Universidade do Porto

FEUP Faculdade de Engenharia

CONDUÇÃO EM TEMPO-REAL DE ALGORITMOS MUSICAIS:

IMPLEMENTAÇÃO DE UM SISTEMA MUSICAL INTERATIVO COM O KIN.RHYTHMICATOR

Diogo Cocharro
(mm09029)

Versão Final

Dissertação realizada no âmbito do
Mestrado em Multimédia
Perfil: Música Interativa e Design de Som

Orientador: Carlos Guedes

31 de Outubro de 2012

Página em branco

Condução em Tempo-Real de Algoritmos Musicais: Implementação de um Sistema Musical Interativo com o kin.Rhythmicator

Diogo Miguel Filipe Cocharro

Mestrado em Multimédia

Aprovado em provas públicas pelo Júri:

Presidente: Doutora Carla Susana Lopes Morais, Professora Auxiliar convidada da Faculdade de Ciências da Universidade do Porto;

Vogal Externo: Doutor Luís Gustavo Martins, Professor Assistente da Escola das Artes da Universidade Católica Portuguesa;

Orientador: Doutor Carlos Alberto Barbosa da Cunha Mendonça Guedes, Professor Coordenador da Escola Superior de Música, Artes e Espetáculo.

_____

31 de Outubro de 2012

ÍNDICE

ÍNDICE DE FIGURAS ..... **v**
ÍNDICE DE TABELAS ..... **VII**
CAPÍTULO 1 ..... **1**
1.1. Introdução 1
1.2. Motivação 3
1.3. Objetivos da Dissertação 4
1.4. Metodologia e Ferramentas 5
1.4.1. Módulo de análise 7
1.4.2. Módulo de Interação 7
1.5. Ferramentas 7
1.5.1. MaxMSPJitter 7
1.5.2. Max4Live 8
1.5.3. Ableton Live 8
1.5.4. Instrumentos MIDI 8
1.5.5. Controlador Gestual WiiMote (Nintendo) 9
1.5.6. Kin.Rhythmicator 9
CAPÍTULO 2 ..... **10**
2.1. Enquadramento Teórico e Estado da Arte 10
2.2. Sistemas Musicais Interativos 10
2.2.1. Interatividade 10
2.2.2. Human Computer Interaction 15
2.3. Implementação e Design de Sistemas Musicais Interativos 16
2.3.1. Paradigmas de inter-relação em sistemas interativos 16
2.3.2. Etapas e Componentes do desenvolvimento de Sistemas Musicais Interativos 19
2.4. Paradigmas de Interação Musical 22
2.4.1. Mapeamentos (*Mappings*) 22
2.4.2. Score Following 25

2.4.3. Sistemas com Inteligência Artificial e Capacidade de Aprendizagem 30

CAPÍTULO 3 38

3.1. Arquitetura e Implementação do Sistema - Kinductor 38

3.2. Hardware 39

3.3. Software 41

3.3.1. Análise da Performance a Partir de Instrumentos Digitais MIDI 42

3.3.2. Análise da Performance a Partir do Controlador Gestual Wiimote 67

3.3.3. "Conductor" – Condução do Kin.Rhythmicator 71

CAPÍTULO 4 83

4.1. Conclusão e Trabalho Futuro 83

4.1.1. Resumo do Trabalho 84

4.1.2. Resumo das contribuições e limitações 85

4.1.3. Trabalho futuro 86

BIBLIOGRAFIA 88

ÍNDICE DE FIGURAS

Figura 1 - Diagrama geral do projeto desenvolvido, o *Kinductor* 6

Figura 2 - Interface gráfico da aplicação Kin.Rhythmicator 9

Figura 3 - Ciclo de Interação (Bongers, 2000) 18

Figura 4 - Componentes básicos de um sistema musical interativo. (Winkler, 2001) 21

Figura 5 - Esquema geral de um sistema "score following". Este analisa a performance do músico e gera o acompanhamento que coincide com a direção e andamento do músico. (Roads, 1996) 27

Figura 6 - Programação síncrona entre *Antescofo* e o programa *NoteAbility Pro,* mostrando em simultâneo a linha instrumental acompanhada pelos programas a serem executados naquele instante pelas partes eletrónicas. 29

Figura 7 - Esquema geral do sistema GenJam. 33

Figura 8 - Representação do interface do software Cypher (Rowe, 2001, in Winkler, 2001) 36

Figura 9 - Diagrama geral do sistema implementado, o *Kinductor* 39

Figura 10 - Alguns exemplos de instrumentos MIDI (korg nanoseries) 40

Figura 11 - *Osculator*, um exemplo de um programa que permite gerir vários dispositivos por Open Sound Control. 41

Figura 12 - Diagrama geral da estrutura do software implementado. 42

Figura 13 - Diagrama da análise da performance a partir de instrumentos digitais MIDI 43

Figura 14 - (A) Cálculo dos tempos delta entre eventos; (B) Percentagem de desvio; (C) Deteção do valor rítmico e cálculo do desvio em relação ao valor rítmico atual. 45

Figura 15 - Live Quantize: representação do método para associar cada evento rítmico («onset») à grelha temporal. 46

Figura 16 - O patch "Live Performance Quantizer" 48

Figura 17 - Patch responsável por calcular a média de amplitudes ao longo da performance. 49

Figura 18 - Exemplos de cálculos da média de amplitudes: (A) Neste caso a densidade é alta mas a média de amplitudes é baixa 0.18; (B) Neste caso a densidade de eventos é baixa mas a média de amplitudes é alta 0.85. 49

Figura 19 - Visão global do módulo (patch) para medir as variações. 50

Figura 20 - Cálculo das diferenças entre o padrão atual (A) e o padrão passado (B), que foi tocado antes do A. 51

Figura 21 – Representação de duas etapas importantes, (1) filtragem de eventos e cálculo das diferenças e (2) filtragem de variações que não contribuem para o cálculo de variações. 53

Figura 22 - Método 2: cálculo das diferenças entre padrões e filtragem de eventos. Todos os eventos sombreados a verde não foram ignorados, caso contrário estariam a negro. 54

Figura 23 - (A) Estratificação dada pelo objeto [kin.stratify] para um compasso 4/4 ao nível da semicolcheia, 16n; (B) cinzento = nível hierárquico de cada pulsação, amarelo = padrão rítmico tocado, laranja = notas do padrão rítmico que contribuem para a sincopação. 55

Figura 24 – Ex. das listas em rotação sincronizadas ao longo de 5 passos. Cada passo corresponde ao intervalo definido na quantização (16n). A lista amarela corresponde à performance musical e a lista cinzenta corresponde ao modelo hierárquico do compasso musical (4/4) e a laranja corresponde à contribuição de cada pulsação para a sincopação. 55

Figura 25 - Exemplo do cálculo da densidade. Foram detetados 7 eventos e o número máximo de eventos possível são 32. O resultado normalizado da densidade é 0.2 57

Figura 26 - Módulo de "idle state detection" 58

Figura 29 - «TWDA» (A) Cálculo das diferenças entre o novo valor recebido e os valores sucedidos; (B) Armazenamento do novo valor. 62

Figura 30 - Recorte do patch «getStableScore» 64

Figura 31 - Patch «EventsPrediction_v01» 65

Figura 32 - Diagrama geral da análise da performance a partir do controlador Wiimote. 67

Figura 33 - «get_wii_motion_analysis», a abstração que filtra e deteta «onsets» a partir dos dados do wiimote. 70

Figura 34 - Patch para deteção de «onsets» 70

Figura 35 - Interface do «Conductor» versão para instrumento MIDI. 72

Figura 36 – (Acima) «Patch» que calcula a interpolação entre 2 padrões rítmicos; (Abaixo) Exemplo de interpolação entre o padrão rítmico A e B. 74

Figura 37 - Diagrama geral do modo de funcionamento do módulo «Conductor» 76

Figura 38 - Interface dos 6 modos de modulação 77

Figura 39 - Exemplo dos diferentes modos de modulação, acima com o âmbito de modulação definido a 100% e em baixo com o mesmo definido a 50%. 78

Figura 40 - Interface do módulo de configuração do instrumento MIDI e gestual 79

Figura 41 - Interface do módulo de configuração da resolução de quantização. 79

Figura 42 - Interface do módulo de análise. 80

Figura 43 - Interface do módulo «Conductor» 81

ÍNDICE DE TABELAS

Tabela 1 - Classificação das aplicações multimédia (Ribeiro, 2007) 11

AGRADECIMENTOS

“Não há nada que concluamos sozinhos, mesmo que o caminho seja solitário.**”**

Esta página encerra um período da minha vida cheio de peripécias, e desta forma expresso a minha profunda gratidão a todas as pequenas coisas que contribuíram para o culminar deste momento. Há sempre alguém que fica de fora por lapso, peço desde já as sinceras desculpas se deixei alguém excluído. Não querendo fazer destaques especiais nem seguindo nenhuma ordem especifica quero agradecer a todos aqueles que apoiaram, incentivaram, ensinaram, pressionaram, conversaram, amaram, criticaram, cozinharam, limparam, seguraram, esperaram, debateram, acarinharam, ajudaram..... . . . . .

susana pai pedro mãe escrevente eduardo nó ricardo sioros carlos queiroz tyrania silva bolas picado manuel gonçalo guedes rui tiago henrique marques rodrigues cv cocharro magalhães josé gustavo ferreira galveias costa telmo george catela salvador da neta noémia fonseca kinetic dias flausino deolinda queimado casa clites páteo café

ABSTRACT

Musical interaction is reciprocal communication. Computer by itself doesn't have any listening skills.

This thesis is about the study of conducting in real-time automatic music generation algorithms with MaxMSP, and the development of a musical interaction system with *kin.rhythmicator* as a case study. *Rhythmicator* is a stochastic rhythm generator that creates rhythm in real-time based in meter.

It was intended to develop several tools in MaxMSP for conducting *kin.rhythmicator* performance in real-time through the user performance with a MIDI instrument or a gestural controller like wiimote. The project is divided in two stages, develop some listening capabilities for the computer and apply this analysis data as performance conductor.

CAPÍTULO 1

1.1. INTRODUÇÃO

O interesse neste tema para uma dissertação de mestrado surge no âmbito de um longo percurso pessoal entre música e tecnologia o qual culminou em 2011 com a participação no projeto de investigação *Kinetic* que decorreu no INESC[1], no qual, entre outras tarefas, tinha como função testar as versões *beta* do software de geração automática de ritmo *Kin.Rhythmicator*.

Esse trabalho consistiu na utilização prática da forma mais exaustiva possível do *kin.rhythmicator* enquanto este se encontrava ainda na sua fase de desenvolvimento, como por exemplo compor música, preparar demonstrações, detetar *bugs*, e por fim manipulá-lo utilizando controladores ubíquos.

Esta ultima tarefa baseou-se em diversos testes e experiências na manipulação da performance em tempo-real do software *Kin.Rhythmicator* tendo em conta as suas idiossincrasias, introduzindo o ponto de partida para o tema e motivação desta tese.

Este trabalho discute o desenvolvimento de uma aplicação em MaxMSP que permite interagir em tempo-real com o gerador de ritmo *kin.rhythmicator*. Este sistema oferece ao computador algumas capacidades de ouvinte, ou seja, efetua uma análise em tempo-real da performance musical e utiliza essa informação para interagir com a performance do *kin.rhythmicator*.

Começa-se por enquadrar teoricamente o contexto deste projeto e apresenta-se várias definições e pontos de vista do estudo da interação musical.

---

[1] INstituto de Engenharia de Sistemas e Computadores

Aborda-se também alguns dos componentes e etapas de desenvolvimento de um sistema musical interativo e quais os paradigmas de interação musical que existe atualmente.

Faz-se também uma apresentação do software musical utilizado na performance em tempo-real e das ferramentas utilizadas para este projeto e por fim também é apresentado o *Kin.Rhythmicator* que foi objeto de estudo das experiências e implementações efetuadas.

A segunda parte desta dissertação aborda os métodos e experiências efetuados na performance em tempo-real com o *Kin.Rhythmicator*, apresentando-se detalhadamente o software desenvolvido para este projeto.

Por último, discute-se a experiência e conclusões obtidas no desenvolvimento deste projeto, e fazem-se sugestões para futuro trabalho.

Por vezes a tradução do jargão técnico de certas áreas de estudo pode ser difícil e até ser dúbio, por isso optou-se por um código que permita distinguir tais palavras ao longo deste trabalho.

- Sempre que ocorrer uma referência a um objeto de MaxMSP a palavra aparece entre parêntesis retos. Ex. [multislider]
- Sempre que se recorrer a uma palavra de jargão técnico esta aparece entre aspas. Ex. «onset»

1.2. MOTIVAÇÃO

Existe uma série de fatores que conduzem à seleção deste tema, a "condução em tempo-real de algoritmos generativos", utilizando como objeto de estudo o *kin.rhythmicator*. São fatores pessoais que consistem num percurso ligado à música e à tecnologia e fatores académicos/profissionais, como já foram descritos na introdução, nomeadamente a participação no projeto *Kinetic* .

A nível pessoal existe uma ligação à música e à performance musical ao longo de vários anos como músico, com a participação em diversos projetos musicais, os quais têm permitido desenvolver francamente o interesse por esta área e por tudo o que a rodeia.

Paralelamente, existe um interesse ligado à tecnologia musical, que se estende desde as tecnologias de captação e gravação de som, e tecnologias de produção, até às tecnologias associadas a sistemas musicais interativos, desde controladores, sensores, algoritmos, computadores, etc.

A colaboração no projeto de investigação *KINETIC* que envolveu o INESC e a FEUP permitiu um contacto muito próximo com algumas das vertentes do projeto, nomeadamente o *Kin.Rhythmicator*, este gerador estocástico de ritmo que proporciona e estimula uma nova abordagem à criação musical, e que incentiva a percorrer caminhos novos e desconhecidos no mundo da performance musical, seja ela improvisação ou não.

Por esta razão a escolha deste tema tornou-se óbvia e natural.

1.3. OBJETIVOS DA DISSERTAÇÃO

A interação é uma forma de comunicação recíproca, e o computador por si só não possui tais capacidades, sobretudo no que respeita à criação musical.

Um dos principais objetivos para esta tese será estudar a interação musical entre músico e computador implementando um sistema que permita controlar a aplicação *kin.rhythmicator* a partir de uma performance musical.

Esse sistema deverá oferecer ao computador algumas capacidades de "ouvinte" que permitam que este explore e analise a performance musical humana, dada a partir do sinal áudio de um instrumento, sensores, instrumentos MIDI ou controladores gestuais.

Com o desenvolvimento de algumas capacidades de "ouvinte" no computador, pretende-se extrair informação que seja musicalmente relevante, e a partir dessa análise gerar os parâmetros que serão mapeados ao gerador de ritmo *Kin.Rhythmicator*, de forma que interaja musicalmente em resposta à performance, criando um canal de influência mútua entre *performer* e *Kin.Rhythmicator*.

Na base da sua conceção, o *kin.rhythmicator* não necessita de controlo direto do utilizador para realizar a sua função - gerar ritmo, pois trata-se de um gerador estocástico de ritmo que através de um interface muito acessível a qualquer utilizador, permite que este defina todos os parâmetros do ritmo sem necessidade de introduzir padrões rítmicos nem de definir tabelas de probabilidades. Basta dizer à aplicação o contexto musical onde este está inserido, por exemplo num compasso 4/4, 7/8, ou 3/4, ou qualquer outro compasso, e o tipo de articulação do ritmo, etc. O *kin.rhythmicator* adapta automaticamente todos os parâmetros para gerar ritmo nesse contexto.

Espera-se também que os resultados desta pesquisa possam ser adaptados a outros contextos musicais, por exemplo manipulação em tempo-real de material musical preconcebido ou gerado automaticamente pelo computador através de outras aplicações. Por exemplo, também a manipulação de parâmetros de efeitos e síntese sonora poderá reagir à performance do músico e não a mapeamentos diretos a controladores.

Numa fase posterior pretende-se implementar estes algoritmos na API *Max4Live* introduzida recentemente pela parceria entre *Ableton* e *Cycling'74*. Esta plataforma permite combinar os pontos fortes do *Ableton Live* para organizar material musical e controlo de processamento com as do *MaxMSPJitter* (interligar sensores, realizar análise de sinal, e criação de algoritmos interativos), proporcionando também um contexto mais prático destes algoritmos tendo em conta a vasta comunidade de utilizadores do *Ableton*.

Assim, explorando estas tecnologias procura-se encontrar respostas a algumas questões:

- Esta prática constitui um novo paradigma de composição/sequenciação utilizando o Live e o MaxMSP em conjunto? Se sim quais as características?
- Quais as vantagens e desvantagens da utilização de sistemas musicais interativos?
- Quais as dificuldades e cuidados a ter no design de sistemas musicais interativos?
- Como controlar algoritmos generativos em MaxMSP?
- Que tipo de mapeamentos e manipulação se pode dar aos dados da análise?
- Como extrair informação musical relativa à perceção humana?
- Como aplicar estes dados no controlo de algoritmos musicais generativos?

