transformed (by means of reproduction, i.e., constant variations, mutations, and/or cross-overs), respecting a pattern chosen by the user. The default pattern is *!_x(Signal)*, which means a function presenting any number of operations but a single value as result (for more information about EDS syntax, look at [Zils and Pachet 2004]). The populations of functions keep reproducing until no improvement is achieved, or until the user intervenes. At this point, the best functions are available to be combined. A selection can be made both manually or automatically. The final step is to choose and compute a model (linear regression, model trees, knn, locally weighted regression, neural networks, etc.) that combines all features. As an output, EDS creates an executable file, which classifies an audio file passed as argument.

In short, the user needs to 1) create the database, in which each recording is labeled with the correspondent class; 2) write one or more general patterns for the features; 3) launch the genetic search; 4) select the appropriate features; 5) choose a model to combine the features. Some of the choices taken in these steps are crucial to the process. They delimit how the user can interfere in the search for features, as explained next.

### 5.1. Pattern Choice

The pattern encapsulates the architecture of the feature. They are represented by the output of the chain of functions to be found. As single values, this output can be a frequency (*f*), amplitude (*a*), time (*t*), or any of them (*x*). Additionally, the output can be a relation, such as time and amplitude (*t:a*), or frequency and amplitude (*f:a*). Moreover, a recent improvement of the system allowed outputs to be a vector of any of the previous types. The patterns may include intermediate outputs, which will be inputs of the outer function. Along with these symbols, a *_ or a !_ is placed to express if the user wants to search a single operator or a sequence of them. At last, the patterns may include specific operators.

For example, *!_f(f:a(Signal))* means that the signal is initially converted into the frequency domain (*f:a*), then some operation is applied to get a frequency as a result (*!_f*).

### 5.2. Genetic Search

Given a set of patterns, a genetic search is launched. It means that a population of features is created, and the capacity of each one to separate the examples in the database is evaluated. The best features are then selected as seeds to a new population. This process evolves the features until no improvement is found.

Although the genetic search can be performed fully automatically, the user can supervise and interfere in the search. This intervention is even desired, since the space of possibilities is enormous, and heuristics are hard to express in most cases. Therefore, the user can lead the system through some specific paths by 1) stopping and restarting the search if it is following a bad path; 2) selecting specific features for future populations; 3) removing ineffective features from the search. Additionally, the stop condition itself is an important factor frequently left to the user.

The choice of the population size may also influence the search, since larger populations may hold a bigger variety of features (which will converge slower), whereas smaller populations will perform a more in depth (faster) search, (which will be most likely to terminate at local maxima). At last, the user can optimize features, finding the values for their arguments which maximize the class separation. For example, the split function (which divides a signal in sub-signals) has the size of the sub-signals as a parameter. Depending on the case, a tiny value can be notably better than large values, for example.

### 5.3. Feature Selection

After many features were found, possibly in different genetic searches, they can be combined to create the final descriptor (eventually with a single feature). The selection of which features to combine is left to the user, even if one useful tool is available: the *expert selection* picks up the features that are better than a customizable threshold and less correlated than another customizable threshold. In fact, as [Herrera et al. 2002] shows, choosing features in a list is as complicated as finding the features themselves, so that is maybe the point at which the quality of the result is more dependent on the user.

### 5.4. Descriptor Creation and Evaluation

Finally, in order to create the descriptor, the learning method that will combine the features must be chosen (normally KNN or GMM). The resultant descriptor is then evaluated on a test database. The results are presented class by class, along with the precision rates.

**Figure 3 – Snapshot of EDS screen givin g the results of a chord recognition descriptor classified with KNN, and evaluated on the test database.**

## 6. Case Study One: F0 Estimation

Our first experiment compares the results of EDS with those of two of the most widely used techniques for the f0 estimation, one in the time-domain, other in the frequency-domain. In order to evaluate the algorithms, we created a database of 1570 wave files, each one containing the sound of a note, ranging from A0 to C9. The wave files were rendered

from midi files using SoundFonts [Timidity 2006], where each one encapsulated one note played by one specific melodic instrument.

The two techniques examined were the autocorrelation via *AMDF* and the filtering of the spectrum at specific frequencies, both explained in section 2. The autocorrelation feature was implemented as in [Pachet and Briot 2004], with 100 candidate frequencies starting from 27.5. The correlation function was defined as the difference between samples, as shown in the formula below.

$$FP_{AMDF,l}(\tau) = \frac{1}{Norme} \sum_{k=0}^{M-1} |s_l(k) - s_l(k+\tau)|$$

**Figure 4 – formula of the AMDF autocorrelation function**

The second one is called here *FPl*, and uses a comb filter on the spectrum of the DFT of the sound (see formula in Figure 5), this filter using the same candidate frequencies as in the previous example.

$$FP_l(f_0) = \int_f \perp\!\!\!\perp_{f_0}(f) |\widetilde{s_l}(f)|\, df = \sum_i |\widetilde{s_l}(i f_0)|$$

**Figure 5 – formula of frequency-based FPl function**

The result of these functions can be interpreted as probabilities of each candidate to be the fundamental. As a further step, machine learning algorithms can be used to map the patterns of these density curves to specific pitch classes. We have implemented these 4 possibilities, called here *Pure AMDF*, *Pure FPl*, *AMDF+Knn*, and *FPl+Knn*. Machine learning algorithms could be used as well to combine more than one of these features into a single solution. We have not implemented this combination, but still calculated its upper limit, defined as follows: if one of the algorithms gives the good solution, the combined algorithm will also do it. The results are presented in the Table below:

**Table 1. Results of Traditional Techniques for F0 Estimation**

| Method | Precision |
|---|---|
| Pure AMDF | 45.10% |
| Pure FPl | 43.64% |
| AMDF+Knn | 71.51% |
| FPl+Knn | 44.18% |
| Upper Limit | 77.90% |

Noticeably the results reflect the deficiency of the techniques in specific ranges. While the correlation-based works better for low-frequencies, the frequency-based works better for high-frequencies. The learning phase corrected some misclassifications but still did not work for the frequency-based solution.

One must notice that, including the research and reading of specialized papers and books, we took 2 weeks to implement the first solution (Pure AMDF). 1 more day was necessary for the second solution, and 2 more days for the Knn versions. Altogether, the experiment took 2 weeks and 4 days (14 working days). Even if this information may not be rigorously scientific, given that other people with different background, programming skills, and dedication might perform differently, we find it useful to give an idea of the order of the time which is needed for its implementation. One must not forget that this is a part of major systems, and not a problem by itself. The developers are experienced programmers, with medium level sound processing skills, that devoted 6 hours a day to this particular problem.

The goal of the experiment is to verify if the automatic extraction can pass the 71.51% rate of correctly classified instances, hopefully approximate the 77.90%, and ideally pass the 77.90%, as well as to monitor how much time it takes.

### 6.1. Automatic Extraction of F0

The system was taken from scratch, and we performed some searches using the general patterns: "*_a(x)*", "*_f(x)*", "*_Va(x)*", and "*_Va(*_Va(t:a(x)))*", the more specific patterns: "*PitchBands(*_t:a(x), 120.0)*", "**_Va (PitchBands(*_t:a(x), 120.0)*", "**_a(Autocorrelation (*_t:a(x)))*", "**_f(Autocorrelation(*_t:a(x)))*", "**_Va (Autocorrelation(*_t:a(x)))*", "*BarkBands(x, 25.0)*", "*Chroma(x)*", "**_Va(Chroma(*_t:a(x)))*", "**_Va (SplitOverlap(Autocorrelation(x), 441, 0))*", and the very specific (optimization) patterns "*PitchBands(x, 120.0)*" and "*Correlation(x, t:a(x))*".

After 3 days of 12 genetic searches, some of them long and intensive, EDS found over 20 features superior to 70%, even though most of them were extremely correlated. On the other hand, they were found in different paths (from different patterns), needing more or less time to be found. We must mention that the correlation-based features scored very badly (19% at best, against 75% from the frequency-based), indicating some malfunctioning in the system. Until the present moment, we are not aware if this is a problem with the genetic search or an error in the operator itself.

We pre-selected 11 features for further evaluation. Many permutations of these features were tried. We hoped that some features would be complementary to the best one(s), improving the overall result. Strangely, the more features in the descriptor the worst was its performance. In fact, weaker features can bring down the quality of the descriptor depending on the method chosen to combine them, and the best descriptor in fact used only 1 feature.

```
1.  Derivation (PitchBands (x, 120.0))
2.  BarkBands (x, 120.0)
3.  Chroma (Derivation (BpFilter (x, 488.0, 26.0)))
4.  Integration (Integration (Hamming (Zcr (Split (Autocorrelation (x), 882.0)))))
5.  PitchBands (x, 120.0)
6.  Derivation (BarkBands (x, 100.0))
7.  BarkBands (x, 150.0)
8.  Chroma (Hann (BpFilter (x, 488.0, 26.0)))
9.  Triangle (SpectralRolloff (Split (Autocorrelation (x), 3307.0)))
10. BarkBands (Abs (Autocorrelation (Autocorrelation (x))), 5.0)
11. SpectralCentroid (Autocorrelation (x))
```

The Figure 6 shows the final results, comparing them to those from the traditional algorithms. It is interesting to notice that the expert selection facility actually selected just the best feature, based on a minimum quality and a maximum correlation among all the features found. The best descriptor scored 75.08%, using a frequency-based approach. The experiment lasted 4 _ days.

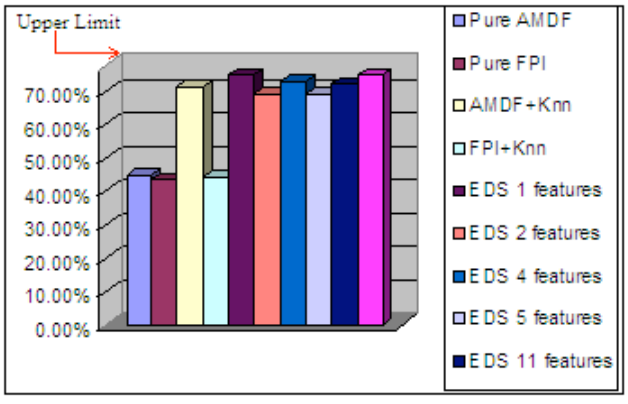



**Figure 6 – Results of the F0 Estimators.**

## 7. Case Study Two: Chord Recognition

The final goal of our chord recognizer is to create a guitar accompanier in Brazilian "bossa nova" style. Consequently, our database has examples of chords played with nylon guitar. The data was taken from D'accord Guitar Chord Database [Cabral et al. 2001], a guitar midi based chord dataset. The purpose of using it was the richness of its symbolic information (chord root, type, set of notes, position, fingers, etc.), which was very useful for labeling the data and validating the results. Each midi chord was rendered into a wav file using SoundFonts [Timidity 2006] and a free nylon guitar patch. The EDS database was created according to the information found in D'accord Guitar database. The database divided the chords into 60 classes, 5 types per root note: major, minor, seventh, minor seventh and diminished. From 1885 samples, 80% was settled on as the training dataset and 20% as the testing dataset. We implemented the traditional KNN over Pitch Class Profile algorithm in order to make a comparison. More details about it can be found at [Cabral et al. 2005]. Considering research and implementation, we took almost 4 weeks to implement it.

### 7.1. Automatic Chord Recognizer

The same databases were loaded in EDS. In our work from 2005, cited above, we found some middling features after having run the system in a fully automated way. Our strategy here is to merge new specialized features with the previous ones. In order to find new and better features, we used specific patterns, appropriate to the problem, mainly: "*_Va(*_t:a(x))", "chroma(x)", "*_Va(PitchBands (*_t:a(x), 120)", "*_Va(BarkBands(*_t:a(x), 120))" and insistently the pattern: "*_Va(chroma (*_t:a(x)))". Grosso modo, we intended to find features which firstly transformed the signal into meaningful information, like the *chroma* and the *pitchbands* do. The *chroma* EDS operator must not be confounded with the *chromagram* or the *PCP* features. These are ready-to-use features, more or less complex, comprehending many processing tasks, including pre and post-processing, while the *chroma* simply folds each bin from the DFT into a single octave, storing the average value for each note.

The search was launched over 40 times, but as well as for the F0 Estimation case, the majority of searches converged to similar results. Notably, the chroma-based features surpassed their concurrent in part due to their adequacy to the problem, in part due to the fact that it does not have extra (internal) parameters, such as the PitchBands or the BarkBands does[1]. In fact, an internal variable can lead the feature to poor results, even if they are potentially good, renouncing their persistent evolution.

Finally, we selected 14 features (11 from the previous work plus 3 just discovered). The 3 new features are:

```
1. Derivation (Power (Chroma (Blackman (x)), -0.3))
2. Log10 (Chroma (Hamming (x)))
3. Hann (PitchBands (x, 120.0))
```

As in the previous experiment, combining many features did not work better than using a single one. However, the single best one worked significatively better than the traditional solution (72.68% against 63.93%), as illustrated in Figure 7. The time needed to finish the experiment was 11 days.

[1] Both the pitchbands and the barkbands have the number of bands as parameter.

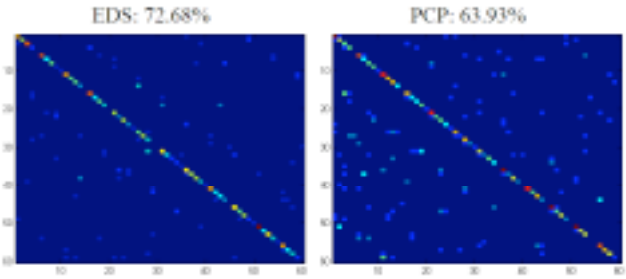



**Figure 7 – confusion matrix and precision rate of EDS (in the left) and the traditional PCP method (in the right).**

## 8. Case Study Three: Percussive Sound Classification

The *pandeiro*, a Brazilian variant of the tambourine, is a very important instrument in the musical tradition of the country. Our aim is to automatically classify the different types of strokes played on the *pandeiro* in order to build a reactive system. It is possible to distinguish among three main categories of sounds: low strokes, slap strokes and jingles strokes, although they are not completely mutually exclusive (the jingles are usually played along with the other ones). We have not found any specific work about this instrument in the scientific literature. However, the strong parallel between this instrument and a drum kit conducts us to adapt algorithms initially conceived for the last one. The low strokes can be related to the kick drum due to its low frequency content, around 100 to 250Hz, whereas the slap strokes have the same strong loudness attack and fast decay characteristics of the snare drum. Also, the jingles strokes have its spectral information located at higher frequencies (around 10 to 15 KHz) similar to the hi-hat.

The examples in our database are divided in six classes: two types of low sounds, named '*tung*' and '*ting*'; two types of slap sounds, named '*pa*' and '*grand pa*'; and two types of jingle sounds, named '*tchi*' and the '*tr*'. Noticeably, it is much harder to distinguish between classes in each pair than among the 3 pairs. We recorded several minutes of *pandeiro* solo containing all the six types of sounds, using different microphones and locations in order to preserve some inherent analysis difficulties, such as variations on the room reverberation and different frequency responses of the equipment. We built our database by automatically segmenting these recordings via a peak detector through the derivative of the convolved loudness curve of the signal, as described in [Pachet and Briot 2004]. The database was split into a training part, with 155 sound samples, and a testing one, with 288.

In this case study, we used EDS not only to automatically find a descriptor, but to assess the traditional solutions as well. In fact, a big part of the traditional features are equivalent to EDS built-in operators, such as the zero-crossing rate (ZCR) proposed by Gouyon [2000]. Thus, EDS showed to be useful as a try-and-test tool, allowing the user to instantaneously evaluate these features. Our first experiment was to evaluate if the ZCR, which Gouyon demonstrated to well discriminate between the kick and the snare drum, is suitable for the pandeiro. The unconvincing result of 47.0% is shown in detail in Table 2.

Our second experiment used the fact that each class has most of its spectral information located in a different region. The sum of band-pass filters, the spectral centroid, or any operator that divides the spectrum in sub-bands seem to be appropriate to capture this aspect. The result of dividing the spectrum in 20 Bark bands was considerably better (80.9%), and is also presented in Table 2.

These experiments spent just a couple of minutes to be done. After some manually created features, we did a first attempt to automatically generate features, by launching the genetic search with the following patterns: "!_a (x)", "!_Va (x)", "!_Va (SplitOverlap (x, 220.0, 0.1))", "!_Va (BarkBands (x, 20.0))", and "!_a (BarkBands (x, 20.0))". This search was stopped after 3 populations of 50 features, which took about 30 minutes to be calculated. We selected the 11 features listed in the next page, resulting in 87.2% of correctly classified instances, as shown in Table 2 under the designation EDS1.

```
SpectralDecrease(SplitOverlap(x,220.0, 0.1))
Rms (SplitOverlap (x,220.0,0.1))
RHF (SplitOverlap (x,220.0,0.1))
Mean (SplitOverlap (x,220.0,0.1))
SpectralSpread(SplitOverlap (x,220.0,0.1))
SpectralDecrease(SplitOverlap(x,220.0,0.3))
Rms (SplitOverlap(x,220.0,0.3))
Chroma (x)
SpectralDecrease (x)
HFC(Mean(SplitOverlap(x,220.0,0.1)))
Range(SplitOverlap(x,220.0,0.1))
```

Finally, we left the genetic search run for twelve hours, reaching 618 populations. Among the great number of features found, we selected the features scoring more than 80% and less correlated than 50%, as listed below. The final rate was 89.2%, showing that the system can converge very quickly, but may take a long time to make slight improvements. The detailed results are also presented in Table 2, under the label EDS2.

```
Rms(SplitOverlap(x,220.0,0.1))
SpectralSpread(SplitOverlap(x,220.0,0.1))
SpectralDecrease(SplitOverlap(x,220.0,0.1))
SpectralKurtosis(SplitOverlap(x,220.0,0.1))
Variance(SplitOverlap(x,220.0,0.1))
Mean(SplitOverlap(x,220.0,0.1))
Sqrt(Sqrt(PitchBands(x,5.0)))
Log10 (PitchBands(x,5.0))
Square (Mfcc0(x,10.0))
Square (Chroma (x))
Power(Hanning (BarkBands (x, 20.0)), -0.5)
```

**Table 2. Results for the Percussive Sound Classification**

| Method | Low | | Slap | | Jingle | | Overall |
|---|---|---|---|---|---|---|---|
| | Tung | Ting | Pa | Gr Pa | Tchi | Tr | |
| ZCR | 68.2% | 38.1% | 10.9% | 22.5% | 72.0% | 18.8% | 47.0% |
| SubBands | 92.2% | 42.9% | 93.5% | 72.5% | 89.2% | 81.2% | 80.9% |
| EDS1 | 84.3% | 92.9% | 84.3% | 80.0% | 91.4% | 81.2% | 87.2% |
| EDS2 | 90.2% | 61.9% | 91.3% | 95.0% | 97.8% | 87.5% | 89.2% |

Besides the bad cost-benefit relationship between performance and computational time, the data shows that the mixture of features enhanced the robustness of the solution (visibly the '*ting*' class, a kind of low stroke frequently misclassified by the traditional solutions). Additionally, it became clear that a split overlap analysis followed by a post-processing technique such as Rms or SpectralSpread overwhelmed other features, becoming predominant inside the populations. Finally, the addition of other kinds of features such as Bark bands, Chroma, Mfcc and Pitchbands apparently increased the quality of the descriptor.

## 9. Discussion

EDS revealed itself as a good feature exploration mechanism, and as so it seems especially appropriate to new or scarcely explored problems. The case study three illustrated this exploratory characteristic of EDS, as the user was able to try-and-test many possibilities, some of them fairly complicated, serving to validate ideas and/or build prototypes. The results in sections 6 and 7 pointed out that EDS can achieve satisfactory results for real world well-known problems. It exceeded or improved traditional techniques, either by optimizing existing ones or launching searches from scratch. On the other hand, the system did not find any especially innovative feature. Even when started from scratch, EDS-created features corroborated those from specialists.

For these specialists, EDS appeal relies on being: 1) an automatic tool to investigate interesting possibilities of work, which can afterward be refined by the specialist; as well as 2) a tool to optimize existing features.

We consider the feature selection methods could be improved, for example by using the procedure described in [Herrera et al. 2002], since it lacks precise control of the features and operators (ultimately the impossibility to build new operators).

At last, the quality of the final solution depends highly on the user. For instance, in a previous work on the chord recognition problem (in Section 7), we explored the same chord recognition database, while EDS was operated by a naïve user. The final descriptor scored 40.31%, in contrast to the 72.68% achieved in our current study (see Figure 7).

Even with these disadvantages, we were quite satisfied with the performance of the system, especially with its capacity to join the entire feature creation procedure into a single piece of software, including the database creation, the search and selection of good features, and the machine learning algorithms to combine them. A whole set of programs which would be very expensive to implement. For this work, the results of traditional approaches were obtained by re-implementing the algorithms described in the literature, hence the performance could obviously be improved by cutting-edge technicians. However, to remember the initial question, we consider these case studies realistic to mimic someone who would start to develop some of these systems.

One extra asset of EDS is the visualization of the result. The rather simple interface, showing the sound samples colored by their class, easily shows the correctly classified instances, making the evaluation/analysis of the result more intuitive and direct. For example, in the chord recognition problem we noticed that 18.94% of the errors referred to equivalent chords, like C° and A°, F#m7/C# and A6/C#, or C/G and Am7/G. Moreover, the system grouped together some chords (11.57% of the errors) with the same function, such

as F7 and B7(#11), or Em7 and A4/7. The system even corrected some faults in the database, placing chords like C/Bb in the right class (C7 instead of Cmaj). This situation represented 6.31% of the errors. In the end, the actual error rate would be of 12.47% (instead of the 27.32% previously mentioned), from which only 0.79% refers to flagrant errors.

## 10. Conclusion and Future Work

This paper presented 3 case studies illustrating the use of an automatic extraction tool called EDS. The work intended to validate the effectiveness of the tool in creating descriptors sufficiently good to be used in real-world applications. The case studies referred to the f0 estimation, the chord recognition, and the percussive sound classification problems. The results showed that the system was able to achieve and even surpass traditional descriptors.

These descriptors were needed for two systems currently being developed (an accompaniment system and an interactive rhythmic tool), and we found it very useful to share our experience with the community. We commented the usage of the system, and suggested some desirable improvements, which can be envisaged as future work.

## 11. Acknowledgments

We would like to thank the whole music team at Sony CSL for the support and encouragement and SBCM'07 reviewers for their comments.

## References

Bartsch, M. A. and Wakefield, G. H. (2001) To Catch a Chorus: Using Chroma-based Representation for Audio Thumbnailing, *Proceedings of International Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, USA.

Cabral, G., Zanforlin, I., Santana, H., Lima, R., and Ramalho, G. (2001) D'accord Guitar: An Innovative Guitar Performance System, *Proceedings of Journées d'Informatique Musicale (JIM'01)*, Bourges, France.

Cabral, G., Pachet, F., and Briot, J.-P. (2005) Automatic x Traditional Descriptor Extraction: The Case of Chord Recognition, *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'2005)*, London, U.K., September.

Cheveigné, A. de, and Kawahara, H. (1999) Multiple Period Estimation and Pitch Perception Model, *Speech Communication*, 27:175–185.

Fitzgerald, D. (2004) *Automatic Drum Transcription and Source Separation*, PhD Thesis, Conservatory of Music and Drama, Dublin Institute of Technology.

Fujishima, T. (1999) Real-time Chord Recognition of Musical Sound: a System using Common Lisp Music, *Proceedings of International Computer Music Conference (ICMC'99)*, Beijing, China.

Gómez, E. and Herrera, P. (2004) Estimating the Tonality of Polyphonic Audio Files: Cognitive versus Machine Learning Modeling Strategies, *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain.

Gouyon, F., Pachet, F., and Delerue, O. (2000) On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds, *3rd Digital Audio Effect Conference (DAFX'00)*, Verona, Italy.

Herrera, P., Yeterian, A., and Gouyon, F. (2002) Automatic Classification of Drum Sounds: a Comparison of Feature Selection Methods and Classification Techniques, *Proceedings of the 2nd International Conference on Music and Artificial Intelligence (ICMAI'02)*, Edimburgh, U.K.

Herrera, P., Dahamel, A., and Gouyon, F. (2003) Automatic Labeling of Unpitched Percussive Sounds, *Proceedings of the Convention of the Audio Engineering Society (AES'04)*, Banff, Canada.

Klapuri, A. (2004) *Signal Processing Methods for the Automatic Transcription of Music*, Doctoral Dissertation, Tampere University of Technology, Tampere, Finland.

Koza, J. R. (1992) *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, The MIT Press.

Kunieda, N., Shimamura, T., and Suzuki, J. (1996) Robust Method of Measurement of Fundamental Frequency by ACLOS – Autocorrelation of Log Spectrum, *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Mitchell, T. (1997) Machine Learning, McGraw-Hill.

Pachet, F and Briot, J.-P. (2004) *Informatique Musicale : du Signal au Signe Musical*, Hermès/Lavoisier.

Pachet, F. and Roy, P. (2007) Exploring Billions of Audio Features, *Proceedings of 5th International Workshop on Content-Based Multimedia Indexing (CBMI'07)*, Bordeaux, France, June.

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 (ISMIR'03)*, Baltimore, USA.

Timidity (2006) Website: http://timidity.sourceforge.net/

Yoshioka, T., Kitahara, T., Komatani, K., Ogata, T. and Okuno, H. (2004) Automatic Chord Transcription with Concurrent Recognition of Chord Symbols and Boundaries, *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain.

Zils, A. and Pachet, F. (2004) Automatic Extraction of Music Descriptors from Acoustic Signals using EDS, *Proceedings of the 116th Convention of the Audio Engineering Society (AES'04)*, Berlin, Germany, May.

# Comparing audio descriptors for singing voice detection in music audio files

**Martín Rocamora**[1]*, **Perfecto Herrera**[2]

[1]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

[2]Music Technology Group – Universitat Pompeu Fabra
Pg. Circumval·lació 8 – 08003 Barcelona, Spain

rocamora@fing.edu.uy, pherrera@iua.upf.edu

***Abstract.*** *Given the relevance of the singing voice in popular western music, a system able to reliable identify those portions of a music audio file containing vocals would be very useful. In this work, we explore already used descriptors to perform this task and compare the performance of a statistical classifier using each kind of them, concluding that MFCC are the most appropriate. As an outcome of our study, an effective statistical classification system with a reduced set of descriptors for singing voice detection in music audio files is presented. The performance of the system is validated using independent datasets of popular music for training, validation and testing, reaching a classification performance of 78.5% on the testing set.*

## 1. Introduction

One of the most memorable and representative features of a song in popular western music is the singing voice melody. For this reason, a lot of research being carried out on Music Information Retrieval deals with it (singer identification, singing voice separation, singing voice melody transcription, query by humming, lyrics transcription, etc). This kind of research would benefit from a reliable segmentation of a song into singing voice fragments. Furthermore, singing voice detection has many other applications in audio processing, such as to generate "lyrics-centred" summaries of songs, or to apply a certain singing voice filter (e.g. de-esser, de-breather) only to the automatically identified vocal fragments of a music audio file.

In this paper we address the problem of processing a music audio file and segmenting it into fragments containing singing (with or without musical accompaniment) and purely instrumental (non-singing) content, referred as vocal and non-vocal respectively. Contextual information must be considered when partitioning to ignore pauses (short non-vocal fragments) during a singing melody or to correct classification errors. One of the most troublesome characteristics of the problem is the variability of music, both with regards to the singing performance and to the accompaniment. In order to limit the scope of the problem, in this work we focus on popular music (in particular music genres such as rock, pop, folk, funk and jazz).

To address the singing voice detection problem we can take advantage of the knowledge on research fields such as musical instruments classification [Martin, 1999] [Herrera et al., 2006] and speech processing [Rabiner and Schafer, 1978] [Chilton, 1999]. The former studies our ability to distinguish different musical instruments. Singing voice detection can be considered a particular case of musical instruments classification in complex mixtures, so many features used in this field may be useful for characterizing vocal and non-vocal portions of a song. Given the similarities between speech and singing voice it is reasonable to apply techniques and descriptors used to segment and recognize speech to singing voice problems. Speech/music discrimination, singing voice separation and singer identification are closely related problems, as many systems developed to perform these tasks try to identify those fragments of the audio file containing vocals.

Although singing voices resembles speech to a certain extent there are significant differences between them that need to be taken into account. To sing a melody line with lyrics it is usually necessary to stretch the voiced sounds and shrink the unvoiced sounds to match notes durations.[1] For this reason, singing voice is more than 90% voiced, where as speech is only aproximately 60% voiced [Cook, 1990]. The majority of the singing voice energy falls between 200 Hz and 2000 Hz, but in speech the unvoiced sounds

*Supported by Comisión Sectorial de Investigación Científica, UdelaR.

[1]Vocal sounds are usually divided by speech researchers in voiced and unvoiced. The vibration of the vocal folds produces quasi-periodic sound referred to as voiced. Those vocal sounds generated by the turbulence of air against the lips or tongue (such as the consonants "s" or "f") are known as unvoiced and its waveform appears random though with some limited spectral shaping [Chilton, 1999].

are more common and tend to raise this energy limit up to 4000 Hz. Speech has a characteristic energy modulation peak around the 4 Hz syllabic rate usually considered as an evidence of it presence in automatic speech processing [Scheirer and Slaney, 1997]. With regards to pitch, the pitch contour of the singing voice tends to be piece-wise constant with abrupt pitch changes in between, while in natural speech pitch slowly drifts down with smooth pitch changes. Besides this, speech pitch is normally between 80 to 400 Hz whereas singing has a wider pitch range that can reach 1400 Hz in a soprano singer [Li and Wang, 2005] [Gerhard, 2002]. Morover, singing voice is highly harmonic, that means that the partials of the sound are located at multiples of the fundamental frequency. Several musical instruments are also harmonic, so some partials of the singing voice are overlapped with those from the accompainment. Additionally, a known feature of operatic singing is the precense of an additional formant (resonance of the vocal tract), called the singing formant, in the frequency range of 2000 to 3000 Hz, that enables the voice to stand out from the accompainment [Sundberg, 1987]. However, the singing formant does not exist in many other types of singing such as the ones in pop or rock.

The approach proposed in this work is to build a statistical classifier trained on descriptors of accompanied singing voices and accompaniments alone. Much effort is put in the study of descriptors taking into account the great majority of those reported in previous work and comparing them in equivalent conditions. The rest of this paper is organized as follows. In the next section the different approaches proposed to address the singing voice detection problem are summarized. Section 3 describes the method we applied for the selection of descriptors and the classification approach. Experimental results are presented in section 4. The paper ends with a discussion on the present work and the main conclusions.

## 2. Previous work

The common procedure followed for partitioning a song into vocal and non-vocal portions is to extract feature paremeters from audio signal frames (near stationary block of samples) at each few tens milliseconds, and then to classify them to one of each class using a threshold method or a statistical classifier.

Threshold methods tend to be simple but need descriptors that clearly discriminate between classes in order to be successful. The methods proposed to classify audio frames into vocal and non-vocal apply a threshold on only one descriptor [Maddage et al., 2004] [Shenoy et al., 2005] or compute different descriptors and apply a set of thresholds on them [Zhang, 2002]. On the other hand, statistical classifiers are trained using accompanied singing voices and pure instrumentals and can learn complex boundaries between classes combining several descriptors. This has the drawbacks of a certain amount of time spent on training and the difficulty of obtaining ground truth annotations. In the singing voice detection problem, finding the exact boundaries over the entire song can be time-consuming, and sometimes could be difficult in case of slow decays or masking. Moreover, special attention has to be paid to ensure generalization beyond the training set avoiding overfitting. Several statistical classifiers have been explored to address the problem of singing voice detection, such as Hidden Markov Models (HMM) [Berenzweig and Ellis, 2001] [New et al., 2004], Gaussian Mixture Models (GMM) [Tsai and Wang, 2006] [Li and Wang, 2007], Artificial Neural Networks (ANN) [Berenzweig et al., 2002] [Tzanetakis, 2004] and Support Vector Machines (SVM) [Maddage et al., 2003] [Maddage et al., 2004].

The short-term classification of each signal frame considers only local information so it is prone to errors, and the classification obtained is typically noisy changing from one class to the other. For this reason, usually long-term information is introduced, by smoothing the classification [Tsai and Wang, 2006] or by partitioning the song into segments (much longer than frames) and assigning the same class to the whole segment. This partitioning of the song is performed based on tempo [New et al., 2004] [Maddage et al., 2003], chord change [Maddage et al., 2004] or spectral (timbre) change [Li and Wang, 2007]. More global temporal aspects of a song are also taken into account by modeling the song structure (intro, verse, chorus, etc) by means of a HMM [New et al., 2004].

With regards to descriptors, research on musical instruments classification has demonstrated the importance of temporal and spectral features [Martin, 1999] [Herrera et al., 2006], and the speech processing field has contributed on well known techniques (such as Linear Prediction) to compute voice signal attributes [Rabiner and Schafer, 1978] [Chilton, 1999]. A broad group of descriptors has been used for the purpose of singing voice detection. Singing voice carries the main melody and the lyrics of a popular song, so it is is usually one of the most salient instruments of the mixture. Therefore, vocal frames can be identified by an energy increase of the signal, and energy or power descriptors are often used [Zhang, 2002] [Tzanetakis, 2004] [New et al., 2004] [Shenoy et al., 2005] [Maddage et al., 2003]. The timbre of different instruments is partially dictated by its spectral characteristics and when new sounds enters a mixture they usually introduce significant spectral changes. Among the descriptors computed to represent this are Mel Frequency Cepstral Coefficients (MFCC) [Tsai and Wang, 2006] [Li and Wang, 2007]

[Maddage et al., 2003], Linear Prediction Coefficients (LPC) (warped LPC or perceptually derived LPC) [Berenzweig and Ellis, 2001] [Berenzweig et al., 2002] [Kim and Whitman, 2002] [Maddage et al., 2003], Log Frequency Power Coefficients (LFPC) [New et al., 2004], and spectral Flux, Centroid and Roll-Off [Zhang, 2002] [Tzanetakis, 2004]. The delta coefficients of the previous features or variances of them are also used to capture temporal information [Berenzweig et al., 2002]. As stated previously singing voice is highly harmonic, thus the harmonicity of the signal, usually computed as an Harmonic Coefficient (HC), is used as a clue for singing voice detection [Chou and Gu, 2001] [Zhang, 2002] [Kim and Whitman, 2002].

Regarding harmonicity a pre-processing technique was proposed that consist in filtering the signal by an inverse comb filter to attenuate the harmonic sounds in the mixture [New et al., 2004] [Shenoy et al., 2005]. Harmonic accompaniment instruments have a very regular harmonic structure and can be partially removed by this filtering. Although being harmonic, singing voice has some features (vibrato, intonation) that deviate the frequency of partials from perfectly harmonic and cause it to remain after filtering.

The following is a summary of the most relevant research work on singing voice detection in chronological order. To our knowledge, the first work that focused and described the problem of locating the singing voice segments in music signals was [Berenzweig and Ellis, 2001]. Posterior probability features and their statistics are derived from an ANN trained on a phone classes database to work with speech. A HMM with two states is used to discriminate singing from accompaniment. Close in time, an approach for singing detection in speech/music discrimination is described in [Chou and Gu, 2001]. It employs a set of features that comprises MFCC as well as HC, 4Hz modulation and energy based features to train a GMM.

Following this early work, new proposals for singing detection were suggested. Artist classification is improved in [Berenzweig et al., 2002] by using only voice segments detected with a multi-layer perceptron that is fed with Perceptual LPC (PLPC), plus deltas and double deltas. An harmonic sound detector is presented in [Kim and Whitman, 2002] to identify vocal regions of an audio signal for singer identification. It works under the hypothesis that most harmonic sounds correspond to regions of singing. By means of an inverse comb filter bank, the signal is attenuated and the Harmonicity is computed as the ratio between the total energy in a frame over the energy of the most attenuated signal. The automatic singer identification system described in [Zhang, 2002] identifies the starting point of the singing voice in a song using energy features, zero-crossing rate (ZCR), HC and Spectral Flux and classifying with a set of thresholds. In [Maddage et al., 2003] a study which aims to establish if there are significant statistical differences between vocal music, instrumental music and mixed vocal and instruments is described. Descriptors set contains LPC, LPC derived Cepstrum, MFCC, Spectral Power, Short Time Energy and ZCR. Classification performance of a SVM is shown to be superior to an ANN and a GMM.

Subsequent research explored some other descriptors and classification approaches. A technique for singing voice detection is proposed in [Maddage et al., 2004]. Musical signals are segmented into beat-length frames using a rhythm extraction algorithm, the Fast Fourier Transform (FFT) of each frame is calculated, and after filtering to a narrow bandwith containing mostly voice, another FFT is applied (Twice Iterated Composite Fourier Transform, TICFT). Based on a threshold on the energy of the TICFT, singing voice frames are separated from instrumental frames. Performance is improved by some frame-correction rules based on chord pattern changes. In [Tzanetakis, 2004] singing voice detection is performed by a bootstrapping process that consists in manually annotating a few random fragments of the song being processed to train a classifier. The feature set includes Mean Relative Energy, and Mean and Standard Deviation of Spectral Centroid, Roll-off, Flux and Pitch. Different classifiers are tested, being Logistic Regression and ANN those which performed best. In the work described in [New et al., 2004] , based on the observation that a more regular harmonic structure is present in non-vocal sections as compared to vocal sections, a key estimation is performed to attenuate the harmonics of the spectrum and LFPC are computed. The energy distribution of the LFPC shows that vocal segments have relatively higher energy values in the higher frequency bands. Classification is done with a multi-model HMM that models the different sections of a typical song structure (intro, verse, chorus, bridge, outro). A classification refinement is provided by a verification step based on classification confidence and an automatic bootstrapping process (similar to that proposed in [Tzanetakis, 2004]). The work in [Shenoy et al., 2005] also addresses the singing voice segmentation problem by estimating the key of the song in order to perform an inverse comb filtering to attenuate the harmonic sounds. Singing voice is retained after filtering due to vibrato and intonation. The energy in different frequency sub-bands is computed and the highest energy frames are classified as vocal.

Most recent works use MFCC as feature vectors and GMM as classifiers. A vocal/non-vocal detection algorithm is proposed in [Tsai and Wang, 2006] applied to singer recognition. The class of each frame is hypothesized according to log-likelihoods and the final decision is made at homogeneous segments. In [Li and Wang, 2007], the problem of singing voice separation from music accompaniment is addressed and a singing voice detection procedure is used. The audio is partitioned by detecting large spectral changes, and segments are classified according to the log-likelihoods of all the frames of a portion.

# 3. Method

## 3.1. Databases

Independent datasets of popular music recordings were used for training, validation and testing. Audio of all datasets is stereo at a sampling rate ($f_s$) of 44.1 kHz. The training database was build extracting short audio excerpts from music recordings and manually classifying them into vocal and non-vocal. Three different excerpt-length sets were constructed of 0.5, 1 and 3 seconds length. Each set contains 500 instances of each class. Music utilized belongs to a music genres database comprising alternative, blues, classical, country, electronic, folk, funk, heavy-metal, hip-hop, jazz, pop, religious, rock and soul. In addition, approximately 25% of pure instrumental and a capella music was also added. The validation database consists of 63 fragments of 10 seconds that were manually annotated. Music was extracted from Magnatune[2] recordings belonging to similar genres as the ones used for training. Once the features and the classifier were set and its parameters finely tuned, an independent evaluation was conducted on a testing database of 46 manually annotated songs, for a total duration of 3 hours. Music in this set comprises 7 different singers performing in genres that were used for training and validation.

## 3.2. Descriptors implemented

Based on the existing literature the following timbre descriptors were implemented: Mel Frequency Cepstral Coefficients (MFCC), Perceptually derived LPC (PLPC), Log Frequency Power Coefficients (LFPC) and Harmonic Coefficient (HC). In addition, a general purpose musical instruments classification feature set was built, including Spectral Centroid, Roll-off, Flux, Skewness, Kurtosis and Flatness. Pitch was also included, being the only non-spectral feature reported that was considered relevant (4Hz modulation is appropriate for speech but not for singing [Chou and Gu, 2001], ZCR strongly correlates with the Spectral Centroid [Herrera et al., 2006] and other power or energy features can be regarded as variants of spectral descriptors such as LFPC).

Audio signal is processed in frames of 25 ms using a Hamming window and a hop size of 10 ms. Considering that the majority of energy in the singing voice falls between 200 Hz and 2000 Hz [Kim and Whitman, 2002], the frequency range is established in 200 Hz to 16 kHz for all spectral descriptors.

Implementation of MFCC derives 13 coefficients from 40 mel scale frequency bands for each signal frame. An FFT is applied to each signal frame and the magnitude spectrum is obtained by taking the absolute value. After that, the magnitude spectrum is processed by a filter bank whose center frequencies are spaced according to the mel scale. Then, energy on each band is computed and the logarithm is taken. The elements of these vectors are highly correlated so a Discrete Cosine Transform (DCT) is applied to finally obtain the MFCC.

Some psychoacoustic concepts are introduced in the PLP analysis technique that make it more consistent with human hearing in comparison with conventional LP analysis. PLP coefficients are obtained by a critical band integration of the signal, followed by equal loudness weighting and intensity to loudness conversion. Finally, a Linear Prediction analysis of order 12 is applied. Both MFCC and PLPC are implemented using [Ellis, 2005][3].

To derive LFPC the signal frames are passed through a bank of 12 band-pass filters spaced logarithmically, and the coefficient for each band is obtained by computing the power of the band divided over the band bandwith and expressed in decibels [New et al., 2004], as follows,

$$\mathrm{LFPC}_t(m) = 10\ \log_{10}\left|\frac{S_t(m)}{N_m}\right|, \quad S_t(m) = \sum_{k=f_{m-1}}^{f_m} X_t(k)^2, \quad m = 1, 2, \ldots, 12$$

where $t$ is the frame number, $m$ indicates the band number, $S_t(m)$ is the power of the band, $N_m$ is the number of spectral components in the band, $X_t(k)$ is the $k^{th}$ spectral component of the signal frame, and $f_m$ are the indexes of the band boundaries corresponding to frequencies spaced logarithmically from 200Hz to 16kHz as represented in figure 1.

Implementation of HC follows the procedure described in [Chou and Gu, 2001], where temporal and spectral autocorrelation of the signal frame are computed (TA and SA respectively), and the HC is obtained as the maximum of the sum of the autocorrelation functions, $\mathrm{HC}_t = \max_\tau [\ \mathrm{TA}_t(\tau) + \mathrm{SA}_t(\tau)\ ]$, where $t$ is the frame number, and $\tau$ is the temporal delay. Temporal and spectral autocorrelation functions

[2] http://magnatune.com/

[3] http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/

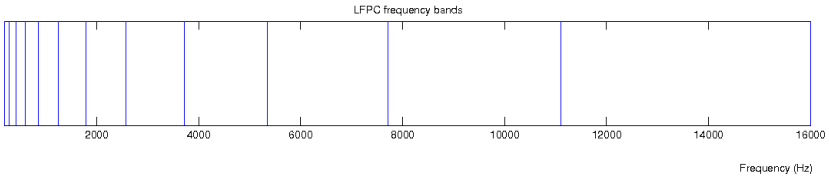



**Figure 1: Frequency bands used for LFPC computation.**

are calculated as,

$$\text{TA}(\tau) = \frac{\sum_{n=1}^{N-\tau} \bar{x}_t(n)\ \bar{x}_t(n+\tau)}{\sqrt{\sum_{n=1}^{N-\tau} \bar{x}_t^2(n) \sum_{n=1}^{N-\tau} \bar{x}_t^2(n+\tau)}},\ \text{SA}(\tau) = \frac{\sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t(k)\ \bar{X}_t(k+k_\tau)}{\sqrt{\sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t^2(k) \sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t^2(k+k_\tau)}}$$

where $x_t(n)$ is a signal frame of $N$ samples, $X_t(k)$ is the magnitude spectrum of the signal frame computed by an $M$ point FFT, $\bar{x}_t(n)$ and $\bar{X}_t(k)$ are the zero mean versions of $x_t(n)$ and $X_t(k)$, and $k_\tau = \frac{M}{\tau f_s}$ is the frequency bean index that corresponds to the time delay $\tau$.

Spectral Flux, Roll-off, Centroid, Skewness, Kurtosis and Flatness are implemented based on [Herrera et al., 2006]. The Spectral Flux (SFX) is a measure of local spectral change, and it is computed as the spectral difference of two consecutive frames as,

$$\text{SFX}_t = \sum_{k=1}^{\frac{M}{2}} \left(\hat{X}_t(k) - \hat{X}_{t-1}(k)\right)^2$$

where $\hat{X}_t(k)$ is the energy normalized magnitude spectrum of the signal frame. Spectral Roll-off is computed as the frequency index $R$ below which the majority of the spectral energy is concentrated ($\gamma = 0.85$ is used),

$$\sum_{k=1}^{R} X_t(k)^2 \leq \gamma \sum_{k=1}^{\frac{M}{2}} X_t(k)^2.$$

The rest of the spectral descriptors consider the spectrum as a distribution, which values are the frequencies and the probabilities are the normalized spectral amplitude, and compute measures of the distribution shape. The Spectral Centroid is the barycenter or center of gravity of the spectrum and is computed as,

$$\text{SC} = \frac{\sum_{k=1}^{\frac{M}{2}} k\ X_t(k)}{\sum_{k=0}^{\frac{M}{2}-1} X_t(k)}.$$

The Skewness is a measure of the asymmetry of a distribution around its mean, while the Kurtosis is a measure of the flatness of a distribution around its mean. They are computed as the normalized moments of order 3 and 4 respectively by,

$$\text{SS} = \frac{\sqrt{\frac{M}{2}} \sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^3}{\left(\sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^2\right)^{\frac{3}{2}}},\quad \text{SK} = \frac{\frac{M}{2} \sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^4}{\left(\sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^2\right)^2} - 3.$$

where $\tilde{X}_t$ is the mean value of the magnitude spectrum $X_t(k)$. The Spectral Flatness is a measure of how flat or similar to white noise the spectrum is. It is computed as the ratio of the geometric mean to the arithmetic mean of the spectrum,

$$\text{SF} = \frac{\sqrt[\frac{M}{2}]{\prod_{k=1}^{\frac{M}{2}} X_t(k)}}{\frac{1}{\frac{M}{2}} \sum_{k=1}^{\frac{M}{2}} X_t(k)}.$$

A low value indicates a tonal signal, while for a noisy signal the value is close to 1. It is typically computed for several frequency bands. We used the following four overlapped frequency bands: 200 to 1000 Hz, 800 to 2500 Hz, 2000 to 3500 Hz and 2500 to 5000 Hz.

Reliable pitch estimation in polyphonic music signals remains up to the moment a very complex open problem, so for the sake of simplicity pitch estimation is performed by applying a monophonic fundamental frequency estimation algorithm based on [A. de Cheveignè, H. Kawahara, 2002]. This has the

drawback of an unreliable estimation, noisy and prone to octave errors due to the many sounds present at the same time. The implemented algorithm uses the Difference Function (DF), a variation of the autocorrelation function that calculates the difference between the signal frame and a delayed version of it,

$$DF_t(\tau) = \sum_{n=1}^{N-\tau} \left(x_t(n) - x_t(n+\tau)\right)^2 .$$

For a periodic signal this function is zero at values of $\tau$ multiples of the signal period, while in case of quasiperiodic signal it has minimuns close to zero. The pitch of the signal is estimated as the inverse of the delay value of the first minimun.

As stated previously, the audio signal is processed in overlapped frames, so the audio features computed describe the signal at each 10 ms. To take into account descriptors information over several consecutive frames, an audio fragment is considered, of 0.5, 1 or 3 seconds. Mean, median, standard deviation, skewness and kurtosis are extracted for this fragments. Additionally, deltas and double deltas are computed for each descriptor coefficient and the same statistical measures are calculated.

### 3.3. Validation procedures

To compare descriptors, in order to select them, classification performance on the training dataset is considered. The training database is composed of audio excerpts, each of them labelled as belonging to either one class. Performance is computed as the percentage of correctly classified instances using 10-fold cross-validation (CV). Different classifiers provided by [Witten and Frank, 2005][4] were considered and, after detailed comparisons, SVM was selected (see table 3).

The validation database is used to adjust system parameters such as the fragment length to be used to process the music audio file, or the classifier options. It is also used for evaluating the inclusion of different post-processing strategies to the system. On the other hand, testing database provides an independent dataset to test the final system performance. In both cases, music audio files manually labelled must be processed. The adopted procedure divides the file into 50% overlapped fragments, descriptors for each fragment are computed at each 10 ms and statistical measures over the whole fragment are calculated. After that, each fragment is classified as vocal or non-vocal. Finally, vocal and non-vocal regions of the audio file are build considering that the class of each fragment extends from its center to half hop-size on either side (due to overlapping). Performance is computed as the percentage of time that the classification and the manual annotation coincides. It is important to note that, due to the rough discretization of the audio in fragments, even in a correctly classified case there is always a small divergence between the annotated and the computed boundaries (see figure 2).

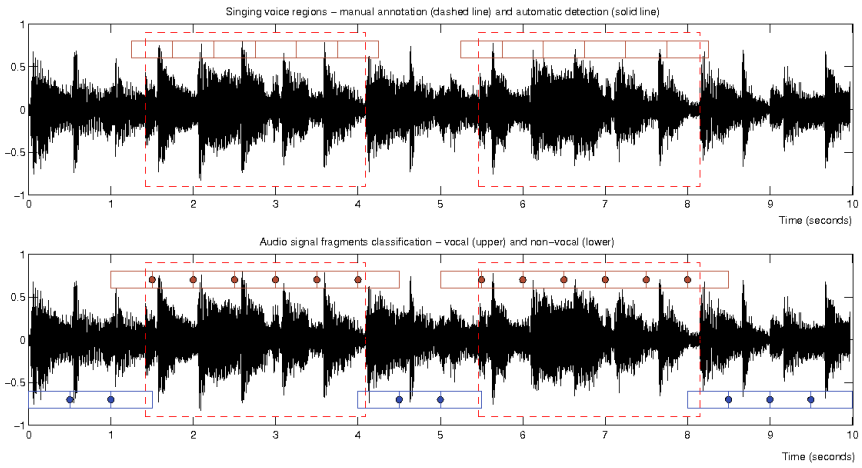



**Figure 2: Singing voice detection example. Manual annotation (dashed line) versus automatic detection (solid line) is shown at the top. Audio signal is divided into 50% overlapped fragments of 1 second length. The classification of the fragments into vocal (upper) and non-vocal (lower) is depicted at the bottom. This rough discretization of the audio in fragments produces a small divergence between the annotated and the computed boundaries of the vocal regions. Although this example could be considered a correctly classified case, performance computed as the percentage of time that the classification and the manual annotation coincides is 93.3%.**

[4] http://www.cs.waikato.ac.nz/ml/weka/

### 3.4. Selection of descriptors

As stated previously, for each descriptor coefficient (as well as deltas and double deltas), mean, median, standard deviation, skewness and kurtosis are computed. Adding irrelevant attributes to a dataset is not practical and often confuses a machine learning system [Witten and Frank, 2005]. Thus, feature selection is performed to discard less significant descriptors among each category (e.g. MFCC). For this task a correlation-based feature subset selection method provided by [Witten and Frank, 2005] is applied. However, automatic selection is not reliable when the number of features is not sufficiently small compared to the available observations, as spurious correlations between features are more likely. In those cases, a selection procedure is applied that is motivated by practical considerations concerning the inclusion of the selected features on the final system. It would be desirable to relieve the system of computing whole groups of attributes (double deltas for instance) if their contribution is not so relevant. However this is not generally ensured when appling automatic selection or dimensionality reduction techniques. For this reason, the selection firstly determines the individual classification performance of a SVM classifier for each group of statistical attributes (e.g MFCC medians). Then a procedure similar to backward elimination is applied, that is, starting with the full set and deleting a group of attributes one at a time, trying to leave out as much as possible without reducing classification performance significantly. Individual performance of each group is used as a clue for selecting those to eliminate. As a result of this feature selection, each attribute category is finally represented by a reduced set of descriptors.

### 3.5. Classification strategy

In order to select a statistical classifier for the system, different methods provided by [Witten and Frank, 2005] were applied to the training set, and 10 times 10-fold CV classification performance was compared. The best performing classification model is finally incorporated into our singing detection system. Frame length used by the system is selected by comparing the performance of the selected model trained with the different training sets (of 0.5, 1 and 3 seconds excerpt length) and applied to the validation dataset.

Some post-processing strategies were considered to improve classification performance of the system based on classification confidence and contextual information. The first one is motivated by the fact that in some cases a fragment lies over a transition between vocal and non-vocal. For this reason, if a frame has a low classification confidence (based on probability estimates for each class) it is subdivided into two new fragments and each of them is re-classified. In case of a transition each new fragment could be classified to a different class (see figure 3). Additionally, two simple context rules are proposed. If a low probability fragment is surrounded by elements of the same class re-classification is avoided. On the other hand, if one of the half size fragments produced by re-classification is surrounded by elements of the other class, it is deleted.

## 4. Results

### 4.1. Selection of descriptors

Descriptors selection results are summarized in table 1. The selected features and the total number of coefficients for each category are presented. MFCC and PLPC set includes delta coefficients, while LFPC set includes delta and double delta coefficients. The only discarded type of descriptor in the Spectral set was Flux. The complete Spectral set includes: Centroid mean and median, Roll-off mean, median and standard deviation, Skewness mean and median, Kurtosis mean and median and Flatness mean, median and standard deviation.

| Category | Performance | # Coefficients | Features selected |
|---|---|---|---|
| MFCC+D | 84.5% | 52 | median, stdev |
| LFPC+D+DD | 78.7% | 72 | median, stdev |
| Spectral | 76.0% | 21 | whole set without Flux |
| PLPC+D | 70.8% | 78 | median, stdev, skewness |
| HC | 63.6% | 2 | mean, median |
| Pitch | 59.1% | 3 | mean, stdev, kurtosis |

**Table 1: Feature selection results and 10 times 10-fold CV classification performance of a SVM on the training dataset of 1 second length audio excerpts. Performance is measured as percentage of correct decisions.**

Descriptor sets are ranked in table 1 according to the performance obtained with a SVM using 10 times 10-fold CV over the training set (note that random guess rate is 50% in this task). The results

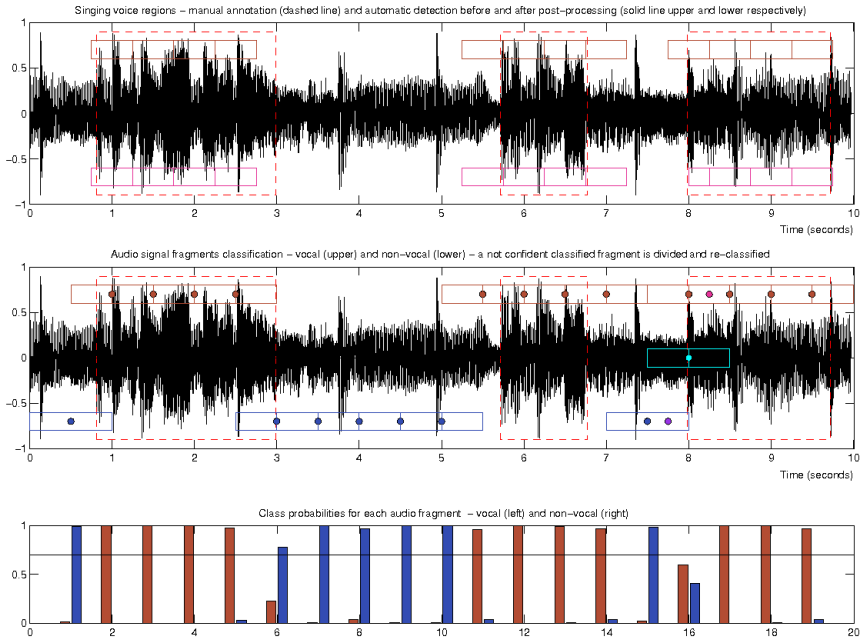



**Figure 3: Post-processing based on classification confidence. An audio frame with low probability estimates for each class is subdivided into two new fragments and each of them is re-classified. In this example, fragment centered on second 8 lies over a transition between non-vocal and vocal and its probability estimates fall below a threshold of 0.7. The fragment is divided and each new portion is correctly classified. Classification is improved after post-processing (from 84.2% to 86.5%) as it is shown in the manual anotation versus automatic detection comparison at the top.**

of paired t-tests (corrected resampled t-test [Witten and Frank, 2005]) comparing the performance of the various feature categories show that there are statistically significant differences between the MFCC set and each of the other sets (at a significance level of $p < 0.05$). The confussion matrix of the classification model based on MFCC features over the training set is presented in table 2.

| | | classified as | |
|---|---|---|---|
| | | vocal | non-vocal |
| class | vocal | 428 | 72 |
| | non-vocal | 77 | 423 |

**Table 2: Confusion matrix of the classification on the training dataset of 1 second length audio excerpts using MFCC descriptors, obtained by 10-fold CV. The rows of the matrix correspond to the ground-truth of the audio excerpt and the columns indicate the hypothesis.**

### 4.2. Classification strategy

Different classifiers were considered using the MFCC feature set on the training database comparing 10-fold CV classification performance. Table 3 shows the results obtained with a SVM trained using the Sequential Minimal Optimization method, a backpropagation ANN, a decision tree classifier implementing the well-known C4.5 algorithm and two different K-Nearest Neighbors (KNN).

Based on the classification performance on the validation set, 1 second turned out to be the most suitable fragment length. Results obtained were 73.3%, 74.2% and 73.0%, for 0.5, 1 and 3 seconds respectively. According to these results, our singing detection system was built with a SVM model using the MFCC feature set and a fragment length of 1 second.

The post-processing strategies proposed were added to the system one at a time in order to evaluate them on the validation database. Results obtained are reported in table 4.

| Classifier | SVM | ANN | KNN-3 | KNN-1 | C4.5 |
|---|---|---|---|---|---|
| Performance | 85.1% | 82.6% | 76.5% | 73.5% | 73.1% |

**Table 3: Performances of different classifiers after 10-fold CV on the training database of 1 second length audio excerpts using the MFCC set.**

| Post-processing | none | re-classify | add rule 1 | add rule 2 |
|---|---|---|---|---|
| Performance | 75.7% | 76.3% | 76.6% | 76.8% |

**Table 4: Evaluation of the post-processing strategies on the validation dataset using the MFCC descriptors. Performance is computed as the percentage of time that the classification and the manual annotation coincides.**

| Post-processing | none | all |
|---|---|---|
| Performance | 77.6% | 78.5% |

**Table 5: Performance of the system on the testing dataset with no post-processing and performing all the post-processing strategies proposed, computed as the percentage of time that the classification and the manual annotation coincides.**

Finally the system was used to process the testing set achieving a performance similar to that obtained in the validation set. Table 5 shows the results with no post-processing and considering all the post-processing strategies.

## 5. Discussion and conclusions

Analysis of the results obtained indicate that those descriptors that model the spectral content of the audio signal were the most appropriate for the problem (MFCC, LFPC, Spectral, PLPC). It is interesting that such a simple descriptor as LFPC outperformed all other features sets but MFCC, and that the general purpose Spectral outperformed PLPC. The results confirm that HC is not able to discriminate singing voice sounds from other harmonic musical instruments. The poor performance obtained with the Pitch descriptors is due to the utilization of a monophonic fundamental frequency estimation algorithm. We plan to apply other pitch estimation methods in our future work that could deal with polyphonic audio and to develop pitch descriptors that exploit singing voice pitch contour distinctive features such as intonation. Regarding MFCC, the feature selection performed points out that considering delta coefficients can boost performance (2% on the training set). However, they are generally not used when applying MFCC to this type of problem [Li and Wang, 2007] [Tsai and Wang, 2006]. Classification performance decreased significantly in validation and testing compared to 10-fold CV on the training set, which is not surprising because of the different origins and data variances of the databases. Additionally, the developed system roughly divides the audio file in fragments, so given our validation approach, we can take these results as worst-case or lower-bound estimations of the system performance.

We have studied the singing voice detection problem in music audio files by a statistical classification approach and we have compared, under equivalent conditions, the performance of several types of acoustic descriptors reported to be used for the problem. The results obtained confirm the usefulness of MFCC for this problem. As an outcome of our study, an effective singing voice detection system to process popular music audio files with a reduced set of descriptors has been developed. It is difficult to compare the performance achieved with other research work because there is no standard dataset for evaluation, but results obtained are promising and similar to the ones reported. Although our primary intention was to compare already used descriptors for this task, we have attempted to combine different descriptors as well as different classifiers. The overall classification performance was not improved so the results are not included. Also some other descriptors were tested without success. Our future work will follow this direction as it is reasonable to expect better results by combining different sources of information.

## 6. Acknowledgments

This work was carried out during an internship at the Music Technology Group (MTG) financially supported by Comisión Sectorial de Investigación Científica, UdelaR. First author is very grateful to all the people at the MTG for their support and to Dr. Alvaro Pardo for his careful reading of this article and his suggestions.

## References


A. de Cheveignè, H. Kawahara (2002). YIN, a fundamental frecuency estimator for speech and music. *Journal Acoustic Society of America*, 111:1917–1930.

Berenzweig, A. and Ellis, D. (2001). Locating singing voice segments within music signals. *Proc. IEEE Workshop on Apps. of Sig. Proc. to Acous. and Audio, Mohonk NY, October 2001*, page 4pp.

Berenzweig, A., Ellis, D., and Lawrence, S. (2002). Using voice segments to improve artist classification of music. *AES 22nd International Conference*.

Chilton, T. (1999). Speech analisys. School of Electronic and Physical Sciences, University of Surrey.

Chou, W. and Gu, L. (2001). Robust singing detection in speech/music discriminator design. *International Conference on Acoustics, Speech, and Signal Processing*.

Cook, P. R. (1990). *Identification of control parameters in an articulatory vocal tract model, with applications to the synthesis of singing*. PhD thesis, Stanford Univ., Stanford, CA.

Ellis, D. P. W. (2005). PLP and RASTA (and MFCC, and inversion) in Matlab. Online web resource: `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`.

Gerhard, D. (2002). Pitch-based acoustic feature analysis for the discrimination of speech and monofonic singing. *Journal of the Canadian Acoustical Assosiation*, pages 152–153.

Herrera, P., Klapuri, A., and Davy, M. (2006). Automatic Classification of Pitched Musical Instrument Sounds. In Klapuri, A. and Davy, M., editors, *Signal Processing Methods for Music Transcription*, pages 163–200. Springer, New York.

Kim, Y. E. and Whitman, B. P. (2002). Singer identification in popular music recordings using voice coding features. *In Proceedings of Interational Conference on Music Information Retrieval*, pages 164–169. Paris, France.

Li, Y. and Wang, D. (2007). Separation of singing voice from music accompaniment for monaural recordings. *IEEE Transactions on Audio, Speech and Language Processing*.

Li, Y. and Wang, D. L. (2005). Separation of singing voice from music accompaniment for monaural recordings. Technical Report OSU-CISRC-9/05-TR61, Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA.

Maddage, N. C., Wan, K., Xu, C., and Wang, Y. (June 2004). Singing voice detection using twice-iterated composite fourier transform. *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference*, pages Page(s):1347 – 1350 Vol.2.

Maddage, N. C., Xu, C., and Wang, Y. (2003). A svm-based classification approach to musical audio. *Proc. ISMIR*.

Martin, K. D. (1999). *Sound-Source Recognition: A Theory and Computational Model*. PhD thesis, MIT. Cambridge, MA.

New, T. L., Shenoy, A., and Wang, Y. (2004). Singing voice detection in popular music. Technical report, Department of Computer Science, University of Singapore, Singapore, October 2004.

Rabiner, L. R. and Schafer, R. W. (1978). *Digital Processing of Speech Signals*. Prentice Hall, New Jersey.

Scheirer, E. D. and Slaney, M. (1997). Construction and evaluation of a robust multifeature speech/music discriminator. *ICASSP, Munich, Germany*.

Shenoy, A., Wu, Y., and Wang, Y. (2005). Singing voice detection for karaoke application. *Visual Communications and Image Processing 2005, Proc. of SPIE*, page Vol. 5960.

Sundberg, J. (1987). *The science of the singing voice*. De Kalb, Il., Northern Illinois University Press.

Tsai, W. H. and Wang, H. M. (2006). Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signal. *IEEE Transactions on Speech and Audio Processing, January 2006.*, pages Vol. 14, No 1.

Tzanetakis, G. (2004). Song-specific bootstrapping of singing voice structure. Technical report, Department of computer science, University of Victoria.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. 2nd Edition, Morgan Kaufmann, San Francisco.

Zhang, T. (2002). System and method for automatic singer identification. HP Labs Technical Report.

# Analyzing Harmonic Progressions with HarmIn: the Music of Antônio Carlos Jobim

**Giordano Cabral[1], Robert Willey[2]**

[1]Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie

[2]School of Music
University of Louisiana at Lafayette

Giordano.Cabral@lip6.fr, willey@louisiana.edu

***Abstract.*** *This paper describes a tool designed for the analysis, retrieval, and visualization of harmonic progressions which aim to provide the user with valuable statistics and graphs. This output is intended to help the users to better understand the harmonic content of a music dataset, hopefully causing the emergence of interesting findings. The analysis of the music of Antônio Carlos Jobim motivated the creation of our tool, and for this reason is presented as our case study. We also describe how this tool can be integrated into an accompaniment system, going beyond analysis into the fields of interactive music and composition.*

## 1. Preamble

Last year, Robert Willey contacted me. He was looking for IT tools to assist in analyzing the music of Antônio Carlos ("Tom") Jobim (Willey 2005). As I was working on a bossa-nova accompaniment system, which needed exactly the kind of information he was able to provide, I became immediately interested in his project. He is an American music researcher seeking hints that might point out the way Jobim composed, especially how his chords were arranged on the piano, and ultimately how to play the piano in Jobim's style.

We then started to build a set of tools using some techniques inherited from my previous work (Cabral et al 2006 and Cabral 2005). We began by testing the tools with the large corpus containing the transcription of a set of authoritative Jobim's songbooks: the Cancioneiro Jobim (Jobim 2005). As we were not actually sure about what those tools were intended to find, we developed one functionality at a time, reevaluating the utility and abilities of the tools iteratively. In the end, the tools were found to be flexible enough to work with different corpuses.

The purpose of this paper is to present such tools in their current stage. We hope it will be useful for other researchers, from whom we look forward to getting feedback and suggestions about new features to be implemented.

## 2. Introduction

One of the first applications of computer programs to the making of music was the automatic composition by Lejaren Hiller of the *Illiac Suite* in 1957 (Hiller and Isaacson

1993). Since that time computer programs have been used to analyze the compositional style of a number of composers. What makes our tool special is the way it is adjusted to respond to specific queries. We not only developed a tool that extracts statistical information from the data, but also invented a new way of plotting harmony that facilitates the understanding of the harmonic path traced by a song, a set of songs, a style or a composer's work (in our case study, Jobim's work). Beyond that, we integrated our tool to an accompaniment system, which allowed us to musically interact with the dataset, as well as to compose new Jobim-like harmonies.

This paper describes this tool, and illustrates its applications in the study of Jobim's music. It is divided into two main sections: the first presents a description of the tool's architecture, its main functionalities, and data formats; the second lays out the experiments and the results which were obtained.

## 3. HarmIn 1.0

The ensemble of tools developed is named HarmIn. It is made of a number of features, as mentioned above. The system takes a harmonic sequence as input and provides several outputs, intended to help the discovery of the rules, patterns, and harmonic shapes enclosed in the sequence. The following subsections explain each part of the HarmIn.

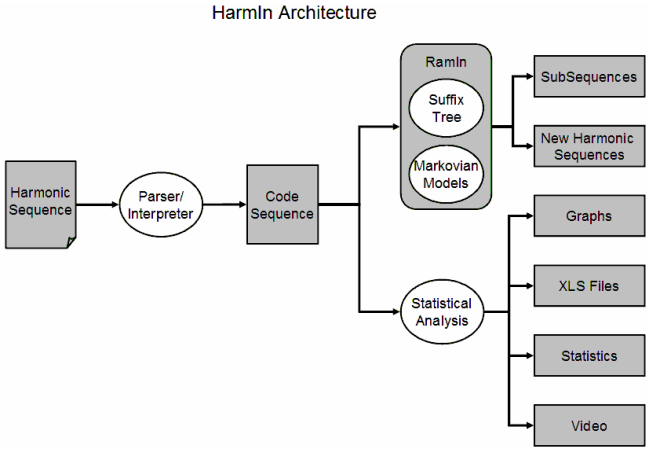



**Figure 1. HarmIn architecture.**

### 3.1 The Input: Harmonic Sequences and the Parser

The input of the system is a harmonic sequence. This harmonic sequence is a text file, containing a sequence of chord names. Symbols in the database that are not relevant for this study, such as repetition of a chord "/", or a rest "x" are ignored. The parser is responsible for reading and interpreting the input file. It was originally conceived to read files in the format defined by (Willey and Cabral 2007), but in fact it is flexible to allow other notations. The expected data format is formed by one or more songs, each one composed of a header and a body. The header contains general information (e.g. title, tempo, key, composer), and the body contains the list of chords. The header can be omitted, so a simple file with only a sequence of chords is also accepted by the system. Even tablatures may be correctly read, if the user chooses to ignore the chords not recognized by the system (so it will ignore the lyrics part). Additionally, the user can

redefine the chord names understood by the system. Normally he or she will only add to the list of known chords, but he/she can also specify not to accept certain names. This can be done in two different ways: either the user edits a text file (chords.ini), or he/she answers a question when the system finds a chord it does not recognize.

Figure 2 shows an example of a harmonic sequence input. The part going from "title" to "chord_symbol_misprint_m88" is the header. The body is enclosed by the "Page" and "Done" tags.

```
title       time        42                          Cmaj7       G7(b9)      C
Imagina     3/4         bars                        /           /           /
tempo       composer    100                         /           /           /
moderato    TomJobim&ChicoBuarque   comment         G7(b9)      C           E7(b9)
comment     arranged    chord_symbol_misprint_m88   /           /           Done
FIX_KEY     PauloJobim  Page                        /           /
key         volume      G7(b9)                      Cmaj7       G7(b13b9)
C           1           /                           /           /
major       page        /                           /           /
```

**Figure 2. Example of an input file containing a harmonic sequence.**

## 3.2 Interpreting Rules

To perform the statistical analysis, the system must have a set of interpreting rules, in order to convert the chord names into actual chord classes. This is defined by the user in the same "chords.ini" file described above. In fact, when defining an accepted chord name, the user must also indicate the chord class it refers to. Figure 3 shows an example of such a "chords.ini" file. In the case the system does not find the interpreting rule for a chord in the input harmonic sequence, the user is requested to classify it, continuously increasing the number of chords defined in the chords.ini file.

```
case {'', '7M', '7M(9)', '6', '6/9', '6(9)', '7M(#11)',
'7M(#5/9)', '7M(6/9)', 'maj7', 'maj7(9)', 'maj7(6)', '96',
'maj7(#9)', '96(#11)', '(#5)', '(omit3)', '9', '9(#11)',
'(add9)', '6(#11)', 'maj7(#119)', 'maj7(#11)', '(#119)',
'(13)', '(#9)', '(#11)', '(maj7)', '(b13)', 'maj7(96)',
'maj7(13#119)', 'maj7(#5)', 'maj7(#9#5)', '(b9)', '(13#9)',
'maj7(#5', 'maj7(b5)', 'maj7(#116)', '6(#9)', '(#9#5)',
'maj7(69)', 'add9(#5)', '(b5)', '69', '69(#11)', '9(no3)',
'(add9omit3)', 'maj7(139)', '6(11)', '(b6)'}
type = 0;

case {'7', '7(b9)', '7(9)', '7(#9)', '7(13)', '7(b13)',
'7(b9', '7(b5)', '7(#11)', '7(#5)', '7(9', '7(b9/b13)',
'7(9/#11)', '7(b9/#11)', '7(b5/b9)', '7(#119)', '7(13b9)',
'7(b13b9)', '7(#11b9)', '7(b9b5)', '7(#11#5)', '7(9#5)',
'7(b9#5)', '7(#9#5omit3)', '7(b13#9)', '7(13#9)',
'7(13#119)', '7(13#9)', '7(13#11)', '7(#11#9)', '7(139)',
'7(#9#5)', '7(b139)', '7(9#)', '7(11)', '7(#9b9)',
'7(13#11b9)', '7(omit3)', '(b13b9)', '7(13#11#9)',
'7(#9b5)', '7(11#13b9)', '(b13#9)', '7(13omit3)'}
type = 1;

case {'m', 'm6', 'm(b6)', 'min', 'm6(9)', 'm7M', 'm7M(9)',
'm(7M)', 'm(9)', 'm(maj7)', 'm96', 'm(11maj7)', 'm(add9)',
'm(9maj7)', 'm(11)', 'm9(maj7)', 'm(#5)', 'm9', 'm7',
'm7(b5)', 'm7(9)', 'm7(11)', 'm7(b5', 'm7(b5/9)', 'min7',
'm74', 'm7(119)', 'min7(9)', 'm7(6)', 'm7(9b5)', 'm6(11)',
'm7(b9)', 'm(96)', 'm7(b13)', 'm7(96)', 'm7(69)',
'm6(119)', 'm7(b6)', 'm(7)', 'm(6maj7)', 'm7(13)',
'm7(139)', 'm7(11b5)', 'm7(#5)', 'm7(9#5)', 'm(b6add9)',
'm(9b6)', 'm7(omit5)', 'm7(13119)', 'm(119maj7)',
'm(maj76)'}
type = 2;

case{'°', '°(b13)', 'dim7', 'dim', 'dim7(b13)',
'dim(maj7)', 'dim(maj7)_Fm', 'dim7(9)'}
type = 3;

case {'7/4', '4/7(9)', '4/7', '4/7(13)', '7/4(b9)',
'4/7(b9)', 'aug', '74', '74(9)', '74(b9)', '74(b13)', '4',
'(134)', '4(add9)', '74(139)', '7(sus)', '4(96)', '4(b9)',
'74(1393)', '4(b9#5)'}
type = 4;

case{'7_C', '_C', '_Cb', '_C#', '_D', '_Db', '_D#', '_E',
'_Eb', '_E#', '_F', '_Fb', '_F#', '_G', '_Gb', '_G#', '_A',
'_Ab', '_A#', '_B', '_Bb', '_B#', '_Eb(add9)',
'_Ebm(add9)', '_Eb6', '_C6', 'm_Ab6', '_F6'}
type = 5;
```

**Figure3. Examples of interpreting rules defined in the chords.ini file.**

## 3.3 RamIn and the Code Sequence

The parser/interpreter generates a file which is readable by the RamIn system. RamIn is an accompaniment system designed to find and play chords to accompany a melody input via microphone. RamIn generates the chord lines by simulating a Markovian model which encapsulates the probability distribution of chord transitions learned from an input harmonic sequence. Since our tool transforms the initial input harmonic

sequence into the RamIn input sequence, we can use all of its capabilities, which include finding the most frequent subsequences, and creating new harmonies coherent with the input harmony. These functionalities are possible thanks to the use of suffix trees, for details see (Cabral 2006). Section 4 presents some illustrative examples.