1.4. METODOLOGIA E FERRAMENTAS

O projeto desenvolvido para esta dissertação é dividido em diversos módulos ou subsistemas, que em conjunto formam a aplicação que constitui o sistema para interagir com *Kin.Rhythmicator*. Uma das estratégias implementadas nesta aplicação é a representação abstrata da performance rítmica através de atributos que se relacionam com a perceção humana em vez da utilização da linguagem e símbolos musicais típicos.

> *"Objects in the visual world have six perceptual attributes: size, color, location, orientation, luminance, and shape. What do we mean by "object"? (…) I propose that an object is something that maintains its identity across changes (or transformations) in these attributes. In other words, as we move an object through space, it is still the same object. (…)*
>
> *A performance of music contains the following seven perceptual attributes: pitch, rhythm, tempo, contour, timbre, loudness, and spatial location (one might add reverberant environment as an eighth). Technically speaking, pitch and loudness are psychological constructs that relate to the physical properties of frequency and amplitude. The term contour refers to the shape of a melody when musical interval size is ignored, and only the pattern of "up" and "down" motion is considered. Each one of these eight attributes can be changed without changing the others. With the exception of contour, and sometimes rhythm, the recognizability of the melody is maintained when each of these attributes is changed. In fact, for many melodies, even the rhythm can be changed to some degree and the melody will still be recognizable. (…)"*
>
> (Cook, 2001)

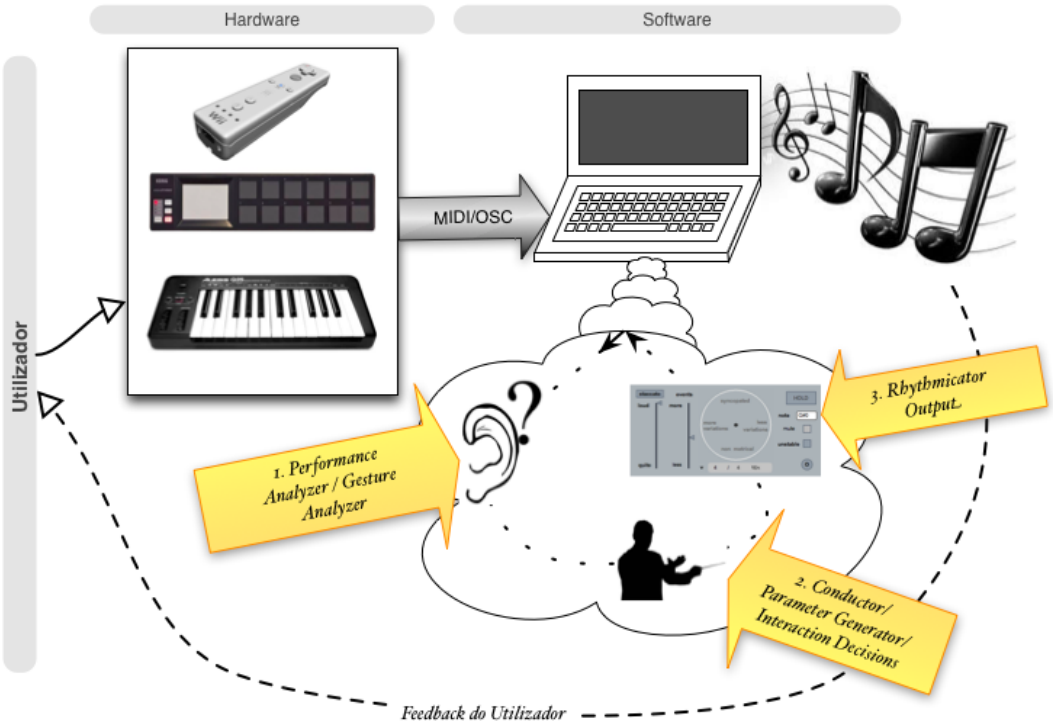


Figura 1 - Diagrama geral do projeto desenvolvido, o *Kinductor*

A abordagem implementada para análise da performance inspira-se na memória sensorial humana, que nos auxilia na perceção do mundo.

> *One kind of memory is the immediate sensory memory we experience as image persistence. For example, if you look outside the window on a bright day and then close your eyes, an afterimage stays on your retine for a few moments. This has been called iconic memory by Ulric Neisser (1967). We talk about the auditory equivalent of this as echoic memory: for a few moments after hearing a sound (such as a friend's voice) we are usually able to "hear" a trace of that sound in our mind's ear. (...)*
>
> *When you are holding a thought inside your head such as what you are about to say next in a conversation, or as you're doing some mental arithmetic - it stands to reason that this requires some type of short-term, or immediate, memory. This kind of memory, the contents of your present consciousness and awareness, has been called "working memory" by Alan Badelley (1990), and is similar to what Atkinson and Shiffrin called short-term memory.*
>
> *Long-term memory is the kind of memory that most of us think of as memory - the ability to remember things that happened some time ago, or that we learned some time ago (usually more than a few minutes ago, and up to a lifetime ago).*
>
> (Cook, 2001)

1.4.1. MÓDULO DE ANÁLISE

Este módulo é responsável por extrair informação de baixo nível relativa à perceção humana da performance rítmica, isto é, a performance é descrita através de atributos abstratos que de certa forma se assemelham à descrição que a perceção humana faz das coisas que vemos e ouvimos.

Este facto é notoriamente observado nas crianças, por exemplo ao nível da melodia, estas respondem primeiro ao contorno antes de adquirirem capacidade de percecionar a melodia, isto é, estas não distinguem entre uma canção e uma alteração melódica dessa canção se o contorno melódico for preservado. O contorno de uma melodia é processado mais rapidamente do que os intervalos melódicos, logo é mais fácil de ser recordado. (Cook, 2001, p. 216)

1.4.2. MÓDULO DE INTERAÇÃO

Este módulo gere os mapeamentos, reações, e interpretação dos dados da análise. É neste módulo que serão concebidas as relações entre a análise da performance e a geração de valores de parâmetros que serão enviados para o kin.rhythmicator.

1.5. FERRAMENTAS

1.5.1. MAXMSPJITTER

(Cycling74, 2012)

O MaxMSP é uma linguagem de programação visual desenvolvida pela Cycling74 orientada para músicos, artistas visuais e multimédia. Esta é modular tendo em conta a forma como os programas são desenvolvidos interligando objetos num formato que chamam " patches ". O MaxMSP é expansível proporcionando aos programadores uma API[2] em linguagem C para desenvolver os chamados objetos externos.

[2] Application Programming Interface

Originalmente o MaxMSP foi desenvolvido no IRCAM[3] por *Miller Puckette* na década de 80 para dar acesso aos compositores a sistemas musicais interativos com o computador. Em 1989 foi adaptado ao computador NeXT e às unidades de processamento ISPW (*IRCAM Signal Processing Workstation*), que mais tarde evoluiu para MAX/FTS (*Faster Than Sound*). Ainda em 1989 a licença passou para a empresa de *David Zicarelli*, a *Opcode Systems* que lançou uma versão comercial em 1990. Depois em meados dos anos 90 a empresa cessou atividade e desde então passou a ser mantido e desenvolvido pela *Cycling74*.

(Wikipedia, 2012)

1.5.2. MAX4LIVE

O *Max4Live* é um dos produtos desenvolvidos pela *Cycling74*. Foi desenvolvido em colaboração com a empresa *Ableton*, famosa pelo seu sequenciador/instrumento de produção musical *Live*. Esta plataforma permite importar para o *Ableton Live* as potencialidades do MaxMSP. Isto significa que o utilizador pode criar os seus sintetizadores, sequenciadores personalizados, manipular MIDI[4], e criar efeitos áudio e interligar sensores e controladores.

(Cycling74, 2012)

1.5.3. ABLETON LIVE

O *Ableton Live* é um sequenciador para produção musical desenvolvido e mantido pela empresa alemã *Ableton*, que tem a particularidade de poder ser utilizado também como instrumento. (Wikipedia, 2012)

1.5.4. INSTRUMENTOS MIDI

- Teclados, pads, guitarra MIDI, etc. Praticamente qualquer instrumento MIDI pode ser utilizado.
- Sendo o MIDI uma norma padrão, à partida a aplicação não terá qualquer problema em funcionar com qualquer instrumento MIDI, desde teclados a *pads* de percussão, guitarras MIDI, etc.
- Hoje em dia existe uma grande variedade de instrumentos MIDI, ou sistemas MIDI que podem ser adaptados a outros instrumentos convencionais.

[3] *Institut de Recherche et Coordination Acoustique/Musique*

[4] Musical Instrument Digital Interface

1.5.5. CONTROLADOR GESTUAL WIIMOTE (NINTENDO)

Este foi o primeiro controlador gestual desenvolvido para a consola Wii da Nintendo. A principal característica é a sua capacidade de captura de movimentos através de sensores, mas também permite apontar através de um sensor ótico. (Wikipedia, 2012)

1.5.6. KIN.RHYTHMICATOR

Kin.Rhythmicator trata-se de uma aplicação que gera automaticamente ritmo baseado na assinatura de compasso definida, constituindo uma das principais ferramentas desta dissertação, sendo que o titulo se encontra diretamente relacionado com esta aplicação, a partir da qual se realizam experiências e se desenvolve este projeto.

Esta aplicação foi desenvolvida no âmbito do projeto "Kinetic – Gestural controller-driven, adaptive, and dynamic music composition systems" que envolveu entre outros parceiros a FEUP e o INESC.

Os ritmos gerados em tempo-real não têm em consideração o estilo musical, estes são antes característicos do compasso musical selecionado dentro do contexto musical (peça, musica, ou instalação). O algoritmo é baseado num modelo estocástico do compasso musical que permite criar uma distribuição probabilística de cada divisão do compasso. Este permite, através de um interface acessível a qualquer utilizador (quer tenha experiência musical ou não), controlar, facilmente e em tempo-real, parâmetros e qualidades do ritmo que está a ser gerado, como por exemplo, densidade de eventos dentro do compasso musical, quantidade de variações e sincopação e a força métrica do ritmo gerado.

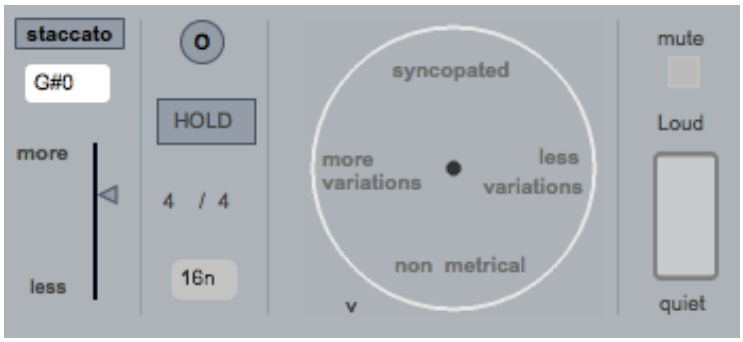


Figura 2 - Interface gráfico da aplicação Kin.Rhythmicator[5]

[5] (Sioros & Guedes, Automatic Rhythmic Performance in Max/MSP: the kin. rhythmicator, 2011)

CAPÍTULO 2

2.1. ENQUADRAMENTO TEÓRICO E ESTADO DA ARTE

Esta secção serve como introdução a conceitos e problemáticas relacionados com sistemas interativos dentro dos quais se tem levantado algumas discussões. É também apresentado alguns dos projetos que têm servido como ponto de referência neste campo de investigação e que servem também como referência e inspiração a esta tese.

O controlo de algoritmos musicais tem sido bastante explorado, e existe uma grande variedade de projetos que abordam diferentes técnicas e estratégias. Todos os algoritmos musicais generativos têm características muito próprias, o que dificulta a criação de um modelo genérico para interação com estes. As técnicas utilizadas têm que ser adaptadas às idiossincrasias de cada sistema musical interativo.

2.2. SISTEMAS MUSICAIS INTERATIVOS

2.2.1. INTERATIVIDADE

Atualmente, a vasta capacidade de resposta dos computadores para processamento de dados em tempo-real tem incentivado o seu uso em diversos contextos e com diversas tecnologias multimédia, (entretenimento, artes digitais, áudio, vídeo, animação), contribuindo assim para que a palavra interatividade apareça associada a este dentro dessa diversidade de aplicações.

Graças às tecnologias multimédia[6] e ao desenvolvimento de aplicações multimédia[7], as aplicações interativas têm vindo a proliferar, uma vez que permitem uma apresentação dos conteúdos de forma não linear, em que o utilizador desempenha um papel ativo. (Ribeiro, 2007)

As tecnologias multimédia em conjunto com aplicações interativas podem ser aplicadas numa grande variedade de situações, por essa razão podem ser classificadas de acordo com diversos critérios, por exemplo o tipo de utilizador (ex.: crianças ou adultos), mercado a que se destina (ex.: doméstico ou profissional), e a área de utilização das aplicações na atividade humana. Nuno Ribeiro (2007) propõe a seguinte classificação das aplicações multimédia:

Área de utilização	Exemplos	Tipos de aplicações multimédia
Educação	Universidades Escolas Em casa	Livros eletrónicos Aplicações de ensino interativo Aplicações de ensino à distancia (E-learning)
Empresarial	Indústria Serviços	Aplicações de formação profissional Aplicações de vendas interativas e marketing Apresentações e comunicações multimédia
Entretenimento e Lazer	Em casa	Revistas e jornais eletrónicos Jogos interativos Aplicações de televisão interativa Aplicações de realidade virtual Aplicações musicais interativas
Informação ao Público	Locais públicos	Quiosques multimédia

Tabela 1 - Classificação das aplicações multimédia (Ribeiro, 2007)

A utilização dos computadores no processo de criação e performance musical tem desvendado novos caminhos e criado novos desafios sociais e culturais nesta prática.

Esta dissertação pretende focar-se sobretudo em sistemas musicais interativos. Esta prática permite explorar técnicas especificas de interação, ao mesmo tempo que desenvolve novas e desafiantes formas de interação entre humanos e computadores.

[6] Tecnologias Multimédia: Ferramentas que auxiliam o utilizador na divulgação de ideias, conceitos ou serviços. (Ribeiro, 2007) Os média incluem texto, som, imagem, etc.

[7] Aplicação multimédia: Programa ou aplicação informática que assiste o utilizador na consulta da informação multimédia. (Ribeiro, 2007)

Segundo a classificação proposta por Nuno Ribeiro (2007), pode-se considerar a sua área de utilização como Entretenimento e Lazer destinado ao público, visto que pode ser aplicado na performance ao vivo ou em instalações. De qualquer forma, o trabalho desenvolvido com sistemas musicais interativos também pode ser adaptado a um contexto educacional, para o ensino e desenvolvimento musical.

Segundo o dicionário da Priberam, pesquisando as palavras interativo, interação e interatividade surge diversas definições, cada uma com ligeiras diferenças conforme o contexto onde cada palavra é apresentada.

> *Interativo (adjetivo):*
>
> *"Diz-se de um suporte de comunicação que favorece uma permuta com o público."*
>
> *"Diz-se de fenómenos que reagem uns sobre os outros."*
>
> *Interação (substantivo):*
>
> *"Influência recíproca de dois ou mais elementos."*
>
> *Interatividade (substantivo):*
>
> *"Faculdade de permuta entre o utilizador de um sistema informático e a máquina, por meio de um terminal dotado de um ecrã de visualização."*
>
> (Priberam, 2011)

Em qualquer uma das definições observa-se que para que haja interação, tem de haver um ou mais canais de comunicação, que permitam que esta ocorra em dois sentidos, e a participação de dois ou mais intervenientes. Desta forma uma conversa entre duas pessoas pode ser considerada um exemplo de interação, enquanto se apenas uma das pessoas falar e a outra não, já não há interação, poderá ser apenas um discurso. Assim como suporte de comunicação temos a fala e a audição, o idioma e os sujeitos intervenientes na conversa.