### 3.4 Statistical Analysis

The main objective of this study was to analyze the harmonic content of Jobim's songs. So, the most important aspect of the tools was to support the analysis by extracting data and statistics from the input, and constructing graphs. The output of this module of the system is simple statistical values, such as the number of chord changes, the standard deviation of the chords, or their distribution. Tabbed text files (XLS files), readable by sheet programs (such as Microsoft Excel) as well as by scientific ones (such as Matlab) are also exported, so that the user can perform his own queries and generate graphs.

### 3.5 Graphs and Videos

The system can also generate a set of graphs and videos. The graphs include a histogram of the single chords in the input harmony or the subsequences found by RamIn (Figure 4), a plot of the chord transitions as a two-dimensional matrix, where the two dimensions represent the two chords in the transitions (Figure 5), and a specially designed graph, where the chords follow the circle of fifths and are connected by lines representing the transitions (Figure 6). This graph gives a visual signature to the input harmony. This signature may refer to a single song or a group of songs from a composer or a style. It provides a new way of comparing them.

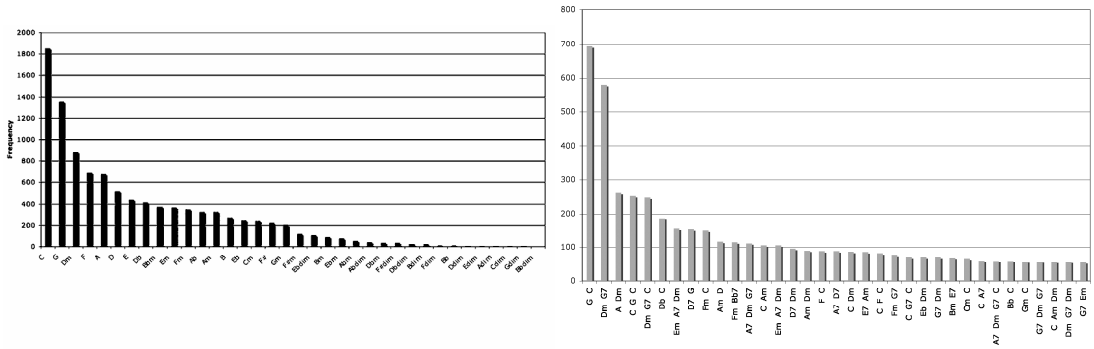
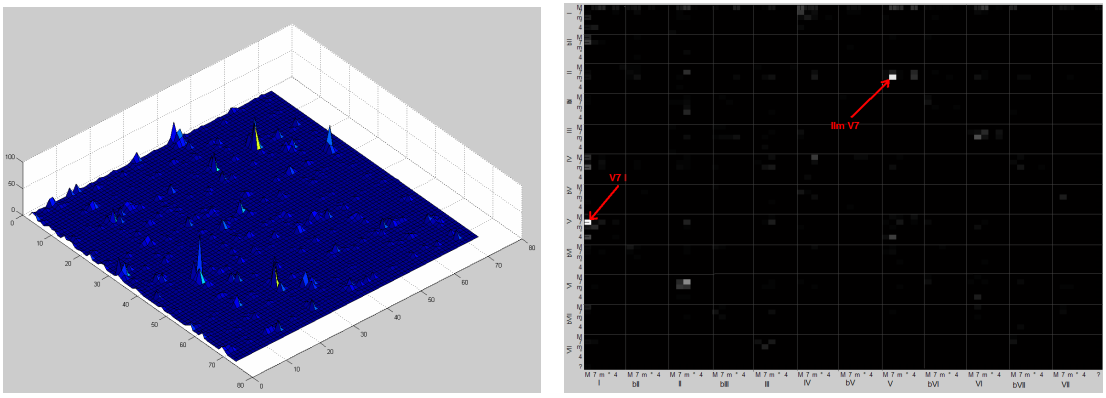



**Figure 4. Histogram of the chords (at the left) and of the subsequences of chords (at the right).**

Figure 5 shows the frequency in which the chord transitions occur. A chord transition is a change from a first chord to a second chord. The graphs represent the first chord as the row and the second chord as the column. Higher points (in Figure 5a) and brighter pixels (in Figure 5b) indicate a higher incidence of that transition. For example, the pixels representing the transitions from *ii* to *V7* and from *V7* to *I* are clearly brighter, and are annotated in figure 5b.




**Figures 5a and 5b. Distribution of chord transitions, plotted as a 3D graph (on the left) or a 2D graph (on the right).**

Figure 6 shows the same information but in a more legible way, as one can also follow the sequence of transitions. The little circles represent the chords, and the lines connecting them represent the transitions. The chord roots are displayed following the circle of fifths clockwise, so that the final graph generally becomes simpler. The quality of the chord varies according to its distance from the center, and has a different color (in the case illustrated in Figure 6, major=blue, dominant=green, minor=yellow, diminished=salmon, suspended=purple). The thickness and transparency of the lines indicate the frequency in which each chord transition occurred. Progressions that occurred less than 1% of the time were omitted to make the graph more readable. The direction of the transition is indicated by the coordinate of the line vertex inside the circle. Each line starts in the upper part of a circle and points to the lower part. For example, in Figure 6, the thick line between *Vdom* and *Imaj* indicates a frequent transition from chord *Vdom* to *Imaj*.

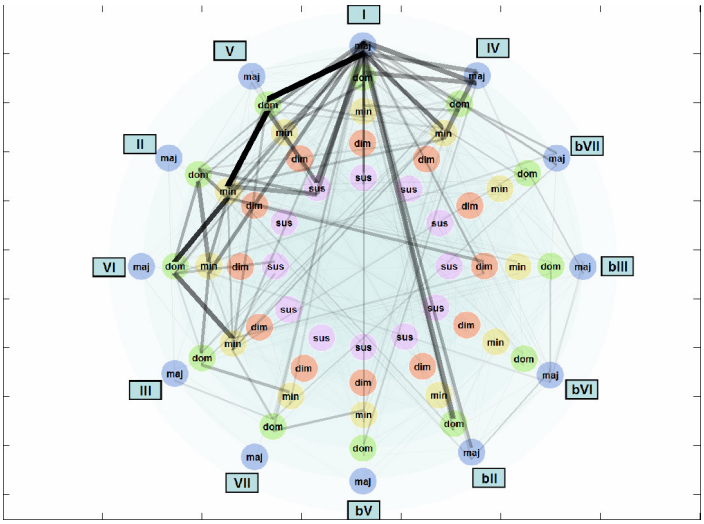



**Figure 6. HarmIn chord transition graph.**

Besides being used for music analysis, this graph can be used in composing music, since a chord sequence can be produced by following a path of chord connections. The more one chooses thicker and more opaque lines, the more the resulting sequence will be coherent with the input harmony.

Video animations can also be created showing the evolution of these graphs over the songs, the time, the key, or any other characteristic the user judges to be pertinent. Some examples of these videos are available at http://willshare.com/jobim.

## 4. Case Study: The Music of Antônio Carlos Jobim

For this study, the music of Antônio Carlos Jobim was used, initially to quantify the distribution of chords and the variety of progressions. Experiments were performed in order to determine which approaches to analysis would show the best results. Separating the analysis into groups of songs in major and songs in minor reduced the number and frequency of significant chord transition pairs, clarifying the text and graphic analysis output.

We also wished to investigate the effect of considering internal modulations in the songs or not. The data was analyzed and represented graphically, leading to observations concerning how his harmonic language changed over the course of his career, and we eventually developed graphing techniques that made the data easy to read.

### 4.1. Cancioneiro Jobim

The data for this study was entered into Microsoft Word files corresponding to the *Cancioneiro Jobim* songbook series, an authoritative five-volume set of approximately 250 songs covering Jobim's complete recorded works, written by Antônio Carlos Jobim, Paulo Jobim, and a team of arrangers under their supervision (Jobim 2001). Then, a VBA program converted it into the harmonic sequence serving as the input of the system.

The details of the data representation, as well as the process of converting the piano arrangements and accompanying chord symbols, and a preliminary comparison of Jobim's work with that of the Beatles is described in a separate paper submitted for the conference (Willey and Cabral 2007). The database can be obtained from the authors for research purposes.

### 4.2 Tonality and Modulation

All chords are considered relative to key of the song in which they occur. For example, a D major chord in a song in A major is considered to be the same as a B major chord in a song in F# major. This follows the method used in KG Johansson's study of the recorded works of the Beatles (Johansson 1999) of transposing all the songs to the key of C major or minor before beginning the analysis. This way, the data from songs in different keys can be combined.

Modulations in the music are indicated in the database embedded amid the chord data. The analysis can either include or ignore them. Figure 7 shows an example of this. If modulations are considered, the *Eb*, *Ab* and *Db7* would be respectively a *I*, a *IV*, and a *bVII7*. Otherwise, this sequence would be seen as a *bIII*, a *bVI*, and a *bII7*. However, if modulations are considered, the transitions from *C* to *Bb7*, and from *Bb7* to *Eb* would be respectively [*I*→*bVII7*] and [*V7*→*I*]. This means that the same chord (*Bb7*) would be considered differently in the two consecutive transitions (as a *bVII7* in the first, as a *V7* in the second). The current implementation ignores the modulation, in order to maintain the accurate representation of the input chord sequences.

```
                Key: C         Eb                    C
              Chord: C F  G C Bb7   Eb   Ab  Db7   C F  G
  With Modulation: I IV V I bVII7 I    IV  bVII7 I IV V
Ignoring Modulation: I IV V I bVII7 bIII bVI bII7  I IV V
```

**Figure 7. How tonality and modulation may be considered in HarmIn.**

### 4.3 Chords

One of the innovations of the bossa nova was its inclusion of extended and altered chord tones, increasing the variety of possible chord types. For example, there are twenty three varieties of minor chords present due to permutations of added *6*, *9*, *11*, and/or *13* and their alterations. In the analysis for the study these extensions were ignored, and the chords were grouped into five types: major, minor, 7 (dominant), diminished, and suspended. The reduction of music to a series of chord symbols acts as a filter to the complexity of compositions, creating a generalization that allows the statistical analysis to find significant trends. If every possible pitch combination were allowed as input there would be so many possibilities that the analysis would not be useful. Slash chords, such as *F#7/D* which did not resolve to simple inversions, unusual polychords, such as an $E^b$ triad over a $C\#m^7$ chord (written as one chord symbol separated from another by a horizontal line), or any other unrecognized type were left out of the analysis.

The most common in songs in major keys, in order of decreasing frequency, were found to be *I, IIm, V7, VI7, IVm, IV, II7, IIIm, VIm, Vsus, III7, bVI, bII, bVII7, bIII, bIII7, VIsus, III, and VI*. An analysis of the transitions from one chord to another explained the presence of non-diatonic chords such as *bII (subV7), bVII7* (*V7* of *bIII*), and *bVI* (*subV7* of *V*). Yet, the order holds some surprises, specially the *IVm* befor the *IV*, and the low positions of the *III* and *VI*.

### 4.4 Chord progressions

The focus of the statistical analysis was to analyze the transitions between pairs of chords, and to see how the chord progressions changed over the course of the five periods in Jobim's compositional life. The arrangement of chords in songs in major keys display a preponderance of root movement by fourths and fifths, as seen in Figure 8a by the number of short lines connecting chords on adjacent spokes (i.e. between chords in the *I* or *IV* groups). The most dramatic exception to this appears to be the distant move from *bII* to *I* nearly opposite on the circle. This is misleading for the eye, however, since *bII* is actually just a substitute for *V,* the neighbor of *I*.

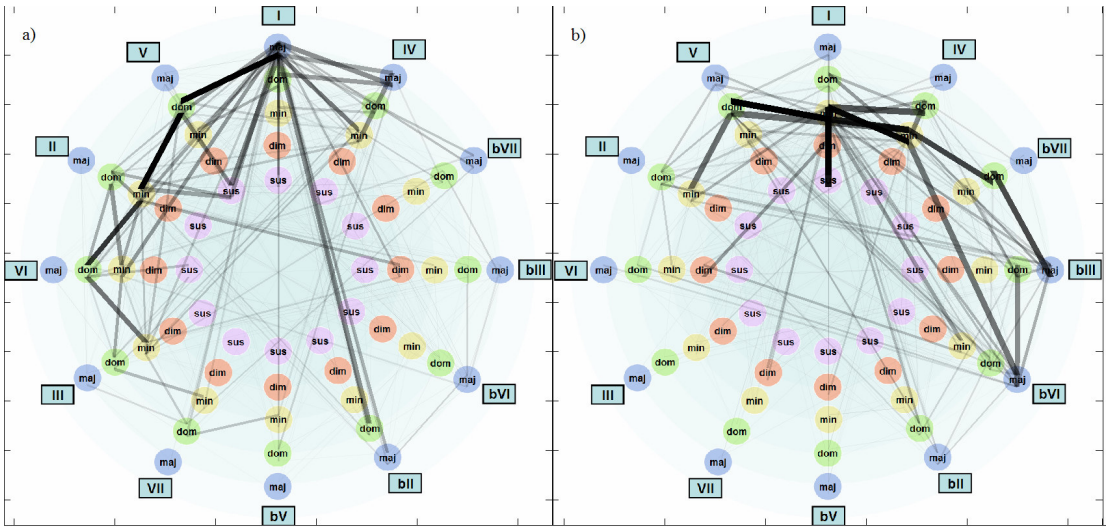



**Figure 8. Chord transitions in all songs in major keys (a) and in minor keys (b).**

Like those in major keys, the chord progressions used in songs in minor keys also have movement by fourths and fifths and their substitutions. However, there is more root in these songs by step and third, which shown in Figure 8b as connections with longer lines which cross neighboring spokes.

## 4.5 Subsequences

A list of the most common subsequences found in Jobim's songs in major keys is shown in Figure 9. As expected, transitions like *<VI7 ii V7 I>* were found among the most frequent. This group of frequently used elements is typical of popular music and jazz standards of from the second half of the 20th century, indicating that chord progressions alone are not sufficient to explain the unique characteristics of Jobim's style. The chord symbols in the database generally do not include the non-chord tones that Jobim often used in his melodies. Another important element of his style that is missing is the description of his smooth voice leading and countermelodies.

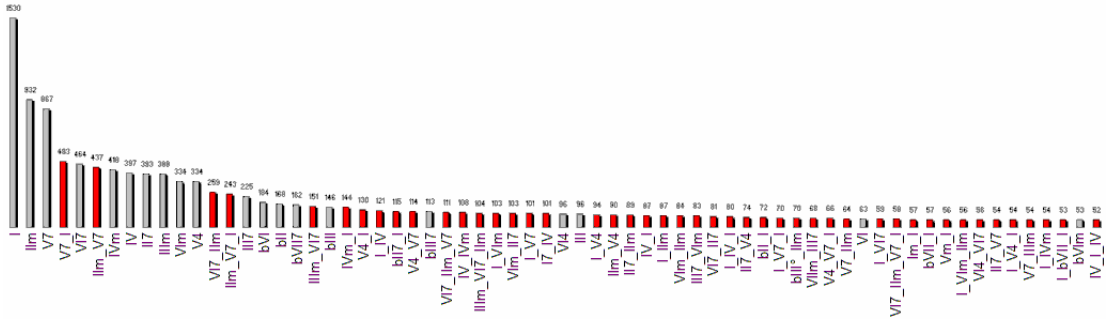


**Figure 9. Subsequences in songs in major keys occurring more than fifty times The bars in gray indicate single chords, which can be omitted from the graph.**

Still, some surprises emerged, like the progression *<II7 IIm>* among the most usual. Another important finding is that some single chords appear after entire sequences. For example, the *III* (35th position, 96 occurrences) has less occurrences than the sequence *<iii V7/ii ii>* (29th position, 104 occurrences), and the *VI* (52nd position, 63 occurrences) has less occurrences than sequences like *<bIII° ii>* (48th position, 70 occurrences) and *<vii III7>* (49th position, 68 occurrences).

### 4.6 Change in progressions over the years

Figure 10 shows the evolution of the harmony in the music of Jobim over the volumes of the Cancioneiro series. The last graph refers to the sum of all five volumes. The most apparent change in harmonic progressions over time was observed to occur between volumes 1 and 2, shown in Figures 10a and 10b. This marks a major change in Jobim's vocabulary, beginning in volume 2 which covered the bossa nova period, recognized for its rich harmony. The HarmIn tools also generated video animations showing the evolution volume by volume and song by song. These videos, as well as other material is available at the website http://willshare.com/jobim.

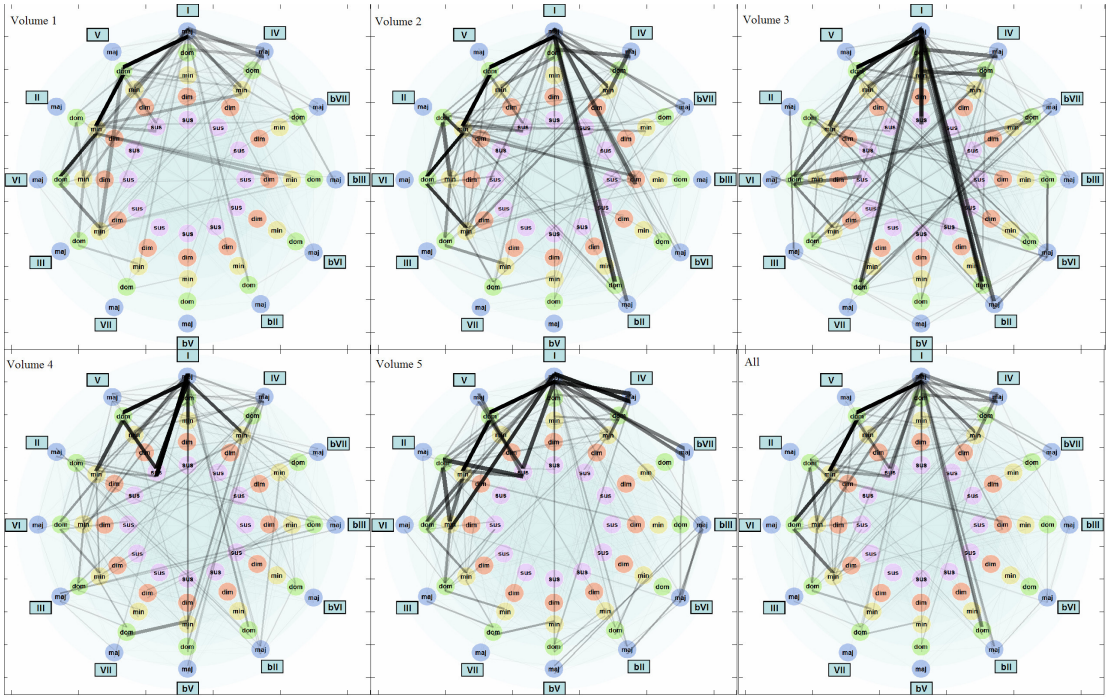



**Figure 10. Chord transition graphs related to songs in major keys from the 5 Volumes of the Cancioneiro Jobim.**

### 4.7 Analysis of most famous songs

Figure 11 shows HarmIn ability to compare datasets. It can display the graphs side by side, compute statistical data, and generate graphs showing the differences between the datasets. Figure 11 presents a selection of sixteen of Jobim's most famous songs in major keys (the Hits dataset, in Figure 11b) compared with the entire group of songs in major keys (Figure 11a). The Hits dataset appears to have more modulations to distantly related keys, and shows increased use of substitute dominants, resulting in smooth bass lines. Figure 11c shows the graph of the difference between them. The red lines represent a higher incidence of certain chord transitions in the first dataset, green lines represent a higher incidence in the second one.

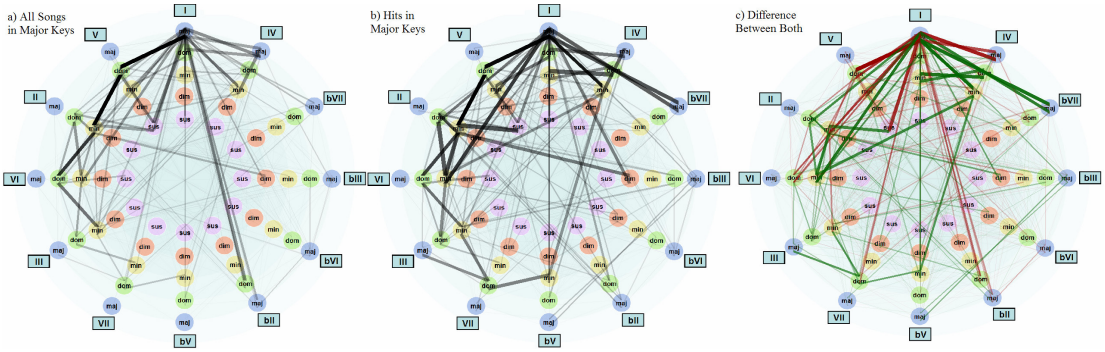



**Figure 11. Comparison between the chord transitions in 16 of Jobim's most famous songs and the rest of his work. The 3rd graph plots the difference between them.**

## 4.8 Comparing Datasets

Section 4.8 already exposed how HarmIn can be used to compare datasets. We performed another experiment in this sense, intended to compare the music of Antônio Carlos Jobim with other composers or styles. We gathered many tablatures from a website specially dedicated (Mvhp 2007). Then, we created six datasets, three for composers (The Beatles, Chico Buarque, Caetano Veloso), and three for music genres (respecting the classification given by the site: Axé, Forró, Brazilian Funk). The Beatles' dataset is composed of 107 songs, Chico Buarque's of 121, and Caetano Veloso's of 122. The Forró dataset contains 121 songs by 15 artists, the Axé dataset contains 396 songs by 16 artists, and the Funk 25 songs by 8 artists. Although these datasets are not authoritative and parsing them may lead to errors, introducing the results still seems worthwhile.

Figure 12 shows comparative graphs between his music and that of The Beatles. Figure 12a shows the difference in the number of times each chord has been used. This graph reveal that The Beatles used much more certain few types of chords (negative values in the histogram), while Jobim used a wider range of chords (positive values in the histogram). Figure 12b shows the difference between the number of times each chord transition has occurred. White pixels indicate a higher incidence of a particular chord transition in the music of The Beatles, black pixels indicate a higher incidence in the music of Jobim. Figure 12c shows the same difference in the chord transition graph. Green lines represent chord transitions used more often by The Beatles, red lines are the ones used more often by Jobim.

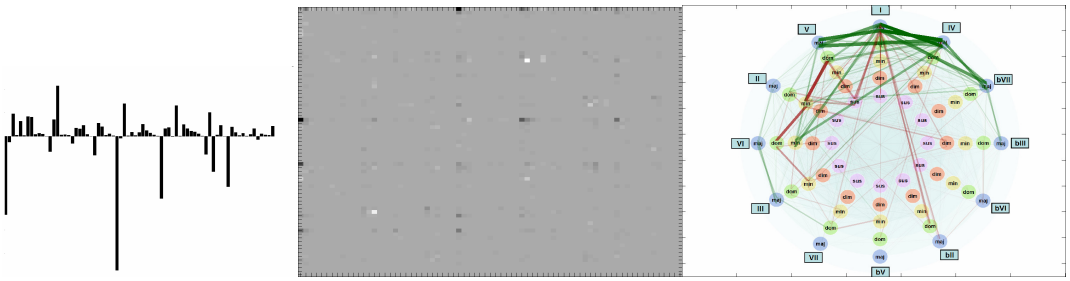


**Figure 12. Comparison between the music of Jobim and the Beatles.**

HarmIn can also give some statistics about the comparison. For example, the correlation or the covariance of two datasets, the standard deviation of a group of datasets, the sum

of the absolute differences between two datasets, or the sum of their product. The table below presents the correlation coefficients between them. We can see how the Axé and the Brazilian Funk harmonies are highly correlated. Also, the data point Chico Buarque's harmonies being the most correlated to Jobim's, and interestingly a certain similarity between Caetano Veloso's harmony and that of The Beatles.

| Major | Tom | Beatles | ChicoBuarque | Caetano | Forro | Axé | Funk |
|---|---|---|---|---|---|---|---|
| Tom | 1,00000 | 0,47868 | 0,68656 | 0,61271 | 0,28819 | 0,57131 | 0,61775 |
| Beatles | 0,47868 | 1,00000 | 0,69512 | 0,85065 | 0,69558 | 0,83029 | 0,63504 |
| Chico Buarque | 0,68656 | 0,69512 | 1,00000 | 0,78441 | 0,51290 | 0,72072 | 0,62971 |
| Caetano | 0,61271 | 0,85065 | 0,78441 | 1,00000 | 0,69159 | 0,83953 | 0,68011 |
| Forro | 0,28819 | 0,69558 | 0,51290 | 0,69159 | 1,00000 | 0,84511 | 0,68511 |
| Axé | 0,57131 | 0,83029 | 0,72072 | 0,83953 | 0,84511 | 1,00000 | 0,87386 |
| Funk | 0,61775 | 0,63504 | 0,62971 | 0,68011 | 0,68511 | 0,87386 | 1,00000 |

| Minor | Tom | Beatles | ChicoBuarque | Caetano | Forro | Axé | FunkCarioca |
|---|---|---|---|---|---|---|---|
| Tom | 1,00000 | 0,44540 | 0,68752 | 0,59144 | 0,33371 | 0,34375 | 0,21177 |
| Beatles | 0,44540 | 1,00000 | 0,27671 | 0,65560 | 0,45967 | 0,39702 | 0,38435 |
| Chico Buarque | 0,68752 | 0,27671 | 1,00000 | 0,51849 | 0,38903 | 0,36033 | 0,23657 |
| Caetano | 0,59144 | 0,65560 | 0,51849 | 1,00000 | 0,48903 | 0,53617 | 0,37496 |
| Forro | 0,33371 | 0,45967 | 0,38903 | 0,48903 | 1,00000 | 0,62917 | 0,55129 |
| Axé | 0,34375 | 0,39702 | 0,36033 | 0,53617 | 0,62917 | 1,00000 | 0,52432 |
| Funk Carioca | 0,21177 | 0,38435 | 0,23657 | 0,37496 | 0,55129 | 0,52432 | 1,00000 |

**Figure 13. Table with the correlation coefficients between each pair of datasets considered.**

The correlation between chord transition matrices, however, does not always explain harmonic similarities or divergences. Even if all matrices are normalized by the overall number of chords in each database, the scaling may affect the results. Sometimes a big difference in a certain chord transition hides many similarities. The main harmonic characteristic of a composer may be the use of certain chords in specific situations, and as these chords may seldon appear, they won't impact the final result. Thus, the analysis must comprise all graphs and statistics: the number of chords used, the variety and shape of the chord transition paths, etc. HarmIn does not give an answer to questions like "Which composer is most similar to Jobim?", but it provides the user with the tools to analyze and reach its own conclusions. To demonstrate that, the relation of the three genres we studied is shown in Figure 14. In fact the graphs seem quite similar while very different from the others. Check our website for the full graphics and data.

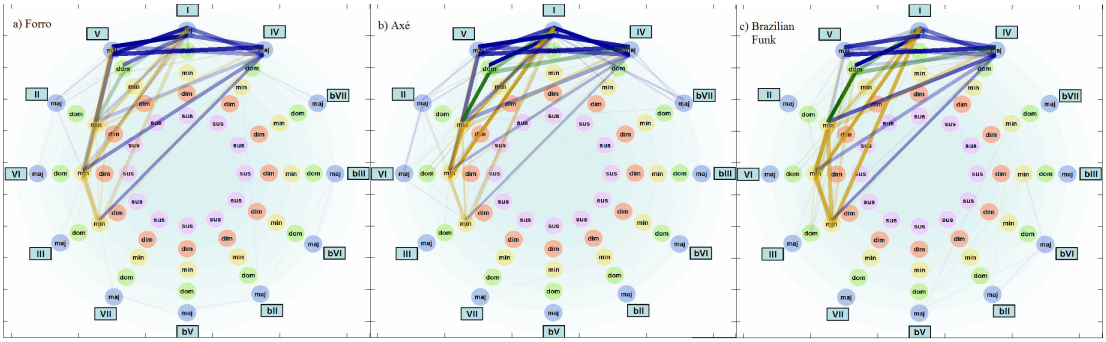



**Figure 14. Chord transition graphs for Forro, Axé, Brazilian Funk genres.**

## 4.9 Generation of new progressions

A final feature of HarmIn we experimented was its capability to integrate with RamIn, then to use its power to create new chord progressions, coherent with the harmony used as input. RamIn allows an iterative and interactive composition while the user repeatedly accepts or rejects continuations for a given harmonic input sequence. Figure 15 below shows a progression made with the help of RamIn.

```
Dm7(9) / G7(13) / Dm7(9) /
G7(13) / Dm7(9) / G7(13) /
Dm7(9) / G7(13) /

C7M(9) / G(add9) / G/A Em7
F#m7 Em6/G G#m7(11) / C#7(9)
/ G#m7 / C#7 /

C#7(b13) / Bm7 / Bm / Bm7 /
E7(13) / Em7 / A7 A6(9) Dm7
/
/ / Gm7(9) / C7 / Gm7 /
A7/E / Dm7(9) / / /

Dm7(9) / / /
G7/B / Gm7/Bb / Am7 / / /
Am7 D7 Am7 D7 Am7 D7 Am7 D7

Dm7 G7(13) Dm7 G7(13) C / D
/ D / C#7 / D / Bm7 /
Em7 / A7 A6(9) D6(9) / / /
```

**Figure 15. A new chord progression based on the analysis of Jobim's, made with the help of RamIn.**

## 5. Conclusion

This paper presented a set of tools grouped under the name of HarmIn, intended to help the analysis of harmony, especially chord progressions. The main features of HarmIn were presented, along with a case study with the music of Antônio Carlos Jobim, as well as some comparisons between him and other composers. A website accompanying this paper (http://willshare.com/jobim) includes sample song data, color graphics and videos, and audio examples of generated sequences.

## 6. References


Cabral, G. and Briot, J.-P. and Pachet, F. *Incremental parsing for real time accompaniment systems*. In Special Track on Artificial Intelligence in Music and Art (AIMA'2006) Florida Artificial Intelligence Research Society Conference 19th International FLAIRS'2006 Conference, Florida Artificial Intelligence Research Society, Melbourne Beach, Florida, USA, 2006.

Cabral, G. *Comping by Humming : Adaptation x Continuité dans l'Interaction Musicale*. Rapport de Pré-Soutenance de Thèse, Paris, 2005.

Cabral, G., Briot, J.-P., and Pachet, F. *Impact of Distance in Pitch Class Profile Computation*. In Proceedings of the 10th Brazilian Symposium on Computer Music (SBCM 2005). Belo Horizonte, Brazil, 2005.

Fujita, T. *The Beatles: Complete Scores*. Milwaukee: Hal Leonard Publishing Corporation, 1993.

Hiller, L. and Isaacson, L. "Musical Composition with a High-Speed Digital Computer", in Machines Models of Music, Schwanauer, M. And Levitt, D. Eds, MIT Press, pp. 9-21, 1993. Reprint of original article in Journal of Audio Engineering Society, 1958.

Jobim, A. C. *Cancioneiro Jobim: complete works, arrangements for piano*, 5 volumes, Jobim, P., coordination. Rio de Janeiro: Jobim Music, 2001.

Johansson, KG. *The Harmonic Language of the Beatles*, STM-Online Vol. 2, 1999. http://www.musik.uu.se/ssm/stmonline/vol 2 1/KGJO/index.html

Mvhp. *MV Portal de Cifras*, Website: http://www.mvhp.com.br, 2007.

Willey, R. *"Antônio Carlos ('Tom') Jobim"*, Encyclopedia of Recorded Sound in the United States, edited by Frank Hoffmann. New York: Routledge, 2005.

Willey, R., and Cabral, G. *Representing Antônio Carlos Jobim's Harmonic Progressions*. Submitted to the 11th Brazilian Symposium on Computer Music (SBCM 2007), São Paulo, 2007.

# Comparing audio descriptors for singing voice detection in music audio files

**Martín Rocamora**[1]*, **Perfecto Herrera**[2]

[1]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

[2]Music Technology Group – Universitat Pompeu Fabra
Pg. Circumval·lació 8 – 08003 Barcelona, Spain

rocamora@fing.edu.uy, pherrera@iua.upf.edu

***Abstract.*** *Given the relevance of the singing voice in popular western music, a system able to reliable identify those portions of a music audio file containing vocals would be very useful. In this work, we explore already used descriptors to perform this task and compare the performance of a statistical classifier using each kind of them, concluding that MFCC are the most appropriate. As an outcome of our study, an effective statistical classification system with a reduced set of descriptors for singing voice detection in music audio files is presented. The performance of the system is validated using independent datasets of popular music for training, validation and testing, reaching a classification performance of 78.5% on the testing set.*

## 1. Introduction

One of the most memorable and representative features of a song in popular western music is the singing voice melody. For this reason, a lot of research being carried out on Music Information Retrieval deals with it (singer identification, singing voice separation, singing voice melody transcription, query by humming, lyrics transcription, etc). This kind of research would benefit from a reliable segmentation of a song into singing voice fragments. Furthermore, singing voice detection has many other applications in audio processing, such as to generate "lyrics-centred" summaries of songs, or to apply a certain singing voice filter (e.g. de-esser, de-breather) only to the automatically identified vocal fragments of a music audio file.

In this paper we address the problem of processing a music audio file and segmenting it into fragments containing singing (with or without musical accompaniment) and purely instrumental (non-singing) content, referred as vocal and non-vocal respectively. Contextual information must be considered when partitioning to ignore pauses (short non-vocal fragments) during a singing melody or to correct classification errors. One of the most troublesome characteristics of the problem is the variability of music, both with regards to the singing performance and to the accompaniment. In order to limit the scope of the problem, in this work we focus on popular music (in particular music genres such as rock, pop, folk, funk and jazz).

To address the singing voice detection problem we can take advantage of the knowledge on research fields such as musical instruments classification [Martin, 1999] [Herrera et al., 2006] and speech processing [Rabiner and Schafer, 1978] [Chilton, 1999]. The former studies our ability to distinguish different musical instruments. Singing voice detection can be considered a particular case of musical instruments classification in complex mixtures, so many features used in this field may be useful for characterizing vocal and non-vocal portions of a song. Given the similarities between speech and singing voice it is reasonable to apply techniques and descriptors used to segment and recognize speech to singing voice problems. Speech/music discrimination, singing voice separation and singer identification are closely related problems, as many systems developed to perform these tasks try to identify those fragments of the audio file containing vocals.

Although singing voices resembles speech to a certain extent there are significant differences between them that need to be taken into account. To sing a melody line with lyrics it is usually necessary to stretch the voiced sounds and shrink the unvoiced sounds to match notes durations.[1] For this reason, singing voice is more than 90% voiced, where as speech is only aproximately 60% voiced [Cook, 1990]. The majority of the singing voice energy falls between 200 Hz and 2000 Hz, but in speech the unvoiced sounds

*Supported by Comisión Sectorial de Investigación Científica, UdelaR.

[1]Vocal sounds are usually divided by speech researchers in voiced and unvoiced. The vibration of the vocal folds produces quasi-periodic sound referred to as voiced. Those vocal sounds generated by the turbulence of air against the lips or tongue (such as the consonants "s" or "f") are known as unvoiced and its waveform appears random though with some limited spectral shaping [Chilton, 1999].

are more common and tend to raise this energy limit up to 4000 Hz. Speech has a characteristic energy modulation peak around the 4 Hz syllabic rate usually considered as an evidence of it presence in automatic speech processing [Scheirer and Slaney, 1997]. With regards to pitch, the pitch contour of the singing voice tends to be piece-wise constant with abrupt pitch changes in between, while in natural speech pitch slowly drifts down with smooth pitch changes. Besides this, speech pitch is normally between 80 to 400 Hz whereas singing has a wider pitch range that can reach 1400 Hz in a soprano singer [Li and Wang, 2005] [Gerhard, 2002]. Morover, singing voice is highly harmonic, that means that the partials of the sound are located at multiples of the fundamental frequency. Several musical instruments are also harmonic, so some partials of the singing voice are overlapped with those from the accompainment. Additionally, a known feature of operatic singing is the precense of an additional formant (resonance of the vocal tract), called the singing formant, in the frequency range of 2000 to 3000 Hz, that enables the voice to stand out from the accompainment [Sundberg, 1987]. However, the singing formant does not exist in many other types of singing such as the ones in pop or rock.

The approach proposed in this work is to build a statistical classifier trained on descriptors of accompanied singing voices and accompaniments alone. Much effort is put in the study of descriptors taking into account the great majority of those reported in previous work and comparing them in equivalent conditions. The rest of this paper is organized as follows. In the next section the different approaches proposed to address the singing voice detection problem are summarized. Section 3 describes the method we applied for the selection of descriptors and the classification approach. Experimental results are presented in section 4. The paper ends with a discussion on the present work and the main conclusions.

## 2. Previous work

The common procedure followed for partitioning a song into vocal and non-vocal portions is to extract feature paremeters from audio signal frames (near stationary block of samples) at each few tens milliseconds, and then to classify them to one of each class using a threshold method or a statistical classifier.

Threshold methods tend to be simple but need descriptors that clearly discriminate between classes in order to be successful. The methods proposed to classify audio frames into vocal and non-vocal apply a threshold on only one descriptor [Maddage et al., 2004] [Shenoy et al., 2005] or compute different descriptors and apply a set of thresholds on them [Zhang, 2002]. On the other hand, statistical classifiers are trained using accompanied singing voices and pure instrumentals and can learn complex boundaries between classes combining several descriptors. This has the drawbacks of a certain amount of time spent on training and the difficulty of obtaining ground truth annotations. In the singing voice detection problem, finding the exact boundaries over the entire song can be time-consuming, and sometimes could be difficult in case of slow decays or masking. Moreover, special attention has to be paid to ensure generalization beyond the training set avoiding overfitting. Several statistical classifiers have been explored to address the problem of singing voice detection, such as Hidden Markov Models (HMM) [Berenzweig and Ellis, 2001] [New et al., 2004], Gaussian Mixture Models (GMM) [Tsai and Wang, 2006] [Li and Wang, 2007], Artificial Neural Networks (ANN) [Berenzweig et al., 2002] [Tzanetakis, 2004] and Support Vector Machines (SVM) [Maddage et al., 2003] [Maddage et al., 2004].

The short-term classification of each signal frame considers only local information so it is prone to errors, and the classification obtained is typically noisy changing from one class to the other. For this reason, usually long-term information is introduced, by smoothing the classification [Tsai and Wang, 2006] or by partitioning the song into segments (much longer than frames) and assigning the same class to the whole segment. This partitioning of the song is performed based on tempo [New et al., 2004] [Maddage et al., 2003], chord change [Maddage et al., 2004] or spectral (timbre) change [Li and Wang, 2007]. More global temporal aspects of a song are also taken into account by modeling the song structure (intro, verse, chorus, etc) by means of a HMM [New et al., 2004].

With regards to descriptors, research on musical instruments classification has demonstrated the importance of temporal and spectral features [Martin, 1999] [Herrera et al., 2006], and the speech processing field has contributed on well known techniques (such as Linear Prediction) to compute voice signal attributes [Rabiner and Schafer, 1978] [Chilton, 1999]. A broad group of descriptors has been used for the purpose of singing voice detection. Singing voice carries the main melody and the lyrics of a popular song, so it is is usually one of the most salient instruments of the mixture. Therefore, vocal frames can be identified by an energy increase of the signal, and energy or power descriptors are often used [Zhang, 2002] [Tzanetakis, 2004] [New et al., 2004] [Shenoy et al., 2005] [Maddage et al., 2003]. The timbre of different instruments is partially dictated by its spectral characteristics and when new sounds enters a mixture they usually introduce significant spectral changes. Among the descriptors computed to represent this are Mel Frequency Cepstral Coefficients (MFCC) [Tsai and Wang, 2006] [Li and Wang, 2007]

[Maddage et al., 2003], Linear Prediction Coefficients (LPC) (warped LPC or perceptually derived LPC) [Berenzweig and Ellis, 2001] [Berenzweig et al., 2002] [Kim and Whitman, 2002] [Maddage et al., 2003], Log Frequency Power Coefficients (LFPC) [New et al., 2004], and spectral Flux, Centroid and Roll-Off [Zhang, 2002] [Tzanetakis, 2004]. The delta coefficients of the previous features or variances of them are also used to capture temporal information [Berenzweig et al., 2002]. As stated previously singing voice is highly harmonic, thus the harmonicity of the signal, usually computed as an Harmonic Coefficient (HC), is used as a clue for singing voice detection [Chou and Gu, 2001] [Zhang, 2002] [Kim and Whitman, 2002].

Regarding harmonicity a pre-processing technique was proposed that consist in filtering the signal by an inverse comb filter to attenuate the harmonic sounds in the mixture [New et al., 2004] [Shenoy et al., 2005]. Harmonic accompaniment instruments have a very regular harmonic structure and can be partially removed by this filtering. Although being harmonic, singing voice has some features (vibrato, intonation) that deviate the frequency of partials from perfectly harmonic and cause it to remain after filtering.

The following is a summary of the most relevant research work on singing voice detection in chronological order. To our knowledge, the first work that focused and described the problem of locating the singing voice segments in music signals was [Berenzweig and Ellis, 2001]. Posterior probability features and their statistics are derived from an ANN trained on a phone classes database to work with speech. A HMM with two states is used to discriminate singing from accompaniment. Close in time, an approach for singing detection in speech/music discrimination is described in [Chou and Gu, 2001]. It employs a set of features that comprises MFCC as well as HC, 4Hz modulation and energy based features to train a GMM.

Following this early work, new proposals for singing detection were suggested. Artist classification is improved in [Berenzweig et al., 2002] by using only voice segments detected with a multi-layer perceptron that is fed with Perceptual LPC (PLPC), plus deltas and double deltas. An harmonic sound detector is presented in [Kim and Whitman, 2002] to identify vocal regions of an audio signal for singer identification. It works under the hypothesis that most harmonic sounds correspond to regions of singing. By means of an inverse comb filter bank, the signal is attenuated and the Harmonicity is computed as the ratio between the total energy in a frame over the energy of the most attenuated signal. The automatic singer identification system described in [Zhang, 2002] identifies the starting point of the singing voice in a song using energy features, zero-crossing rate (ZCR), HC and Spectral Flux and classifying with a set of thresholds. In [Maddage et al., 2003] a study which aims to establish if there are significant statistical differences between vocal music, instrumental music and mixed vocal and instruments is described. Descriptors set contains LPC, LPC derived Cepstrum, MFCC, Spectral Power, Short Time Energy and ZCR. Classification performance of a SVM is shown to be superior to an ANN and a GMM.

Subsequent research explored some other descriptors and classification approaches. A technique for singing voice detection is proposed in [Maddage et al., 2004]. Musical signals are segmented into beat-length frames using a rhythm extraction algorithm, the Fast Fourier Transform (FFT) of each frame is calculated, and after filtering to a narrow bandwith containing mostly voice, another FFT is applied (Twice Iterated Composite Fourier Transform, TICFT). Based on a threshold on the energy of the TICFT, singing voice frames are separated from instrumental frames. Performance is improved by some frame-correction rules based on chord pattern changes. In [Tzanetakis, 2004] singing voice detection is performed by a bootstrapping process that consists in manually annotating a few random fragments of the song being processed to train a classifier. The feature set includes Mean Relative Energy, and Mean and Standard Deviation of Spectral Centroid, Roll-off, Flux and Pitch. Different classifiers are tested, being Logistic Regression and ANN those which performed best. In the work described in [New et al., 2004] , based on the observation that a more regular harmonic structure is present in non-vocal sections as compared to vocal sections, a key estimation is performed to attenuate the harmonics of the spectrum and LFPC are computed. The energy distribution of the LFPC shows that vocal segments have relatively higher energy values in the higher frequency bands. Classification is done with a multi-model HMM that models the different sections of a typical song structure (intro, verse, chorus, bridge, outro). A classification refinement is provided by a verification step based on classification confidence and an automatic bootstrapping process (similar to that proposed in [Tzanetakis, 2004]). The work in [Shenoy et al., 2005] also addresses the singing voice segmentation problem by estimating the key of the song in order to perform an inverse comb filtering to attenuate the harmonic sounds. Singing voice is retained after filtering due to vibrato and intonation. The energy in different frequency sub-bands is computed and the highest energy frames are classified as vocal.

Most recent works use MFCC as feature vectors and GMM as classifiers. A vocal/non-vocal detection algorithm is proposed in [Tsai and Wang, 2006] applied to singer recognition. The class of each frame is hypothesized according to log-likelihoods and the final decision is made at homogeneous segments. In [Li and Wang, 2007], the problem of singing voice separation from music accompaniment is addressed and a singing voice detection procedure is used. The audio is partitioned by detecting large spectral changes, and segments are classified according to the log-likelihoods of all the frames of a portion.

# 3. Method

## 3.1. Databases

Independent datasets of popular music recordings were used for training, validation and testing. Audio of all datasets is stereo at a sampling rate ($f_s$) of 44.1 kHz. The training database was build extracting short audio excerpts from music recordings and manually classifying them into vocal and non-vocal. Three different excerpt-length sets were constructed of 0.5, 1 and 3 seconds length. Each set contains 500 instances of each class. Music utilized belongs to a music genres database comprising alternative, blues, classical, country, electronic, folk, funk, heavy-metal, hip-hop, jazz, pop, religious, rock and soul. In addition, approximately 25% of pure instrumental and a capella music was also added. The validation database consists of 63 fragments of 10 seconds that were manually annotated. Music was extracted from Magnatune[2] recordings belonging to similar genres as the ones used for training. Once the features and the classifier were set and its parameters finely tuned, an independent evaluation was conducted on a testing database of 46 manually annotated songs, for a total duration of 3 hours. Music in this set comprises 7 different singers performing in genres that were used for training and validation.

## 3.2. Descriptors implemented

Based on the existing literature the following timbre descriptors were implemented: Mel Frequency Cepstral Coefficients (MFCC), Perceptually derived LPC (PLPC), Log Frequency Power Coefficients (LFPC) and Harmonic Coefficient (HC). In addition, a general purpose musical instruments classification feature set was built, including Spectral Centroid, Roll-off, Flux, Skewness, Kurtosis and Flatness. Pitch was also included, being the only non-spectral feature reported that was considered relevant (4Hz modulation is appropriate for speech but not for singing [Chou and Gu, 2001], ZCR strongly correlates with the Spectral Centroid [Herrera et al., 2006] and other power or energy features can be regarded as variants of spectral descriptors such as LFPC).

Audio signal is processed in frames of 25 ms using a Hamming window and a hop size of 10 ms. Considering that the majority of energy in the singing voice falls between 200 Hz and 2000 Hz [Kim and Whitman, 2002], the frequency range is established in 200 Hz to 16 kHz for all spectral descriptors.

Implementation of MFCC derives 13 coefficients from 40 mel scale frequency bands for each signal frame. An FFT is applied to each signal frame and the magnitude spectrum is obtained by taking the absolute value. After that, the magnitude spectrum is processed by a filter bank whose center frequencies are spaced according to the mel scale. Then, energy on each band is computed and the logarithm is taken. The elements of these vectors are highly correlated so a Discrete Cosine Transform (DCT) is applied to finally obtain the MFCC.

Some psychoacoustic concepts are introduced in the PLP analysis technique that make it more consistent with human hearing in comparison with conventional LP analysis. PLP coefficients are obtained by a critical band integration of the signal, followed by equal loudness weighting and intensity to loudness conversion. Finally, a Linear Prediction analysis of order 12 is applied. Both MFCC and PLPC are implemented using [Ellis, 2005][3].

To derive LFPC the signal frames are passed through a bank of 12 band-pass filters spaced logarithmically, and the coefficient for each band is obtained by computing the power of the band divided over the band bandwith and expressed in decibels [New et al., 2004], as follows,

$$\mathrm{LFPC}_t(m) = 10\ \log_{10}\left|\frac{S_t(m)}{N_m}\right|, \quad S_t(m) = \sum_{k=f_{m-1}}^{f_m} X_t(k)^2, \quad m = 1, 2, \ldots, 12$$

where $t$ is the frame number, $m$ indicates the band number, $S_t(m)$ is the power of the band, $N_m$ is the number of spectral components in the band, $X_t(k)$ is the $k^{th}$ spectral component of the signal frame, and $f_m$ are the indexes of the band boundaries corresponding to frequencies spaced logarithmically from 200Hz to 16kHz as represented in figure 1.

Implementation of HC follows the procedure described in [Chou and Gu, 2001], where temporal and spectral autocorrelation of the signal frame are computed (TA and SA respectively), and the HC is obtained as the maximum of the sum of the autocorrelation functions, $\mathrm{HC}_t = \max_\tau [\ \mathrm{TA}_t(\tau) + \mathrm{SA}_t(\tau)\ ]$, where $t$ is the frame number, and $\tau$ is the temporal delay. Temporal and spectral autocorrelation functions

[2] http://magnatune.com/

[3] http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/

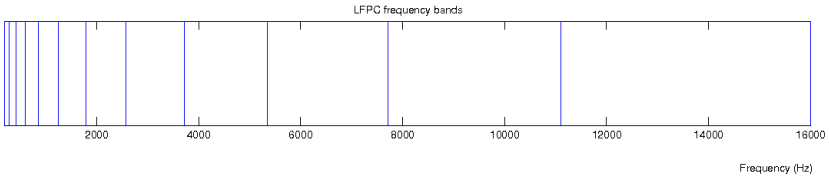



**Figure 1: Frequency bands used for LFPC computation.**

are calculated as,

$$\text{TA}(\tau) = \frac{\sum_{n=1}^{N-\tau} \bar{x}_t(n)\ \bar{x}_t(n+\tau)}{\sqrt{\sum_{n=1}^{N-\tau} \bar{x}_t^2(n) \sum_{n=1}^{N-\tau} \bar{x}_t^2(n+\tau)}},\ \text{SA}(\tau) = \frac{\sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t(k)\ \bar{X}_t(k+k_\tau)}{\sqrt{\sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t^2(k) \sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t^2(k+k_\tau)}}$$

where $x_t(n)$ is a signal frame of $N$ samples, $X_t(k)$ is the magnitude spectrum of the signal frame computed by an $M$ point FFT, $\bar{x}_t(n)$ and $\bar{X}_t(k)$ are the zero mean versions of $x_t(n)$ and $X_t(k)$, and $k_\tau = \frac{M}{\tau f_s}$ is the frequency bean index that corresponds to the time delay $\tau$.

Spectral Flux, Roll-off, Centroid, Skewness, Kurtosis and Flatness are implemented based on [Herrera et al., 2006]. The Spectral Flux (SFX) is a measure of local spectral change, and it is computed as the spectral difference of two consecutive frames as,

$$\text{SFX}_t = \sum_{k=1}^{\frac{M}{2}} \left(\hat{X}_t(k) - \hat{X}_{t-1}(k)\right)^2$$

where $\hat{X}_t(k)$ is the energy normalized magnitude spectrum of the signal frame. Spectral Roll-off is computed as the frequency index $R$ below which the majority of the spectral energy is concentrated ($\gamma = 0.85$ is used),

$$\sum_{k=1}^{R} X_t(k)^2 \leq \gamma \sum_{k=1}^{\frac{M}{2}} X_t(k)^2.$$

The rest of the spectral descriptors consider the spectrum as a distribution, which values are the frequencies and the probabilities are the normalized spectral amplitude, and compute measures of the distribution shape. The Spectral Centroid is the barycenter or center of gravity of the spectrum and is computed as,

$$\text{SC} = \frac{\sum_{k=1}^{\frac{M}{2}} k\ X_t(k)}{\sum_{k=0}^{\frac{M}{2}-1} X_t(k)}.$$

The Skewness is a measure of the asymmetry of a distribution around its mean, while the Kurtosis is a measure of the flatness of a distribution around its mean. They are computed as the normalized moments of order 3 and 4 respectively by,

$$\text{SS} = \frac{\sqrt{\frac{M}{2}} \sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^3}{\left(\sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^2\right)^{\frac{3}{2}}},\quad \text{SK} = \frac{\frac{M}{2} \sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^4}{\left(\sum_{k=1}^{\frac{M}{2}} (X_t(k) - \tilde{X}_t)^2\right)^2} - 3.$$

where $\tilde{X}_t$ is the mean value of the magnitude spectrum $X_t(k)$. The Spectral Flatness is a measure of how flat or similar to white noise the spectrum is. It is computed as the ratio of the geometric mean to the arithmetic mean of the spectrum,

$$\text{SF} = \frac{\sqrt[\frac{M}{2}]{\prod_{k=1}^{\frac{M}{2}} X_t(k)}}{\frac{1}{\frac{M}{2}} \sum_{k=1}^{\frac{M}{2}} X_t(k)}.$$

A low value indicates a tonal signal, while for a noisy signal the value is close to 1. It is typically computed for several frequency bands. We used the following four overlapped frequency bands: 200 to 1000 Hz, 800 to 2500 Hz, 2000 to 3500 Hz and 2500 to 5000 Hz.

Reliable pitch estimation in polyphonic music signals remains up to the moment a very complex open problem, so for the sake of simplicity pitch estimation is performed by applying a monophonic fundamental frequency estimation algorithm based on [A. de Cheveignè, H. Kawahara, 2002]. This has the

drawback of an unreliable estimation, noisy and prone to octave errors due to the many sounds present at the same time. The implemented algorithm uses the Difference Function (DF), a variation of the autocorrelation function that calculates the difference between the signal frame and a delayed version of it,

$$DF_t(\tau) = \sum_{n=1}^{N-\tau} \left(x_t(n) - x_t(n+\tau)\right)^2 .$$

For a periodic signal this function is zero at values of $\tau$ multiples of the signal period, while in case of quasiperiodic signal it has minimuns close to zero. The pitch of the signal is estimated as the inverse of the delay value of the first minimun.

As stated previously, the audio signal is processed in overlapped frames, so the audio features computed describe the signal at each 10 ms. To take into account descriptors information over several consecutive frames, an audio fragment is considered, of 0.5, 1 or 3 seconds. Mean, median, standard deviation, skewness and kurtosis are extracted for this fragments. Additionally, deltas and double deltas are computed for each descriptor coefficient and the same statistical measures are calculated.

### 3.3. Validation procedures

To compare descriptors, in order to select them, classification performance on the training dataset is considered. The training database is composed of audio excerpts, each of them labelled as belonging to either one class. Performance is computed as the percentage of correctly classified instances using 10-fold cross-validation (CV). Different classifiers provided by [Witten and Frank, 2005][4] were considered and, after detailed comparisons, SVM was selected (see table 3).

The validation database is used to adjust system parameters such as the fragment length to be used to process the music audio file, or the classifier options. It is also used for evaluating the inclusion of different post-processing strategies to the system. On the other hand, testing database provides an independent dataset to test the final system performance. In both cases, music audio files manually labelled must be processed. The adopted procedure divides the file into 50% overlapped fragments, descriptors for each fragment are computed at each 10 ms and statistical measures over the whole fragment are calculated. After that, each fragment is classified as vocal or non-vocal. Finally, vocal and non-vocal regions of the audio file are build considering that the class of each fragment extends from its center to half hop-size on either side (due to overlapping). Performance is computed as the percentage of time that the classification and the manual annotation coincides. It is important to note that, due to the rough discretization of the audio in fragments, even in a correctly classified case there is always a small divergence between the annotated and the computed boundaries (see figure 2).

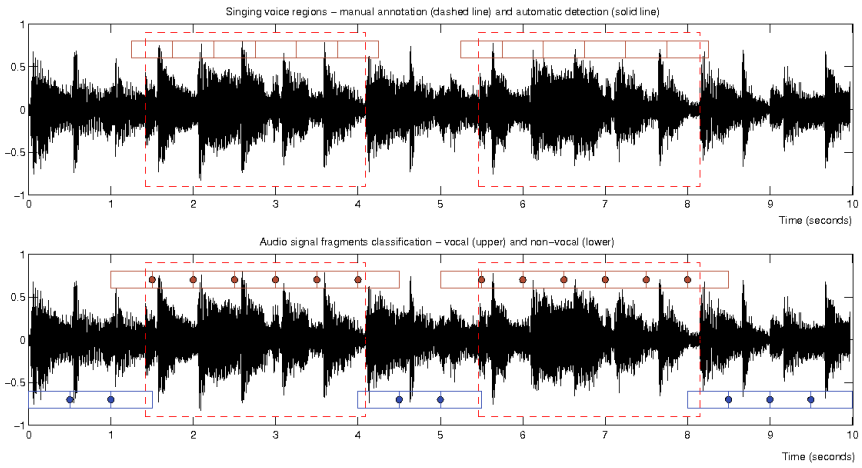



**Figure 2: Singing voice detection example. Manual annotation (dashed line) versus automatic detection (solid line) is shown at the top. Audio signal is divided into 50% overlapped fragments of 1 second length. The classification of the fragments into vocal (upper) and non-vocal (lower) is depicted at the bottom. This rough discretization of the audio in fragments produces a small divergence between the annotated and the computed boundaries of the vocal regions. Although this example could be considered a correctly classified case, performance computed as the percentage of time that the classification and the manual annotation coincides is 93.3%.**

[4] http://www.cs.waikato.ac.nz/ml/weka/

### 3.4. Selection of descriptors

As stated previously, for each descriptor coefficient (as well as deltas and double deltas), mean, median, standard deviation, skewness and kurtosis are computed. Adding irrelevant attributes to a dataset is not practical and often confuses a machine learning system [Witten and Frank, 2005]. Thus, feature selection is performed to discard less significant descriptors among each category (e.g. MFCC). For this task a correlation-based feature subset selection method provided by [Witten and Frank, 2005] is applied. However, automatic selection is not reliable when the number of features is not sufficiently small compared to the available observations, as spurious correlations between features are more likely. In those cases, a selection procedure is applied that is motivated by practical considerations concerning the inclusion of the selected features on the final system. It would be desirable to relieve the system of computing whole groups of attributes (double deltas for instance) if their contribution is not so relevant. However this is not generally ensured when appling automatic selection or dimensionality reduction techniques. For this reason, the selection firstly determines the individual classification performance of a SVM classifier for each group of statistical attributes (e.g MFCC medians). Then a procedure similar to backward elimination is applied, that is, starting with the full set and deleting a group of attributes one at a time, trying to leave out as much as possible without reducing classification performance significantly. Individual performance of each group is used as a clue for selecting those to eliminate. As a result of this feature selection, each attribute category is finally represented by a reduced set of descriptors.

### 3.5. Classification strategy

In order to select a statistical classifier for the system, different methods provided by [Witten and Frank, 2005] were applied to the training set, and 10 times 10-fold CV classification performance was compared. The best performing classification model is finally incorporated into our singing detection system. Frame length used by the system is selected by comparing the performance of the selected model trained with the different training sets (of 0.5, 1 and 3 seconds excerpt length) and applied to the validation dataset.

Some post-processing strategies were considered to improve classification performance of the system based on classification confidence and contextual information. The first one is motivated by the fact that in some cases a fragment lies over a transition between vocal and non-vocal. For this reason, if a frame has a low classification confidence (based on probability estimates for each class) it is subdivided into two new fragments and each of them is re-classified. In case of a transition each new fragment could be classified to a different class (see figure 3). Additionally, two simple context rules are proposed. If a low probability fragment is surrounded by elements of the same class re-classification is avoided. On the other hand, if one of the half size fragments produced by re-classification is surrounded by elements of the other class, it is deleted.

## 4. Results

### 4.1. Selection of descriptors

Descriptors selection results are summarized in table 1. The selected features and the total number of coefficients for each category are presented. MFCC and PLPC set includes delta coefficients, while LFPC set includes delta and double delta coefficients. The only discarded type of descriptor in the Spectral set was Flux. The complete Spectral set includes: Centroid mean and median, Roll-off mean, median and standard deviation, Skewness mean and median, Kurtosis mean and median and Flatness mean, median and standard deviation.

| Category | Performance | # Coefficients | Features selected |
|---|---|---|---|
| MFCC+D | 84.5% | 52 | median, stdev |
| LFPC+D+DD | 78.7% | 72 | median, stdev |
| Spectral | 76.0% | 21 | whole set without Flux |
| PLPC+D | 70.8% | 78 | median, stdev, skewness |
| HC | 63.6% | 2 | mean, median |
| Pitch | 59.1% | 3 | mean, stdev, kurtosis |

**Table 1: Feature selection results and 10 times 10-fold CV classification performance of a SVM on the training dataset of 1 second length audio excerpts. Performance is measured as percentage of correct decisions.**

Descriptor sets are ranked in table 1 according to the performance obtained with a SVM using 10 times 10-fold CV over the training set (note that random guess rate is 50% in this task). The results

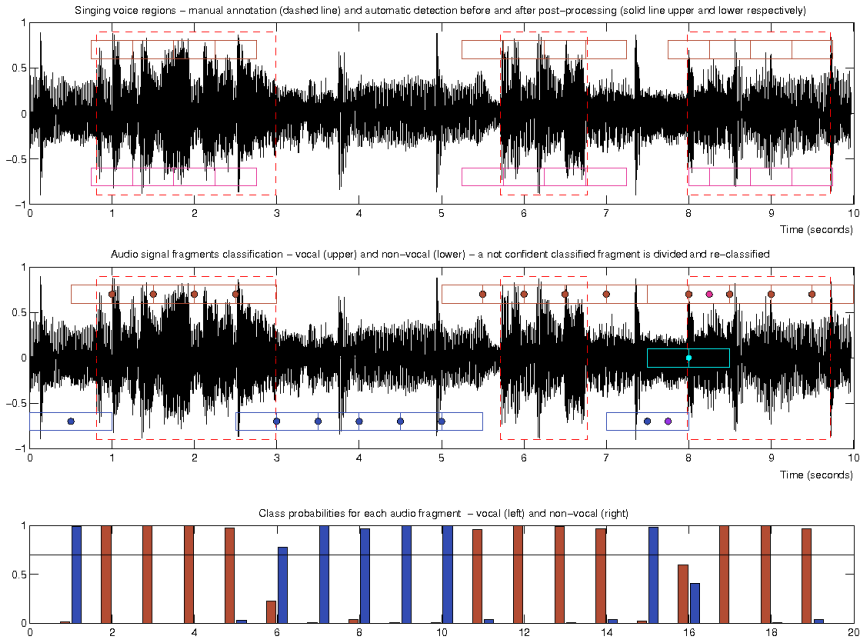



**Figure 3: Post-processing based on classification confidence. An audio frame with low probability estimates for each class is subdivided into two new fragments and each of them is re-classified. In this example, fragment centered on second 8 lies over a transition between non-vocal and vocal and its probability estimates fall below a threshold of 0.7. The fragment is divided and each new portion is correctly classified. Classification is improved after post-processing (from 84.2% to 86.5%) as it is shown in the manual anotation versus automatic detection comparison at the top.**

of paired t-tests (corrected resampled t-test [Witten and Frank, 2005]) comparing the performance of the various feature categories show that there are statistically significant differences between the MFCC set and each of the other sets (at a significance level of $p < 0.05$). The confussion matrix of the classification model based on MFCC features over the training set is presented in table 2.

| | | classified as | |
|---|---|---|---|
| | | vocal | non-vocal |
| class | vocal | 428 | 72 |
| | non-vocal | 77 | 423 |

**Table 2: Confusion matrix of the classification on the training dataset of 1 second length audio excerpts using MFCC descriptors, obtained by 10-fold CV. The rows of the matrix correspond to the ground-truth of the audio excerpt and the columns indicate the hypothesis.**

### 4.2. Classification strategy

Different classifiers were considered using the MFCC feature set on the training database comparing 10-fold CV classification performance. Table 3 shows the results obtained with a SVM trained using the Sequential Minimal Optimization method, a backpropagation ANN, a decision tree classifier implementing the well-known C4.5 algorithm and two different K-Nearest Neighbors (KNN).

Based on the classification performance on the validation set, 1 second turned out to be the most suitable fragment length. Results obtained were 73.3%, 74.2% and 73.0%, for 0.5, 1 and 3 seconds respectively. According to these results, our singing detection system was built with a SVM model using the MFCC feature set and a fragment length of 1 second.

The post-processing strategies proposed were added to the system one at a time in order to evaluate them on the validation database. Results obtained are reported in table 4.

| Classifier | SVM | ANN | KNN-3 | KNN-1 | C4.5 |
|---|---|---|---|---|---|
| Performance | 85.1% | 82.6% | 76.5% | 73.5% | 73.1% |

**Table 3: Performances of different classifiers after 10-fold CV on the training database of 1 second length audio excerpts using the MFCC set.**

| Post-processing | none | re-classify | add rule 1 | add rule 2 |
|---|---|---|---|---|
| Performance | 75.7% | 76.3% | 76.6% | 76.8% |

**Table 4: Evaluation of the post-processing strategies on the validation dataset using the MFCC descriptors. Performance is computed as the percentage of time that the classification and the manual annotation coincides.**

| Post-processing | none | all |
|---|---|---|
| Performance | 77.6% | 78.5% |

**Table 5: Performance of the system on the testing dataset with no post-processing and performing all the post-processing strategies proposed, computed as the percentage of time that the classification and the manual annotation coincides.**

Finally the system was used to process the testing set achieving a performance similar to that obtained in the validation set. Table 5 shows the results with no post-processing and considering all the post-processing strategies.

## 5. Discussion and conclusions

Analysis of the results obtained indicate that those descriptors that model the spectral content of the audio signal were the most appropriate for the problem (MFCC, LFPC, Spectral, PLPC). It is interesting that such a simple descriptor as LFPC outperformed all other features sets but MFCC, and that the general purpose Spectral outperformed PLPC. The results confirm that HC is not able to discriminate singing voice sounds from other harmonic musical instruments. The poor performance obtained with the Pitch descriptors is due to the utilization of a monophonic fundamental frequency estimation algorithm. We plan to apply other pitch estimation methods in our future work that could deal with polyphonic audio and to develop pitch descriptors that exploit singing voice pitch contour distinctive features such as intonation. Regarding MFCC, the feature selection performed points out that considering delta coefficients can boost performance (2% on the training set). However, they are generally not used when applying MFCC to this type of problem [Li and Wang, 2007] [Tsai and Wang, 2006]. Classification performance decreased significantly in validation and testing compared to 10-fold CV on the training set, which is not surprising because of the different origins and data variances of the databases. Additionally, the developed system roughly divides the audio file in fragments, so given our validation approach, we can take these results as worst-case or lower-bound estimations of the system performance.

We have studied the singing voice detection problem in music audio files by a statistical classification approach and we have compared, under equivalent conditions, the performance of several types of acoustic descriptors reported to be used for the problem. The results obtained confirm the usefulness of MFCC for this problem. As an outcome of our study, an effective singing voice detection system to process popular music audio files with a reduced set of descriptors has been developed. It is difficult to compare the performance achieved with other research work because there is no standard dataset for evaluation, but results obtained are promising and similar to the ones reported. Although our primary intention was to compare already used descriptors for this task, we have attempted to combine different descriptors as well as different classifiers. The overall classification performance was not improved so the results are not included. Also some other descriptors were tested without success. Our future work will follow this direction as it is reasonable to expect better results by combining different sources of information.

## 6. Acknowledgments

This work was carried out during an internship at the Music Technology Group (MTG) financially supported by Comisión Sectorial de Investigación Científica, UdelaR. First author is very grateful to all the people at the MTG for their support and to Dr. Alvaro Pardo for his careful reading of this article and his suggestions.

## References


A. de Cheveignè, H. Kawahara (2002). YIN, a fundamental frecuency estimator for speech and music. *Journal Acoustic Society of America*, 111:1917–1930.

Berenzweig, A. and Ellis, D. (2001). Locating singing voice segments within music signals. *Proc. IEEE Workshop on Apps. of Sig. Proc. to Acous. and Audio, Mohonk NY, October 2001*, page 4pp.

Berenzweig, A., Ellis, D., and Lawrence, S. (2002). Using voice segments to improve artist classification of music. *AES 22nd International Conference*.

Chilton, T. (1999). Speech analisys. School of Electronic and Physical Sciences, University of Surrey.

Chou, W. and Gu, L. (2001). Robust singing detection in speech/music discriminator design. *International Conference on Acoustics, Speech, and Signal Processing*.

Cook, P. R. (1990). *Identification of control parameters in an articulatory vocal tract model, with applications to the synthesis of singing*. PhD thesis, Stanford Univ., Stanford, CA.

Ellis, D. P. W. (2005). PLP and RASTA (and MFCC, and inversion) in Matlab. Online web resource: `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`.

Gerhard, D. (2002). Pitch-based acoustic feature analysis for the discrimination of speech and monofonic singing. *Journal of the Canadian Acoustical Assosiation*, pages 152–153.

Herrera, P., Klapuri, A., and Davy, M. (2006). Automatic Classification of Pitched Musical Instrument Sounds. In Klapuri, A. and Davy, M., editors, *Signal Processing Methods for Music Transcription*, pages 163–200. Springer, New York.

Kim, Y. E. and Whitman, B. P. (2002). Singer identification in popular music recordings using voice coding features. *In Proceedings of Interational Conference on Music Information Retrieval*, pages 164–169. Paris, France.

Li, Y. and Wang, D. (2007). Separation of singing voice from music accompaniment for monaural recordings. *IEEE Transactions on Audio, Speech and Language Processing*.

Li, Y. and Wang, D. L. (2005). Separation of singing voice from music accompaniment for monaural recordings. Technical Report OSU-CISRC-9/05-TR61, Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA.

Maddage, N. C., Wan, K., Xu, C., and Wang, Y. (June 2004). Singing voice detection using twice-iterated composite fourier transform. *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference*, pages Page(s):1347 – 1350 Vol.2.

Maddage, N. C., Xu, C., and Wang, Y. (2003). A svm-based classification approach to musical audio. *Proc. ISMIR*.

Martin, K. D. (1999). *Sound-Source Recognition: A Theory and Computational Model*. PhD thesis, MIT. Cambridge, MA.

New, T. L., Shenoy, A., and Wang, Y. (2004). Singing voice detection in popular music. Technical report, Department of Computer Science, University of Singapore, Singapore, October 2004.

Rabiner, L. R. and Schafer, R. W. (1978). *Digital Processing of Speech Signals*. Prentice Hall, New Jersey.

Scheirer, E. D. and Slaney, M. (1997). Construction and evaluation of a robust multifeature speech/music discriminator. *ICASSP, Munich, Germany*.

Shenoy, A., Wu, Y., and Wang, Y. (2005). Singing voice detection for karaoke application. *Visual Communications and Image Processing 2005, Proc. of SPIE*, page Vol. 5960.

Sundberg, J. (1987). *The science of the singing voice*. De Kalb, Il., Northern Illinois University Press.

Tsai, W. H. and Wang, H. M. (2006). Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signal. *IEEE Transactions on Speech and Audio Processing, January 2006.*, pages Vol. 14, No 1.

Tzanetakis, G. (2004). Song-specific bootstrapping of singing voice structure. Technical report, Department of computer science, University of Victoria.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. 2nd Edition, Morgan Kaufmann, San Francisco.

Zhang, T. (2002). System and method for automatic singer identification. HP Labs Technical Report.