Ao falar de sistemas interativos utilizando a intervenção de computadores, Todd Winkler refere 3 características que ocorrem na interação com computadores (Winkler, 2001, p. 3):

- *Interaction is a two-way street (…);*
- *Computers simulate interaction (…);*
- *Interaction means action(…)*

A primeira característica refere-se ao exemplo já mencionado de duas pessoas conversando, partilhando palavras e ideias, demonstrando a comunicação em dois sentidos e a forma como os conteúdos evoluem consoante a interpretação da informação. A interação é entendida como uma forma de comunicação recíproca. Como exemplo desta dialética pode-se considerar os dados que o utilizador fornece (*inputs*) ao sistema através de sensores ou de um instrumento musical, ou no caso do exemplo anterior temos a mensagem de um dos interlocutores da conversa. Por sua vez, a informação recebida é processada segundo as regras definidas no sistema como uma transformação ou novo estimulo para o utilizador. Este obtém esse resultado através do interface escolhido para o sistema, por exemplo os altifalantes, ou no caso do exemplo acima a interpretação da mensagem e o envio da resposta da parte do interlocutor que a escutou. Este processo encerra um ciclo que influência simultaneamente o utilizador e o sistema. Os processos de tratamento da informação podem variar bastante conforme os objetivos pretendidos para o sistema e o comportamento desejado para este. Esses processos podem incluir diversos tipos de software e algoritmos, mapeamentos, algoritmos de aprendizagem e de inteligência artificial e/ou algum tipo de cognição, influenciando assim a forma como a informação recebida é interpretada.

A segunda característica refere-se à forma como acontece interatividade nos computadores, na medida em que estes permitem que o utilizador altere certos parâmetros do seu estado atual e comportamento, sendo que o ciclo de interação dá-se por sua vez quando os computadores afetam as ações futuras dos seus utilizadores.

No entanto essa relação de interatividade depende sempre das funcionalidades, algoritmos implementados e tecnologias utilizadas por quem desenvolve o sistema interativo com computadores, tal como descreve Nuno Ribeiro (2007):

> *Um sistema interativo permite que o utilizador controle o conteúdo e o fluxo de informação do sistema, contudo a quantidade de controlo que é oferecida ao utilizador não é ilimitada, pois esse controlo encontra-se restringido pelos parâmetros que os autores do sistema incluem no projeto, e que ficam embebidos no código do programa que controla a combinação e a apresentação dos media.*

A simulação de interação depende dos processos utilizados para processar os dados que entram no sistema (*inputs*), e do tipo de comportamento e cognição desejado pelos autores para o sistema.

A terceira característica refere-se aos programas informáticos, isto é, estes de uma forma geral contêm um certo grau de interatividade, dependendo da forma como respondem às ações humanas e o grau de influência e incentivo nas reações dos utilizadores.

> *"Interactivity comes from a feeling of participation, where the range of possible actions is known or intuited, and the results have significant and obvious effects, yet there is enough mystery maintained to spark curiosity and exploration." (Winkler, 2001, p. 3)*

> *"Na sua essência, os sistemas musicais interativos compreendem as várias formas pelas quais se permite que o utilizador* (performer, músico ou não músico) *se relaciona com a informação* (neste caso informação musical)*, sendo este relacionamento mediado pelo computador." (Ribeiro, 2007)*

Todd Winkler apresenta a seguinte definição de música interativa:

> *Interactive music is defined here as a music composition or improvisation where software interprets a live performance to affect music generated or modified by computers. (Winkler, 2001)*

E Joel Chadabe apresenta uma distinção entre os conceitos peça e atividade:

> *"A 'piece', whatever its content, is a construction with a beginning and end that exists independent of its listeners and within its own boundaries of time. An 'activity' unfolds because of the way people perform; and consequently, an activity happens in the time of living; and art comes closer to life." (Chadabe, 2001)*

Sobrepondo as definições de ambos os autores podemos considerar a música interativa (que tem a particularidade de ocorrer em tempo-real) como uma atividade onde para conquistar vida exige a participação obrigatória dos intervenientes, os *performers* (músicos ou não), e que estes façam parte de uma rede interligada por um sistema que faz mediação e se influência mutuamente.

> *"Here's a case in point from some recent work. I composed 'Many Times ...', where the elipsis is the name of the performer, for a concert of my music at Engine 27, a sound gallery / performance space in New York City, in the spring of 2001. The underlying idea of the use of technology is that technology expands our capabilities. In 'Many Times ...', the instrument takes the sound produced by a performer and from it produces many different transformed instances of it throughout the performance space, multiplying the performer's actions so that it comes from loudspeakers on the left, on the right, above, behind, from here, there, everywhere.*
>
> *Those transformations, generated by the software that animates the instrument, are essentially unpredictable. Because of their unpredictability, they provide the performer with something to react to. In other words, the performer is influencing the electronic system by performing, vocally or by playing an acoustic instrument, and the electronic*

system is influencing the performer by giving the performer something to react to. This is what I call 'interactivity', where the word interactive means 'mutually influential'. I find it a wonderful way to make music. It brings out the best in everyone. And considering that anyone can play the role of performer, it could bring out the best in anyone."

(Chadabe, 2001)

2.2.2. HUMAN COMPUTER INTERACTION

A utilização de computadores para design de sistemas musicais interativos envolve também a área de estudo da Interação Humano-Computador (*HCI - Human Computer Interaction*), tendo em conta as entidades que podem estar envolvidas num sistema deste tipo, isto é, performer (pessoas, utilizadores, audiência), sistema (hardware, software, interface).

Wikipédia apresenta a seguinte definição de HCI:

> *Human–computer interaction (HCI) is the study, planning and design of the interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioral sciences, design and several other fields of study. Interaction between users and computers occurs at the user interface (or simply interface), which includes both software and hardware; for example, characters or objects displayed by software on a personal computer's monitor, input received from users via hardware peripherals such as keyboards and mice, and other user interactions with large-scale computerized systems such as aircraft and power plants. The Association for Computing Machinery defines human-computer interaction as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."* *(Wikipédia, 2011)*

É importante realçar o facto de HCI ser uma área de estudo onde ocorre a interseção de diferentes áreas de estudo, contribuindo certamente para o desenvolvimento de sistemas interativos. Por exemplo em Sistemas Musicais Interativos podemos considerar várias disciplinas como composição, psicoacústica, *sound design*, ciências computacionais, programação, design de interfaces, eletrónica (sensores) e síntese sonora.

2.3. IMPLEMENTAÇÃO E DESIGN DE SISTEMAS MUSICAIS INTERATIVOS

> *"O design de um sistema interativo pode dividir-se em duas grandes fases de desenvolvimento: a interação física entre pessoas e o sistema, e o comportamento interativo como resultado cognitivo da máquina."* (Bongers, 2000)

Bongers (2000) divide o design de sistemas interativos em duas fases. A primeira relaciona-se com o tipo de hardware utilizado/desenvolvido para o sistema, e o contexto físico e social da interação. O sistema comunica com o seu ambiente através transdutores e atuadores (*transducers and actuators*), estes transformam sinais do mundo real para o domínio do sistema e vice-versa.

A segunda fase de desenvolvimento está relacionada com o software e processos que processam os dados (*inputs*) do utilizador e o *feedback* que este obtém. Nesta fase encontra-se, por exemplo, os algoritmos desenvolvidos/utilizados para o sistema.

Bert Bongers (2000) tal como Todd Winkler (2001) referem que interação é um processo com dois sentidos. No entanto Bongers descreve esse processo utilizando as palavras controlo e feedback. O controlo ganha forma através de um instrumento ou interface, que atua como um transdutor das ações do mundo real para o domínio digital no caso de o sistema ser um computador. Os sensores (transdutores) funcionam assim como os órgãos dos sentidos, permitindo que a máquina comunique com o mundo exterior.

Segundo Bert Bongers (2000) o sistema é controlado pelo utilizador, e este pode fornecer dois tipos de resposta ao utilizador, uma resposta do tipo *feedback* ou *feedforward*. *Feedback* ajuda o utilizador a articular o controlo, ou seja, trata-se de uma reação a um estímulo. *Feedforward* guia de forma ativa o utilizador, a resposta é gerada ativamente pelo sistema de forma a revelar informação sobre o seu estado interno, ou por outras palavras indicando o caminho ao utilizador.

2.3.1. PARADIGMAS DE INTER-RELAÇÃO EM SISTEMAS INTERATIVOS

Bert Bongers (2000) identifica a interação entre humano e sistemas eletrónicos segundo várias categorias:

- *Performer* - Sistema

- Ex: um músico a tocar um instrumento;

- Sistema - audiência
  - Ex: uma instalação artística;

- *Performer* - Sistema - Audiência

- A interação entre *Performer* - Audiência pode estar sempre presente, mas neste caso não existe a mediação através de um sistema eletrónico;

- A interação entre *Performers* pode ocorrer num *ensemble* ou numa banda, com ou sem mediação por um sistema eletrónico. (ex: The Hub; Sensorband utilizam um sistema como mediação da interação);

Joel Chadabe (2001) define a relação entre o *performer* e as suas "atividades", neste caso o sistema, da seguinte forma:

> *"How does art function in the life of the performer that is participating in one of my musical activities? Well, for one thing, I view the performer as a human being and friend, and not as the executer of a construction; and so I try to design a role for the performer that is challenging, creative, and comfortable. By challenging, I mean that I put the performer in a situation where the demands on the performer's listening and reacting skills are not routine. By creative, I mean that the performer has to make compositional decisions, as against, in a more traditional role, executing the notes in a musical score. By comfortable, I ask the performer to participate in a way that the performer finds agreeable, both physically and musically." (Chadabe, 2001)*

A interação entre humano e sistema deve influenciar mutuamente ambas as entidades. Essa relação é ilustrada por Bert Bongers como um ciclo de interação (*interaction-loop*).

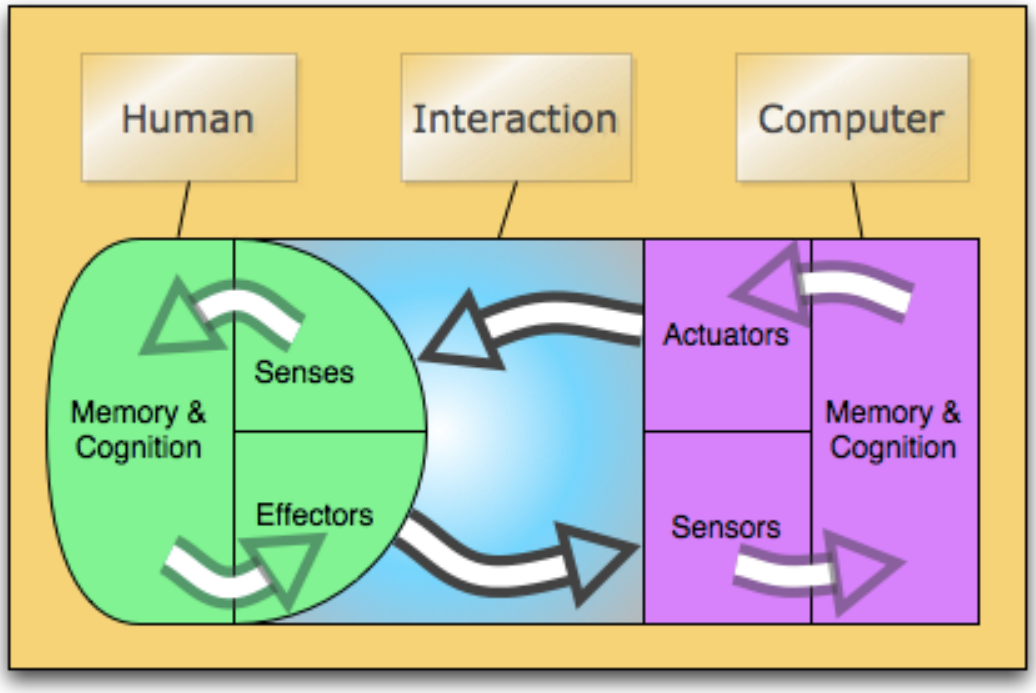


Figura 3 - Ciclo de Interação (Bongers, 2000)

Por definição as partes quadradas correspondem à maquina e as partes arredondadas correspondem ao humano. A máquina ou sistema pode consistir em vários elementos interligados, como por exemplo o caso de computadores em rede, protocolos como o MIDI[8] ou OSC[9]. O sistema pode também referir-se a um instrumento musical. (Bongers, 2000)

O ciclo de interação pode iniciar-se quando o utilizador quiser ativar o sistema, o qual pode ser controlado pelos *inputs* dados pelo utilizador. O sistema processa então a informação e apresenta o resultado.

Por exemplo, ao pressionar uma tecla num instrumento eletrónico musical, o som é apresentado pelos altifalantes, o humano (utilizador) perceciona essa informação do sistema, processa-a e controla-a novamente.

[8] **Musical Instrument Digital Interface**, protocolo de comunicação que permite a interação entre diferentes instrumentos ou componentes.

[9] **Open Sound Control**, protocolo de comunicação para dispositivos multimédia. Desenvolvido em CNMAT, este protocolo apresenta melhores vantajagens que o MIDI devido à maior resolução, rapidez, precisão, capacidade organização e flexibilidade para aplicar em diferentes tipos de dispositivos e redes.

Ao retirar cognição de um dos lados do ciclo de interação, essa parte reage em vez interagir, neste caso só ocorre uma parte do ciclo.

Muitos sistemas interativos nas artes dos novos media são na realidade sistemas reativos, o ideal será a interação influenciar ambos humano e sistema mutuamente. (Bongers, 2000)

2.3.2. ETAPAS E COMPONENTES DO DESENVOLVIMENTO DE SISTEMAS MUSICAIS INTERATIVOS

O design de um sistema musical interativo implica a interligação de diferentes sistemas e componentes de forma a que o sistema apresente uma resposta credível, isto é, que pareça apropriada para a ação executada e apropriada ao estilo de música.

> *"Interactive software simulates intelligent behavior by modeling human hearing, understanding and response."*
>
> *(Rowe, Machine Musicianship, 2001)*

Este processo é de certa forma análogo às atividades distintas que ocorrem durante uma improvisação jazz ou em qualquer outro idioma musical: ouvir, interpretar, compor, e tocar/executar.

Todd Winkler (2001) identifica os seguintes componentes num sistema musical interativo:

1. Performance humana, instrumentos: a atividade humana é traduzida para informação digital e enviada para o computador;
2. Análise da performance, audição computacional: o computador recebe os dados da performance humana e analisa-os podendo extrair diferentes tipos de informação como altura, dinâmica, ritmo, etc;
3. Interpretação: o software interpreta a informação na análise gerando novos dados que irão influenciar a composição ou performance musical;
4. Composição computacional: processos computacionais responsáveis por todos os aspetos da composição algorítmica, baseiam-se nos resultados da interpretação da performance;

5. Performance, geração de som: o computador toca a música utilizando sons gerados internamente ou então envia a informação musical para outros dispositivos que geram som.

Winkler (2001) faz ainda uma separação destas componentes, considerando as duas primeiras como questões práticas, que estão relacionadas com factos e precisão das análises e as três ultimas componentes, lidam com questões artísticas, limitadas apenas pelo talento e imaginação dos compositores.

A respeito do software, o interface e o instrumento, Joel Chadabe (2001) faz o seguinte comentário:

> *"The musical 'instrument' I use is defined by the software I design. It is the software that articulates the interface between instrument and performer, determines how the instrument will react to a performer's actions, and generates the sounds. The activitity that I design, then, is defined by the interface, the way the instrument responds to a performer, and the nature of the sounds. In short, as against a musical score that is played on an instrument, in my music it is the instrument itself that is the work of art. The instrument is inseparable from the music it produces. As Yeats wrote, "How can we know the dancer from the dance?" (Chadabe, 2001)*

O interface faz parte do sistema e possibilita a comunicação entre o utilizador e o sistema, sendo constituído pelos sensores e os atuadores (Bongers, 2000). Os sensores podem ser considerados como os órgãos dos sentidos, na medida em que possibilitam e atuam como transdutores do mundo real para energia elétrica de forma que possa ser lida pelo sistema. Existe uma grande variedade de sensores que permitem captar a mesma informação que os humanos conseguem percecionar e até outros dados que não são percetíveis aos humanos.

Os atuadores têm a função oposta dos sensores, estes convertem a energia elétrica do sistema para outras formas que possam ser percecionados pelos humanos. Por exemplo, um altifalante que emite som ou um projetor de vídeo.

Em suma, um sistema interativo musical funciona utilizando um computador para interpretar a performance musical que posteriormente poderá influenciar parâmetros musicais ou ser utilizado para gerar música em tempo-real.