# Microtiming in "Samba de Roda" —Preliminary experiments with polyphonic audio

**Fabien Gouyon**[1]

[1]INESC Porto (Institute for Systems and Computer Engineering of Porto),
Campus da FEUP, Rua Dr. Roberto Frias, 378
4200-465 Porto, Portugal

fgouyon@inescporto.pt

***Abstract.*** *In this paper we report on preliminary experiments in the study of microtiming features in Samba music. Focusing on polyphonic audio music with almost constant tempo, we propose an algorithm to discover from data in a bottom-up manner systematic timing deviations at the 16th-note level in quarter-note-long temporal patterns.*
*Our analysis of the data highlights a systematic shift of third and fourth 16th-note beats, slightly ahead of their corresponding quantized positions.*

## 1. Introduction

Music styles can often be characterized by specific repeating rhythmic patterns. There are two important aspects to such patterns. First, the relative time instants of musical notes as measured on a discrete, quantized time grid (specified by integer multiples or divisions of a basic pulse). This is how a pattern would be noted on a score. And second, the small and yet systematic timing deviations between instants where notes are actually played and their corresponding quantized positions. In some cases, these deviations can be represented as series of tempo changes, while in other cases they are better represented as *event shifts* at a *constant tempo* [Desain and Honing, 1991, Bilmes, 1993]. In this paper, we focus on the latter type of deviations, which we will refer to as "microtiming."

Such deviations occur in many different musical styles and depend on the position in the metrical structure. For instance, Jazz is characterized by a particular pattern of deviations: the "swing," where "consecutive eighth-notes are performed as long-short patterns" [Friberg and Sundström, 2002]. Traditional Irish fiddle music also shows comparable timing deviations at the eighth-note level [Rosinach and Traube, 2006]. One characteristic of Viennese Waltz is also a certain amount of swing, but this time at the quarter-note level (each third quarter-note in a bar is shorter). In this paper, we are interested in a particular type of Samba music (see Section 2), and focus on systematic deviations at the 16th-note level. That is, we consider quarter-note-long patterns and seek whether systematic timing deviations occur around each of the four 16th-note beats.

There is a number of computational approaches to the study of performers' timing expressiveness [Widmer, 2002]. The work of [Bilmes, 1993] is of special interest here. He analyzes percussion-based audio data,[1] introduces the notion of "tatum" (fastest metrical pulse) and focuses on deviations with respect to this level of the metrical hierarchy. He proposes a semi-automatic[2] transcription system relying on onset detection and stroke classification, then, the metrical position of each stroke is determined and timing deviations are computed. Similar phrases are then clustered and systematic deviations can be

[1]with a separate audio track for each instrument
[2]i.e. with complete knowledge of the metrical structure

induced via machine learning algorithms. The learned microtiming deviations can then be applied to quantized phrases in order to generate expressive musical phrases. Synthesis is done via the triggering of isolated percussion samples.

[Wright and Berdahl, 2006] propose a system for percussion-based MIDI data. Their system learns deviations from quantized positions for 9 different Brazilian rhythms (none of them being Samba de Roda) via diverse machine learning algorithms and then apply these learned patterns to quantized data, with satisfactory results.

Finally, some authors proposes algorithms for determining the swing of audio signals, e.g. [Laroche, 2001]. Additionally, the swing of such signals can be modified by time-scaling techniques [Gouyon et al., 2003, Janer et al., 2006].

The paper is structured as follows. First we provide details of the data used for experiments. We then propose an algorithm to highlight patterns of microtiming deviations in quarter-note-long segments. We then discuss some findings and propose lines of future work.

## 2. Data

For these preliminary experiments, we collected a relatively small number of audio excerpts, namely 49, of length ranging between around 10 to 30 s. Audio data were ripped from commercial CDs to 44.1 kHz mono. These excerpts are representative of traditional Samba music in the particular style of Rio de Janeiro's "Samba de Roda," with acoustic guitar, four-stringed small Brazilian guitar (i.e. "cavaquinho"), and a percussion section (tambourine —i.e. "pandeiro"—, friction drums, etc.), following a characteristic duple rhythm with second and fourth beats in a bar often marked by a low-frequency percussion sound. In this style of music, it is very common that the tambourine and "cavaquinho" follow a rhythmic pattern at the 16th-note level. Artists and bands are Teresa Cristina & Grupo Semente (albums "A vida me fez assim", "A música de Paulinho da Viola" vol. 1 & 2 and "O mundo é o meu lugar"), Renascença Clube (album "Samba do trabalhador"), Velha guarda da Portela (album "Tudo azul"), Grupo Fundo do Quintal (album "Seja sambista também"), Elton Medeiros, Nelson Sargento and Galo Preto (album "Só Cartola") and Paulinho da Viola and Elton Medeiros (album "Samba na madrugada").

There are between 15 to 73 beats per excerpt (e.g. around 5 to 18 bars per excerpt), reaching a total number of 1803 beats.

## 3. Algorithm

In order to discover *systematic* patterns of deviations with respect to quantized positions, we make use of an algorithm to compute rhythmic patterns inspired from [Dixon et al., 2004]. Among other differences, detailed below, we focus on patterns at a different level of the metrical hierarchy (quarter-notes instead of bars), use a different beat tracking algorithm and use a different signal representation (complex spectral difference instead of amplitude envelope).

### 3.1. Beat tracking

Studying timing patterns in quarter-note segments requires knowledge of individual beats at this level. We segmented the data with the use of the semi-automatic software described in [Gouyon et al., 2004]. When necessary, we manually oriented tracking towards quarter-note level.

## 3.2. Rhythmic patterns

### 3.2.1. Onset detection function

Audio data is processed into a representation of lower dimensionality highlighting note onsets. Instead of using the signal amplitude envelope, as in [Dixon et al., 2004], we chose to use one of the onset detection functions proposed in [Bello et al., 2004]: the "complex spectral difference" (the spectral difference between consecutive signal frames computed in the complex domain, i.e., accounting for magnitude and phase). Frames are 23.2 ms-long and hop size is set to 11.6 ms. The resulting sampling frequency is $44100/512 = 86.1$ Hz. See an example in Figure 1.

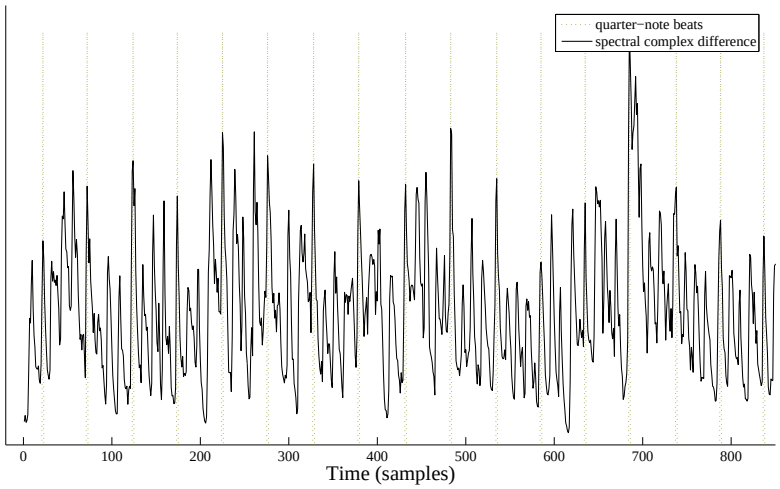



**Figure 1: Example of complex spectral difference, and quarter-note beats (beginning of the excerpt "Tive Sim" from the album "Só Cartola" by Elton Medeiros, Nelson Sargento and Galo Preto).**

### 3.2.2. Beat recentering

Beat positions are then slightly corrected (automatically) so that they would correspond precisely to note onsets. For this, we use a tolerance window of 50 ms around beats and reset beats to the closest maximal onset in this window.[3]

### 3.2.3. Resampling and normalization

As written above, in the data we use, tempi are roughly constant. However, some slight differences can appear in Inter-Beat Intervals (IBIs). In order to be able to accurately compare patterns in quarter-note segments of slightly different lengths, we must resample data in each segment so that they would all have the exact same length. We chose to resample to 40 points per quarter-note segment, using a polyphase implementation.

Segment amplitudes are then normalized to unity.

[3]The length of the tolerance window is not critical.

### 3.2.4. Extraction of typical quarter-note segment patterns

One way of computing the typical quarter-note pattern is to compute the average, for each point in the segments, over all $1803 - 49 = 1754$ segments. Figure 2 shows an illustration of this average pattern together with some individual patterns (randomly selected).

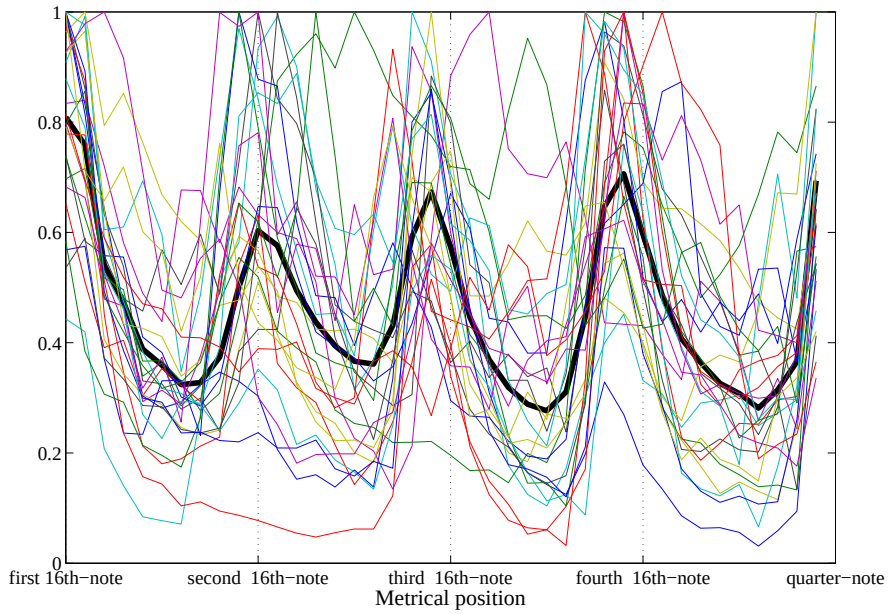



**Figure 2: 27 randomly selected patterns, and average pattern (thick line).**

However, we can expect some patterns to be outliers (as e.g. those corresponding to quarter-note segments in an excerpt's introduction, or fill-ins). We therefore remove outliers by clustering patterns with a $k$-means algorithm, as in [Dixon et al., 2004].[4] Figure 3 illustrates three of the typical patterns in our data, obtained by $k$-means clustering. They account for 35%, 31% and 34% of the patterns, respectively.

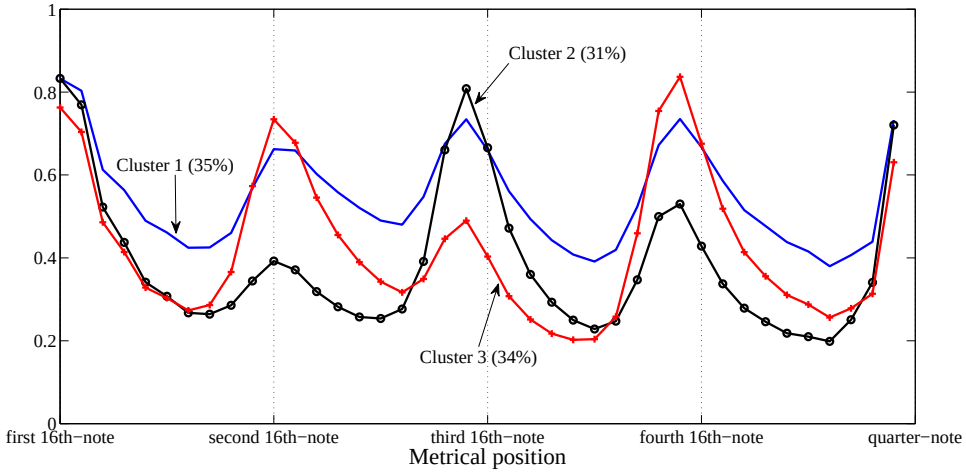



**Figure 3: Three patterns obtained via $k$-means clustering (with $k = 3$).**

[4] with $k = 3$, we used Weka for this, see `http://www.cs.waikato.ac.nz/ml/weka` and [Witten and Frank, 2000].

## 4. Discussions

We can see in Figures 2 and 3 that most of the patterns show local maxima around each of the four 16th-note beats in a quarter-note. This corresponds to the fact that in Samba, there is usually an explicit metrical grid at the 16th-note level, mostly set by the tambourine and other percussive instruments, or harmonic-percussive instruments, as the "cavaquinho."

We can also see in Figure 3 that there are three typical patterns in our data: one with accents (by "accent," we mean local maxima in the onset detection function) on all 16th-notes (cluster 1, where the onset detection function has similar amplitudes around all 16th-notes), one with accents on the first and third 16th-notes (cluster 2), and one with accents on the first, second and fourth 16th-notes (i.e. cluster 3).

We can retrieve representative instances of these clusters by correlating clusters with all quarter-note segments in the data. For instance, the waveform and spectrogram[5] of the most representative instance of cluster 3 (with accents mostly on the first, second and fourth 16th-note, this is especially clear on the waveform) is shown in Figure 4.

### 4.1. Systematic deviations

More interestingly, we can also see, in the average pattern, and even more so in the typical patterns obtained by $k$-means clustering, that both the third and fourth 16th-note beats are slightly *ahead* of their corresponding quantized positions. Notes seem to be played typically (on average) slightly before their quantized positions, with an advance of around 1/40 of the IBI. This corresponds to almost 20 ms at a tempo of 90 BPM (typical for this kind of music).

We can see in Figure 4 and example of such a pattern, where the third and fourth 16th-notes are played almost 30 ms ahead of their quantized positions.

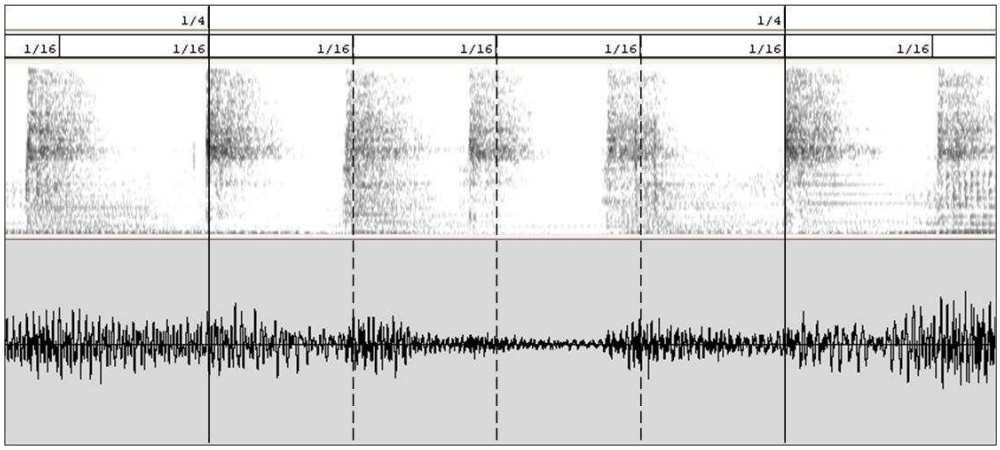



**Figure 4: Illustration, from bottom to top, of the audio waveform, spectrogram and metrical structure at the 16th-note and quarter-note levels for one of the 1754 quarter-note-long patterns in our data (i.e. one of the patterns in the excerpt "Alvorada" from the album "Só Cartola" by Elton Medeiros, Nelson Sargento and Galo Preto). This pattern is typical of the *strong-strong-weak-strong* form found in our data (i.e. cluster 3). Note also the timing shift of the third and fourth 16th-notes, slightly ahead of their corresponding quantized positions.**

## 5. Future work

The work reported here should be continued and cross-validated by further experiments with a larger dataset. A complementary avenue for future work would be to purchase

[5] frequencies are represented on a linear scale from 0 to 22050 Hz

MIDI-matched audio data in the same music style, this would open the way to a finer precision in the analysis which would be necessary to capture subtle microtiming differences in different quarter-note segments, or different measures [Wright and Berdahl, 2006].

It would also be interesting to study the sensitivity of these findings to the number of clusters. Further experimentations with different machine-learning algorithms, or different front-ends (e.g. spectral centroid normalized by energy as suggested in [Paulus and Klapuri, 2002]), could also be of interest. In complement, one could also explore outliers properties.

In these experiments, we purposely normalized the length of each quarter-note pattern. It would be interesting in further experiments to study a possible dependency of the found deviations with respect to tempo (as has been demonstrated in the particular case of swing [Friberg and Sundström, 2002]).

A cross-disciplinary effort could also be done in comparing findings reported here to Samba-specific musicological literature, e.g. [Sandroni, 1996].

There are diverse applications to the findings reported here. First, one might think of taking advantage of these characteristic features for music similarity and music genre classification [Dixon et al., 2004]. Second, it would also be interesting to develop a system for music transformation where such deviations could be explored (either augmented or subtracted). This would permit to study their perceptual relevance. This would also open the way to change the expressiveness of quantized data, and make it sound "more human" [Bilmes, 1993, Gouyon et al., 2003, Janer et al., 2006, Wright and Berdahl, 2006, Widmer, 2002, Ramirez and Hazan, 2006]. Finally, further studies could be dedicated to the relation between microtiming deviations and perception of "Groove" in Samba, as well as in other music styles.

## Acknowledgments

Carioca da Gema, Claudia Vargas, Guy Madison, Simon Dixon and Kalle Hörnström. Thanks to two anonymous reviewers for their insightful comments.

## References

Bello, J., Duxbury, C., Davies, M., and Sandler, M. (2004). On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 11(6):553–556.

Bilmes, J. (1993). *Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm*. Master Thesis, MIT, Cambridge.

Desain, P. and Honing, H. (1991). Tempo curves considered harmful. A critical review of the representation of timing in computer music. In *Proc. International Computer Music Conference*.

Dixon, S., Gouyon, F., and Widmer, G. (2004). Towards characterisation of music via rhythmic patterns. In *Proc. International Conference on Music Information Retrieval*, pages 509–516.

Friberg, A. and Sundström, J. (2002). Swing ratios and ensemble timing in jazz performances: Evidence for a common rhythmic pattern. *Music Perception*, 19(3):333–349.

Gouyon, F., Fabig, L., and Bonada, J. (2003). Rhythmic expressiveness transformations of audio recordings: Swing modifications. In *Proc. Digital Audio Effects Conference*.

Gouyon, F., Wack, N., and Dixon, S. (2004). An open-source tool for semi-automatic rhythmic annotation. In *Proc. International Conference on Digital Audio Effects*, pages 193–196.

Janer, J., Bonada, J., and Jordà, S. (2006). Groovator — An implementation of real-time rhythm transformations. In *Proc. 121st Convention of the Audio Engineering Society*.

Laroche, J. (2001). Estimating tempo, swing and beat locations in audio recordings. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 135–138.

Paulus, J. and Klapuri, A. (2002). Measuring the similarity of rhythmic patterns. In *Proc. International Conference on Music Information Retrieval*.

Ramirez, R. and Hazan, A. (2006). A tool for generating and explaining expressive music performances of monophonic jazz melodies. *International Journal on Artificial Intelligence Tools*, 15(4):673–691.

Rosinach, V. and Traube, C. (2006). Measuring swing in irish traditional fiddle music. In *Proc. International Conference on Music Perception and Cognition*, pages 1168–1171.

Sandroni, C. (1996). Mudanças de padrão rítmico no samba carioca, 1917-1937. *Revista Transcultura de Música*, 2.

Widmer, G. (2002). Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research*, 31(1):37–50.

Witten, I. and Frank, E. (2000). *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco.

Wright, M. and Berdahl, E. (2006). Towards machine learning of expressive microtiming in Brazilian drumming. In *Proc. International Computer Music Conference*.

# Posters

11º Simpósio Brasileiro de Computação Musical

SBCM

11th Brazilian Symposium on Computer Music

# O Theremin Virtual: Usando Dispositivos de Realidade Virtual em Experimentos Musicais

**Marcelo Soares Pimenta, Evandro Manara Miletto, Carlos Augusto Dietrich, Luciana Nedel**

Laboratório de Computação & Música e Laboratório de Computação Gráfica
Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{mpimenta,miletto,cadietrich,nedel}@inf.ufrgs.br

***Abstract.*** *This paper describes the design, implementation and experimental tests of the Virtual Theremin, a device for musical expression. In this work we propose the assembling of the Virtual Theremin using a data glove and a magnetic motion tracker for the user interaction. The sound generation is provided integrating these devices with a programming environment (MaxMSP) The features of this project and the consequences of the use of a greater quantity of degrees of freedom are briefly discussed.*

***Resumo.*** *Este artigo descreve o projeto, implementação e experimentos do Theremin Virtual, um instrumento para expressão musical. Neste trabalho propõe-se a construção de um Theremin Virtual utilizando-se uma luva de dados e um rastreador de movimento magnético, sendo que a geração de som é feita através da integração destes dispositivos com o ambiente MaxMSP. Características do projeto e conseqüências de um aumento significativo de graus de liberdade e possibilidades de uso são resumidamente discutidos.*

## 1. Introdução

O objetivo deste artigo é descrever como foi construído o *Theremin Virtual* e quais os resultados alcançados nos experimentos sonoros e musicais a partir deste dispositivo.

O Theremin virtual resulta da combinação de um dispositivo de rastreamento de posições e uma luva de dados virtual usados em realidade virtual para o mapeamento de gestos do usuário para sons. Investigou-se as conseqüências do aumento substancial de graus de liberdade do Theremin (originalmente com 2 graus de liberdade para modificar tom e volume) e dos possíveis diferentes mapeamentos entre gestos e sons.

Como objetivos secundários, desejava-se também (a) investigar a ampliação do número de graus de liberdade para usuários especializados e avaliar a combinação entre eles, bem como as facilidades e complicações decorrentes deste aumento de possibilidades como forma de expressividade sonora/musical; e (b) realizar experimentos e avaliações com os dispositivos e com as soluções desenvolvidas visando confirmar ou refutar as combinações definidas.

## 2. Theremin Virtual

O Theremin virtual pode ser conceitualmente dividido em três componentes: a interface com o usuário, o dispositivo de síntese de som e o mapeamento lógico entre a interface e o dispositivo de síntese de som. O objetivo dos dois primeiros componentes é proporcionar ao usuário um ambiente de execução semelhante ao do próprio Theremin, reproduzindo as características do aparelho original. Já o mapeamento entre estes componentes traz a possibilidade de novos experimentos musicais/interativos, pois novos tipos de mapeamentos entre gestos e sons (ou filtros de sons) podem ser explorados.

No Theremin Virtual, a posição da mão do usuário é capturada através da combinação de dois dispositivos comumente utilizados em RV: uma luva de dados e um dispositivo de rastreamento de posição. A luva de dados utilizada é o modelo Data Glove 5, da empresa 5DT. Ela disponibiliza cinco sensores de curvatura dos dedos, além de dois sensores de rotação. O rastreador de posição foi fixado sobre a luva de dados, formando um dispositivo capaz de rastrear a posição de cada dedo do usuário (Figura 1). O rastreador de posição informa a translação (deslocamento e rotação) da mão do usuário, e a luva de RV informa a curvatura de cada dedo A, B, C e D.

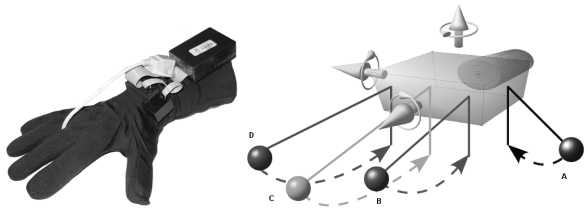



**Figura 1. Modelo da interface proposta para o Theremin Virtual.**

Para viabilizar a comunicação dos dispositivos de RV com o Max/MSP (ambiente utilizado para síntese sonora) foram criados três objetos específicos para este experimento, o *glove* (envio e recebimento de mensagens ao *driver* do dispositivo Data Glove 5), o *FOB* e o *save2file*. Estes objetos são implementados na forma de *plugins* para o ambiente, através da interface definida pela API Max/MSP. A interface de cada *plugin* Max/MSP define apenas um procedimento de inicialização, chamado *main*. Neste procedimento são fixados os *callbacks* que irão responder as entradas/saídas do *plugin*, assim como para a sua criação e destruição. *Main* é chamado no momento em que o objeto é inserido pelo usuário dentro de um *patch* (programa), ou um *patch* previamente salvo é aberto dentro do ambiente Max/MSP. Os *callbacks* do *plugin* são então registrados, e passam a interagir com os demais objetos do ambiente.

O objeto implementado recebe dois parâmetros no momento de sua criação: o número da porta serial ao qual o dispositivo está conectado e o intervalo de tempo entre as requisições de dados. O usuário também pode fixar um terceiro parâmetro opcional, que determina o intervalo no qual os dados capturados do dispositivo são normalizados. Caso este parâmetro não seja informado, os dados fornecidos pelo objeto *glove* variam entre 256 valores possíveis no intervalo [0, 1], onde o valor 1 indica a curvatura máxima de um dedo. O objeto *FOB* foi criado para enviar e receber mensagens ao *driver* do dispositivo *Flock of Birds*. Este dispositivo também é ligado à interface serial, portanto recebe os mesmos parâmetros de configuração do objeto *glove*. Neste objeto os dados de

saída indicam a posição e a rotação no espaço em relação a origem do dispositivo, medidas em centímetros e graus, respectivamente. O fluxo de dados do objeto FOB é o seguinte (ver Figura 2).

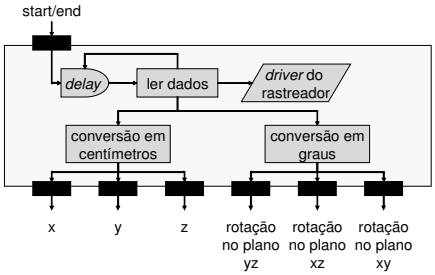



**Figura 2. Fluxo de dados no objeto *FOB*.**

O objeto lê e converte continuamente os dados disponibilizados pelo rastreador de posição, em intervalos de tempo fixados pelo usuário. Os dados coletados nos experimentos são armazenados em arquivos de texto através do objeto *save2file*. O objetivo deste objeto é manter um registro dos primeiros *n* dados do experimento (posição das mãos do usuário e parâmetros de som controlados). Os dados armazenados são gravados em arquivo no término da execução do ambiente.

Mapeou-se, como no original, a variação da tonalidade (pitch) de acordo com o movimento horizontal da mão (pra frente - mais agudo, ou pra trás – mais grave, mais ou menos próximo da base do rastreador) e a variação da amplitude (volume) de acordo com o movimento vertical da mão (para cima – mais volume, ou para baixo, menos volume).

## 3. Experimentos

Um aspecto da avaliação realizada com o Theremin Virtual tentou provar a fidelidade do instrumento virtual ao original. Uma das questões a ser respondida era a seguinte: *é possível utilizar o Theremin Virtual como instrumento musical para a obtenção de notas musicais da escala temperada (as 7 notas musicais e seus acidentes) com intensidades determinadas (volume do som quantificável)*?

Nos experimentos realizados usando o ambiente Max/MSP (Cycling'74 2007), cada usuário operava sobre a interface e sobre o ambiente, ajustando os parâmetros quando fosse necessário. A avaliação da performance do usuário considerou os resultados gravados pelo ambiente (log de uso) e a opinião do usuário, relatada em protocolo concorrente (verbalização simultânea) durante a realização do experimento e em entrevistas realizadas imediatamente após o experimento.Considerando os diferentes perfis específicos dos usuários necessários para uma avaliação relevante, optou-se – além de três usuários leigos em música envolvidos no projeto - pela participação adicional de dois músicos com conhecimentos profundos de instrumentos eletrônicos e tecnologia musical.

Os principais problemas relatados pelo usuários foram: a) a falta de precisão dos dispositivos (o rastreador opera baseado na variação de um campo magnético, que sofre constantes interferências do ambiente - a luva de dados também mostra uma variação significativa em suas medições de curvatura, resultante de suas limitações tecnológicas),

b) a troca de mensagens entre o ambiente Max/MSP e os *drivers* dos dispositivos (introdução de um *overhead* significativo no processamento de áudio), c) a falta de um referencial físico (fundamental para a qualidade sua interpretação) e d) a limitação dos intervalos de freqüências possíveis nos mapeamentos adotados pelo instrumento (proveniente da parametrização do ambiente Max/MSP). Os usuários sugeriram que o ambiente poderia ter uma flexibilidade maior ao permitir a associação de várias amostras sonoras ao dispositivo de síntese de som.

Esta avaliação dos usuários enfatizou a grande flexibilidade e capacidade de expressão da interface proposta para o Theremin Virtual devido ao grande número de graus de liberdade. Na opinião de ambos, a associação de gestos a parâmetros de síntese e controle sonoros possibilita a realização de um grande número de tarefas, limitadas apenas pela criatividade do performer. Paradoxalmente, esta é exatamente uma das dificuldades maiores apontadas pelos usuários leigos que não conhecem nem usam adequadamente os diferentes parâmetros a serem controlados e acabam não aproveitando esta potencialidade.

## 4. Considerações finais

A idéia usar o Theremin Virtual como '*instrumento musical para a obtenção de notas musicais da escala temperada* foi refutada de acordo com o mapeamento proposto. No entanto, em face de outras possibilidades de aplicação sugeridas a partir das opiniões e observações dos usuários especialistas, pretendemos continuar com muitos experimentos futuros.

Alguns dos maiores problemas a serem resolvidos envolvem por exemplo novos algoritmos para tratamento das informações processadas pelo ambiente Max/MSP. Este processamento ocasiona um pequeno atraso em termos de *feedback* sonoro, o que é significativo e determinante em termos de performance musical.

Pretende-se também investigar composição musical interativa através da manipulação de amostras sonoras através de gestos do usuário. Esperamos relatar os resultados em futuras publicações.

## Bibliografia

Cycling74, Max/MSP. 2007. Disponível em http://www.cycling74.com/index.html

Hunt, A., Wanderley, M. M., and Kirk, R. 2000. Towards a model for instrumental mapping in expert musical interaction. In In Proceedings of the 2000 International Computer Music Conference, International Computer Music Association, 209–212.

Preece, J. et alli. Design da Interação , Bookman, 2004.

Serafin, S., Dudas, R., Wanderley, M. M., and Rodet, X., 1999. Gestural control of a real-time physical model of a bowed string instrument. In Proceedings of the International Computer Music Conference – ICMC99. Beijing, China.

Wanderley, M. , Depalle, P. Gestural Control of Sound Synthesis. Proceedings of IEEE, vol 92, N. 4 (April), Johannsen, G. (ed), Special Issue on Engineering and Music – Supervisory Control and Auditory Communication, pp. 632-44, 2004.

Wanderley, M.; Orio, N. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. Computer Music Journal, MIT . 1 September 2002, vol. 26, no. 3, pp. 62-76.

# Sistema Adaptativo de Reescrita Musical Estocástica

**Rafael Baquini Bueno, Ricardo Luis de Azevedo da Rocha**

Escola Politécnica – Universidade de São Paulo (USP)
Av. Prof. Luciano Gualberto. Trav. 3 N. 158 - 05508-900 - São Paulo - SP - Brasil
rafael.bueno@poli.usp.br, luis.rocha@poli.usp.br

***Abstract.*** *This paper describes a musical composing system based on stochastic rewriting rules that can change during computational steps. This rewriting allows the music composing to change the three main sound properties: pitch, duration and intensity. The rules composition can be changed to fit the different musical styles.*

***Resumo.*** *Este artigo visa descrever um sistema de composição musical baseado em regras estocásticas de reescrita, que são alteradas em tempo de execução. Esta reescrita permite a composição de música alterando as três principais propriedades do som: altura, duração e intensidade. A composição das regras pode ser alterada de maneira a se adequar aos diferentes estilos musicais.*

## 1. Introdução

A composição musical pode ser vista como um processo baseado em regras discretas e bem comportadas. Uma linha melódica trabalha a manipulação de algumas propriedades do som tais como altura, duração ou ritmo e intensidade, e estas devem ser levados em conta durante a composição. Este estudo se baseia no Sistema Tonal, assim, também é preciso levar em conta a relação que os diferentes sons apresentam entre si.

Desta forma, pode-se utilizar um modelo computacional para auxiliar neste processo, simulando a estrutura musical e gerando estocasticamente linhas melódicas estruturadas, através da manipulação das propriedades do som acima citadas.

## 2. Modelagem Musical

Uma melodia pode ser definida por um conjunto de compassos. A estruturação deste conjunto depende de aspectos como a Escala escolhida, o Estilo desejado para a música, e o tamanho total ou número de compassos. Aqui, é preciso definir, também, um Fator de Substituição, que controla o número de iterações no processo de reescrita.

Cada compasso é composto por um acorde principal, a partir do qual são geradas as demais notas; da sua duração e de um conjunto de notas. Para simplificação do modelo, adota-se que, cada compasso é formado por apenas um acorde, que será a base de substituição do compasso.

Uma nota é definida por suas propriedades sonoras: duração, intensidade, tom e altura. Além disso, define-se um campo de Taxa de Substituição (TS), que determina a probabilidade de uma nota ser substituída durante o processo de reescrita. Assim, as três estruturas definidas ficam como a seguir:

**Quadro 1 - Estrutura da música**

| Atributo | Descrição |
|---|---|
| Escala | Determina a tonalidade da música |
| Estilo | Define o conjunto de regras de substituição a ser utilizado |
| Tamanho | Determina o número de compassos |
| Fator de Substituição | Indica a diminuição na taxa de substituição a cada iteração |
| Compassos | Seqüência de compassos que compõem a música |

**Quadro 2 - Estrutura de um compasso**

| Atributo | Descrição |
|---|---|
| Acorde | Acorde central do compasso |
| Duração | Duração total do compasso, em quantidade de semifusas |
| Notas | Seqüência de notas que compõem o compasso |

**Quadro 3 - Estrutura de uma nota**

| Atributo | Descrição |
|---|---|
| Duração | Duração da nota, medido em quantidade de semifusas |
| Tom | Determina o tom/semitom da nota. Indica se é Pausa |
| Altura | Complemento do Tom. Indica em qual Oitava a nota é tocada |
| Intensidade | Indica a intensidade da nota |
| TS | Taxa de Substituição da nota |

### 2.1. Geração da Estrutura Inicial

Inicialmente, devem-se escolher o estilo musical, o tamanho da música, o fator de substituição e definir a escala, ou tonalidade, da música.

Em seguida, é preciso verificar a cadência para determinar os próximos compassos. Aqui, são utilizadas algumas regras musicais, referente à Cadência, para obter o acorde dos demais compassos estocasticamente a partir do estilo e tamanho desejados para a música. Neste momento, cada compasso é preenchido apenas com uma nota de altura igual à Tônica do acorde e de duração igual ao tamanho do compasso.

Após isso, é permitido ao usuário manipular a estrutura, adicionando novos compassos, ou alterando os existentes. Esta manipulação pode ser usada para gerar estruturas como refrões ou introduções.

## 3. Reescrita Musical

O processo de reescrita utiliza regras de substituição, que se baseiam em métodos adaptativos [Neto 1994] para alterar a estrutura da cadeia (seqüência de notas). Estas regras podem ser divididas em três tipos: rítmica, estrutural e de combinação.

As regras rítmicas são aquelas que alteram a duração das notas, dividindo-as em notas de menores durações, gerando pausas e manipulando a intensidade dos sons. As regras estruturais são aquelas que substituem os tons das notas por outros tons relativos, como terças, quintas ou outras, de acordo com o acorde do compasso. As regras de combinação são aquelas que ligam duas notas semelhantes em uma nota só.

Também é necessário estabelecer um critério para uso destas regras. Aqui, é utilizado o mesmo conceito de Taxa de Substituição aplicado às notas, para definir a probabilidade de uma regra ser utilizada.

Definem-se por TSr, TSe e TSc, respectivamente, as taxas de substituição para as regras rítmicas, estruturais e de combinação. Cada uma dessas taxas possui um valor entre 0 e 1, de maneira que a soma das três totalize 1. Com esta limitação, somente uma regra é utilizada a cada substituição. Em seguida, determina-se a TS para cada regra de substituição. Desta maneira, pode-se compor de um Estilo Musical, que é representado por um conjunto de regras de substituição definidas com seus respectivos pesos.

Definida a estrutura inicial da melodia, a reescrita é um processo iterativo que ocorre em cada compasso de uma vez, como descrito pelo seguinte algoritmo:

Algoritmo 1: Reescrita de um compasso

```
      alteração = 1;
      Enquanto (alteração = 1)
Laço:    Enquanto (Compasso não terminou)
            Nota = Próxima_Nota();
            x = random1[0-1]; y = random2[0-1]; z = random3[0-1];
            Se (x>Nota.TS)
               Nota.TS = 0; Vá para Laço;
            Se (y<=TSr)
               Substituição = Ritm; Regra = Escolhe(RegrasRítmicas[],z);
            Senão Se (y<=TSr+Tse)
               Substituição = Estr; Regra = Escolhe(RegrasEstruturais[],z);
            Senão
               Substituição = Comb; Regra = Escolhe(RegrasCombinação[],z);
            Substitui(Nota, Regra);
            Nota.TS = Nota.TS * Fator de Substituição
         Fim Enquanto
         Se (Não houve substituição)
            alteração = 0;
      Fim Enquanto
```

O processo deve ser repetido até que se atinja a estabilidade. Pode-se entender esta estabilidade pela não-ocorrência de substituições em uma dada iteração. A estabilidade acontece quando os atributos TS das notas se tornam reduzidos o suficiente, diminuindo as ocorrências de substituições. Uma vez que um compasso atingiu a estabilidade, ele é encerrado e a operação passa ao compasso seguinte.