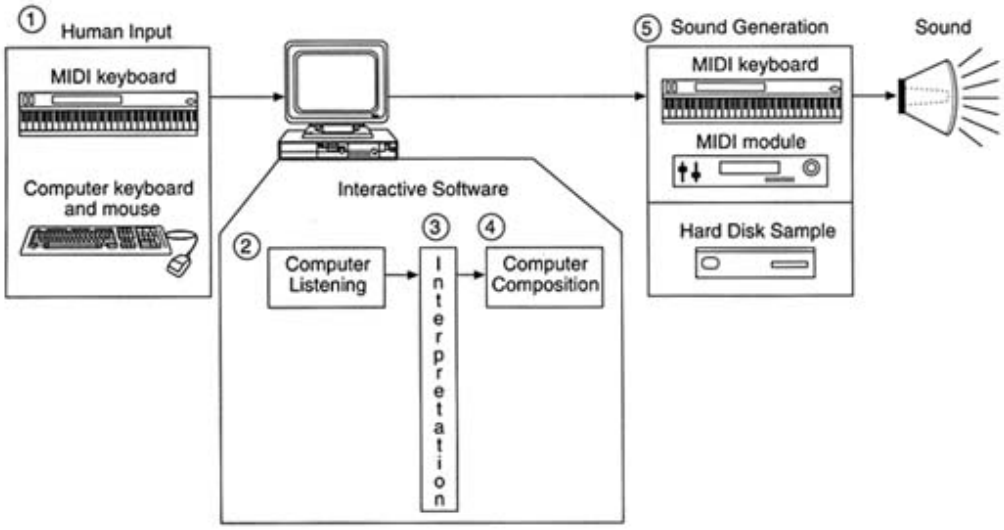


Figura 4 - Componentes básicos de um sistema musical interativo. (Winkler, 2001)

2.4. PARADIGMAS DE INTERAÇÃO MUSICAL

Atualmente já existe uma grande variedade de técnicas e sistemas que podem ser aplicados no desenvolvimento de sistemas musicais interativos. Cada um deles tem características e finalidades diferentes, devendo o compositor ou artista que desenvolve o sistema escolher o paradigma que melhor se adequa aos seus objetivos.

Neste capítulo pretende-se fazer uma representação global de alguns dos paradigmas mais conhecidos e utilizados para o desenvolvimento de sistemas musicais interativos. Convém realçar que nenhum destes paradigmas/modelos é fechado em si mesmo, ou seja, é normal que ocorra interseção com outros paradigmas/modelos, aliás, muitos paradigmas e técnicas são definidos por essas interseções e cruzamentos de técnicas/processos, o que torna difícil, por vezes, identificar onde começa um modelo e acaba outro.

2.4.1. MAPEAMENTOS (*MAPPINGS*)

Os mapeamentos podem ser considerados uma das técnicas mais eficazes e mais utilizadas em sistemas musicais interativos e não só. Tipicamente, os mapeamentos são usados como métodos de transformação para fazer o elo de ligação entre as ações do *performer* com algoritmos de geração automática de música, ou algoritmos de síntese sonora. (Winkler, 2001)

Esta é uma prática comum a qualquer programa de sequenciação ou sintetizador ou processamento de efeitos. É fácil encontrar utilizadores que mapeiam os seus controladores MIDI a parâmetros de um sintetizador, ou a um parâmetro de um efeito para processar o sinal áudio.

Existe outras áreas musicais abrangidas por esta prática, nomeadamente o desenvolvimento de novos instrumentos eletrónicos e interfaces e também os instrumentos aumentados. Os instrumentos eletrónicos podem ser equipados com todo o tipo de sensores, desde acelerómetros, giroscópios, sensor de pressão, sensor de flexão, sensores de distância, botões, etc., que depois podem ser mapeados a quaisquer parâmetros de sintetizador, algoritmo ou efeito sonoro.

Numa definição mais generalista, o mapeamento de parâmetros, geralmente associa uma característica musical proeminente a outra. (Winkler, 2001)

O software *MaxMSP* desenvolvido pela *Cycling74* é uma linguagem perita e ideal para este tipo de práticas. Ali, encontram-se todo o tipo de ferramentas e algoritmos à disposição do utilizador para esta prática, estando este limitado apenas pelo seu conhecimento e imaginação.

Existe duas arquiteturas muito utilizadas para suportar o mapeamento de parâmetros a sensores e controladores, o MIDI e o OSC (Open Sound Control).

2.4.1.1. TIPOS DE MAPEAMENTO E EXEMPLOS

Sem pretender esgotar o assunto, esta secção apresenta alguns exemplos comuns de tratamento dos mapeamentos.

A escala de variação dos parâmetros MIDI dentro da norma GM (*General MIDI*), varia num âmbito entre 0 e 127 valores, este facto deve-se ao tamanho da palavra MIDI, isto é, o número de bits restantes para passar o valor do parâmetro, oferece apenas uma resolução de 7 bits (2^7=128). Neste caso, o mapeamento de um controlador MIDI (por exemplo um potenciómetro), ao parâmetro Volume da norma GM faz-se de forma direta, porque não é necessário aplicar qualquer tipo de escalamento.

Habitualmente mapeamentos entre parâmetros do mesmo tipo não requerem nenhum tipo de tratamento, mas existe casos onde isso não se aplica, como por exemplo quando os parâmetros têm escalas diferentes (âmbito dos parâmetros não corresponde), ou quando se pretende controlar a transição entre valores. Nestas situações aplica-se algum tipo de escalamento, ou recorre-se a expressões matemáticas, tipo função transferência, para obter o resultado desejado.

O MaxMSP, compreende por defeito, uma série de objetos que permitem fazer escalamentos e mapeamentos.

- [zmap]: Este objeto permite realizar escalamentos de forma muito simples, aplica-se habitualmente quando os âmbitos entre dois parâmetros não são iguais. Por exemplo imagine o parâmetro A que varia entre [0.~1.] e o parâmetro B que varia entre [0~127], o [zmap] aplica o escalamento necessário de forma que o parâmetro A varie também entre [0~127].

- [scale]: Este objeto tem um comportamento muito semelhante ao [zmap] mas difere dois aspetos. Em oposição ao [zmap], este não trunca os valores dos parâmetros, isto

é, imagine os mesmos parâmetros A e B com os mesmo âmbitos referidos anteriormente, se por alguma eventualidade o [scale] recebesse um valor superior a 1., por exemplo 2., este continuaria a escalar esse mesmo valor, enquanto o [zmap] truncaria o valor em 1 conforme definido por argumento, na instanciação do objeto. Outro aspeto é que o [scale] permite receber um 5º argumento, esse valor irá definir uma exponencial aplicada no escalamento, passando de escalamento linear para exponencial. Esta característica permite adicionar uma curva ao mapeamento de parâmetros.

- [line]: Este objeto não realiza escalamentos, mas sim interpolação entre valores. Utiliza-se habitualmente para suavizar as transições entre os valores dos parâmetros. Este requer a duração da rampa e o período de tempo entre cada passo na rampa, para ser instanciado. Se os dados a serem interpolados forem recebidos a uma frequência constante é possível obter resultados semelhantes com o objeto [slide], mas neste caso passa a ter uma curva logarítmica.

- [linedrive]: Este objeto funciona como uma combinação entre [scale] com exponencial e [line] que realiza interpolação entre valores num dado tempo definido pelo utilizador. Basicamente o objeto realiza um escalamento exponencial, e envia o valores formatados para serem aplicados diretamente no [line] para interpolação.

- [table]: Este objeto consiste basicamente numa tabela gráfica que permite armazenar valores enviando o índice e o valor a armazenar. Os valores podem ser chamados mais tarde pelo índice.

- [function]: Permite que o utilizador desenhe funções sem conhecer a expressão matemática correspondente.

- [expr]: Permite que o utilizador manipule expressões matemáticas.

2.4.2. SCORE FOLLOWING

O acompanhamento automático utilizando o computador é conhecido como *Score Following*. Esta técnica acompanha o progresso da performance de um músico utilizando uma partitura conhecida à partida pelo sistema, isto é, através do reconhecimento da performance do músico em comparação com a mesma partitura que este executa, o computador reconhece o ponto na partitura que o músico executa naquele instante. Este sistema, permite a sincronização em tempo-real de um músico tocando uma partitura com a partitura em si mesma (esta contida no computador).

A sincronização em tempo-real de sinais áudio ou midi com uma partitura simbólica abre uma série de possibilidades ao compositor/artista que desenvolve o sistema, nomeadamente na interpretação de peças com eletrónica, e permite a ornamentação e o acompanhamento automático de peças com o auxilio do computador, ou atribuindo ao computador também um papel de instrumentista, tocando em conjunto com o(s) músico(s). O tipo de utilização dado a esta técnica depende dos objetivos do compositor ou músico para a peça que está a desenvolver, mas esta abre possibilidades para acionar eventos, controlar algoritmos de geração automática de música, controlo e manipulação do som a ser gerado ou sintetizado, etc.

Em 1983 *Barry Vercoe* do IRCAM[10] e *Roger Dannenberg* da CMU[11] foram pioneiros na introdução e desenvolvimento de sistemas "score following". Devido às limitações na época em termos computacionais para processamento de sinais áudio, os computadores interpretavam apenas dados simbólicos. Por exemplo no caso da peça "synthetic performer" de Barry Vercoe, o flautista *Larry Beauregard* utilizava uma série de sensores no seu instrumento para enviar informação simbólica sobre a sua performance para o computador. (IRCAM - imtr, 2011)

Com a introdução e evolução do protocolo MIDI na década de 80 e a sua integração nos instrumentos comercializados na época, os sistemas de "score following" foram adaptados para suportar este protocolo. O protocolo MIDI representa a altura das notas com números[12], desta

[10] **IRCAM** - Instituto de Pesquisa e Coordenação de Música e Acústica ("Institut de Recherche et Coordination Acoustique/Musique"), instituição dedicada à pesquisa e criação de música contemporanêa. (Wikipédia, 2011) (IRCAM, 2011)

[11] **CMU** - Carnegie Mellon University

[12] Na realidade uma mensagem MIDI é um pouco mais complexa do que a descrição dada no exemplo, esta é na realidade composta por vários bytes", o "status byte" e o data byte. O primeiro indica o tipo mensagem e o segundo representa os dados enviados nessa mensagem. No texto a descrição foi simplificada para facilitar a leitura mas no entanto, no MaxMSP é possivél extrair um número das notas representando apenas a altura.

forma uma linha melódica é representada no computador como uma sequência de números. Imaginando que a mesma linha melódica está a ser tocada por um músico utilizando um teclado MIDI, por cada nova nota que este toca é enviado para o computador um número. Se a linha melódica for tocada sem erros e seguindo a sequência certa como representado na partitura, então o computador deteta que a sequência coincide com aquela armazenada na memória do computador, extrapolando assim a informação da localização do músico na partitura.

Como já foi referido nenhum destes paradigmas é fechado em si mesmo. O *score following* continua a ser uma área de investigação ativa onde ocorre a interseção com outras áreas como inteligência artificial, reconhecimento de padrões, processamento de sinal e musicologia.

O advento de unidades hardware para processamento dedicado de sinal áudio (por exemplo os módulos 4X[13] e ISPW[14] do IRCAM) permitiu que emergisse os primeiros sistemas *score following* baseados em deteção de altura, isto é, recebem como entrada um sinal áudio e a partir da sua análise detetam a altura da(s) nota(s) que o músico executa.

Muitas das primeiras composições com eletrónica em tempo-real utilizaram esta tecnologia para performances entre músicos tocando uma partitura e composição eletrónica. Um dos pioneiros foi *Philippe Manoury* que compôs uma peça para flauta e eletrónica em tempo-real intitulada *Júpiter.* Esta peça é considerada uma das primeiras compostas em MaxMSP com eletrónica em tempo-real, e foi concebida inicialmente para flauta MIDI mas, em 1992 foi modificada para suportar deteção de altura. (IRCAM - imtr, 2011)

[13] **Sogitec 4X** foi uma unidade de processamento digital de som desenvolvida no IRCAM por *Giuseppe Di Giugno* em 1980. Foi o último processador hardware de grande dimensões a ser desenvolvido (Schutterhoef, 2007) (Wikipédia, 2011)

[14] **ISPW** (IRCAM Signal Processing Workstation) foi uma unidade hardware DSP desenvida pelo IRCAM e a Ariel Corporation no final na década de 80. Esta unidade funcionava acopulada ao computador NeXT e tinha capacidade de processar áudio e sintese em tempo-real, e podia trabalhar com 8 entradas e saídas de sinal. (Schutterhoef, ISPW, 2007) (Wikipédia, 2011)

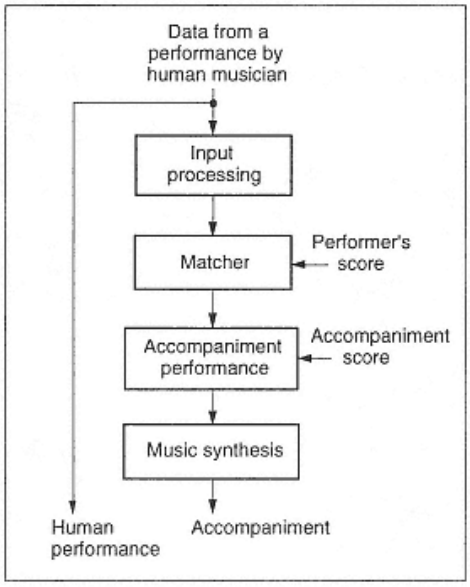


Figura 5 - Esquema geral de um sistema "score following". Este analisa a performance do músico e gera o acompanhamento que coincide com a direção e andamento do músico. (Roads, 1996)

Na década de 90 a investigação na aplicação de modelos probabilísticos para o processamento de áudio e voz deu um salto significativo. Ocorrem erros e incertezas na performance e na perceção da máquina que não devem ser ignorados por um sistema de *score following*. Desta forma, a intersecção desta área de investigação com os modelos probabilísticos (por exemplo cadeias de Markov Escondidas, HMM) que vinham a ser desenvolvidos para o processamento de voz e áudio, permitiu desenvolver sistemas de *score following* mais robustos. (IRCAM - imtr, 2011)

Em 2000, no IRCAM foi desenvolvido por *Nicola Orio* e *Diemo Schwarz* o módulo *suivi~* para *score following* equipado com um sistema de aprendizagem artificial. Este sistema estreou em concerto em 2005 com uma peça de *Pierre Boulez* para flauta, orquestra e eletrónica. A peça intitula-se *explosante fixe*. (IRCAM - imtr, 2011)

O desenvolvimento musical e cientifico de *score following* tem objetivos diferentes. Enquanto o ponto de vista científico tem entre os seus objetivos a procura da precisão exata entre o músico e a partitura, no ponto de vista musical exige acesso em tempo-real a parâmetros de interpretação que possam ser sincronizados com os artistas.

Nos finais de 2007 em colaboração com o compositor *Marc Stroppa* procurou desenvolver-se um novo paradigma de *score following* que permitisse antecipar eventos. Este sistema extrai simultaneamente a posição e o tempo em tempo-real, antecipando parâmetros da performance dos músicos. Este projeto conduziu ao que é atualmente a plataforma padrão

no IRCAM para *score following* - o sistema *Antescofo*. A robustez deste sistema ajudou a tornar explícito os objetivos musicais da utilização de *score following* e isso conduziu a uma evolução do paradigma para o que é intitulado *Synchronous Programming* (programação síncrona), isto é, os eventos eletrónicos passam a ser programas polifónicos executados em paralelo com o músico. Este paradigma tenta colmatar as lacunas entre os aspetos relacionados com performance e a composição musical da música por computador. Uma das vantagens é que exige relativamente pouco tempo a serem desenvolvidos. (IRCAM - imtr, 2011)

A linguagem de programação síncrona do sistema Antescofo tem evoluído no sentido de responder às exigências dos compositores.

2.4.2.1. ANTESCOFO~ E PROGRAMAÇÃO SÍNCRONA

As linguagens de programação síncrona são linguagens orientadas para a música por computador e sistemas reativos em tempo-real. Estas têm vindo a ser desenvolvidas e aplicadas desde há algum tempo em sistemas de aviação e sistemas incorporados. A importância deste paradigma de programação tem crescido bastante ao longo dos tempos. (IRCAM - imtr, 2010)

Este paradigma tem sido integrado no contexto do sistema *Antescofo* para a composição musical e a performance, procurando rever, testar e verificar os conceitos atuais de programação e paradigmas aplicados em sistemas musicais interativos.

Antescofo foi desenvolvido em colaboração com o compositor *Marco Stroppa* e o soxofonista *Claude Deelangle*, e estreou em público em 2007 com a peça *Of Silence* para saxofone e eléctrónica. *Antescofo* consiste num sistema de *score following* modular polifónico, tendo integrado o paradigma de linguagem de programação síncrona orientado para a composição musical e performance na música por computador. (IRCAM - imtr, 2011)

Este sistema permite o reconhecimento automático da posição na partitura e do tempo a partir de um sinal de entrada áudio vindo do músico, tornando possível a sincronização da performance instrumental com os eventos gerados pelo computador.

Por exemplo, *Antescofo* pode trabalhar em simultâneo com o programa *NoteAbility Pro*, que é uma ferramenta gráfica para edição de partitura. Uma das funcionalidades deste

programa é que permite a sincronização em tempo real da posição na partitura entre ambos os programas, com os objetos de *Antescofo* para *MaxMSP* e *PD*[15].