### 3.1. Regras de substituição rítmica

As regras rítmicas substituem a estrutura rítmica da nota, fazendo sua fragmentação em notas menores, gerando pequenas alterações no atributo Intensidade, ou criando pausas. As regras rítmicas podem ser representadas pela seguinte estrutura:

$$( Nota_0\ (Dur_0,Tom_0,Alt_0,Int_0,TS_0) \rightarrow \{ Nota_1\ (Dur_1,Tom_1,Alt_0,Int_1,TS_1), Nota_2\ (Dur_1,Tom_2,Alt_0,Int_2,TS_2), \ldots Nota_n\ (Dur_1,Tom_n,Alt_0,Int_n,TS_n) \}, TS_r )$$

Onde:

$Dur_0 = n * Dur_1$

$Tom_x \in \{Tom_0, Pausa\}, 1 \leq x \leq n$

$TS_x = 0$, se $Tom_x = Pausa$; $TS_x = TS_0$, se $Tom_x = Tom_0$

$Int_x \in \{Int_0 - 1, Int_0, Int_0 + 1\}, 1 \leq x \leq n$

$TS_r$ é a taxa de substituição da regra

### 3.2. Regras de substituição estrutural

As regras estruturais alteram o tom da nota por outro tom do acorde, levando em conta, principalmente, o estilo de música escolhido. Como a Altura se relaciona diretamente com o tom da nota, as regras estruturais também permitem a alteração deste atributo.

$$( Nota_0 (Dur_0,Tom_0,Alt_0,Int_0,TS_0) \rightarrow Nota_1 (Dur_0,Tom_1,Alt_1,Int_0,TS_0), TS_r )$$

Onde:

$Tom_1$ = Função($Tom_0$, Acorde)
$Alt_1 \in \{Alt_0 - 1, Alt_0, Alt_0 + 1\}$
Função(Tom, Acorde) é uma função que devolve uma nota relativa ao tom, dentro do Acorde especificado.

### 3.3. Regras de substituição de combinação

As regras de combinação são utilizadas para unir duas notas consecutivas que tenham o mesmo tom e altura, criando uma única nota, de mesmo tom e altura e com duração igual à soma das anteriores. Para isso, é necessário fazer a comparação com a nota imediatamente anterior. Utilizando um Autômato Finito Adaptativo, podemos fazer de forma natural esta verificação. [Neto e Pariente 2002]

$$(\{Nota_0 (Dur_0,Tom_0,Alt_0,Int_0,TS_0), Nota_1 (Dur_1,Tom_0,Alt_0,Int_1,TS_1)\} \rightarrow Nota_2 (Dur_2,Tom_0,Alt_0,Int_2,TS_2), TS_r)$$

Onde:

$Dur_2 = Dur_0 + Dur_1$
$Int_2 \in \{ Int_0, Int_1\}$
$TS_2 \in \{ TS_0, TS_1\}$

## 4. Conclusões

Muitos estudos sobre computação evolutiva aplicada a música se restringem a apenas algumas das propriedades do som, como apenas a altura [Neto e Basseto 1999], a estrutura rítmica [Gimenes, Miranda e Johnson 2005]. Este trabalho propôs a criação de um sistema compositor de melodias utilizando regras estocásticas e adaptativas, que trabalhe as três propriedades do som simultaneamente, e permita a composição automática de linhas melódicas baseadas em regras pré-definidas, utilizando os padrões de linguagem musical conhecidos.

## Referências

[Neto 1994] Neto, J. J. “Adaptive Automata for Context - Sensitive Languages” SIGPLAN NOTICES, Vol. 29, n. 9, pp. 115-124, September, 1994.

[Neto e Pariente 2002] Neto, J. J. and Pariente, C. A. B. “Adaptive Automata - a Revisited Proposal” CIAA 2002, 2002, Vol.2608, July 3-5, Springer-Verlag, 2002, pp. 158-168

[Neto e Basseto 1999] Neto, J. J. e Basseto, B. A. “A Stochastic Musical Composer Based on Adaptative Algorithms.” SBC&M99, Rio de Janeiro, 1999.

[Gimenes, Miranda e Johnson 2005] Gimenes, M., Miranda, E. R. e Johnson C. “A Memetic Approach to the Evolution of Rhythms in a Society of Software Agents” SBCM, 10º Simpósio Brasileiro de Composição Musical, pp. 13-23, Outubro, 2005.

# Síntese de Imagens Controladas por Áudio

**Mariana Zaparolli Martins, Marcelo Queiroz**

Dep. de Ciência da Computação – Universidade de São Paulo (USP)
R. do Matão, 1010 – Cid. Universitária – 05508-090 – São Paulo - SP - Brasil

mzm@ime.usp.br, mqz@ime.usp.br

***Abstract.*** *In this paper a Pd object library is described which combines audio analysis and image synthesis tools for generating visual accompaniments to musical input data. The user establishes connections that determine how musical parameters affect the synthesis of graphical objects, and controls these connections in real-time during performance. The library combines a straightforward communication protocol for exchanging musical and visual parameters with an easy-to-use interface which makes it accessible for users with no computer programming experience. Its potential applications areas include children's musical education and entertainment industry.*

***Resumo.*** *Neste artigo é apresentada uma biblioteca de objetos Pd que combina ferramentas de análise de áudio e síntese de imagens para a geração de acompanhamentos visuais a uma entrada musical. O usuário estabelece conexões que determinam como os parâmetros musicais afetam a síntese de objetos gráficos, e controla estas conexões em tempo-real durante a performance. A biblioteca combina um protocolo de comunicação para intercâmbio de parâmetros musicais e visuais com uma interface fácil de usar, tornando-a acessível a usuários sem experiência em programação de computadores. Suas áreas possíveis de aplicação incluem a musicalização infantil e a indústria de entretenimento.*

## 1. Introdução

Este trabalho tem como objetivo descrever um conjunto de objetos para a ferramenta Pd (Puredata, ver [Puckette 1998]) que possibilite de forma simples e interativa a criação e modificação de associações entre informações de áudio e de objetos gráficos, a fim de permitir a criação de acompanhamentos visuais para uma entrada musical. Entende-se por acompanhamento visual uma animação (em vídeo) que corresponde a uma síntese de imagens onde a definição dos parâmetros que controlam os objetos gráficos utiliza informação musical extraída automaticamente da entrada de áudio e atualizada em tempo-real.

Recentemente, o aumento do poder de processamento dos computadores pessoais e a facilidade de circulação de informação multimídia em banda-larga fizeram com que muitos sistemas computacionais para associação de áudio e vídeo fossem desenvolvidos [Rowe 2001]. Alguns exemplos destes sistemas são: *Isadora*, *PixelToy, AVS, G-Force, Milkdrop Soundscape, Egosound* e o projeto *Conducting a Virtual Orchestra*.

Em particular, alguns sistemas (também chamados *instrumentos visuais* [Collopy 1999]) estão voltados à geração de imagens e permitem o controle interativo da síntese em tempo-real. Um destes instrumentos visuais é o *GEM* [Danks 1997], que utiliza os conceitos do *OpenGL* e fornece objetos Pd para síntese de imagens a partir da descrição de uma cena 3D.

Na próxima seção são discutidos alguns conceitos envolvidos na especificação dos objetos, com ênfase no protocolo de comunicação entre as camadas de processamento de áudio e síntese de imagens. Em seguida são apresentados brevemente os objetos implementados e por fim são feitas algumas considerações sobre aplicações e trabalhos futuros.

## 2. Entre som e imagem

No projeto de objetos gráficos que respondem a uma entrada musical, deve-se considerar em primeiro lugar que tipo de informação será extraída da entrada, levando em conta as restrições impostas pelo requisito de operação em tempo-real. A análise espectral utilizando janelas pequenas (*Short-Time Fourier Transform*) permite a obtenção de diversas características sonoras, tais como freqüência predominante (no caso de sinais quasi-periódicos) e padrões de espalhamento ou concentração de energia em diversas bandas de freqüência. A partir da envoltória dinâmica é possível identificar ataques. Da obsevação destas análises ao longo do tempo é possível detectar *glissandos*, *crescendos* e *descrescendos*, padrões rítmicos e mudanças de andamento.

A síntese de imagens por computador pode ser feita de inúmeras maneiras, sendo uma destas a construção da imagem a partir da descrição dos elementos que compõem uma cena tridimensional (*renderização),* tais como objetos, fontes de luz e câmeras. Os parâmetros que controlam estes elementos podem ser de posição, dimensões, cor, iluminação, posição da câmera e atributos da superfície dos objetos, como rugosidade, transparência, espelhamento, entre outros.

Independentemente da escolha feita em relação ao tipo de objeto que será sintetizado, um problema evidente ao se controlar imagens a partir de informações retiradas do áudio é a tradução dos parâmetros musicais em parâmetros visuais. Não se trata aqui de determinar associações entre estes parâmetros, o que evidentemente é um problema estético cuja solução (particular e subjetiva) será dada pelo usuário do sistema, mas em representar tais parâmetros de modo a permitir ao usuário o máximo de liberdade nestas associações. Neste contexto específico, parece natural propor que a tradução entre som e imagem seja intermediada por um universo de representação abstrato, que não seja nem musical nem visual a priori, mas que permita o estabelecimento de conexões diversas entre estes dois universos.

A partir da observação de características elementares do áudio, tais como envelope dinâmico, altura e ataques, parece natural representar tais características como sinais discretos (funções do tempo) com valores variando num intervalo arbitrário [0...1], ou alternativamente como disparos (*bangs*). Essas duas modalidades de representação permitem um mapeamento convencional de parâmetros sonoros que admitem uma ordem linear (tais como altura e intensidade) através de sinais [0...1] bem como a representação de eventos pontuais (tais como ataques ou detecção de notas específicas) através de disparos. Por outro lado, tamanho e posição de objetos gráficos ou cores no sistema RGB também podem ser representadas por valores entre [0...1]; disparos podem ser úteis para sinalizar o início de uma seqüência qualquer de movimentos de um objeto.

Desta maneira, a comunicação entre objetos de análise de áudio e objetos de síntese de imagens é feita através de sinais discretos de controle que variam no intervalo [0...1] ou através de disparos. Cada valor produzido por um objeto de análise de áudio, bem como cada parâmetro de controle de síntese de imagem, pertencem claramente a uma destas duas categorias, e qualquer conexão entre saídas [0...1] e entradas [0...1] ou

entre saídas *bang* e entradas *bang* é possível, e estabelece uma relação causal entre a entrada de áudio e a síntese de imagens.

## 3. A implementação do sistema AIM

A escolha da ferramenta Pd como plataforma-base para processamento em tempo-real se deve a vários fatores, sendo os mais importantes o fato de se tratar de um software livre, multi-plataforma, que conta com objetos nativos para gerenciamento de áudio e MIDI além de inúmeras bibliotecas que estendem sua funcionalidade, como a biblioteca GEM para síntese de imagens. Desta maneira, foi natural planejar o desenvolvimento do sistema como uma coleção de objetos Pd que oferecesse facilidades para a obtenção de resultados visuais de forma simples e rápida a partir de uma entrada musical, incluídos aí objetos para geração de interfaces que pudessem ser facilmente controladas em tempo-real durante a performance.

Esta coleção de objetos foi batizada de AIM (*Audible IMages*) e os objetos foram divididos em quatro categorias: objetos para análise do áudio de entrada, objetos para síntese de imagens, objetos para síntese e processamento de sinais, e interfaces. Um objeto central *(aim.control*) é responsável pela geração da janela GEM e pela entrada de áudio, possuindo controles para a câmera e o fundo de cena, bem como recursos para gerenciamento de microfone e *playlists*.

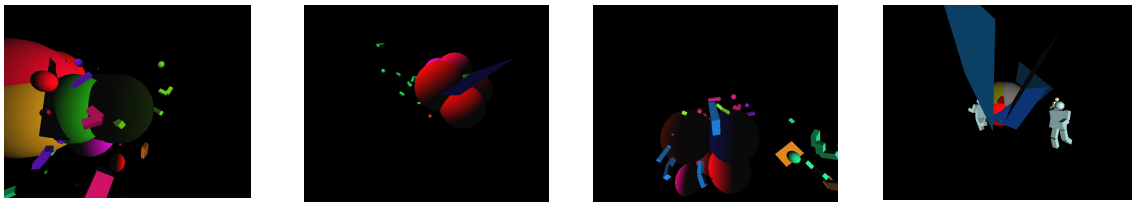
Pertencem à categoria de análise de áudio objetos tais como *aim.pitch* (para altura musical) e *aim.amplitude,* que tanto servem para extrair informações pontuais do sinal como permitem a obtenção de médias ao longo de um período maior de tempo, *aim.attack,* que detecta inícios de notas ou eventos percussivos, *aim.timedensity,* que obtém a densidade temporal de eventos através da contagem do número de ataques por segundo e *aim.spectraldensity,* que faz a análise da densidade espectral através da contagem da proporção de picos espectrais proeminentes.

Atualmente o sistema AIM possui três objetos primitivos e três objetos complexos para a síntese de imagens, todos a partir de figuras geométricas. A título de exemplo serão descritos os três objetos compostos: *aim.balls, aim.squares* e *aim.dolls*. O objeto *aim.balls* produz 28 esferas controladas por 7 parâmetros abstratos, que determinam propriedades distintas (raio, cores, posição) em cada bola. O objeto *aim.squares* controla o lançamento de quadrados que se movimentam em direção ao fundo da cena, sendo seus movimentos e cores controlados independentemente. O objeto *aim.dolls* gerencia o movimento de 6 bonecos formados por paralelepípedos e esferas, oferecendo controles de cor, raio das trajetórias e coesão das partes que compõem os bonecos, permitindo a aproximação visual com os outros objetos de síntese de imagem.

Os objetos para processamento e síntese de sinais permitem modificar um sinal de controle AIM, ou mesmo gerar um sinal de controle a partir de uma descrição formal, e fornecem uma flexibilidade maior nas conexões feitas entre a camada de análise de áudio e a camada de síntese de imagens. Dentro da categoria de objetos para o processamento de sinais estão os objetos *aim.invert*, *aim.chebyshev, aim.compress, aim.expand* e *aim.gate.* Na categoria de síntese de sinais a biblioteca possui os objetos *aim.sine* e *aim.cosine, aim.phasor* e *aim.random.*

As interfaces são objetos da biblioteca AIM que permitem que as conexões entre a camada de áudio e a camada gráfica possam ser alteradas em tempo-real de forma rápida e simples, através de chaves liga-desliga. Uma destas interfaces é o *aim.interface8,* que fornece uma matriz 8x8 de chaves liga-desliga que controlam o fluxo

de 8 sinais [0...1] provenientes de análise de áudio (ou síntese e processamento de sinais), correspondentes às linhas da matriz, para 8 controles de síntese de imagens, correspondentes às colunas. Além do controle direto pelo usuário, a interface também permite a leitura de arquivos contendo configurações das chaves liga-desliga, bem como um modo piloto-automático de geração destas conexões de forma aleatória.



**Exemplos de saida utilizando diversos objetos de síntese de imagem**

## 4. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada a biblioteca *AIM (Audible Images),* cujo objetivo é servir de ferramenta para a exploração de associações entre parâmetros sonoros e parâmetros visuais, permitindo a geração de acompanhamentos visuais a uma entrada de áudio através da constução de patches simples e diretos, podendo ser usado por pessoas sem experiência de programação. Aplicações possíveis dessa biblioteca incluem projeções em festas e casas noturnas bem como o uso pessoal, desenvolvimento de jogos onde os objetos ou personagens relacionam-se com a música de fundo, e como recurso didático na musicalização infantil. O trabalho futuro inclui o desenvolvimento de novos objetos de análise de áudio, síntese de imagens e interfaces, bem como uma avaliação junto a usuários através de questionário, a fim de aprimorar a biblioteca.

## 5. Referências


Collopy, F., Fuhrer, R. e Jamenson, D. (1999) *"Visual Music in a Visual Programming Language"*, In: IEEE 1999 Symposium on Visual Languages, pps. 111-118.

Danks, M. (1997) "*Real-time Image and Video Processing in GEM"*. In: Proceedings of the International Computer Music Conference (ICMC) 1997, pps. 220-223.

Iazzetta, F. (2003) *"A Performance Interativa em Pele"*, In: IX Simpósio Brasileiro de Computação Musical, Campinas, São Paulo.

Nagashima, Y., (1998) *"Real-time Interactive Performance with Computer Graphics and Computer Music"*, In: Proc. of the 7th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Man-Machine Systems.

Puckette, M., Apel, T. e Zicarelli, D. (1998) *"Real-time audio analysis tools for PD and MSP",* In: Proc. of the Intl. Computer Music Conference (ICMC) 1998.

Rowe, R. (2001) *Machine Musicianship*, The MIT Press, Cambridge, Massachusetts.

Tarabella, L., Magrini M. e Scapellato G. (1997) *"Devices for Interactive Computer Music and Computer Graphics Performances"*, In: IEEE First Workshop on Multimedia Signal Processing, pps. 65-70.

# Octopus Music API: Modelling Musical Performance

**[1]Leandro Costalonga, Eduardo Miranda, [2]Evandro Miletto**

[1]Interdisciplinary Centre of Computer Music Research - University of Plymouth
206 Smeaton Building,Drake Circus,Plymouth – UK

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

{leandro.costalonga,eduardo.miranda}@plymouth.ac.uk,
miletto@inf.ufrgs.br

***Abstract.*** Music languages are on the cutting edge of all kinds of work in computer music [Loy and Abbott, 1985]. A satisfactory realization of an encoded work can be reconstituted through the interpretive practice of trained performers, but the knowledge that enables human performers to interpret music notation is extremely difficult to represent in a formal way [Sundberg et al., 1983]. Unlike previous work, the Octopus Music API aims to facilitate the programming of music performance system by providing the means to model not only the music but also the interaction of the performer and the musical instrument.

## 1. Introduction

A computer language presents an abstract model of computation that allows one to write a program without worrying about details that are not relevant to the problem domain of the program [McCartney, 2002]. Many are the ways in which computers can be employed in the service of music however the development of programming languages specifically for musical applications seems to have been concentrated on the areas of sound synthesis and musical performance [Loy and Abbott, 1985]. These languages are designed to provide a set of abstractions that makes expressing compositional and signal processing ideas as easy and direct as possible [McCartney, 2002].

The Octopus Music API is a Java API design to help in the modelling musical performance applications. Numerous software packages have been written for applications in music composition, music analysis, sound synthesis, and sound manipulation [Pennycook, 1985] but differently from other Java APIs' for music software development, the Octopus API has its focus on the modelling of musical performance elements - mainly the performer and his instrument. Since the limited space available for this paper, only the most important classes of the API will be presented. For a full understanding of the capabilities of the API please refer to the complete documentation at http://cmr.soc.plymouth.ac.uk/software/octopus/index.html.

The sections of this paper are related to the conceptual classification of the structures of the API. Starting with Musical Data Structures in Section 2, followed by Musical Data Interpreters and Instrument Classes. A brief conclusion is presented in Section 6.

## 2. Musical Data Structures

Musical Data Structures is a computational formalization of musical concepts that can be manipulated by the API. This classification applies for classes that represent musical concepts, such as *Note* or *RhythmPattern*.

### 2.1. Class octopus.Note

Most computer music notations define a musical note as the specification of an acoustic event. In the traditional music notation a *Note* specifies a human gesture toward an instrument [Loy and Abbott, 1985]. For us, the *Note* is the smallest audible element handled by the API.

### 2.2. Class octopus.Chord

A *Chord* is a set of *Notes* played together or arpeggiated. The *Notes* that compose a *Chord* are encapsulated in *ChordNote* objects which are the *Notes* with its contextual information, in other words, its role (interval) within the *Chord*.

## 3. Rhythmic Classes

### 3.1. Class octopus.Bar

Notes are sometimes connected by curved lines called slurs to show their grouping into phrases [Loy and Abbott, 1985]. A *Bar, in the context of this work,* is simply a rhythmic phrase. It's a collection of the smallest rhythmic structure designed in the API. The *Bar.RhythmEvent* can be a note or rest with values between 0 and 1 for duration, dynamic, and accentuation. *RhythmEvents* can be linked together through the *tie* attribute.

### 3.2. Class octopus.RhythmPattern

In the Octopus Music API the rhythmic line is defined independently from the *Melody* or *Harmony* facilitating the manipulation of the *Notes* and/or the rhythmic data.

The *RhythmPattern* represents a monophonic rhythmic line. Both the Melody and the Harmony embody *RhythmPattern, which is* composed of *Bars*, *Marks* and *Returning Points*. The B*ars* are inserted sequentially so the input order is relevant.

### 3.3. Class octopus.Arpeggio

An Arpeggio is a set of *RhythmPatterns* played simultaneously; it is used to spread the notes of the *Chord* along its duration (voicing). Often the *Arpeggio* information is omitted on the score and its use varies upon to the technique and expressivity of the *Performer*. When a *Musician* is requested to play a *Harmony* using a particular *Arpeggio*, it will adapt the *Arpeggio* to the *Harmony*, repeating or stretching its duration to match the duration specified for the *Chords*.

### 3.4. Class octopus.Melody

*Melody* is a set of *Notes* disposed in a sequence and played according to a certain *RhythmPattern*.

### 3.5. Class octopus.Harmony

*Harmony* is a set of *Chords* played according to a *RhythmPattern* and *Arpeggio*. If an *Arpeggio* is not assigned for a *Chord* then all *Notes* of the *Chord* will sound simultaneously and lasts for the respective duration assign for the *Chord*. The duration of each *Chord* is the same as the *RhythmEvent* associated with it. There is no information on the *Harmony* class defining how the *Chords* should be played (i.e. chord's shapes). This knowledge belongs to the *Musician*.

### 3.6. Class octopus.Music

*MusicalComponents* are the objects that can be placed in time and played as a single unit known as *Music*. These components can be either a *Melody or Harmony.*

## 4. Musical Data Interpreters

### 4.1. Class octopus.Musician

The *Musician* is basically an interpreter of the playable musical structures like: *Scale, Melody, Harmony, Music, RhythmPattern* and so on. He knows how to read and play these structures in the most basic possible level. No instrument restriction is considered in this computation.

### 4.2. Class octopus.instument.Performer

As a subclass of *Musician*, *Performers* also interpret musical structure but they have to adapt these *MusicalComponents* to the limitations of his *Instrument*. For instance, when a *Guitarist* plays a *Harmony* he will play it respecting the limitation of the *Guitar* he is playing which may sound slightly different when played by the simple *Musician*, although a *Guitarist* is a *Musician* in a higher level.

### 4.3. Class octopus.instument.fretted.Guitarist

The *Guitarist* is a *HarmonicPerformer* that knows how o play *Guitar* (actually, any fretted instrument). The *Guitar* used by the *Guitarist* has a direct influence in the way the *Music* is played.

As a *HarmonicPerformer*, the *Guitarist* needs to know how to play a *Chord* in the *Guitar*. A *Chord* can be played in several different ways (*ChordShape*) and using several different *Arpeggios*. Choose the most adequate *ChordShape* is a responsibility of the *Guitarist*. The *Guitarist* implemented in this API uses a similarity function to take this decision. Basically it chooses the most similar chord shape related to the previous one, minimizing the transitions effort [Costalonga et al. 2006].

## 5. Instrument Classes

### 5.1. Class octopus.instument.Instrument

The abstract class *Instrument* determines the minimum requirements that a new *Instrument* must implement to be able to interact with the other classes of API. This assures the scalability of the API to contemplate new *Instruments* during its development.

### 5.2. Class octopus.instument.string.fretted.FrettedInstrument

The *FrettedInstrument* Class represents the category of *Instruments* that have a string running along its fretted neck. The *Guitar* is an example of a subclass of *FrettedInstrument* that models a standard acoustic guitar, with 6 strings, 12 frets and standard tuning;

## 6. Final Words

The API is currently been hosted by SourceForge.net under Academic Free License. Free download and detailed information is available at http://cmr.soc.plymouth.ac.uk/software/octopus/.

This paper presented an overview of the Octopus Music API which is a Java API designed to assist software developers in the construction of software for musical performance. The API models what we believe to be the 3 elements presented in a musical performance: a) the performer, b) the instrument, and c) the music.

The key point to be observed in this API is the inverse approach of common music performance modelling, this means, instead of encode the performance task in complex musical notations we decided to endow the *Performer* with the ability and freedom to interpreter these structures based on his own knowledge. In the first stage, to prove the idea, the API focuses on a Guitar Musical Performance. Future work aims to extend the idea to different musical *Instruments*.

## 7. Acknowledgement

This research is sponsored by CAPES, Ministry of Education of Brazil.

## References

Costalonga, L., Miranda, E., Matthias, J., Vicari, R. (2006) An Idiomatic Plucked String Player. In: Proc. of FLAIR's Special Track on Artificial Intelligence in Music and Art, Melbourne, Florida, USA.

Loy, G. and Abbott, C. (1985). Programming languages for computer music synthesis, performance, and composition. ACM Computing Surveys (CSUR), 17(2):235-265.

McCartney, J. (2002). Rethinking the Computer Music Language: SuperCollider. Computer Music Journal, 26(4):61-68.

Pennycook, B. (1985). Computer-music interfaces: a survey. ACM Computing Surveys (CSUR), 17(2):267-289.

Sundberg, J., Askenfelt, A., Fryden, L. (1983). Musical performance: A synthesis-by-ruleapproach. Comput. Music J. 7, 1.

# Um Modelo Psicoacústico de Rugosidade

**Alexandre Torres Porres, Jônatas Manzolli**

Instituto de Artes/NICS – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6166 – 13091-970 – Campinas – SP – Brasil

{porres,jonatas}@nics.unicamp.br

***Abstract.*** *We discuss and present on this paper a Roughness model as well as a critical review on the perception of roughness and its theory.*

***Resumo.*** *Apresentamos e discutimos neste artigo um modelo de rugosidade, assim como uma revisão crítica de sua teoria e percepção.*

## 1. Rugosidade/Dissonância Sensorial & Banda Crítica

A sensação de batimentos ocorre para flutuações lentas de amplitude. Já a sensação de Rugosidade (do inglês *Roughness*, também traduzido como aspereza) ocorre para taxas de flutuação entre 20 Hertz até um intervalo que depende da Banda Crítica [Vassilakis 2001]. Como elementos da Dissonância Sensorial, Plomp e Levelt (1965) relacionaram essas percepções com a Banda Crítica (medida em *Bark* [Zwicker 1961]). Seus resultados – para dois tons senoidais – indicam um valor máximo de Dissonância Sensorial no intervalo que corresponde a um quarto da Banda Crítica, e uma sensação mínima para intervalos que a excedem (ver Figura 1).

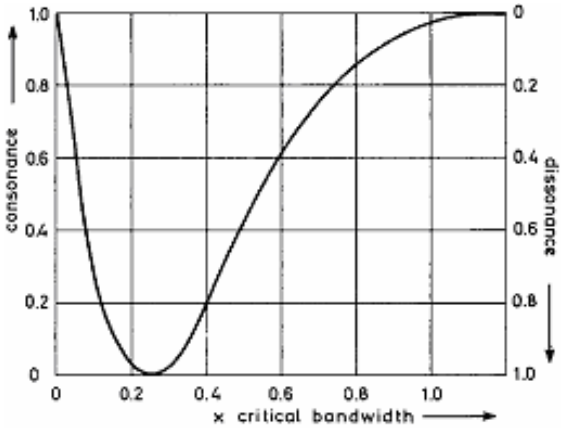


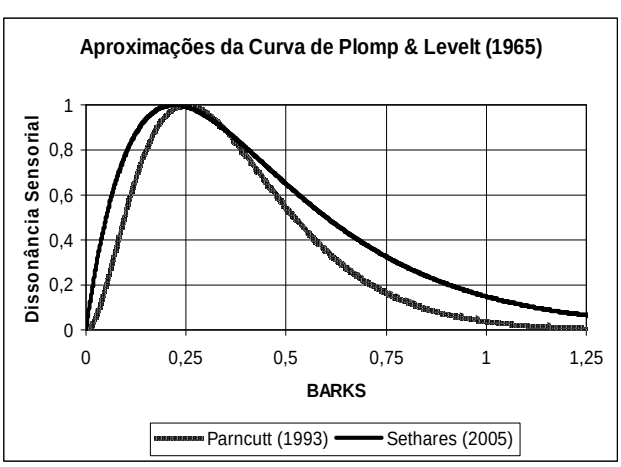

**Figura 1. Curva de Plomp e Levelt (1965) para intervalos em Barks.**

Apesar de seus correlatos físicos e psico-fisiológicos, o termo "Dissonância Sensorial" carrega uma subjetividade conceitual que abre margem para discussão do que seria "Dissonância". Daí advém uma crítica de Vassilakis, que atinge o trabalho de Plomp e Levelt (1965). Segundo esse autor, novas estimativas seriam aferidas mais imparcialmente eliminando conceitos subjetivos. Em suas palavras [Vassilakis 2001:83-84]:

> "Na tradição ocidental [rugosidade] tem sido regularmente ligada a conceitos de consonância e dissonância, (...) [estes] adquiridos esteticamente (...) ou não (...). Estudos que focam essa sensação têm sido ocasionalmente tendenciosos a encontrar uma justificativa aceitável definitiva e universal da 'inevitabilidade natural' e 'superioridade estética' da teoria da musical ocidental (...). Isso tem lhes privado de examinar seriamente os correlatos físicos e fisiológicos da sensação de rugosidade".

## 2. O Modelo de Rugosidade

Estudamos duas funções que a aproximam os resultados de Plomp e Levelt (1965) comparadas na Figura 2a. Como sugerido por Barlow, adotamos uma combinação de duas equações para converter Hertz em Barks: a de Terhardt (1979) – para valores abaixo de 219.5Hz – e Traünmuller (1990) – para valores acima. Descartamos uma segunda função de Sethares (2005) que propõe um ajuste não acurado (Ver Figura 2b) na conversão de Hertz para Barks em função da freqüência de base (*fmin*). Na Figura 2b, o ponto máximo em Barks varia de acordo com o registro do tom fixo (0.9 para 10kHz, 0.12 para 5KHz, 0.16 para 2500Hz, 0.19 para 100Hz e 0.23 para 500Hz), enquanto deveria permanecer constante como na Figura 2a – cujo ponto máximo é 0.22 Barks.



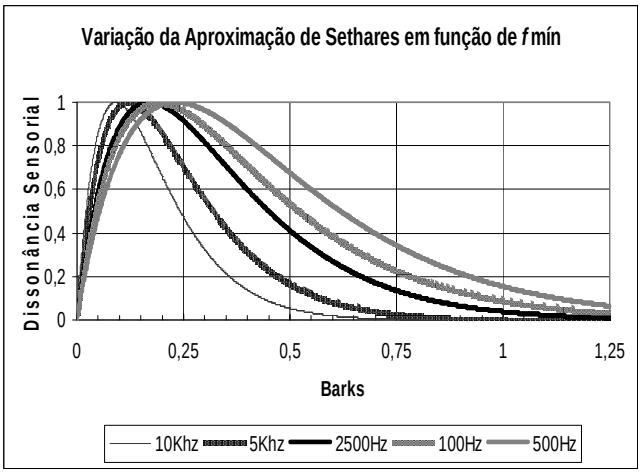




**Figura 2a e 2b. Comparações das aproximações em a) e Variação da Curva na função descartada [Sethares 2005] em b.**

Como no modelo de Barlow (1980), ajustamos os valores das amplitudes relativas por meio das *Curvas de Iso-Loudness* [Fletcher e Munson 1933]. Plomp e Levelt (1965) não mediram a sensação de componentes espectrais com amplitudes relativas diferentes. Em vista disso, Sethares (2005) multiplica o valor de rugosidade pelo menor valor das amplitudes. Entretanto, Vassilakis (2001) critica esse e outros procedimentos similares, e propõe uma equação que contabiliza a relação entre percepção de rugosidade e o *grau de flutuação de amplitude* [Vassilakis 2001]. O diagrama da Figura 3 representa os passos de nosso modelo, que retorna **R**(f1-A1,f2-A2), ou seja, o valor de rugosidade para um par de freqüências em Hz e suas respectivas amplitudes relativas. Como no modelo de Plomp e Levelt (1965), a rugosidade de um tom complexo é calculada somando-se a rugosidade de cada combinação dos pares de tons puros do espectro. A Figura 4 traz as Curvas de nosso modelo de rugosidade e compara os resultados de diferentes parâmetros como o redimensionamento proposto por Vassilakis (2001) e o de Sethares (2005) na Figura 4a, as aproximações de Parncutt (1993) em 4a e Sethares (2005) em 4b), assim como o redimensionamento de Sethares (2005) e a influência das Curvas de Fletcher e Munson (1933) na figura 4b. O espectro analisado é um tom complexo harmônico formado por seis parciais cujas amplitudes relativas decaem na taxa de 88%. O modelo foi implementado em puredata <http://puredata.info> baseado, a princípio, na implementação em *Matlab* do modelo de Sethares (disponível em <http://eceserv0.ece.wisc.edu/~sethares/comprog.html>).

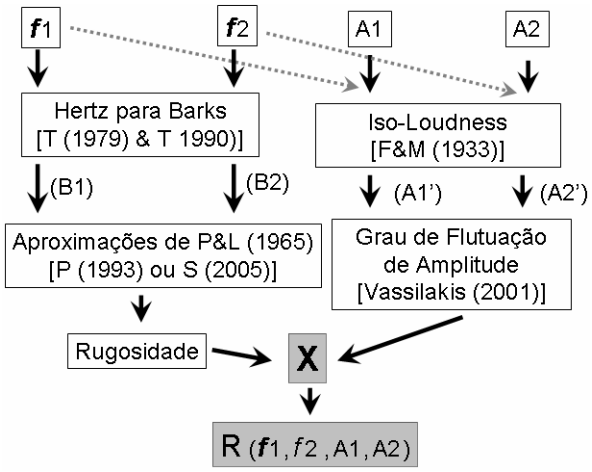



**Figura 3. O Modelo de Rugosidade para um par de tons puros.**

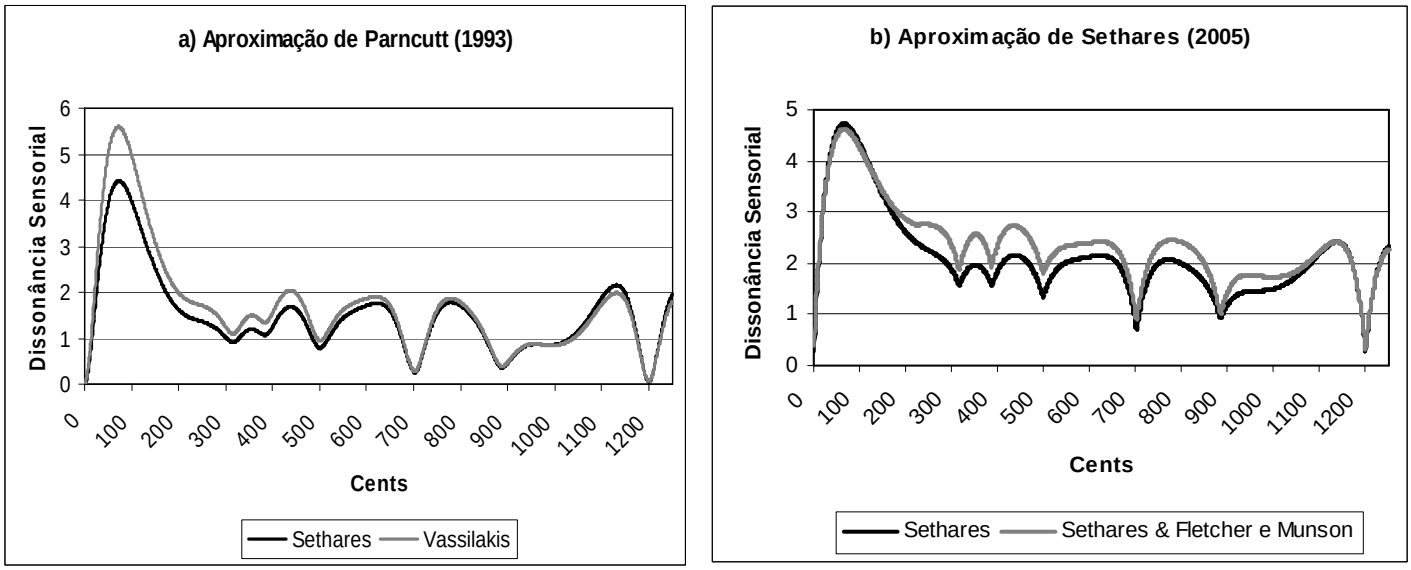



**Figura 4a e 4b. Comparações dos Parâmetros do Modelo. Curvas no registro de uma oitava mais um quarto de tom a partir de 500Hz.**

## 3. Discussão e conclusão.

Os dados de Plomp e Levelt (1965) – testes psicoacústicos subjetivos aferidos por média estatística – são imprecisos por natureza. Sua curva média (Figura 1) é, a priori, uma aproximação simplificada dos dados. Não obstante, a aproximação de Parncutt (1993) gera um gráfico mais semelhante à Figura 1 que a de Sethares (2005). Nesses gráficos em curvas, pontos mínimos representam o alinhamento de alguns parciais – eliminando a sensação de rugosidade. A partir desse princípio, curvas de tons complexos permitem encontrar intervalos musicais "consonantes" de acordo com o conteúdo espectral. Todavia, as variações timbrísticas de um instrumento musical são tantas e ligadas a tantos elementos de controle performático que há um limite intransponível nas análises desta ferramenta capaz apenas de analisar um espectro sonoro estático. De modo que melhorias em nosso modelo devem incluir mais uma nova dimensão temporal nas Curvas, assim como diferentes amostras de sons. Em contrapartida, é possível a análise de um sinal digital no tempo via FFT.

O modelo baseia-se no trabalho de Plomp e Levelt (1965), Parncutt (1993), Sethares (2005), Barlow (1980) e Vassilakis (2001). O elemento mais importante do modelo ora apresentado é a aproximação da curva de resultados de Plomp e Levelt (1965), assim como a necessária conversão de Hertz para a escala Bark [Zwicker 1961]. Uma revisão de nosso modelo poderá ocorrer quando esses dados essenciais forem revisados – o que é esperado em virtude da antiga data dos experimentos de Plomp e Levelt (1965). Um novo passo deve incluir o efeito de mascaramento. A rugosidade é um atributo fisiológico da sensação auditiva que influencia uma dimensão abstrata da percepção de dissonância. Helmholtz [1877: 234-235] apresenta a seguinte conclusão sobre inato *versus* adquirido na percepção de Rugosidade/Dissonância Sensorial:

> "A combinação [de tons] percebida como mais rugosa ou suave que outra depende apenas da estrutura anatômica do ouvido (...). Mas a que grau de rugosidade um ouvinte está inclinado a ... como meio para expressão musical depende do gosto e hábito; por isso a fronteira entre consonância e dissonância tem mudado freqüentemente ... e ainda mudará adiante.

## 4. Agradecimentos

Alexandre Porres tem apoio FAPESP (processo 04/12405-6) e é orientando do Prof. Dr. Jônatas Manzolli – que tem apoio CNPq (projeto 308765/2003-6). Sethares e Vassilakis foram prestativos via e-mail. Barlow tem colaborado com dicas que serviram como inspiração e guia, além de ter fornecido dados para o desenvolvimento de nosso modelo.

## Referências

Barlow, C. (1980). Bus journey to Parametron. *Feedback Papers* vol. 21-23, Feedback Studio Verlag, Köln.

Fletcher, H. e Munson, W.A. (1933), "Loudness, its definition, measurement and calculation" in *Journal of the Acoustical Society of America* N°5, 82-108 .

Helmholtz, H.L.F. (1877) On the Sensations of Tone as a Psychological basis for the Theory of Music. New York, NY: Dover Publications. 2ª edição (1954)

Parncutt, R. (1993) "Parncutt's implementation of Hutchinson & Knopoff roughness model", <http://www-gewi.uni-graz.at/staff/parncutt/rough1doc.html>.

Plomp, R. e Levelt, W.J.M. (1965). Tonal Consonance and Critical Bandwidth. *Journal of the Acoustical Society of America* N° 38, 548-568.

Sethares, W.A. (2005) *Tuning, Timbre, Spectrum, Scale*. London: Springer-Verlag. 2ª Edição com CD-ROM.

Traunmüller, H. (1990) Analytical expressions for the tonotopic sensory scale *Journal of the Acoustical Society of America.* N°88: 97-100.

Terhardt (1979) On the perception of spectral information in speech. In: Hearing Mechanisms and Speech (Creutzfeldt, O., Scheich, H., Schreiner, C., eds.), Springer, Berlin/Heidelberg, 281-291.

Vassilakis, P.N. (2001) Perceptual and Physical Properties of Amplitude Fluctuation and their Musical Significance. *Dissertação de Doutorado.* UCLA.

Zwicker, E. (1961) Subdivision of the audible frequency range into critical bands (Frequenzgruppen), *Journal of the Acoustical Society of America. 33*: 248.

# Arcabouço Orientado a Objetos para simulação Acústica na Plataforma AcMus

**Mário Henrique Cruz Torres**[1], **Fábio Kon**[1]

[1]Departamento de Ciência da Computação - Universidade de São Paulo

{marioct, kon}@ime.usp.br

***Abstract.*** *In this poster we describe an Object Oriented framework, built upon AcMus System, which is being made in Java language and already counts with an ray tracing algorithm implementation. We also show the easinesses that our framework provide for its users, as simplicity in the comparison between calculated acoustic parameters from the measurement and of the simulation of a room.*

***Resumo.*** *Neste poster descrevemos um arcabouço Orientado a Objetos desenvolvido dentro do ambiente AcMus, o qual está sendo feito totalmente na linguagem Java e já conta com a implementação de um algoritmo de traçado de raios. Mostramos também as facilidades que nosso arcabouço proverá para seus usuários, como a simplicidade na comparação entre parâmetros acústicos calculados a partir da medição e da simulação de uma sala.*

## 1. Introdução

Uma das propostas do projeto AcMus é construir um ambiente integrado para medição e simulação acústica de salas, o qual visa também a contribuir com ferramentas computacionais para melhorar a qualidade da acústica de salas [Iazzetta et al. 2001]. O sistema AcMus foi desenvolvido usando a linguagem de programação Java e a plataforma Eclipse. A linguagem Java tem como grandes características ser orientada a objetos e ser independente de plataforma, o que significa que programas escritos nesta linguagem funcionam em qualquer sistema operacional. Já a plataforma Eclipse provê um poderoso ambiente, baseado em componentes, para construção de aplicações com interfaces gráficas complexas [Foundation 2006].

O ambiente AcMus foi usado para realizar a medição acústica de salas de concerto na cidade de São Paulo. Os resultados obtidos contribuem para mostrar como o software está maduro na área de medição e cálculo de parâmetros acústicos de salas, como pode ser visto em [Figueiredo and Iazzetta 2005].

Dentro deste ambiente, construímos um arcabouço orientado a objetos para realizar a simulação acústica de salas. Tal arcabouço já conta com o método de traçado de raios e uma fonte sonora pseudo-aleatória implementados, de forma que é extremamente simples comparar os resultados de uma medição com os resultados de uma simulação, a fim de validar o modelo de simulação usado. O que também é uma novidade é este arcabouço ser licenciado sob a LGPL (*Lesser General Public Licence*), sendo único nesse sentido.

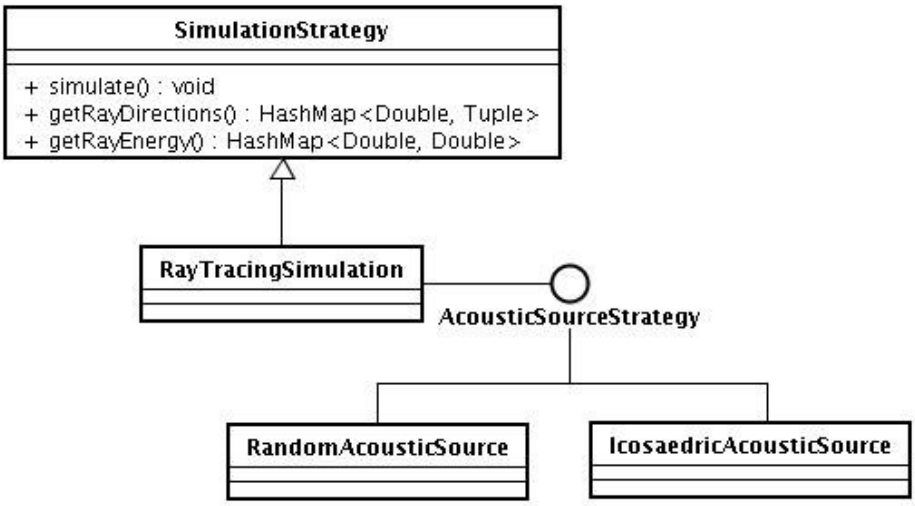



**Figure 1. Diagrama de Classes que mostra a estrutura de simulação do arcabouço e a interface *AcousticSourceStrategy* usada para permitir um baixo acoplamento entre o método de simulação e a fonte sonora.**

## 2. Arcabouço

Nosso arcabouço visa a facilitar a simulação e a comparação dos resultados obtidos com dados de uma medição. Para isso, implementamos uma estrutura orientada a objetos totalmente modularizada que permite ao usuário incluir, de maneira simples, seus próprios algoritmos de simulação do AcMus. Esta modularização é atingida com o uso do padrão de projeto (Strategy) [Gamma et al. 1995], o qual define uma família de algoritmos e nos permite trocar o algoritmo usado num certo momento independentemente do resto do sistema.

Já temos implementado um algoritmo de simulação acústica que utiliza o método de traçado de raios descrito por [Kulowski 1985] e uma fonte sonora pseudo-aleatória, na qual, por sua vez, são gerados raios por um método pseudo-aleatório. Ainda estamos desenvolvendo os métodos que acoplam o resultado obtido com a simulação ao cálculo de parâmetros acústicos e aos métodos de comparação entre a simulação e a medição da sala.

### 2.1. Arquitetura

Descrevemos a seguir as principais classes envolvidas no processo de simulação e a interface que deve ser implementada para criar novas fontes sonoras.

#### *SimulationStrategy*

A classe abstrata *SimulationStrategy* deve ser implementada por todos algoritmos de simulação, pois ela é o ponto de ligação entre nosso arcabouço e os métodos de simulação. Essa classe define somente três métodos, sendo o método `generate()` chamado pelo arcabouço para realizar a simulação. Os algoritmos de simulação implementados em nosso arcabouço devem armazenar três informações a respeito da simulação: (i) o tempo e (ii) correspondente energia - que constituem a Resposta Impulsiva Energética (IR), que

é a resposta da sala a um determinado impulso de curta duração - e (iii) o raio que atinge o receptor sonoro. Estas informações são utilizadas para calcular os parâmetros acústicos e para realizar a auralização da simulação. Os outros dois métodos definidos na classe *SimulationStrategy* são usados justamente para recuperar tais informações. São eles: `getRayDirections()`, que devolve um mapa *hash* com os tempos e raios que atingiram o receptor, e `getRayEnergy()`, que devolve um mapa *hash* com os tempos e correspondentes energia dos raios que atingiram o receptor.

É possível ver na Figura 1 a especialização realizada pela classe *RayTracingSimulation* que implementa um algoritmo de traçado de raios implementado de acordo com o método descrito por Kulowski [Kulowski 1985].

#### *AcousticSourceStrategy*

Os métodos de simulação acústica que utilizam a técnica de traçado de raios são muito sensíveis à forma como é construída a fonte sonora. Kulowski propõe um modelo de fonte sonora pseudo-aleatória, cuja vantagem é a simplicidade de implementação. Já Lewers [Lewers 1993] discute a validade dos modelos de fontes sonoras, principalmente as geradas por algoritmos pseudo-aleatórios, propondo um modelo onde os raios são gerados a partir de várias subdivisões de um icosaedro. Lewers mostra também como a escolha da fonte sonora pode influenciar os resultados das simulações.

Tendo em vista as distintas necessidades para a escolha das características de uma fonte sonora, criamos nosso sistema usando o padrão de projeto *Strategy*. Com este padrão é possível mesclar o algoritmo de simulação com qualquer fonte sonora implementada ou desenvolver uma nova fonte sonora e usar os algoritmos de simulação para a verificação dos resultados. Esta versatilidade torna nosso arcabouço independente do algoritmo usado para a fonte sonora e permite que novos algoritmos sejam avaliados de forma extremamente simples. O que pode ser visto na Figura 1, onde mostramos uma outra fonte sonora, *IcosahedricAcousticSource*, a qual pode ser implementada de acordo com Lewers.

Para desenvolver um algoritmo de uma nova fonte sonora usando nosso arcabouço, basta criar uma classe que implemente a interface *AcousticSourceStrategy*, a qual tem somente dois métodos, que devolvem uma lista com os raios gerados:

- `generate()`
- `generate(int n)`

### 2.2. Histograma com a Resposta Impulsiva

Nosso arcabouço provê também uma maneira simples para se construirem gráficos com a resposta impulsiva da simulação. O Histograma com a Resposta Impulsiva pode ser gerado a partir dos valores obtidos com o método `getRayEnergy()`, implementado pelos algoritmos de simulação. Para visualizar este diagrama basta que o usuário clique sobre "Histograma da Resposta Impulsiva".

### 2.3. Cálculo de Parâmetros Acústicos a Partir da Simulação

Ao realizarmos uma simulação acústica, estamos interessados em saber como é ou como será o comportamento acústico de uma determinada sala. Medidas para descrever tal

comportamento são complexas e muitas vezes não refletem a sensação acústica percebida pelos ouvintes.

O ambiente AcMus apresenta as ferramentas necessárias e já avaliadas para realizar os cálculos de vários parâmetros acústicos, dentre eles: Tempo de Reverberação (RT60, RT30 e RT20), Força Sonora (G), Definição ($D_t$). Como nosso arcabouço foi desenvolvido integrado ao AcMus, utilizamos os parâmetros acústicos deste sistema. Para realizar o cálculo dos parâmetros acústicos, basta clicar na opção "gerar parâmetros acústicos" e todos parâmetros serão calculados e mostrados na tela.

## 3. Conclusão

Nosso arcabouço já permite que sejam feitas simulações acústicas de salas, dentro do ambiente AcMus, usando o método de traçado de raios e uma fonte sonora pseudo-aleatória. A integração entre a simulação realizada e os cálculos de parâmetros acústicos do AcMus é mais uma facilidade para o Engenheiro Acústico que não precisará se preocupar em usar vários programas independentes e em lidar com os formatos de arquivos específicos de cada um destes programas.

Por ser um programa livre, esperamos que, em pouco tempo, o AcMus conte com várias implementações de algoritmos de simulação, fato que permitirá a comparação de maneira simples de algoritmos de simulação, dando mais liberdade para o usuário escolher se precisará de mais qualidade na simulação ou tempo de processamento, por exemplo.

O código fonte de nosso arcabouço está disponível em `http://gsd.ime.usp.br/acmus`

## References

Figueiredo, F. and Iazzetta, F. (2005). Comparative study of measured acoustic parameters in concert halls in the city of São Paulo. *Proceedings of the 2005 International Congress and Exposition on Noise Control Engineering, Rio de Janeiro, Brazil.*

Foundation, E. (2006). The Eclipse Platform. *Online at http://www.eclipse.org. Visitado em: maio 2007*.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Iazzetta, F., Kon, F., and Silva, F. (2001). AcMus: Design and Simulation of Music Listening Enviroments. *Anais do XXI Congresso da Sociedade Brasileira de Computação (CD-ROM), Fortaleza-CE.*

Kulowski, A. (1985). Algorithmic representation of the ray tracing technique. *Applied Acoustics*, 18(6):449–469.

Lewers, T. (1993). A combined beam tracing and radiant exchange computer model of room acoustics. *Applied Acoustics*, 38(2-4):161–178.

# ViMus: Sistemas Interativos de Tempo-real para Processamento Audiovisual Integrado

**Jarbas Jácome, Márcio Dahia, Geber Ramalho e Sílvio Meira**

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7.851 – 50.670-901 – Recife – PE – Brasil
{jjoj, mlmd, glr, srlm}@cin.ufpe.br

***Abstract.*** *This paper presents an approach for usability and expressivity trade-off for real-time multimedia systems. Through a new graphic user interface paradigm and using visual programming languages architecture, ViMus is an expressive tool, but still accessible for beginners.*

***Resumo.*** *Este artigo apresenta uma alternativa para um problema de compromisso entre expressividade e usabilidade em sistemas de tempo-real para multimídia. Através de um novo paradigma de interface gráfica e utilizando a arquitetura de sistemas de linguagens de programação visual, o ViMus é uma opção de ferramenta expressiva porém acessível para usuários iniciantes.*

## 1. Introdução

A popularização crescente das artes digitais (novas mídias) e, em particular, da atividade do visual-jóquei (Makela 2006), estimulam as pesquisas em sistemas multimídia de tempo-real. Esta demanda vem sendo atendida com o surgimento de programas específicos para o controle de exibição de vídeos e efeitos como Resolume; e com a criação de extensões para processamento gráfico como a GEM para linguagens visuais de computação musical como o Pd (Puckette 1997).

Este artigo apresenta uma pesquisa em andamento para dissertação de mestrado que propõe uma alternativa para o problema de balanço entre expressividade e usabilidade nestes sistemas. A sessão 2 apresenta os sistemas existentes, a sessão 3 explica o problema "usabilidade x expressividade" e a sessão 4 descreve o programa ViMus que está sendo desenvolvido como parte desta pesquisa.

## 2. Trabalhos relacionados

Entre as pesquisas acadêmicas relacionadas ao tema podemos citar, além do Pd/GEM, o Max/MSP/Jitter (Jones e Nevile 2005), Aura (Dannenberg 2005), Sonnet+Imager (Collopy 1999). Contudo, existe uma grande quantidade de programas comerciais não-acadêmicos relevantes como Resolume, Arkaos, Isadora, Visual Jockey, Modul8 e vvvv.

Identificamos dois grandes grupos aos quais pertencem a maioria dos sistemas interativos de tempo-real: os Sistemas Orientados a Fluxograma (SOFs) e os Sistemas Orientados a Amostras de Vídeo e Efeitos (SOAVEs).

### 2.1. Sistemas Orientados a Fluxograma (SOFs)

Uma das principais características desse tipo de programa é a grande quantidade de possibilidades de aplicações devido à sua natureza modular, extremamente flexível e dinâmica. A maioria dos SOFs oferece uma linguagem de programação visual e seu conceito está historicamente associado à computação musical (Figura 1).

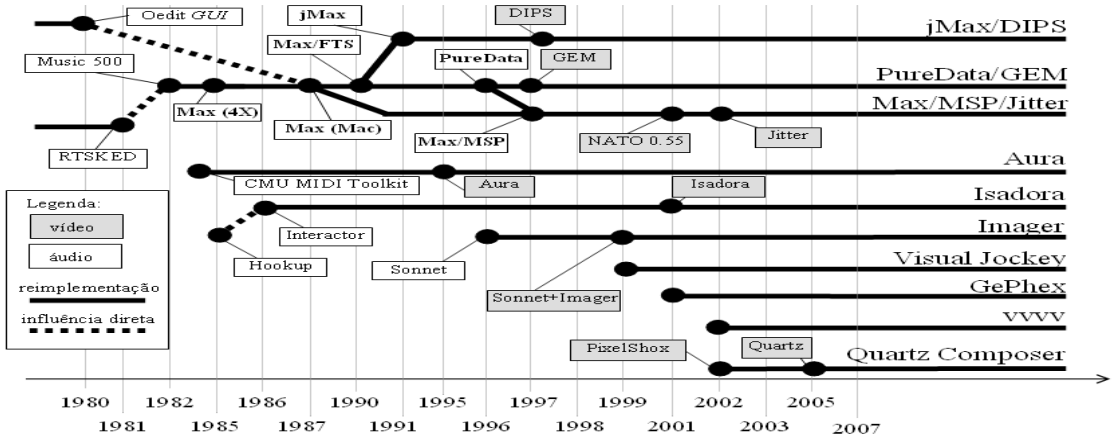



**Figura 1. Linha do tempo dos principais SOFs**

De um modo geral, um "fluxograma" (sugestão de tradução do termo "*patch*") é constituído por objetos e conexões entre os mesmos. Um objeto é um módulo responsável por uma função específica e pode possuir entradas e saídas. O usuário pode interagir com o fluxograma modificando-o em tempo-real (adicionando e removendo objetos e conexões) e/ou através de objetos de interface (como botões e *sliders*) ou objetos que processam MIDI.

### 2.2. Sistemas Orientados a Amostras de Vídeo e Efeitos (SOAVEs)

A maioria dos SOAVEs são programas destinados à atividade do visual-jóquei que consiste basicamente em controlar a execução de vídeos e efeitos de forma improvisada em tempo-real. As amostras de vídeo e efeitos podem ser disparadas a partir de comandos do usuário utilizando o teclado do computador ou um controlador MIDI.

A interface gráfica do usuário dos SOAVEs geralmente apresenta metáforas de equipamentos de controle de vídeo como *mixers* de vídeo do tipo "A e B" e formas de controle de vídeo já consagradas entre os VJs como o teclado de computador e teclado de piano (controlador MIDI). Geralmente é possível associar uma tecla ao comando de disparo de um efeito ou a uma amostra de vídeo. Também é possível fazer um número limitado de associações entre áudio e efeitos de vídeo em tempo-real.

## 3. Problema Usabilidade x Expressividade

Ao analisarmos estes sistemas, observamos um problema que na verdade é comum em tecnologias operadas diretamente por um ser humano: quanto mais funcionalidades e pluralidade de uso, pior é a curva de aprendizado e usabilidade em geral (Nielsen 1994). Os programas mais simples e específicos como Resolume têm uma excelente curva de aprendizado, porém oferecem pouca flexibilidade para criação de novos efeitos e

interações entre as mídias. Já programas como Pd/GEM são extremamente expressivos, porém assustam artistas iniciantes por sua complexidade.

Este problema será chamado nesta pesquisa de “usabilidade x expressividade”. Definiremos expressividade aqui como uma medida para o número de associações de funções e efeitos permitidas. No extremo dessa expressividade estão os programas como Pd que oferecem para o usuário uma linguagem de programação visual. Para facilitar a visão do problema, o gráfico da Figura 2 ilustra a configuração em um plano “usabilidade x expressividade” dos principais *softwares* atuais.

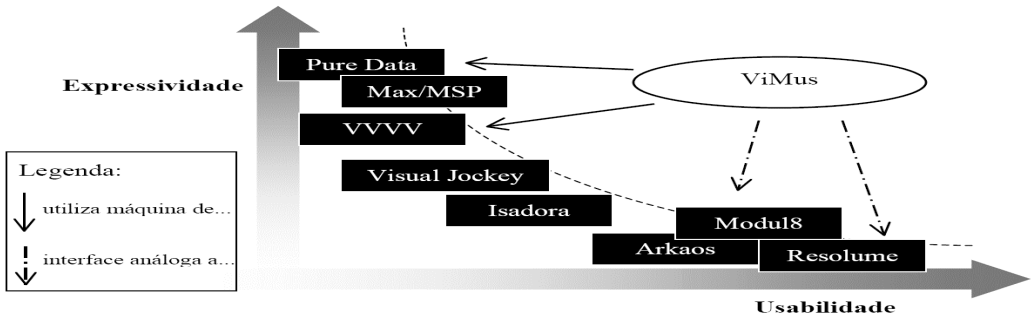



**Figura 2. Principais sistemas em termos de usabilidade x expressividade e posicionamento do ViMus e suas relações.**

## 4. ViMus

ViMus (junção das palavras “visual” e “música”) é uma alternativa para o problema “usabilidade x expressividade”. O ViMus é uma camada de interface amigável para sistemas expressivos, isto é, um metassistema que reúne características boas dos SOFs e dos SOAVEs.

O poder de expressão do ViMus é oferecido ao usuário de forma gradativa e natural. Para isso a arquitetura e interface gráfica do usuário evidenciam dois conceitos já existentes nos SOFs: abstração de fluxogramas e flexibilidade de interface. Uma abstração de fluxograma consiste em um objeto especial que contém um fluxograma. Flexibilidade de interface consiste na possibilidade de composição e modificação de interfaces gráficas para os fluxogramas. Além disso, o ViMus utiliza o conceito de *graph-on-parent* (Puckette 2002), uma opção que determina que objetos de interface (*sliders*, botões, etc) de uma abstração de fluxograma possam ser visualizados dentro do objeto que representa esta abstração no fluxograma “pai”.

Optamos por uma metáfora de caixas tridimensionais abertas em vez de janelas. Uma caixa, assim como uma janela em Pd, contém um fluxograma cujos objetos podem ser outras caixas (abstrações de fluxogramas). Uma caixa possui apenas três lados opacos, os lados restantes são transparentes, permitindo a visualização de seu interior, ou seja, do fluxograma. O lado frontal da caixa é chamado “painel de controle” e exibe elementos de interface gráfica para controle do fluxograma incluindo o painel de controle de caixas “filhas”. O lado superior exibe as entradas e o lado inferior, as saídas.

O ViMus sempre apresenta pelo menos uma caixa contendo um fluxograma. O programa será distribuído com algumas caixas iniciais extremamente simples inspiradas em SOAVEs já consagrados como Resolume e novas caixas poderão ser criadas. O usuário escolhe a caixa que lhe parecer mais apropriada. Uma caixa pode ser girada

permitindo a visualização de seu fluxograma (Figura 3). Da mesma forma, qualquer outra caixa de abstração de fluxograma pode ser aberta e girada. Ativando o modo de edição, as conexões podem ser modificadas, objetos removidos e uma paleta de opções de objetos é exibida para que o usuário possa adicioná-los, modificando a aplicação de acordo com suas necessidades.

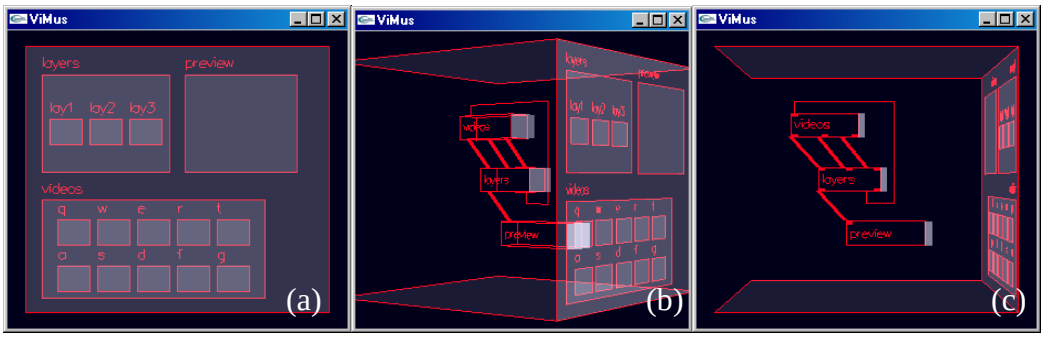



**Figure 3. ViMus: a) ex. de painel de controle, b) girando e c) ex. de fluxograma**

## 5. Considerações finais

Neste artigo apresentamos o ViMus, uma camada de interface amigável para sistemas interativos de tempo-real. A máquina de processamento de vídeo e análise de áudio para o ViMus foi implementada em C++ e OpenGL. Esta máquina será integrada ao Pd/GEM e outros sistemas de código aberto. Um protótipo da interface está sendo avaliado por visual-jóqueis com repostas positivas.

Finalmente, esta pesquisa tem duas contribuições fundamentais: inclusão de mais artistas na cultura de criação de suas próprias ferramentas multimídia para tempo-real e formalização de conhecimento na área.

## References

Collopy, F. (1999). “Visual Music in a Visual Programming Language.” IEEE Sumposium on Visual Music, Tokyo, Japan, IEEE.

Dannenberg, R. B. (2005). "Interactive Visual Music: A Personal Perspective." Computer Music Journal 29(4): 25-35.

Jones, R. and B. Nevile (2005). "Creating Visual Music in Jitter: Approaches and Techniques." Computer Music Journal 29(4): 55-70.

Makela, M. (2006). Live Cinema: Language and Elements. Media Lab. Helsinki, Finlândia, Helsinki University of Art and Design. MA in New Midia: 70.

Nielsen, J. (1994). "Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier." Acessado em nov., 2006, http://www.useit.com/papers/guerrilla_hci.html.

Puckette, M. S. (1997). Pure Data: Recent Progress. Third Intercollege Computer Music Festival, Tokyo, Japan.

Puckette, M. S. (2002). "Max at Seventeen." Computer Music Journal 26(4): 31-43.

# SpotRadio: Uma Ferramenta de Composição Musical Colaborativa em Rede com Suporte a Distribuição e Versionamento de Artefatos.

**João Paulo C. Rolim, Geber Lisboa Ramalho**

Centro de Informática - Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 - CEP 50732-970 – Recife – PE – Brasil

{jpcr,glr}@cin.ufpe.br

***Abstract.*** *This paper presents the SpotRadio, a collaborative tool developed based on a common collaborative model for multimedia production over the Internet. The tool was built to support the musical composition process practiced by groups in which their users are physically distributed and have a need of coordination, communication and cooperation mechanisms to provide quality and group expression for the final artifact.*

***Resumo.*** *Este artigo apresenta o SpotRadio, uma ferramenta colaborativa desenvolvida a partir de um modelo colaborativo que parece ser bastante difundido na a elaboração de obras artísticas multimídia através da Internet. Tendo em vista este modelo, a ferramenta foi idealizada para dar suporte ao processo de composição musical de grupos onde indivíduos estejam fisicamente separados e necessitem de mecanismos de coordenação, comunicação e cooperação que contribuam para a confecção de um artefato final de qualidade e que seja um reflexo da expressividade do grupo como um todo.*

## 1. Introdução

Hoje, podemos interagir com outras pessoas mesmo que estejamos em locais distintos, através do uso de ferramentas de computador que nos proporcionam um ambiente favorável à troca de idéias e de artefatos. Mecanismos de troca de e-mails, vídeo e áudio conferência, fóruns, listas de discussões e bate-papos geram oportunidades de comunicação e coordenação dentro de um grupo de usuários quando envolvidos em um assunto ou tarefa em comum. Estes mecanismos aliados ao uso crescente da Internet no suporte a criação colaborativa potencializam as chances de sucesso do trabalho em conjunto, pois possibilitam o gerenciamento dos recursos envolvidos.

No campo da produção musical não é diferente. Diversos grupos de artistas dispostos na Internet usam ferramentas que facilitam a produção de suas obras. Neste caso, é mais comum o uso de mecanismos como trocas de e-mails e uso de repositórios de arquivos. No entanto, alguns esforços têm sido feitos a fim de promover a utilização de ferramentas mais elaboradas e especializadas para a produção musical. Projetos como FMOL [Jorda and Aguilar 1998], WebDrum [Burk 2000], JamSpace [Gurevich 2006], Radio ReCombo [Rolim 2006] e o PSO [Barbosa and Kaltenbrunner 2002] apresentam diferentes abordagens em aspectos de síntese sonora, sincronização de áudio e interações entre usuários.

Seguindo este mesmo caminho surge o SpotRadio, que através de seus módulos, propicia um ambiente compartilhado onde se permite realizar interações de usuário de modo on-line e off-line.

Este artigo mostra aspectos da concepção e desenvolvimento da ferramenta SpotRadio apresentando na seção 2 o processo colaborativo de composição musical na Internet tomado como base para este projeto, em seguida, na seção 3, teremos uma visão geral do sistema e dos mecanismos de suporte ao processo colaborativo musical. Já na seção 4, serão apresentadas algumas conclusões.

## 2. Processo de Composição Musical Colaborativa na Internet

Tomando como base grupos artísticos como o Re:Combo [Re:Combo 2007] que eventualmente utilizam a Internet como meio de comunicação e de produção de suas obras, observa-se a tentativa de estabelecer um processo colaborativo onde seus membros, com um objetivo em comum, fazem uso de um repositório de arquivos e de um servidor de e-mails. Neste processo, foram identificados pontos de melhoria objetivando a facilidade de comunicação e expressividade do grupo, dando a possibilidade de uma visão geral do artefato em tempo de produção. Estes pontos são:

- Troca de e-mails. Eficaz, mas trás um gargalo de tempo na produção, já que a leitura dos mesmos é feita ao tempo de cada usuário. Não se garante quando serão lidos ou se a tempo.
- Moderação do resultado final. Geralmente feito por um único membro que acaba expressando sua visão mais do que a do grupo. É feito de forma isolada e os demais podem vir a opinar apenas quando o artefato está pronto.
- Troca de arquivos. Comumente são trafegados arquivos de grande tamanho, o que dificulta suas manipulações em rede, desestimulando o processo cooperativo. Em muitos casos, uma pequena modificação do artefato requer a atualização da obra como um todo.

Tendo isso em vista, é proposta uma ferramenta que contém mecanismos síncronos e assíncronos na tentativa de fazer este processo colaborativo mais produtivo.

## 3. SpotRadio

O SpotRadio é uma ferramenta de composição musical colaborativa que está sendo desenvolvida para dar suporte ao processo de composição musical na Internet. A ferramenta é dividida em 2 módulos. O primeiro, desenvolvido em JAVA, trata de interações off-line, com mecanismos de atualização e versionamento de artefatos e o segundo, desenvolvido em C++, trás consigo mecanismos de troca de informações em tempo real.

Os mecanismos presentes no SpotRadio podem se fazer uso de arquivos em diversos formatos, já que seu módulo de atualização pode ser utilizado em qualquer tipo de projeto colaborativo que necessite ou permita a fragmentação da obra em artefatos. Estes artefatos eventualmente podem ser editados em outras ferramentas externas e ainda assim, serem utilizados como parte de um projeto controlado pelo SpotRadio.

Na a edição musical, estes dois módulos trabalham em conjunto promovendo um ambiente compartilhado [Barbosa 2006] através de uma interface, mostrada na Figura 1,

que se utiliza de um seqüenciador musical onde usuários podem organizar seus arquivos de áudio de forma a compor a música em uma linha de tempo.

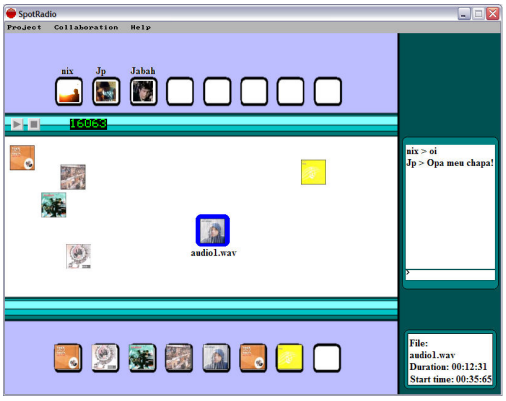



**Figura 1. Protótipo do editor compartilhado do SpotRadio.**

Além de oferecer um mecanismo de bate-papo que facilita a troca de mensagens entre os seus usuários, o seu módulo editor se caracteriza como um ambiente onde ações de um único usuário são percebidas pelos demais em tempo de execução. Nele é possível realizar ações de adição, remoção e modificação de arquivos na obra e de *undo*, para desfazer alterações e restaurar versões anteriores da música. Diferentemente do que acontece em um processo basicamente assíncrono, off-line, o resultado da obra é sabido antes mesmo do grupo fechar uma versão estável da música, já que o produto de cada modificação é percebido em tempo real durante aquela sessão.

Já através do módulo de atualização, mostrado na Figura 2, o usuário pode manter seu projeto atualizado com versões remotas mais recentes. Durante tarefas de atualizações, este módulo tenta diminuir o trafego de arquivos na rede, já que podem ser mantidas versões para cada artefato componente da obra e apenas os artefatos modificados ou ausentes em sua máquina são requisitados para serem baixados.

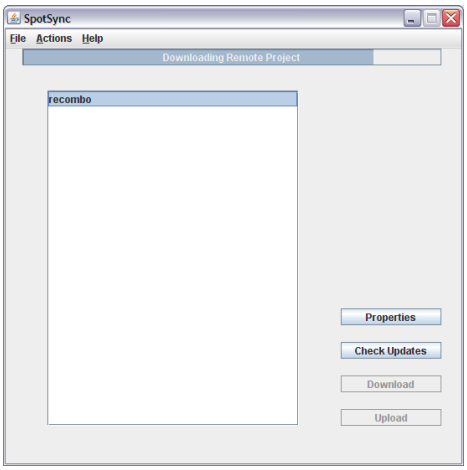



**Figura 2. Protótipo do módulo de atualização de projetos do SpotRadio.**

Adicionalmente, para proporcionar maior flexibilidade à distribuição das obras, o SpotRadio através deste módulo, dá suporte ao uso de vários repositórios dispostos

arbitrariamente na Internet. Cada um dos artefatos pode estar em lugar distinto e pode ser configurado por um usuário quando um desses artefatos é adicionado a um projeto.

## 4. Conclusões

Neste artigo, foi apresentado o SpotRadio, uma ferramenta que tenta tornar mais produtivo um processo colaborativo que parece ser comumente utilizado na composição artística multimídia na Internet. Através de um ambiente compartilhado destinado a composição musical, o SpotRadio insere a este processo mecanismos on-line e off-line que trazem interações em tempo real e dão suporte a atualizações das obras através do versionamento de seus artefatos.