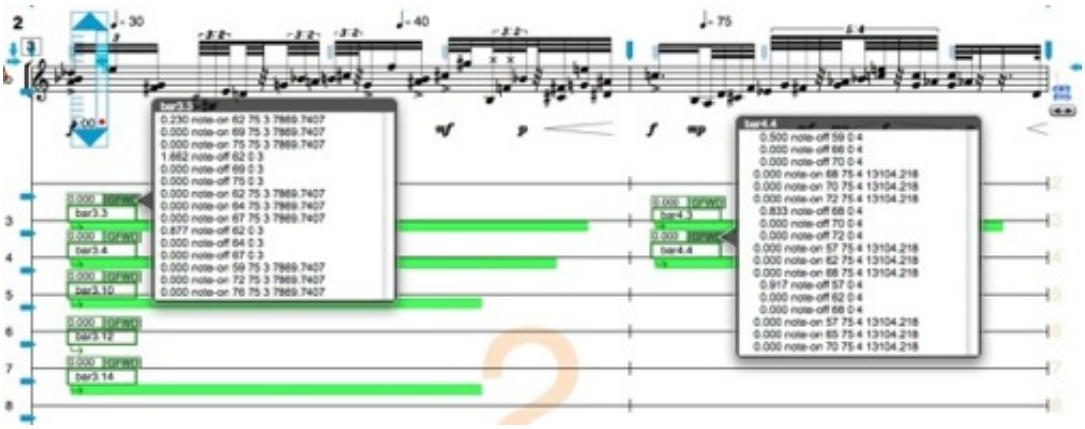


Figura 6 - Programação síncrona entre *Antescofo* e o programa *NoteAbility Pro,* mostrando em simultâneo a linha instrumental acompanhada pelos programas a serem executados naquele instante pelas partes eletrónicas.

[15] Pure Data

2.4.3. SISTEMAS COM INTELIGÊNCIA ARTIFICIAL E CAPACIDADE DE APRENDIZAGEM

Esta secção apresenta alguns exemplos de sistemas musicais interativos que têm como componente principal a capacidade de criar conteúdo de forma independente e/ou capacidade de aprendizagem.

2.4.3.1. CONTINUATOR - FRANÇOIS PACHET

O *Continuator* é um sistema musical interativo desenvolvido por François Pachet, com capacidade de aprendizagem, e que permite gerar música e improvisar coerentemente mantendo o género e estética musical.

Pachet considera que um sistema musical interativo deve:

- Aprender estilos musicais em tempo-real sem conhecimento a priori;
- Ter em conta a performance do músico durante o processo de geração musical de forma a permitir influenciar este processo, criando assim verdadeiros diálogos musicais em oposição a sistemas desencadeados (*triggers*);
- Seguir protocolos de interação que forneçam aos músicos controlo total da música que está a ser gerada, mas mantendo e realçando também a expressividade musical dos utilizadores.

O autor pretende que o sistema seja uma ponte entre os sistemas musicais interativos e os sistemas de imitação musical, um que é limitado na sua capacidade de gerar conteúdo musical dentro de determinado estilo e o outro que fundamentalmente não é interativo (mas é eficiente na representação de informação estilística), respetivamente.

Nas versões iniciais a comunicação do músico com o *Continuator* era feita através de um sistema MIDI que permite interligar qualquer instrumento com capacidade de enviar mensagens MIDI (teclado, guitarra, etc.).

O *Continuator* divide-se em dois grandes módulos, o de análise e o de geração.

O módulo de análise trata a música como sequencias temporais de eventos MIDI, da qual representa a seguinte informação: altura das notas («pitch», dados MIDI de 0 a 127), velocidade das notas (ou amplitude, MIDI de 0 a 127), informação temporal acerca dos tempos de começo e duração das notas, tendo também em conta polifonia, ruído e estruturas rítmicas

arbitrárias. Este módulo é composto por três processos, deteção do fim de frases musicais, analisador de padrões, e analisador de propriedades globais.

A deteção do final de frases musicais assegura que a "continuação" seja temporalmente impercetível. Para tal utiliza um mecanismo de análise temporal que deteta os intervalos entre «onsets» e ajusta automaticamente um «threshold», assim, se a sequência tocada for lenta este aumenta o «threshold», caso contrário diminui automaticamente.

O analisador de padrões constrói o modelo de Markov a partir da sequência musical tocada. Este segmenta essa sequência da esquerda para direita para construir uma árvore de todas as continuações possíveis.

O analisador de propriedades globais analisa a densidade (número de notas por segundo), tempo e compasso métrico (localização de pulsações fortes e fracas) e dinâmica global (forte ou fraco). O sistema utiliza estas propriedades para que a continuação gerada em resposta à sequência tocada seja lógica, fluente e musical. No modo *standard* de funcionamento o *Continuator* recebe a informação musical do músico, e envia para um sintetizador MIDI a performance gerada em tempo-real.

Segundo o autor, o sistema comporta-se como um prossecutor de sequências, onde o fluxo de notas do músico é sistematicamente segmentado em frases utilizando um limite temporal variável (a volta de 250 milissegundos). Cada frase é enviada de forma assíncrona para um processo de análise, a partir do qual se constrói o modelo de padrões recorrentes. O sistema, em reação à frase musical tocada, gera uma nova frase de acordo com a base de dados de padrões aprendidos, como continuação daquela que foi tocada pelo músico.

O coração da arquitetura do *Continuator* consiste num modelo aumentado de cadeias de Markov que considera vários aspetos do estilo musical, como o ritmo, pulsação, harmonia e imprecisão. Esta implementação permite que o sistema possa aprender e gerar música em qualquer estilo musical, quer este funcione em modo autónomo (*standalone*), em modo continuação dos dados musicais recebidos (*continuator*), ou como suporte de improvisação interativa. O sistema possui também capacidade de se adaptar rapidamente e sem intervenção humana a mudanças imprevistas de ritmo, harmonia ou estilo.

O processo de geração produz a continuação nota por nota a partir das probabilidades de Markov inferidas durante o processo de análise. Tecnicamente, esta cadeia de Markov é variável, otimizando a relevância da continuação de cada nota.

(Pachet, The Continuator: Musical Interaction With Style, 2002)

(Pachet, Beyond the cybernetic jam fantasy: The continuator, 2004)

2.4.3.2. GENJAM - JOHN BILES

GenJam consiste num algoritmo genético interativo que modela um músico de jazz improvisador e que atua frequentemente como acompanhador do autor do software John Biles.

GenJam tem a capacidade de aprender improvisação de temas completos sob a guia de um *performer* humano, inclusive *trade fours* [16]em tempo-real.

Os algoritmos genéticos consistem num paradigma computacional onde se aplica os princípios de evolução da genética natural para dar soluções evolutivas a problemas e questões que exigem pesquisa constante e respostas dinâmicas. Um algoritmo genético tradicional mantém à mão uma população de potenciais soluções para o problema em foco, onde cada membro individual da população consiste num cromossoma em formato «string» (genótipo) que é mapeado (descodificado) a uma possível solução (fenótipo).

A adequação (apropriação) de cada potencial solução, determina a aptidão (*fitness*) de cada indivíduo dentro da população, de forma que indivíduos com cromossomas correspondentes (mapeados) a melhores soluções tenham valores mais altos de aptidão (*fitness*). No algoritmo genético a população evolui selecionando os indivíduos mais aptos para serem pais na próxima geração.

Os novos indivíduos nascem realizando uma operação de cruzamento (*crossover*) entre os cromossomas dos pais, isto é, o cromossoma de cada filho contém uma mistura dos cromossomas dos pais e possivelmente uma mutação (*mutation*).

A nova geração que nasceu passa a fazer parte da população como novos indivíduos, iniciando assim novamente o ciclo, onde aqueles indivíduos mais aptos (*fitness*) são selecionados para sobreviver e possivelmente prosperar para futuras gerações. Eventualmente, surgirá um indivíduo na população cujo valores de aptidão correspondem a um critério global, dando a pesquisa por terminada.

Nos algoritmos genéticos tradicionais, a função *fitness* (de aptidão) pode ser codificada num algoritmo, mas no meio artístico, consiste geralmente numa opção estética feita por um humano, geralmente o artista. Isto leva-nos ao modelo de algoritmo genético interativo (IGA), onde um humano, o mentor, deve testar os indivíduos dentro da população e avaliá-los direta ou indiretamente de forma a proporcionar a cada indivíduo um valor de aptidão (*fitness*). Esta tarefa torna-se árdua e dispendiosa em tempo para o mentor.

[16] Nome dado ao processo quando os músicos trocam entre si 4 compassos de solo durante a improvisação.

GenJam baseia-se num algoritmo genético interativo que evolui duas populações de ideias melódicas interligadas hierarquicamente, uma a nível de compassos e outra ao nível de frases musicais, sempre supervisionadas pelo mentor à medida que o sistema aprende a improvisar jazz.

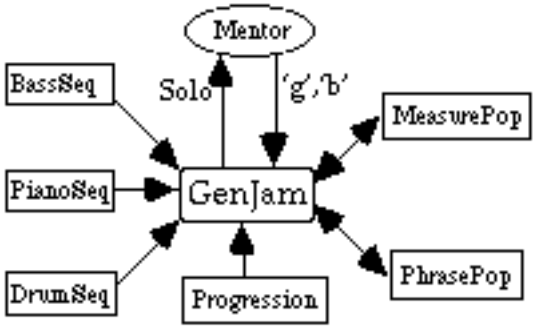


Figura 7 - Esquema geral do sistema GenJam.

A figura acima representa uma vista geral do sistema *GenJam*. Este foi desenvolvido em *Macintosh* em ambiente C utilizando o CMU MIDI Toolkit[17].

Para improvisar sobre um tema, *GenJam* necessita de alguma informação inicial, recolhida pela leitura de alguns ficheiros, como por exemplo a progressão harmónica, tempo, estilo rítmico, número de compassos do tema e ainda as sequências de acompanhamento como o piano, baixo e bateria que são gerados previamente com a ajuda do *Band-In-Box* e importado como ficheiro MIDI.

GenJam constrói sobre a estrutura do tema vários ciclos de eventos MIDI para improvisar sobre este. Os eventos MIDI são descodificados a partir de membros da população de compassos e frases melódicas. Uma frase é implementada, como uma sequência de 4 compassos, logo estas duas populações formam uma hierarquia de estruturas melódicas mutuamente dependente.

Quando o mentor escuta um solo, este pode pressionar a tecla "g" (*good*) quando um trecho lhe agrada, dando assim um reforço positivo à função *fitness* do sistema, ou pode pressionar " b" (*bad*) quando o trecho é considerado fraco, dando assim um reforço negativo ao sistema, desta forma os valores da função *fitness* da população vão sendo atualizados.

[17] Carnegie Mellon University MIDI Toolkit

O *GenJam* tem três modos de funcionamento, aprendizagem (*learning*), *breeding*, e o modo *demo*. O modo de aprendizagem é utilizado pelo utilizador para construir os valores de aptidão (*fitness*), aqui os operadores genéticos estão desativados, o sistema seleciona aleatoriamente frases musicais (ignorando os valores de aptidão) para o mentor classificar.

O modo *demo* é utilizado para performance, as frases musicais são escolhidas através de um processo de seleção tipo torneio, que considera tanto o valor de aptidão da frase como o valor de aptidão dos compassos, no entanto a informação de classificação do mentor é ignorada.

O modo *breeding* é semelhante ao modo *demo*, mas neste são aplicados os operadores genéticos. Antes de um solo ser apresentado, metade de cada população é substituída por uma nova geração de indivíduos para serem classificados.

(Biles A., 1994)

(Biles A., 2003)

2.4.3.3. CYPHER – ROBERT ROWE

Cypher é um sistema musical interativo criado pelo compositor Robert Rowe. Este software produz classificações do comportamento musical através da análise da performance humana, e responde com novo material musical gerado em tempo-real.

Este divide-se em duas grande componentes, o *Listener* que analisa o fluxo de dados MIDI enviado pelo *performer* e o *Player* que gera material musical como resposta. Ambas as componentes do software são hierárquicas, no sentido em que o processamento das abstrações de elevado nível lidam com abstrações produzidas a níveis inferiores, logo correspondendo a intervalos de tempo maiores.

Em termos de análise, o sistema classifica características de baixo nível a partir dos dados MIDI como densidade, registo, sensação de intensidade (*loudness*), tempo, duração, harmonia. A análise a alto nível debruça-se em encontrar grupos e regularidades no comportamento das características de baixo nível ao longo do tempo e também na indução de tonalidade e deteção de pulsação (*beat tracker*).

No que respeita à performance existem três classes com métodos composicionais para gerar as respostas, sequenciação, algoritmos composicionais e transformação. Cabe ao utilizador do software decidir quais os métodos que irão ser utilizados para compor as respostas, através de ligações que se estabelecem entre as mensagens da componente *listener* e os métodos da componente *player*. Quando se estabelece a ligação, esta leva a que os métodos sejam executados sempre que estes recebam uma mensagem da componente *listener* correspondente. Este permite também guardar estados ou coleções dessas ligações

estabelecidas, e invocar diferentes estados ao longo da peça musical através de uma componente *score following* que segue o músico ao longo da performance.

Os resultados produzidos por *Cypher* são também analisados e alterados por um processo critico que segue algumas opções estéticas pré-estabelecidas assegurando também consistência de estilo nas respostas produzidas pelo programa. Nas suas composições com *Cypher*, Rowe evita utilizar sequências musicais armazenadas previamente, em vez disso, o programa é designado a responder ao que "ouve" relativamente à performance de acordo com vários processos algorítmicos.

Segundo Rowe, *Cypher* executa música composta de uma forma que é sensível a certos aspetos da performance humana. É também capaz de responder de forma criativa a música que lhe é desconhecida e pode também compôr música sem a análise da performance musical humana, através da transformação de material musical guardado previamente ou gerando novo material musical.

Desta forma *Cypher* adapta-se perfeitamente a situações de improvisação, em que o computador assume o papel de parceiro musical (acompanhador) do músico, em vez de constituir uma mera extensão de um solista ou *ensemble*. No entanto também foi utilizado na performance de peças compostas.

(Wetzel, 2004)

(NYU Center for Advanced Technology in Multimedia)

(Rowe, Machine Listening and Composing with Cypher, 1992)

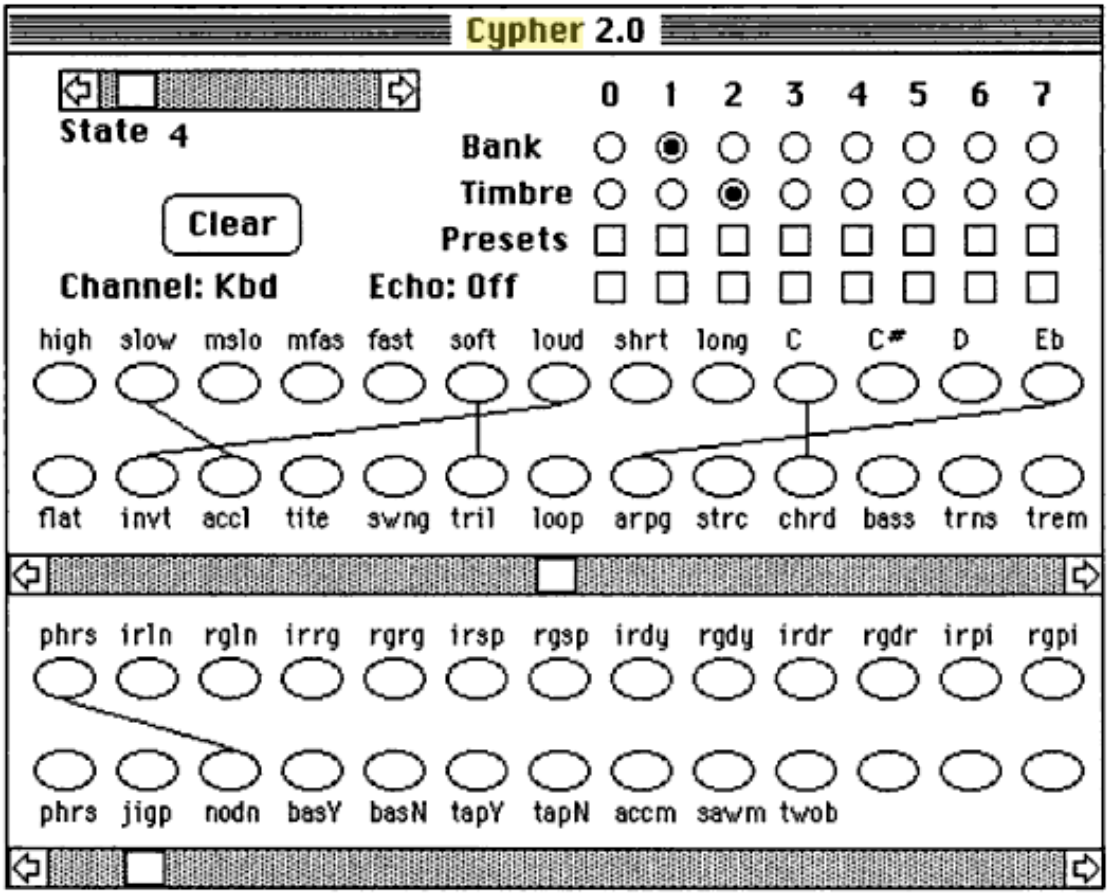


Figura 8 - Representação do interface do software Cypher (Rowe, 2001, in Winkler, 2001)

2.4.3.4. HAILE – GIL WEINBERG

Achou-se relevante incluir esta referência não pela componente robótica do projeto *Haile* que não se relaciona diretamente com o tema desta dissertação, mas sobretudo devido à sua componente software.