## Referências

Barbosa, A. (2006). Survey of Computer-Supported Cooperative Work for Music Applications. Department of Technology, Pompeu Fabra University. Doctor per la Universitat Pompeu Fabra with Mention of European Doctor.

Barbosa, A. and M. Kaltenbrunner (2002). Public Sounds Objects: A shared musical space on the web. Proceedings of International Conference on Web Delivering of Music, Darmstadt, Germany, IEEE Computer Society Press.

Burk, P. (2000). Jammin' on the Web - a new Client/Server Architecture for Multi-User Musical Performance. ICMC 2000.

Gurevich, M. (2006). JamSpace: Designing A Collaborative Networked Music Space for Novices. Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06), Paris, Franca.

Jorda, S. and T. Aguilar (1998). FMOL: a graphical and net oriented approach to interactive sonic composition and real-time synthesis for low cost computer systems. Proceedings of COST G6 Conference on Digital Audio Effects 1998.

Re:Combo (2007). www.recombo.art.br.

Rolim, J. P. (2006). Radio ReCombo: a Web-based Collaborative System for Musical Composition WebMedia 2006, Natal - RN - Brazil.

# Aplicação de redes neurais para auxílio nas composições musicais utilizando compassos como primitivas e inspiração em relevos naturais

**Débora Cristina Corrêa[1], José Hiroki Saito[1]**

[1]Departamento de Computação – Universidade Federal de São Carlos (UFSCAR)
Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brasil

{debora_correa, saito}@dc.ufscar.br

***Abstract.*** *This paper describes a neural network application for helping musical composition using as inspiration the landscapes contour. During the training phase, the neural network learns certain aspects of musical structure by means of measure examples taken from melodies of the training set. The chosen melodies for training are folk melodies, once they are simple and monophonic. During composition, the system uses the measures learned to compose new melodies using as input the extracted data of the landscapes contour. A preliminary version of this system was implemented and the results obtained using one hundred measures.*

***Resumo.*** *Esse artigo descreve uma aplicação de redes neurais para auxílio nas composições musicais utilizando como inspiração as ondulações dos acidentes geográficos. Durante a fase de treinamento, a rede neural aprende certos aspectos da estrutura musical através de exemplos de compassos extraídos de melodias do conjunto de treinamento. As melodias escolhidas para o treinamento são melodias folclóricas, por serem simples e monofônicas. Durante a composição, o sistema utiliza o repertório de compassos usados no treinamento para compor novas melodias, usando como entrada dados extraídos de relevos naturais. Uma versão preliminar desse sistema foi implementada e os resultados foram obtidos para 100 compassos.*

## 1. Introdução

As Redes Neurais Artificiais (RNAs), também conhecidas como sistemas conexionistas, constituem uma forma de computação não-algorítmica inspirada na estrutura e processamento do cérebro humano. A computação é realizada por várias unidades de processamento simples, os neurônios, conectados em rede e atuando em conjunto. As RNAs passam por um processo de aprendizagem que consiste em receber um conjunto de treinamento, para a extração de características necessárias para representar a solução desejada. Nesse procedimento, as redes neurais artificiais adaptam suas interconexões até que os padrões de excitação desejados estejam próximos do comportamento desejado. [HAYKIN, 1999] [BRAGA, LUDERMIR, CARVALHO, 2000]

Para a composição musical, os modelos conexionistas, assim como os outros modelos que envolvem aprendizado de máquina, são capazes de aprender padrões e características presentes nas melodias do conjunto de treinamento e obter

generalizações dessas características para a composição de novas melodias. [TODD, 1989] [MOZER, 1994] [CHEN, MIIKKULAINEN, 2001] [ECK, SCHMIDHUBER, 2002]

O presente trabalho consiste no uso de RNAs para o aprendizado de compassos de melodias, usando como entradas, dados extraídos de relevos naturais, e como saídas as amostras de compassos de melodias. Para a composição, novos dados de relevos são apresentados à rede para a geração de novas melodias.

O artigo está organizado como segue: a seção 2 consiste na descrição do projeto proposto e a seção 3 consiste nos resultados obtidos e conclusão do trabalho.

## 2. Descrição do Trabalho

Além do conhecimento musical, emoções e intenções, o compositor geralmente conta com uma inspiração para o desenvolvimento de uma nova melodia. Um dos objetivos desse trabalho é adicionar ao processo de composição uma forma de inspiração, proveniente da Natureza. Propõe-se que as novas composições sejam inspiradas na Natureza através da utilização das ondulações dos acidentes geográficos durante a fase de treinamento e composição. Na fase de treinamento, a rede neural deve aprender os compassos de melodias escolhidos pelo usuário e, na fase de composição, deve ser capaz de gerar novas melodias com compassos previamente treinados.

O trabalho considera, em princípio, que um compasso consiste numa primitiva melódica que possui condições de se encadear com outros compassos para a formação de uma melodia. Para isso, a restrição no treinamento seria a escolha dos compassos de um mesmo estilo musical.

Na implementação preliminar relativa ao presente trabalho, as notas musicais são representadas por suas alturas e durações. As notações rítmicas estão limitadas em colcheia (♪), semínima (♩) e mínima (𝅗𝅥). As pausas não são usadas. Os compassos são representados pelas notas e pelos seus atributos de duração e formados de modo que possuam quatro tempos cada. Para isso, a colcheia é a unidade de tempo. Assim, a semínima e mínima valem dois e quatro tempos, respectivamente.

As alturas das notas são representadas pela combinação de números inteiros e intervalos musicais. A cada nota é atribuído um número inteiro (Figura 1).

Cada intervalo determina uma distância de freqüência de uma nota para a outra, em semitons (utilizado nesse trabalho) ou em freqüências logarítmicas. Mesmo com um número fixo de neurônios, vários intervalos de notas podem ser atingidos com essa representação. Além disso, essa representação é menos dependente de uma determinada tonalidade e pode descobrir estruturas comuns entre melodias.

O intervalo entre duas notas s e t pode ser determinado pela equação 1:

$$\text{intervalo}(s, t) = t - s \qquad (1)$$

em que s e t são os valores inteiros que representam as notas musicais.

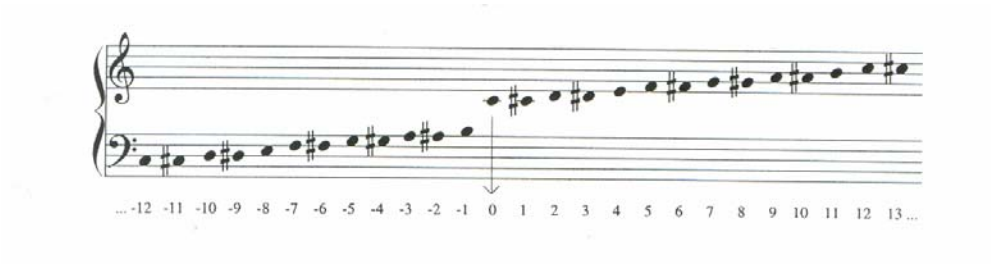
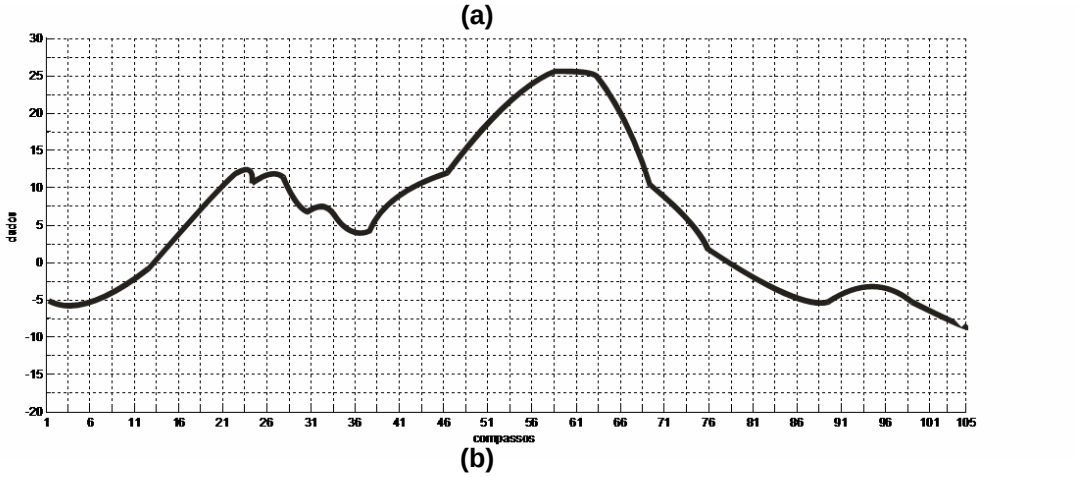



**Figura 1. Representação das notas musicais por valores inteiros**

Os dados que representam os padrões de entrada são codificados a partir do contorno de ondulações geográficas. A Figura 2 apresenta um exemplo dessa codificação.

**(a)**




**(b)**

**Figura 2. Representação da entrada da rede**

O contorno das ondulações é extraído de uma imagem e convertido para valores numéricos inteiros como mostra o gráfico da Figura 2 (b).

A rede utilizada na versão preliminar é uma MLP (*Multi-Layer Perceptron*), fortemente conectada, com 4 entradas de codificações de ondulações geográficas; 4 saídas de altura das notas e 4 saídas de atributos de duração, treinada com o algoritmo *Back-Propagation.* [HAYKIN, 1999]

## 3. Resultados e Conclusão

Como exemplo de aplicação, foram usados 100 compassos extraídos de 11 músicas folclóricas brasileiras como, por exemplo, “Samba lêlê”, “Escravos de Jó”, “Marcha Soldado”, para o treinamento da rede neural MLP.

Usando taxa de aprendizagem, $\alpha = 0.001$, no algoritmo *back-propagation* padrão, um número médio de 500 mil iterações foi suficiente para o treinamento de todos os compassos. Os neurônios da camada escondida e das quatro últimas saídas (neurônios de atributos de duração) usaram funções de ativação sigmóide, enquanto que os quatros primeiros neurônios de saída (notas), funções lineares.

Durante a composição, quando o padrão de entrada coincidiu com um padrão usado para treinamento a saída resultou no compasso treinado, conforme esperado. Para os padrões de entrada com os valores próximos aos padrões de entrada de treinamento, as saídas são próximas aos padrões desejados, revelando assim, a generalização da rede para composições de novas melodias.

O sistema assim constituído pode ser aperfeiçoado com a inclusão de outros atributos para as saídas, como dinâmicas de intensidade e andamento e pausas. Além disso, as notas podem ser polifônicas, a unidade de tempo diferente da colcheia (para utilização de outras notações rítmicas) e os compassos podem ter uma quantidade diferente de tempos.

O sistema é útil para a obtenção automática de melodias para que o usuário possa fazer sua análise posterior.

Como trabalhos futuros, além da implementação dos aperfeiçoamentos já descritos, deve-se incluir os algoritmos de análise dos resultados da composição, melhorando a capacidade de generalização da rede.

## 4. Referências Bibliográficas

BRAGA, A. P.; LUDERMIR, T. B.; CARVALHO, A. P. “Redes Neurais Artificiais – Teoria e Aplicações.” , Rio de Janeiro - RJ, LTC, 2000.

CHEN, C. C. J.; MIIKKULAINEN, R. (2001) “Creating Melodies with Evolving Recurrent Neural Network.” In Proceedings of the International Joint Conference on Neural Networks, IJCNN’01, 2001, p. 2241 – 2246, Washington - DC.

ECK, D.; SCHMIDHUBER, J. (2002) “A First Look at Music Composition using LSTM Recurrent Neural Networks.” Technical Report: IDSIA-07-02.

HAYKIN, S. “Neural Networks – A Comprehensive Foundation.” Prentice Hall, USA, 1999.

MOZER, M. C. (1994) “Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing.” Connection Science, 6(2-3), p. 247 – 280.

TODD, P. M. A (1989) “Connectionist Approach to Algorithmic Composition.” Computer Music Journal: Vol.13, No. 4.

# Observação de acoplamentos entre modos de vibração ortogonais em uma guitarra elétrica

**Nicolau L. Werneck*, Furio Damiani**

[1]DSIF – UNICAMP

nwerneck@gmail.com, furio@fee.unicamp.br

***Abstract.*** *Using a computer and sound card, we captured simultaneously two signals generated by an electric guitar. One of them is the usual one, taken from the pickups. The other is the voltage induced on the extremes of the strings in their motion parallel to the body of the guitar. This allows us to appreciate the dynamics of the string in two dimensions.*

***Resumo.*** *Utilizando um computador e uma placa de som, capturamos simultaneamente dois sinais gerados por uma guitarra elétrica. Um deles é o habitual, saído dos captadores. O outro é a tensão elétrica induzida entre as extremidades das cordas ao se movimentarem paralelamente ao corpo da guitarra. Isto nos permite apreciar a dinâmica da corda vibrando em duas dimensões.*

## 1. Introdução

É cada vez mais comum o uso de modelos físicos na síntese computacional de sinais musicais [Smith III, 2006]. Podemos utilizar estes sintetizadores programado seus parâmetros manualmente, ou ajustando alguns deles a partir de sinais reais. Quando todos parâmetros são extraídos de gravações, temos uma forma de codificação onde se transmite um programa sintetizador a ser executado a partir de parâmetros de controle também transmitidos. É nesta idéia que se baseia o padrão MPEG-4 SA [Lazzaro and Wawrzynek, 1999].

Neste artigo apresentamos análises de gravações feitas de uma guitarra elétrica. Foi possível constatar características importantes do sinal que poderiam ser analisadas com precisão para a obtenção de um codificador como o descrito acima. Em especial, pudemos observar acoplamentos entre modos de vibração transversais.

Em nossos experimentos pudemos amostrar tanto o movimento vertical quanto o horizontal da corda. A guitarra elétrica permite realizar isto com relativa facilidade. Apesar de simples, esta técnica nos parece não ter sido explorada em todo seu potencial.

## 2. Revisão Teórica

### 2.1. Modelos lineares

O modelo mais simples para vibração de cordas é a equação da onda de d'Alembert, dada por $\partial^2 y/\partial t^2 = c^2 \cdot \partial^2 y/\partial x^2$. A solução para um determinado valor de $x$ em uma corda fixada rigidamente é um sinal periódico. Podemos introduzir imperfeições neste modelo, como a rigidez à dobra e a flexibilidade dos suportes. Estes efeitos costumam fazer com que os modos de vibração das cordas deixem de ser harmônicos. Podemos considerar ainda fenômenos que causam perdas, tornando complexas as freqüências que compõem o sinal [Fletcher and Rossing, 1991, cap. 2].

*Financiado pela CAPES.

### 2.2. Modelos não-lineares

O uso de equações diferenciais lineares para se modelar cordas de instrumentos musicais resulta em equações do movimento compostas por somas de exponenciais complexas aproximadamente harmônicas. Podemos ver cada exponencial como sendo um oscilador simples vibrando de forma independente.

Entretanto, é razoavelmente fácil obter equações diferenciais não-lineares para cordas vibrantes. A variação da tensão da corda conforme ela se deforma causa isso. Este problema foi abordado em profundidade pela primeira vez por Kirchhoff [L. A. Medeiros, 2002], e retomado por muitos pesquisadores no século XX.

A princípio estudou-se a vibração planar da corda: principalmente o surgimento de harmônicos em respostas forçadas e a variação da freqüência fundamental de acordo com a amplitude do movimento [Shankland and Coltman, 1939, Carrier, 1945]. Posteriormente a atenção se dirigiu para movimentos da corda nas duas direções [Murthy and Ramakrishna, 1965, Miles, 1965]. Uma importante característica deste modelo não-linear é o surgimento de acoplamentos entre os modos transversais de vibração [Anand, 1969, Elliot, 1980]. Este acoplamento não depende de terminações flexíveis, como o estudado por outros autores [Legge and Fletcher, 1984].

### 2.3. Modelo de uma guitarra elétrica

A guitarra elétrica de corpo sólido se assemelha a um violão, porém não possui caixa de ressonância. Por ser sólida, é de se esperar que as cordas de uma guitarra sejam fixadas de forma mais rígida do que no violão.

O sinal da guitarra vem de seu captador, que é um indutor com um ímã dentro. O campo do ímã aponta na direção normal ao plano da guitarra. O campo induz pequenos dipolos nas cordas. É a variação do campo magnético devido a estes dipolos que induz a tensão na saída do captador. É um processo de indução por variação de relutância.

Enquanto o movimento na direção do campo cria um sinal no captador, o deslocamento da corda no sentido ortogonal ao campo induz uma tensão nas extremidades da corda. Assim, ao analisarmos estes dois sinais, devemos ser capazes de determinar o movimento da corda em duas dimensões.

Medições deste tipo já foram feitas anteriormente, mas sempre em aparatos específicos, e geralmente utilizando sensores opto-eletrônicos [O'Reilly and Holmes, 1992, Hanson et al., 1994]. Nosso experimento utiliza uma guitarra convencional, e não requer sensores externos: apenas o captador já disponível, e a aquisição da tensão nas extremidades da corda. Só é preciso providenciar amplificadores. Utilizamos ainda um ímã externo para criar um campo mais forte e concentrado do que o dos captadores.

## 3. Experimentos

### 3.1. Envelope espectral

Pelo modelo de d'Alembert, podemos criar o sinal de uma guitarra com um trem de impulsos integrado, e passado por um filtro construído com uma multiplicação de senóides na freqüência. Os períodos destas senóides dependem das posições do captador e do ponto da corda em que o músico a toca para puxá-la, geralmente utilizando uma palheta.

Gravações de instrumentos acústicos dificilmente apresentam o espectro esperado. Por isso é difícil estimar o ponto tocado pelo músico ao excitar a corda. Alguns pesquisadores já até expressaram pessimismo quanto a esta possibilidade [Välimäki et al., 1996].

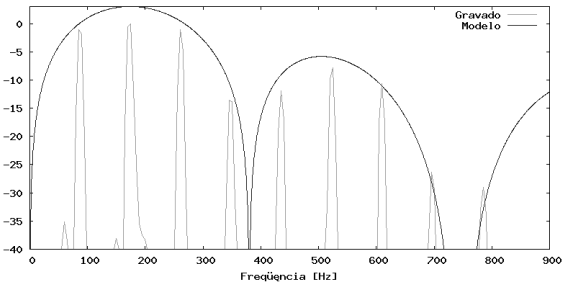



**Figura 1: Espectro da onda gravada, e filtro teórico.**

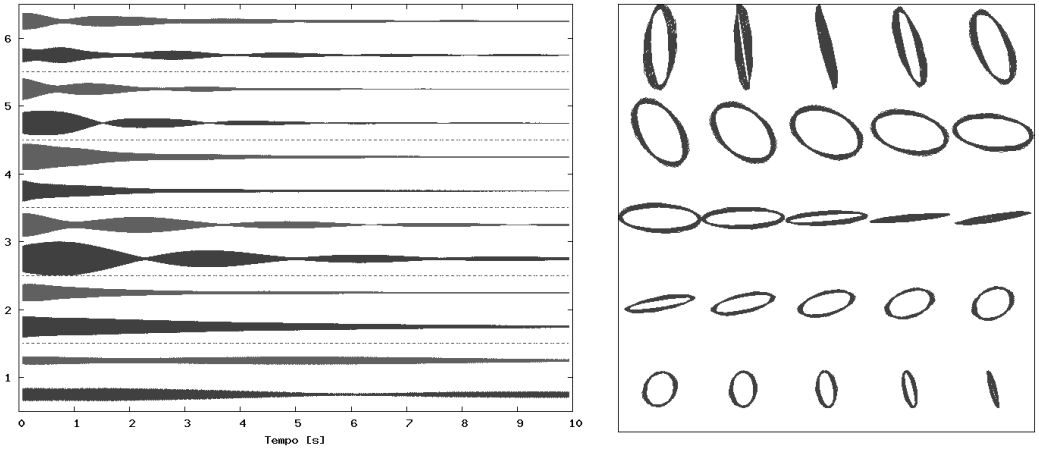



**Figura 2: Decomposição dos dois sinais em suas seis primeiras freqüências constituintes, e espaço de fases para o terceiro modo.**

Enquanto sinais gravados de instrumentos acústicos dificilmente exibem o espectro teórico de forma clara, o sinal obtido por nós parece se aproximar do modelo linear no começo da nota. Obtivemos o sinal diretamente da saída da guitarra, utilizando apenas um amplificador de sinais e a placa de som. A figura 1 traz o espectro obtido de uma gravação e um filtro estimado. Depois do começo da nota os acoplamentos mudam a amplitude das componentes, impossibilitando a análise em instantes posteriores.

Pra encontrar este filtro, partimos dos valores medidos da posição do captador e da palheta ($1/4$ e $1/9$) e os variamos até encontrar uma curva visualmente satisfatória. Os valores encontrados foram de aproximadamente $1/4.5$ e $1/8.7$. Este filtro suposto apenas ilustra o processo pelo qual se pode tentar medir as posições do captador e palheta numa gravação de guitarra elétrica. Ainda é necessário considerar filtros lineares no caminho do sinal e, naturalmente, automatizar o processo e avaliar sua precisão e complexidade.

### 3.2. Visualização dos acoplamentos

Os sinais obtidos foram processados por um banco de filtros FIR projetados com funções $\mathrm{sinc}$ moduladas, e com suavização de Lánczos [Hamming, 1989]. A saída dos filtros nos permite visualizar o comportamento de cada modo de vibração. Na figura 2 (à direita), os números na vertical são o número da componente em questão, sendo que a mais grave é uma nota mi ( $\simeq 82\mathrm{Hz}$). As curvas mais claras, de cima, são as obtidas do captador. Este sinal foi gravado utilizando-se o captador mais próximo à extremidade da corda, e puxando-a em uma posição semelhante na outra extremidade.

A partir deste gráfico, é possível verificar que alguns dos modos de vibração se comportam como um par de osciladores com acoplamento fraco. Apenas para o segundo e quarto modos de vibração este comportamento não foi visível. O gráfico à direita na figura 2 mostra o espaço de fases do terceiro modo de vibração em um trecho de 3 se-

gundos. O resultado se aproxima de uma elipse em precessão [Elliot, 1980], mas existe uma grande variação da proporção desta elipse no tempo, o que só se poderá explicar com análises matemáticas um pouco mais sofisticadas do que as mencionadas aqui.

## 4. Conclusões

Demonstramos a possibilidade de se estudar a dinâmica em duas dimensões das cordas de uma guitarra com gravações do seu sinal tradicional e do sinal induzido nas cordas. As gravações mostram uma boa aproximação a um modelo com dinâmica não-linear. A obtenção dos dois modos de vibração contribui mais para o estudo do instrumento do que para a codificação. Mas conhecendo melhor o instrumento poderemos buscar benefícios numa codificação ao restringir os parâmetros de um sistema que utilize somas de senóides para representar sinais [Hermus et al., 2005].

## Referências


Anand, G. V. (1969). Large-amplitude damped free vibration of a streched string. *J. of the Ac. Soc. of America*, 45(5):1089–1669.

Carrier, G. F. (1945). On the non-linear vibration problem of the elastic string. *Quarterly of Applied Mathematics*, 3(2):157.

Elliot, J. (1980). Intrinsic nonlinear effects in vibrating strings. *Am. J. of Physics*, 48(6).

Fletcher, N. H. and Rossing, T. D. (1991). *The Physics of Mus. Inst.* Springer-Verlag.

Hamming, R. W. (1989). *Digital Filters*. Dover Publications, Inc., third, 1998 edition.

Hanson, R. J., Anderson, J. M., and Macomber, H. K. (1994). Measuremets of nonlinear effects in a driven vibrating wire. *J. of the Ac. Soc. of America*, 96(3):1549–1556.

Hermus, K., Verhelst, W., Lemmerling, P., Wambacq, P., and Huffel, S. V. (2005). Perceptual audio modeling with exponentially damped sinusoids. *Sig. Proc.*, 85:163–176.

L. A. Medeiros, J. Limaco, S. B. M. (2002). Vibrations of elastic strings: Mathematical aspects, part one. *Journal of Computational Analysis and Applications*, 4(2):91–127.

Lazzaro, J. and Wawrzynek, J. (1999). The mpeg-4 structured audio book. [Online; accessed 14-June-2007].

Legge, K. A. and Fletcher, N. H. (1984). Nonlinear generation of missing modes on a vibrating string. *J. of the Ac. Soc. of America*, 76(1):5–12.

Miles, J. W. (1965). Stability of forced oscillations of a vibrationg sting. *J. of the Ac. Soc. of America*, 38:855–861.

Murthy, G. S. S. and Ramakrishna, B. S. (1965). Nonlinear character of resonance in stretched strings. *J. of the Ac. Soc. of America*, 38:461–471.

O'Reilly, O. and Holmes, P. J. (1992). Non-linear, non-planar and non-periodic vibrations of a string. *JSV*, 153(3):413–435.

Shankland, R. S. and Coltman, J. W. (1939). The departure of the overtones of a vibrating wire from a true harmonic series. *J. of the Ac. Soc. of America*, 10(3):161–166.

Smith III, J. O. (2006). *Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects*. `http://ccrma.stanford.edu/~jos/pasp/`.

Välimäki, V., Huopanemi, J., Karjalainen, M., and Jánosy, Z. (1996). Physical modeling of plucked string instruments with application ro teal-time sound synthesis. *Hournal of the Audio Engineering Society*, 44(5):331–352.

# Interfaces Musicais não são Interfaces para Músicos: Discussão e Projeto de uma Interface Musical para Leigos

**Luciano Vargas Flores[1], Evandro Manara Miletto[1], Daniel Eugênio Kuck[1], Jérôme Rutily[2], Marcelo Soares Pimenta[1]**

[1]Instituto de Informática – Laboratório de Computação Musical
Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

[2]ENSIMAG – Institut National Polytechnique de Grenoble (INP)
8 Rue Hébert – 91310 Leuville – France

{lvflores,miletto,dekuck,mpimenta}@inf.ufrgs.br,
jerome.rutily@laposte.net

***Abstract.*** *In this paper, some requirements of user interfaces for musical activities are discussed and proposed, stating that they are not the same as interfaces for musicians. We use our experience in developing CODES, a Web-based environment for cooperative music prototyping, to show some examples of how to apply these requirements in the design of a musical interface aiming mainly at novices in music.*

***Resumo.*** *Neste artigo são discutidos e propostos alguns requisitos de interfaces com o usuário para atividades musicais, mostrando que estas não são necessariamente interfaces para músicos. Nossa experiência com o desenvolvimento do ambiente CODES, um sistema baseado na Web para prototipação musical cooperativa, é usada como exemplo para mostrar como aplicar esses requisitos no design de uma interface musical mais orientada a leigos em música.*

## 1. Introdução

Apesar de músicos experientes e amadores já há muito tempo usarem tecnologia para fazer música, isto ainda não soa tão óbvio quando consideramos pessoas leigas em música. Estamos especialmente interessados em facilitar para *qualquer usuário* (leigo ou não) o acesso a experiências musicais envolventes e significativas. Nos últimos anos temos investigado a idéia de interfaces musicais para leigos, aplicando-a no desenvolvimento do CODES – "COoperative Music Prototype DESign", um ambiente baseado na Web para prototipação musical cooperativa.

No CODES, qualquer pessoa (seja leiga em música ou músico experiente) pode criar seu esboço musical (protótipo), que pode então ser repetidamente ouvido, testado e modificado, tanto pelo autor original quanto por seus parceiros remotos, que estarão cooperando no refinamento desse protótipo. Para que isso seja possível, a interface com o usuário (IU) do CODES foi projetada de modo a satisfazer aspectos relacionados a flexibilidade de interação e a usabilidade, além de fornecer apoio adequado para interações com informação musical complexa, atividades de cooperação e para

percepção de grupo (ou "group awareness", provendo mecanismos de apoio ao entendimento das ações e decisões dos membros de um grupo que compartilha e coopera nos protótipos musicais). Deste modo pretende-se que a IU desenvolvida promova uma interação efetiva entre os usuários e destes com o próprio sistema.

O ponto de vista de nossa pesquisa se enquadra na perspectiva apresentada por Iazzetta [1998], pela qual a interação com meta-instrumentos pode se dar por meio de diversas interfaces alternativas (em contraposição à interação com instrumentos tradicionais). Por mais sofisticado que seja o sistema de produção sonora, uma alternativa de interface – para leigos – exige algumas características para este perfil de usuário diferentes das apresentadas nas interfaces específicas para músicos.

## 2. Projetando Interfaces para Leigos em Música

Se o objetivo é projetar a interação de modo que um sistema musical possa ser útil e usável mesmo para não-músicos, acreditamos que o problema precisa ser abordado do ponto de vista da Interação Humano-Computador (IHC), associado a conceitos das áreas de Computação Musical, Novas Interfaces para Expressão Musical (NIME – "New Interfaces for Musical Expression") e até de Trabalho Cooperativo apoiado por Computador (CSCW – "Computer Supported Cooperative Work"), se a cooperação for um dos requisitos almejados. Sob tal perspectiva interdisciplinar, conceitos como usabilidade, flexibilidade de interação e robustez de interação têm de ser aplicados.

Em geral, os sistemas computacionais para música são projetados para o uso por músicos experientes e, salvo raras exceções (p.ex. os sistemas de "networked music" PitchWeb, Daisyphone e PSO – Public Sound Objects), demandam o aprendizado prévio de conceitos e habilidades específicas para uma boa utilização. Além dos músicos, os leigos em música provavelmente também têm interesse em criar e participar de experiências musicais, mas não possuem essas competências e carecem de ambientes orientados ao seu perfil de usuário.

Músicos conhecem teoria musical. Eles sabem como ler partituras, a notação musical tradicional com sua pauta e símbolos musicais. Mais ainda, eles sabem que estes símbolos referem-se a conceitos como notas, pausas e tonalidades, e leigos podem nem mesmo saber do que se tratam esses conceitos. Músicos também possuem instrumentos musicais e sabem como tocá-los. Conseqüentemente, IUs de software musical comum em geral são *interfaces para músicos*, pois se baseiam em representações musicais tradicionais e em metáforas do dia-a-dia de um músico.

Os músicos de hoje também estão acostumados a lidar com novas tecnologias, o que inclui usar equipamento elétrico/eletrônico, conectar esse equipamento, gravar em múltiplas pistas em estúdios, usar sintetizadores, efeitos, e assim por diante. Isto também se reflete, p.ex., em software de seqüenciamento, onde podemos gravar músicas usando diversas "trilhas".

No outro lado dessa discussão encontram-se os leigos interessados em música. Com base na experiência do nosso grupo de pesquisa em aplicar conceitos de IHC na melhoria de interfaces para sistemas musicais, sugerimos aqui alguns requisitos a serem considerados no design de *interfaces para atividades musicais* em geral, de modo a permitir o uso também por leigos além do uso por músicos:

- *Não basear-se fortemente em notação musical tradicional*, nem exigir dos usuários conhecimento prévio de teoria, conceitos e prática musical para que comecem a se envolver com música. Para usuários leigos, sugere-se representar elementos musicais na forma de símbolos gráficos que reflitam em sua aparência as propriedades musicais que representam e permitam sua manipulação, e esta representação não precisa ser necessariamente a notação musical tradicional.
- Empregar *metáforas musicais da vida real* e não metáforas da vida de um músico. Não se pode, por exemplo, basear o design da interface em informações relativas a um instrumento musical específico, pois o usuário leigo pode não dominar a técnica deste instrumento ou mesmo de instrumento algum.
- Utilizar *mecanismos de interação convencionais*. De preferência, não exigir dispositivos muito sofisticados e sim tecnologias cotidianas e de fácil acesso (mouse, teclado, etc.).
- Evitar conflito com tarefas musicais (que envolvem som), ao *evitar o emprego de feedback sonoro* (que não seja o do som sendo criado).
- Oferecer *alternativas de representação/codificação musical*, facilitando aos usuários a exportação/importação de sua música entre sistemas diferentes.
- Não esquecer os demais requisitos tradicionais de usabilidade, que se tornam ainda mais importantes quando o enfoque são os usuários leigos: *facilidade de aprendizado*, *flexibilidade de interação*, *robustez de interação* e *feedback constante* [Dix et al. 1998; Nielsen 1994; Preece, Rogers e Sharp 2005].
- Construa, se possível, um sistema *multiplataforma*, minimizando os requisitos de utilização e, assim, facilitando seu acesso pelos usuários (este é um requisito de arquitetura/implementação, mas que reflete na usabilidade do sistema).
- Finalmente, para sistemas cooperativos como o CODES, uma característica importante da IU deve ser a possibilidade do usuário perceber e analisar as ações dos membros remotos do grupo sobre o objeto no qual estão trabalhando, e conhecer as razões por trás de cada uma dessas ações. Estes aspectos estão relacionados, respectivamente, a *mecanismos de percepção* (“awareness”) e de *argumentação* (“rationale”), os quais devem então ser implementados na IU.

A seguir discutimos alguns aspectos da IU do CODES, mostrando como buscou-se satisfazer os requisitos de interfaces musicais para leigos apresentados aqui.

## 3. O Projeto da Interface para Leigos do CODES

O ambiente CODES utiliza representações musicais alternativas para que não-músicos manipulem elementos musicais. Na IU do CODES os protótipos musicais aparecem formados por linhas de padrões sonoros, que podem ser editadas. Aqui usamos a metáfora de *linhas musicais*, sem chamá-las de “trilhas” ou “pistas”, nem de nenhum nome em especial. Queremos que o usuário entenda estas linhas como ele quiser, adaptando à vontade essa noção ao modelo mental que ele possui da estrutura da música. A edição é feita por manipulação direta na IU, selecionando-se padrões sonoros entre os diversos que estão pré-definidos e disponibilizados na Biblioteca de Padrões Sonoros do CODES. *Padrões sonoros* são estruturas musicais de alto nível (pequenos

trechos em arquivos MIDI) que facilitam o processo de escolha de materiais e de prototipação pelo usuário.

A busca pelos padrões desejados é auxiliada visualmente pela representação dos materiais na forma de *ícones*, cujo desenho possui relação com o som representado e assim facilita a tarefa do usuário leigo, que "lê" um simbolismo musical mais simples de aprender do que a notação tradicional. Nossa idéia com isso é favorecer um processo de experimentação, sem preocupação com o conhecimento teórico. Cada padrão sonoro pode ser ouvido individualmente, antes de ser selecionado e incorporado a uma linha do protótipo. Cada ícone representa o seu padrão sonoro numa espécie de plano cartesiano, como na metáfora tradicional do "piano roll", onde traços horizontais indicam a duração dos sons, e suas posições verticais a variação de altura (Figura 1).

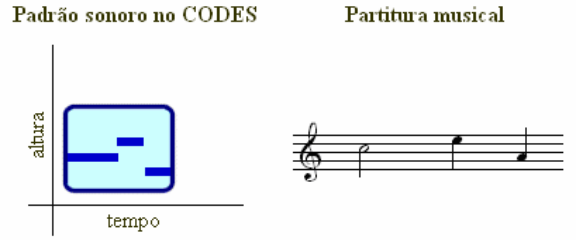



**Figura 1. Representação musical no CODES e a partitura correspondente.**

Para apoiar os aspectos cooperativos da prototipação musical, propomos no CODES três tipos de mecanismos de percepção, ou "awareness" [Dourish e Bellotti 1992]: *Music Prototyping Rationale* (argumentação da prototipação musical); *Action Logging* (registro de ações); e *Modification Marks* (marcas de modificação). No contexto do CODES, a noção adotada de "awareness" é a percepção e compreensão das ações dos outros usuários, o que fornece a um certo usuário um contexto para as suas próprias ações. Maiores detalhes sobre estes mecanismos de percepção do CODES, sua justificativa e seu funcionamento, não serão dados aqui devido à limitação de espaço, mas podem ser encontrados no artigo de Miletto et al. [2006].

## Referências

Dix, A. et al. (1998) "Human-Computer Interaction". Londres: Prentice Hall, 2. ed.

Dourish, P.; Bellotti, V. (1992) "Awareness and Coordination in Shared Workspaces". In: Proceedings of the ACM CSCW '92, Toronto, Canadá. p. 107-114.

Iazzetta, F. (1998) "Interação, Interfaces e Instrumentos em Música Eletroacústica". In: Atas do IHC98 - I Workshop sobre Fatores Humanos em Sistemas Computacionais, Maringá, PR, Brasil. p. 112-120.

Miletto, E. M. et al. (2006) "CODES: Supporting Awareness in a Web-based Environment for Collective Music Prototyping". In: Anais do IHC 2006 - Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais, Natal, Brasil.

Nielsen, J. (1994) "Usability Engineering". San Francisco: Morgan Kaufmann.

Preece, J.; Rogers, Y.; Sharp, H. (2005) "Design de Interação: Além da Interação Homem-Computador". Porto Alegre: Bookman.

# Author Index

## V



## W



## Z