O *Haile* é um robot percussionista que consegue ouvir outros músicos, analisar a sua música em tempo-real e usar o produto da análise para improvisar. Este projeto é muito interessante pela qualidade dos resultados musicais, como se pode verificar nos vídeos[18] que circulam na internet sobre o projeto. (Weinberg & et al., 2012)

[18] http://www.gtcmt.gatech.edu/research-projects/haile

O robot *Haile*, tal como outros projetos descritos anteriormente, contém uma componente de análise (o ouvinte da performance) e uma componente de geração. Em termos de análise, são extraídas algumas características musicais de baixo-nível como «onsets», altura, amplitude e também características de alto-nível como a estabilidade rítmica e semelhança utilizando processos como *dynamic time warping*.

A sua componente de geração musical baseia-se também num algoritmo genético tal como o *GenJam*. A geração musical é baseada na sequência musical tocada por outro músico e analisada pelo sistema. Os dados MIDI ou áudio são segmentados. A partir destes tenta encontrar a melhor resposta musical a partir de uma população de frases musicais geradas por um humano e armazenadas no sistema. Esta população de frases é evoluída através de processos de mutação e cruzamento ao longo de várias gerações para depois serem avaliadas por uma função de adaptação (*fitness function*) que mede a semelhança com o que foi tocado.

CAPÍTULO 3

3.1. ARQUITETURA E IMPLEMENTAÇÃO DO SISTEMA - KINDUCTOR

Neste capítulo será descrita a arquitetura do sistema *Kinductor*[19]. Este divide-se em duas secções principais, hardware e software, sendo que este último também pode ser dividido em duas etapas, a análise em tempo-real da performance musical e a geração e mapeamentos em tempo-real de parâmetros para o *Kin.Rhythmicator*.

Para manter coerência com as designações utilizadas na aplicação desenvolvida em MaxMSP, alguns títulos e termos serão dados em inglês tal como foi feito dentro da aplicação.

[19] http://code.google.com/p/kinductor/

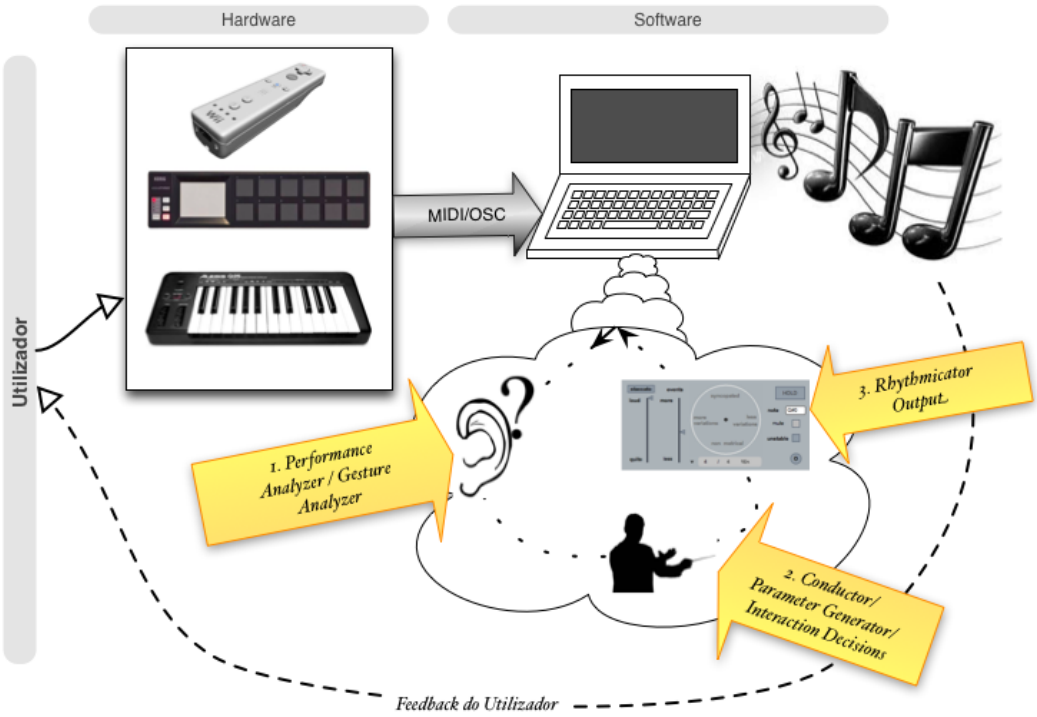


Figura 9 - Diagrama geral do sistema implementado, o *Kinductor*[20]

3.2. HARDWARE

O hardware utilizado neste sistema representa uma componente importante para o seu funcionamento, dado que constitui o principal interface de interação que o utilizador (*performer*) utiliza para comunicar com o sistema, nomeadamente o *kin.rhythmicator*. Este sistema suporta dois tipos de hardware, por um lado os instrumentos MIDI, que são basicamente instrumentos digitais que têm implementado a especificação estandardizada MIDI (*Musical Instrument Digital Interface*). Isto significa que para além de outros tipos de mensagem MIDI também são capazes de enviar mensagens MIDI do tipo Nota. (Wikipedia, 2012)

É a partir deste tipo de mensagem que é feita a análise rítmica como será descrito posteriormente ao nível do software.

[20] http://code.google.com/p/kinductor/

Figura 10 - Alguns exemplos de instrumentos MIDI (korg nanoseries)

Para além de instrumentos digitais MIDI, este sistema também suporta o protocolo OSC (Open Sound Control) para compatibilidade com o controlador gestual da Nintendo, WiiMote. (Wikipedia, 2012) Tendo em conta que a extração de informação da performance é feita pela análise rítmica, assume-se que os controladores gestuais têm conceptualmente um papel semelhante ao *shaker*, um instrumento onde os gestos executados são caracterizados por movimentos periódicos e rítmicos. Outros trabalhos relacionados com a análise de movimento na dança utilizam uma abordagem diferente.

OSC é um protocolo para formatação do conteúdo de mensagens entre computadores, sintetizadores e outros dispositivos multimédia beneficiando das novas tecnologias de suporte de redes. Este foi desenvolvido no CNMAT (*Center for New Music and Audio Technologies*) por Adrian Freed e Matt Wright e é comparável aos formatos XML[21] e JSON[22]. (Wikipedia, 2012)

[21] Extensible Markup Language

[22] JavaScript Object Notation

Figura 11 - *Osculator*, um exemplo de um programa que permite gerir vários dispositivos por Open Sound Control.

3.3. SOFTWARE

> *"My measure of success, however, is not whether these programs match empirical data from research with human subjects, but whether they output structures that make musical sense."* *(Rowe, Machine Musicianship, 2001)*

Baseado nos objetivos propostos desta tese, o software assume uma componente importante como intérprete entre a performance musical do utilizador e o feedback que este receberá do gerador de ritmo *Rhythmicator*. Este decompõe-se em diversos módulos, cada um

com uma função especifica, que trabalham interligados entre si a fim de interagirem com o *Rhythmicator*. Nesta secção será apresentado cada módulo do software implementado.

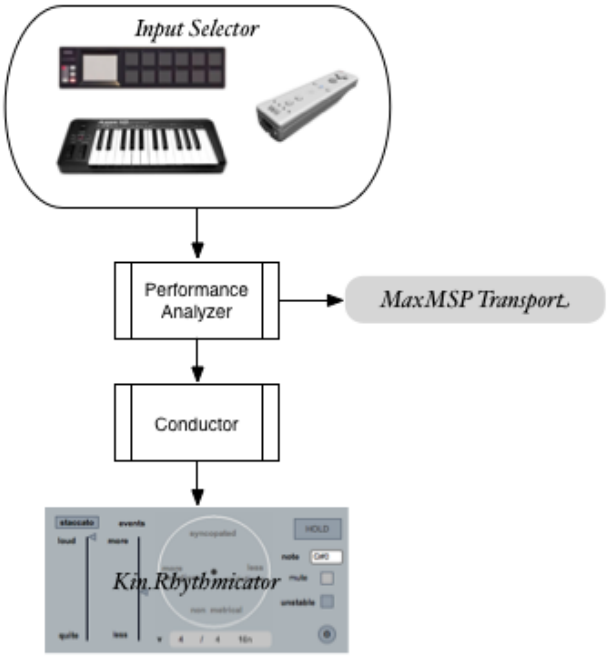


Figura 12 - Diagrama geral da estrutura do software implementado.

Na Figura 12 observamos um diagrama geral da estrutura do software implementado. Como este suporta dois tipos de instrumentos diferentes, instrumentos MIDI e controladores gestuais (*wiimote*), estes requerem duas abordagens diferentes, no entanto a estrutura geral do software implementado é igual para ambas as versões, como se pode observar na Figura 12. As duas versões diferem sobretudo no módulo de análise da performance, dado que o tipo de dados dos instrumentos MIDI é diferente dos controladores gestuais (fluxo de dados mais rápido por OSC).

3.3.1. Análise da Performance a Partir de Instrumentos Digitais MIDI

Numa primeira fase encontramos o módulo de análise que recebe como entrada a performance do utilizador, do qual existe duas versões. Uma lida com instrumentos digitais MIDI e outra lida com o controlador *WiiMote* por OSC. Como cada um deste tipo de

instrumentos gera um formato de dados diferente, logo cada um deles exige um tratamento próprio de acordo com as suas idiossincrasias técnicas.

Nesta etapa a performance rítmica é analisada estatisticamente ao nível da média de variações, amplitudes, sincopação, densidade, estado ocioso (*performer* ativo ou não ativo), e desvio temporal em relação ao andamento musical inicial. Estes são atributos globais da música tocada, e segundo (Cook, 2001, p. 214) podem ser relacionados com atributos percetuais, isto é, constituem uma forma abstrata de representar dados complexos.

No entanto, na análise da performance com controladores gestuais é realizada em simultâneo uma análise do padrão rítmico extraído a partir de «onsets» (ataque do movimento) do movimento gestual.

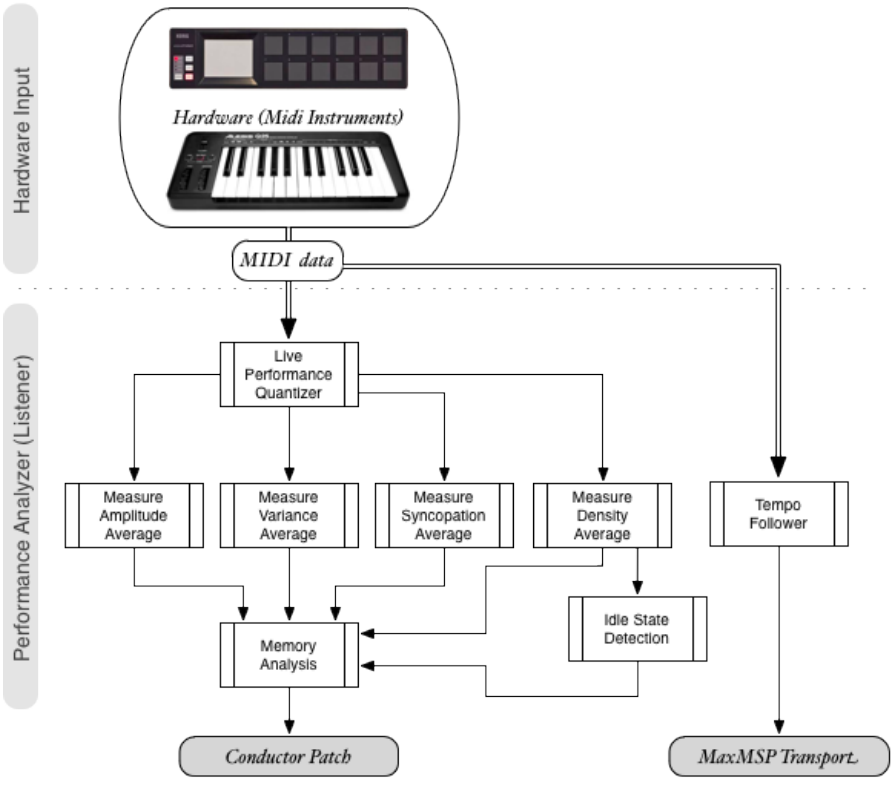


Figura 13 - Diagrama da análise da performance a partir de instrumentos digitais MIDI

3.3.1.1. TEMPO FOLLOWER

Este módulo, baseado num exemplo de Todd Winkler (Winkler, 2001), tem a finalidade de detetar *accelerando* e *ritardando* no andamento musical da performance do utilizador, para criar variações de andamento no contexto da peça musical.

Estes cálculos são realizados em duas fases: primeiro detetando qual o valor rítmico que está a ser executado pelo músico (semínima 4n, colcheia 8n, semicolcheia 16n, etc.), e em segundo calculando qual o desvio temporal em relação à pulsação musical no contexto da peça onde este executa a performance (por exemplo, uma peça musical com andamento inicial 100 bpm). Se esse desvio ocorrer dentro dos limites predefinidos, então o andamento da peça é alterado para seguir o *performer*. O exemplo de Todd Winkler calcula os desvios temporais apenas num valor rítmico predefinido (16n), esta versão foi implementada de forma a suportar 16n, 8nt, 8n, 4n, e 2n.

O cálculo do valor rítmico é feito a partir do chamado *tempo delta*, que consiste no cálculo dos intervalos de tempo entre «onsets» de eventos rítmicos. Tendo como referência o andamento musical predefinido inicialmente na peça, por exemplo 100 bpm, podemos calcular facilmente que o tempo delta da pulsação musical (semínima 4n) é 60000/100 = 600ms. Por sua vez, o tempo delta da colcheia (8n) é 600/2=300ms, da semicolcheia é 600/4=150ms, da tercina de colcheia é 600/3=200ms, e da mínima é 600/0.5=1200ms. Este raciocínio demonstra que os diversos valores rítmicos definidos na música ocidental podem ser calculados como múltiplos da pulsação musical, logo, a deteção do valor rítmico executado a partir dos tempos delta entre «onsets», pode ser calculada a partir do rácio correspondente a cada valor rítmico. Por exemplo, sabendo que a semicolcheia (16n) corresponde a ¼ da pulsação musical (600/4=150ms), podemos inferir então que um tempo delta de 150ms*4=600ms corresponde a uma pulsação musical sem qualquer desvio temporal.

À medida que o utilizador executa a sua performance, é possível ir comparando os seus tempos delta com as durações dos valores rítmicos dados pelo andamento da peça musical. Isto é possível graças ao conjunto de objetos associados ao [Transport] do Max6. Este consiste num relógio que cria um contexto de tempo musical baseado em compasso, tempos, unidades, assinatura de compasso. Todos os objetos que funcionam associados a tempo musical passam a depender deste. Um deles é o objeto [translate] do MaxMSP que permite traduzir os valores rítmicos para milissegundos com os quais se pode ir comparando os tempos delta da execução.

Assim, para detetar que valor rítmico está a ser executado compara-se o tempo delta da execução com todos os valores rítmicos em milissegundos (ms) dados pelos objetos [translate], se estiver dentro do limite de desvio selecionado então será considerado válido.

Para validar os tempos delta existe um filtro dado em percentagem, que corresponde ao desvio da pulsação musical permitido. Se o andamento musical corresponder a 100bpm, então a pulsação musical corresponde a 600ms, logo um desvio de 10% equivale a 60ms, o que corresponde a um desvio permitido entre 540ms e 660ms.

desvio_minimo = ValorRitmicoX – desvio

desvio_maximo = ValorRitmicoX + desvio

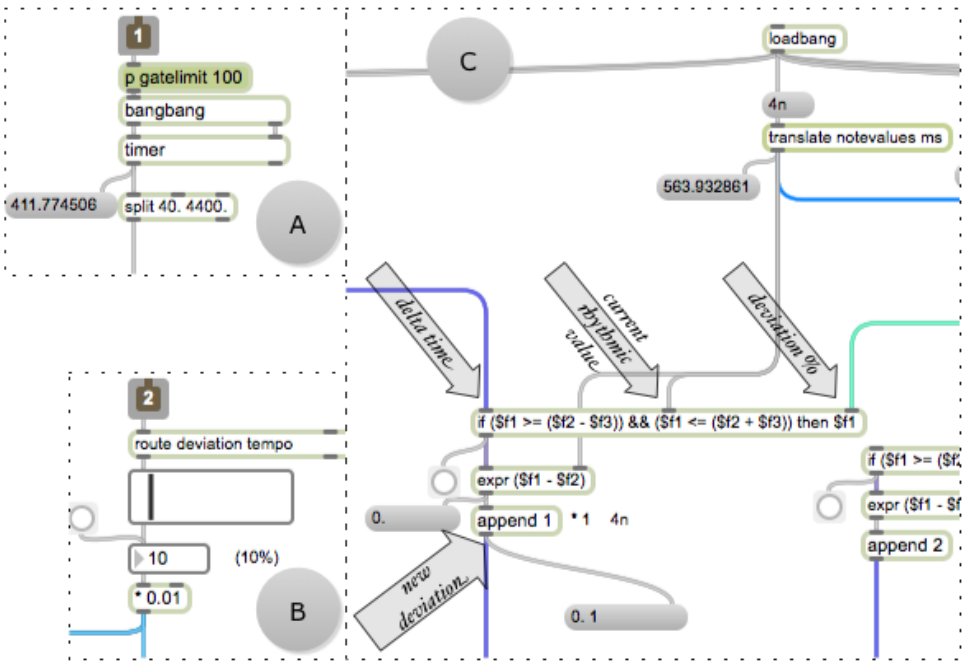


Figura 14 - (A) Cálculo dos tempos delta entre eventos; (B) Percentagem de desvio; (C) Deteção do valor rítmico e cálculo do desvio em relação ao valor rítmico atual.

3.3.1.2. LIVE PERFORMANCE QUANTIZER

Para o correto funcionamento de alguns módulos é necessário que os dados da performance se encontrem no formato apropriado, devidamente quantizados e segmentados. Quantizar significa restringir algo a um número limitado de possibilidades. Musicalmente, apesar das regras e a teoria serem claras acerca do ritmo, um dos aspetos que tornam um ritmo apelativo, é o facto de a performance humana conter imprecisões e micro desvios temporais na execução rítmica. No cenário computacional é diferente, pois como nos encontramos num meio digital, este exige que tudo seja discreto. Isto a nível rítmico significa

que a execução tem de ser alinhada numa grelha de divisões temporais especifica ajustada à resolução e precisão pretendida, logo os desvios micro temporais são excluídos da análise.

Neste contexto, como já sabemos à partida o tempo da pulsação musical da peça (BPM[23]), não é necessário utilizar um algoritmo para indução da pulsação («*beat tracker*»). Neste módulo pretende-se ajustar os «onsets» (*time points*) à grelha temporal definida previamente. Para tal, os «onsets» devem ser associados à divisão da grelha que se encontra mais próxima.

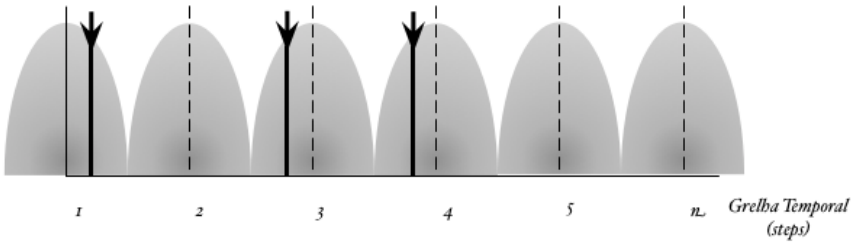


Figura 15 - Live Quantize: representação do método para associar cada evento rítmico («onset») à grelha temporal.

Como se pode observar na figura acima, as linhas verticais numeradas representam cada intervalo da grelha temporal definida previamente, e a área sombreada a cinzento representa o espaço temporal associado a cada intervalo de tempo da grelha. A linha vertical a negrito representa eventos rítmicos que, neste caso são três. Esta implementação garante que os eventos sejam associados ao intervalo da grelha mais próximo do evento, ou seja, quer ocorra antecipação ou atraso dos «onsets» em relação ao intervalo da grelha, este método garante que a associação seja correta.

Se a distribuição temporal (área a cinzento) coincidisse com o inicio de cada intervalo da grelha então o 2º evento na Figura 15 seria associado ao 2º intervalo da grelha em vez do 3º.

A implementação em MaxMSP deste método de quantização começa pela definição da resolução rítmica de quantização adequada à performance musical, que corresponde à grelha temporal, utilizando o objeto [metro] associado ao [transport]. Desta forma sempre que ocorrer uma alteração no andamento musical, o objeto [metro] acompanha essa alteração

[23] Beats Per Minute

automaticamente. Assim, o intervalo temporal do [metro] é definido em valores rítmicos (colcheias, semicolcheias, etc.) em vez de milissegundos.

A distribuição dada a cada intervalo da grelha temporal tal como representado na Figura 15 (responsável por associar cada evento ao intervalo da grelha mais próximo), é feito introduzindo um atraso com metade da duração do [metro] que define a resolução rítmica. Assim, se o módulo estiver configurado com uma resolução de quantização até à semicolcheia (16n) o atraso introduzido deverá ser uma fusa (32n) num compasso 4/4.

A performance rítmica é armazenada num objeto [multislider] através das amplitudes recebidas pelos «onsets». Quando não é recebido nenhum evento é atribuído uma amplitude igual a 0 naquele intervalo de tempo. Para as amplitudes serem armazenadas no intervalo correto, estes têm de ser numerados para estarem identificados. Para tal, é necessário calcular, conforme a resolução de quantização escolhida, quantos intervalos de tempo existe num compasso, isto é recebido pela mensagem *@steps* do objeto externo *kinetic* [kin.stratify].

O compasso é transformado num ciclo que varia entre [0.~1.]. O produto desta variável pelo número de intervalos de tempo de um compasso escolhido na resolução de quantização permite-nos identificar em tempo-real em que intervalo nos encontramos, e assim atribuir o «onset» ao intervalo correto.

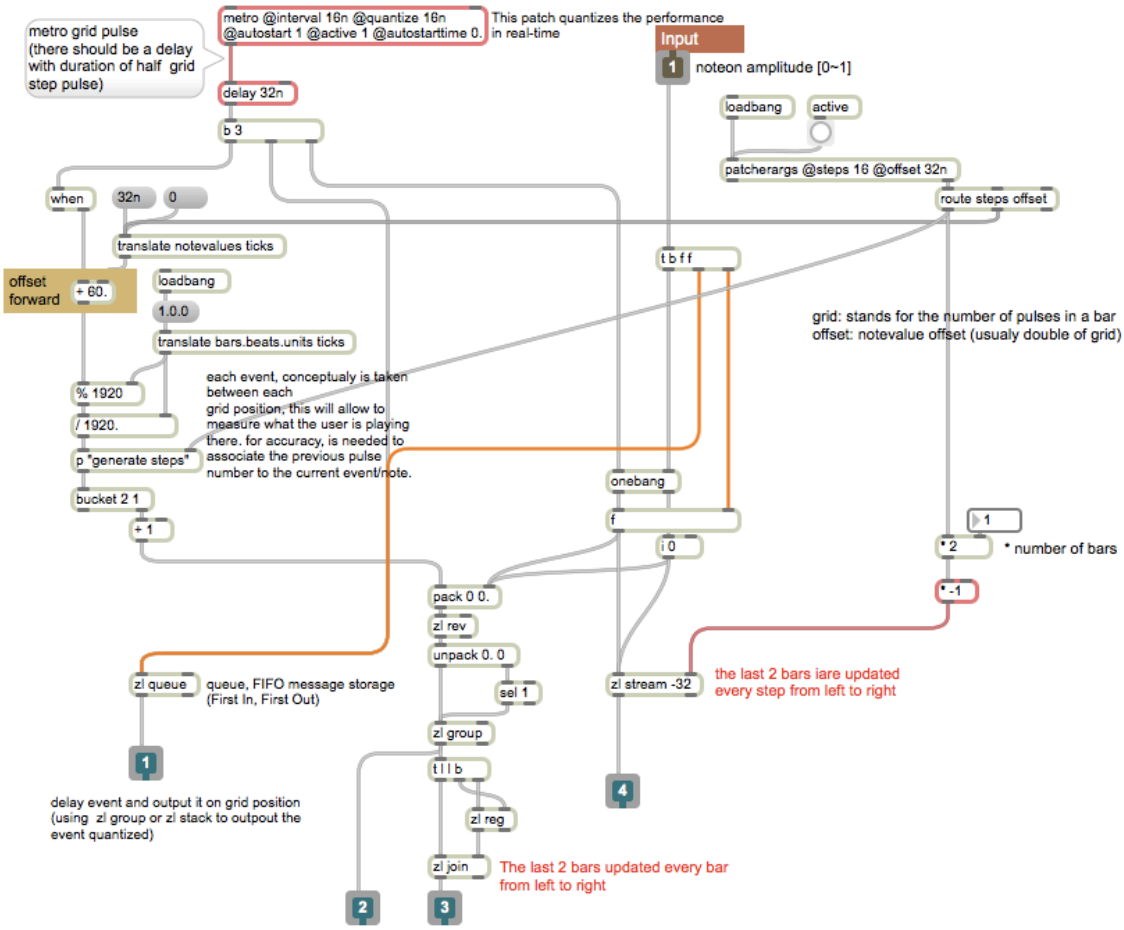


Figura 16 - O patch "Live Performance Quantizer"

No que respeita ao resultado final, este «patch» tem diversas saídas que podem ter diferentes utilidades. O [outlet 4] envia os dois últimos compassos atualizados a cada intervalo de quantização, o [oulet 3] envia os dois últimos compassos atualizados a cada compasso, o [outlet 2] envia apenas o último compasso atualizado a cada compasso e o [outlet 1] envia a ultima nota recebida.

3.3.1.3. Measure Amplitude Average

Este módulo tem a função de calcular a média de amplitudes ao longo da performance. Isto oferece uma noção global em tempo-real da performance em termos de dinâmica ao longo dos 2 últimos compassos, se esta é fortíssimo ou pianíssimo numa escala normalizada entre [0.~1.].

Este patch funciona em resposta ao patch "live performance quantizer", logo para funcionar deve estar ligado a este e receber os dados devidamente formatos como uma lista de MaxMSP. Foram testados diferentes métodos e aquele que apresentou melhores resultados consiste na divisão das soma de todas as amplitudes pela contagem de «onsets».

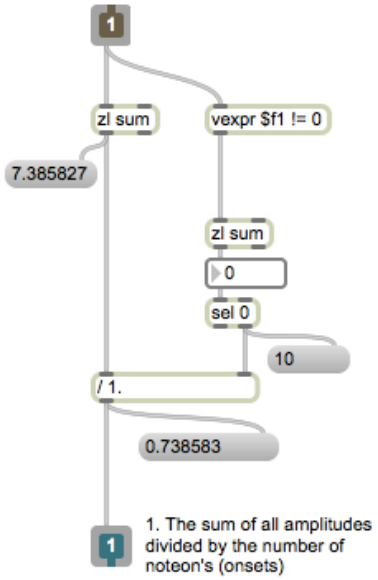


Figura 17 - Patch responsável por calcular a média de amplitudes ao longo da performance.

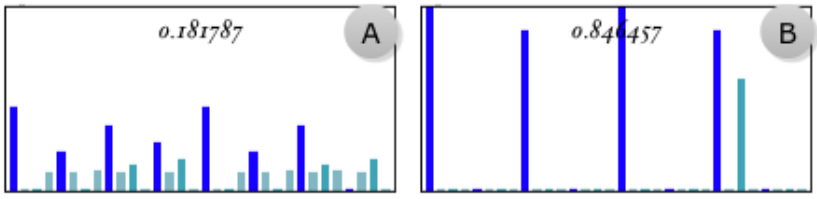


Figura 18 - Exemplos de cálculos da média de amplitudes: (A) Neste caso a densidade é alta mas a média de amplitudes é baixa 0.18; (B) Neste caso a densidade de eventos é baixa mas a média de amplitudes é alta 0.85.

3.3.1.4. Measure Variance Average

Este módulo tem função de calcular a média de variações ao longo da performance, isto é, este módulo deteta se o utilizador executa padrões de forma repetida ou se estes são sempre diferentes ao longo do tempo. Se a performance for repetitiva o rácio baixa, e torna-se 0 se

performance for sempre igual. Por outro lado se os padrões rítmicos estiverem constantemente a variar o rácio sobe até um máximo de 1., o que significa que os últimos padrões executados são totalmente diferentes daqueles executados anteriormente. Para este módulo implementou-se diversos métodos para medir as variações. Será descrito mais adiante as duas implementações com as quais se obteve melhores resultados.

Para a mediação das variações há duas variáveis a ter em consideração, o padrão rítmico (a sequência de «onsets» dentro da janela temporal definida, dois compassos), e a variação de amplitudes (a variação dinâmica dos «onsets» quando comparados uns com os outros).

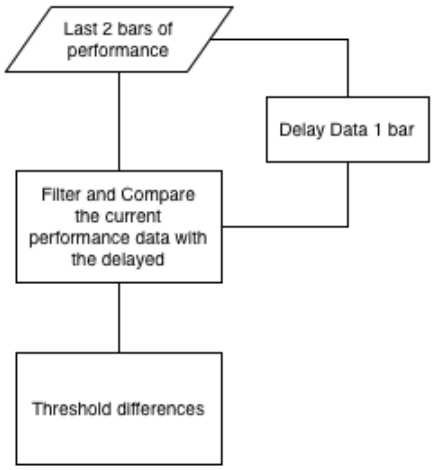


Figura 19 - Visão global do módulo (patch) para medir as variações.

Este recebe como entrada os dois últimos compassos da performance, que são atualizados a cada intervalo de tempo definido pela resolução de quantização no módulo [live performance quantizer]. Estes dados são preservados temporariamente numa memória de transferência (*buffer*) que os envia para o processo seguinte com um compasso de atraso.

Neste processo são medidas as alterações entre o padrão executado atual e o padrão passado (recebido pelo buffer) calculando as diferenças absolutas entre o padrão atual e o padrão passado.

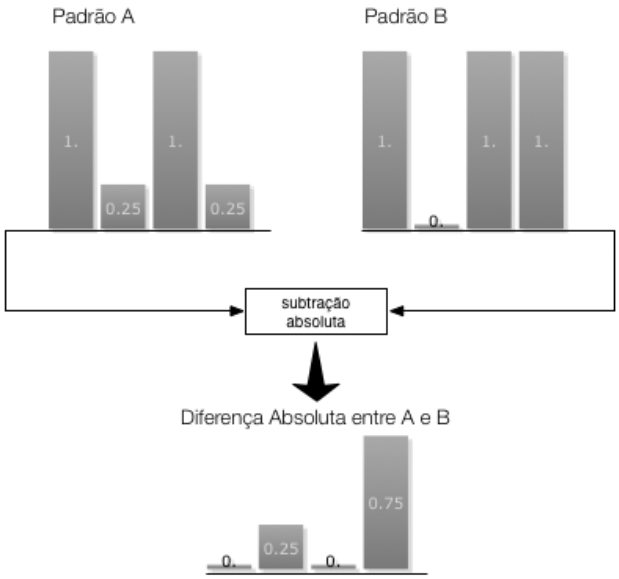


Figura 20 - Cálculo das diferenças entre o padrão atual (A) e o padrão passado (B), que foi tocado antes do A.

Após esta iteração, o padrão A passa para a memória temporária, tornando-se no padrão B na próxima iteração.

3.3.1.4.1. *MÉTODO 1*

Antes do cálculo das diferenças o padrão passado passa por um filtro. Este tem a finalidade de reduzir algumas imprecisões no algoritmo, nomeadamente, para evitar quando o utilizador finaliza a performance (quando passa para o estado «idle mode») que o rácio de variações suba, caso contrário isto iria acontecer um compasso depois. Uma das razões pela qual acontece este fenómeno é que como são calculadas as diferenças absolutas, as diferenças negativas não são ignoradas para não descartar situações em que o padrão se repete mas as amplitudes variam. Desta forma o cálculo de variações depende sobretudo daquilo que está a acontecer no padrão atual. Se não ocorrer nenhum evento, então os eventos do padrão passado serão filtrados e é como se a comparação ocorresse apenas num sentido. Isto evita algum ruído nas medições efetuadas.

A filtragem é feita com o objeto [vexpr] e tem a seguinte expressão:

[vexpr ($f1 != 0.) || ($f2 == 0.)]

Isto significa que:

- Sempre que ocorrer um evento no padrão atual ou sempre que não ocorra um evento no padrão passado, então os eventos no padrão passado serão considerados aceites para a extração das diferenças de amplitude entre os padrões.
- Ou seja, sempre que ocorra um evento no padrão passado que não ocorra na pulsação equivalente no padrão atual, esse evento é filtrado do padrão passado para a subtração de amplitudes.
- Se em ambos padrões e respetivas pulsações não ocorrer nenhum evento, este caso não cria nenhum conflito.

O resultado do cálculo das diferenças é então enviado para outro filtro na etapa seguinte. Este filtro tem a função de decidir quais as diferenças de amplitude que contribuem realmente para a variação. Este consiste em comparar as amplitudes com um limiar («threshold») predefinido. Se a variação de amplitudes for superior ao limiar então é considerado que contribui para a variação da performance.

O limiar predefinido é 0.1 que em valores MIDI corresponde a 0.1*127=13

Assim, todas as variações de amplitudes menores que 0.1 (ou 13 em MIDI) serão ignoradas na etapa seguinte, como se se tratasse de pequenas imprecisões normais na performance humana.

Nesta etapa, todas as variações de amplitudes são somadas e dividas pelo número de intervalos de quantização num compasso, definido previamente pelo utilizador.

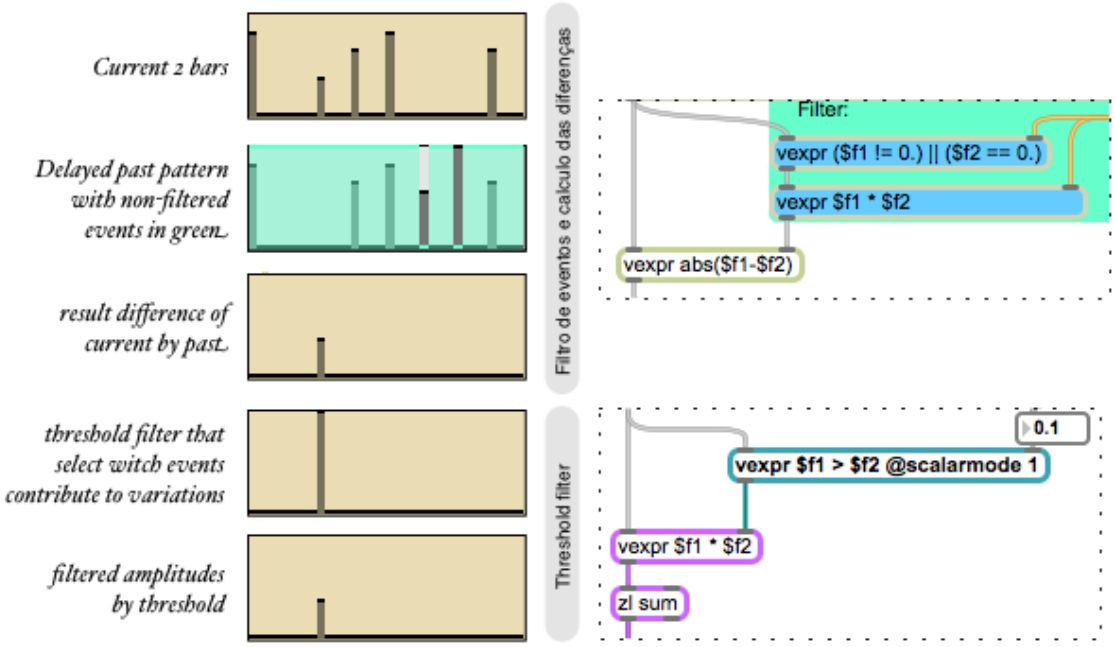


Figura 21 – Representação de duas etapas importantes, (1) filtragem de eventos e cálculo das diferenças e (2) filtragem de variações que não contribuem para o cálculo de variações.

3.3.1.4.2. *MÉTODO 2*

O segundo método é bastante semelhante ao primeiro, diferindo apenas no primeiro filtro implementado.

Este não exclui eventos no padrão passado que não ocorrem no padrão atual na respetiva pulsação, como acontece na primeira implementação. Os eventos serão apenas excluídos do padrão passado quando o utilizador parar a performance, isto é, o padrão atual corresponde apenas a uma lista com amplitudes 0, sem qualquer evento.

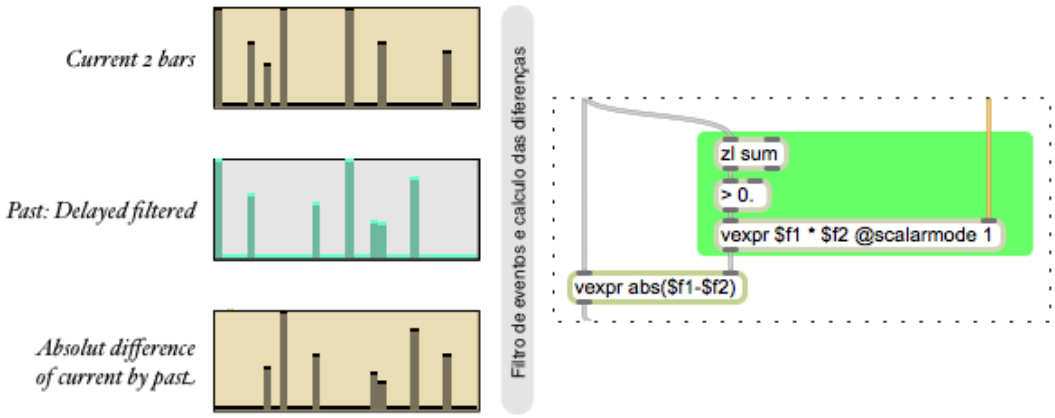


Figura 22 - Método 2: cálculo das diferenças entre padrões e filtragem de eventos. Todos os eventos sombreados a verde não foram ignorados, caso contrário estariam a negro.

3.3.1.5. MEASURE SYNCOPATION AVERAGE

Este modulo foi criado a partir da adaptação de um dos algoritmos utilizados na aplicação *kin.Recombinator*. Esta aplicação, numa descrição geral, permite ordenar uma coleção de loops em formato MIDI de acordo com a sua complexidade, isto é, do mais simples até ao mais complexo, para que o utilizador possa posteriormente criar combinações e variações entre eles. Pode-se consultar informação mais detalhada em (Sioros & Guedes, 2011).

Este módulo tem a função de medir o grau de sincopação dos padrões rítmicos executados pelo utilizador, isto é, de que forma é que a performance rítmica contradiz o modelo que caracteriza o compasso musical em questão. Este cálculo é feito com a ajuda do objeto [kin.OffBeatDetector] que recebe duas listas, uma com o modelo do compasso musical em questão e outra com as amplitudes da performance musical.

O modelo é calculado com a ajuda do objeto [kin.stratify] que calcula a estratificação para cada pulsação do compasso dado o tipo de compasso musical e o nível de estratificação (quantização) selecionado. A partir destes valores é calculado o valor hierárquico de cada pulsação, sendo utilizado depois para a comparação com o padrão rítmico tocado pelo músico.

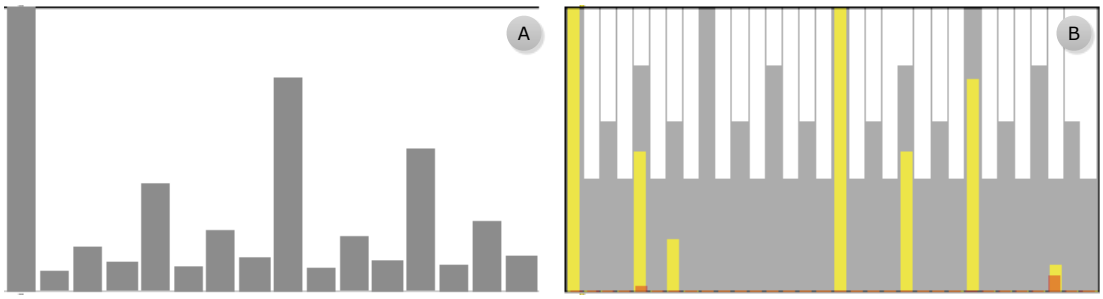
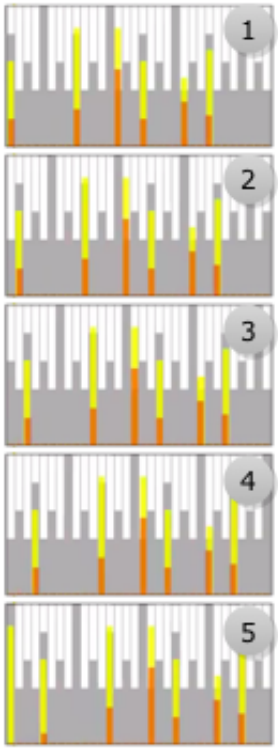


Figura 23 - (A) Estratificação dada pelo objeto [kin.stratify] para um compasso 4/4 ao nível da semicolcheia, 16n; (B) cinzento = nível hierárquico de cada pulsação, amarelo = padrão rítmico tocado, laranja = notas do padrão rítmico que contribuem para a sincopação.

Este objeto faz parte dos algoritmos que compõem a aplicação *kin.Recombinator*.

No caso deste módulo, a novidade consiste na adaptação do algoritmo para permitir a medição da sincopação em tempo-real de forma compatível com os dados recebidos pelo módulo [live performance quantizer].

Este gera os seus dados de saída a uma frequência definida pela resolução de quantização. Por exemplo, se a resolução de quantização foi definida até à semicolcheia (16n), então o resultado da quantização consiste num padrão dos dois últimos compassos tocados em formato de lista de amplitudes, emitido em fluxo, ou seja, a lista é atualizada de forma rotativa, o que significa que a amplitude recebida mais recente encontra-se sempre no primeiro índice da lista, mas que esta mantém sempre o tamanho constante. Assim, para manter coerente a medição de sincopação, o modelo com os níveis hierárquicos usado no objeto [Kin.OffBeatDetector] tem de ser sincronizado com os dados em fluxo recebidos do módulo [live performance quantizer].



A sincronização é feita aplicando rotação aos elementos da lista que contém o modelo do nível hierárquico de cada pulsação, com resolução igual àquela definida na quantização, para que cada elemento da lista com o ritmo tocado corresponda ao respetivo nível hierárquico dado pelo modelo durante a análise.

Figura 24 – Ex. das listas em rotação sincronizadas ao longo de 5 passos. Cada passo corresponde ao intervalo definido na quantização (16n). A lista amarela corresponde à performance musical e a lista cinzenta corresponde ao modelo hierárquico do compasso musical (4/4) e a laranja corresponde à contribuição de cada pulsação para a sincopação.

O resultado final deste módulo, consiste na soma e normalização de todas as pulsações que contribuem para a sincopação. A normalização (valores de sincopação variam entre 0.~1.) é calculada dividindo a soma do padrão de sincopação pelo valor máximo de sincopação. Este valor máximo consiste na soma de todas as pulsações que contradizem o modelo predefinido com amplitude máxima.

3.3.1.6. MEASURE DENSITY AVERAGE

Este módulo tem a finalidade de medir a densidade de eventos dentro da janela temporal utilizada na análise, neste caso dois compassos. Novamente, pretende-se que estes parâmetros de análise correspondam a uma representação global da performance.

A densidade consiste na relação entre a massa ou o peso de um corpo com o seu volume ou com a massa ou peso de outro corpo do mesmo volume. (Priberam, 2011)

Neste caso, são medidos eventos rítmicos e o meio de armazenamento consiste na janela temporal predefinida para a análise (dois compassos) e a sua relação com a resolução de quantização predefinida (por exemplo semicolcheias, 16n). Se o compasso musical for 4/4 então o máximo de densidade possível são 32 notas, porque cada compasso 4/4 divido em semicolcheias consiste em 16 intervalos (pulsações).

Neste módulo, o cálculo da densidade normalizada consiste em dividir a quantidade de eventos detetados pelo número máximo de densidade possível, sendo que o resultado varia entre 0.~1. A amplitude dos eventos rítmicos não é considerada nesta análise.

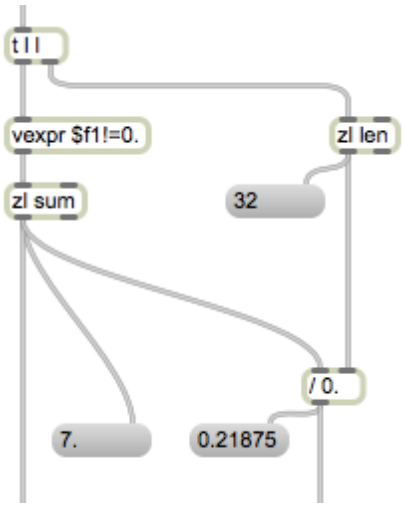


Figura 25 - Exemplo do cálculo da densidade. Foram detetados 7 eventos e o número máximo de eventos possível são 32. O resultado normalizado da densidade é 0.2

3.3.1.7. IDLE STATE DETECTION

Este módulo tem a finalidade de detetar se o utilizador se encontra em estado ócio (*idle state*). Esta informação permite saber quando é que o utilizador cessa ou inicia a performance.

Este módulo divide-se em várias etapas:

- Observação do número de eventos
- Limiar em estado ócio, tempo de decaimento da performance para confirmar o estado ócio.
- Contagem do tempo em estado ócio e envio do resultado.

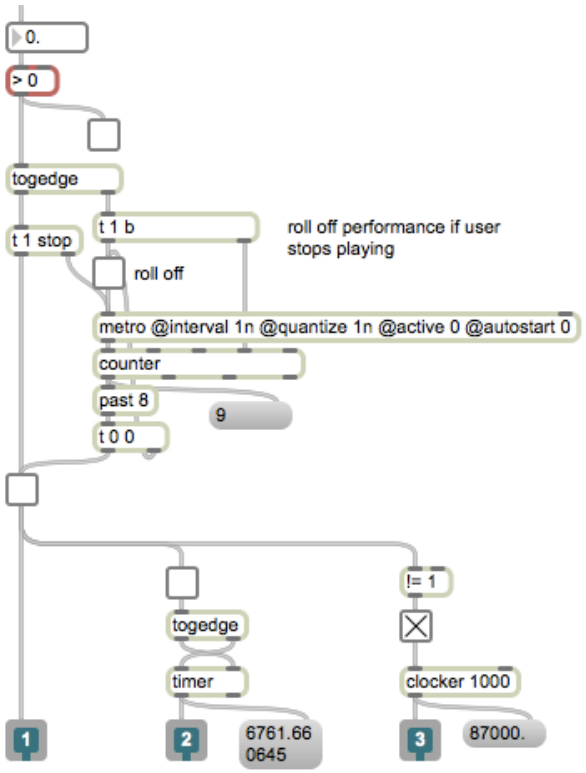


Figura 26 - Módulo de "idle state detection"

A primeira etapa consiste em detetar se não há eventos a ocorrer na performance. Caso seja verdade então a segunda etapa é ativada. Esta observa o número de eventos dado pelo módulo descrito anteriormente, [measure density average].

Na segunda etapa, se não ocorrer nenhum evento durante 8 compassos, então o estado ócio é ativado e por sua vez a terceira etapa.

Este tempo de espera ao longo de 8 compassos permite que ocorra um tempo de decaimento na performance do utilizador, isto é, permite que haja silêncio entre cada frase musical ao longo da performance do utilizador sem que ocorra mudanças de estado de forma aleatória. O tempo de decaimento foi afinado de forma empírica, mas pode ser facilmente alterado pelo utilizador.

Assim, se não ocorrer nenhum evento da parte do utilizador ao longo de 8 compassos, então é considerado que este decidiu parar mesmo e são ativados 2 relógios na terceira etapa.

Um deles reporta em tempo-real o tempo em milissegundos que o utilizador se encontra em estado ócio, e o outro reporta a duração total em estado ócio quando o utilizador mudar novamente para estado ativo.

3.3.1.8. MEMORY ANALYSIS

Neste módulo são realizados uma série de cálculos a partir dos dados analisados anteriormente que representam características globais relativamente à performance (amplitude, variações, sincopação, densidade). Esta análise pretende de certa forma simular alguns processos da memória e cognição humana. Isto é, a partir dos dados extraídos da performance, pretende-se inferir estados e comportamentos da performance observando e analisando a memória a dois níveis, curto prazo e longo prazo. Ao observar o comportamento da performance, pretende-se também calcular a tendência do mesmo no futuro.

3.3.1.8.1. *SHORT TERM AND LONG TERM ANALYSIS*

Como se pode observar na Figura 13 (pág. 43) a análise da performance ocorre em duas etapas. Primeiro extrai-se informação global relativamente aos dados musicais recebidos pela performance do músico, e em segundo procura-se inferir outro nível de informação relativamente à informação analisada anteriormente. A segunda etapa consiste no módulo [Memory Analysis].

Este módulo divide-se em dois tipos de análise: análise a curto prazo e análise a longo prazo. Na realidade elas até diferem muito pouco entre si exceto numa variável, a duração da janela temporal de análise. Esta depende de duas variáveis, o intervalo de análise e a capacidade da memória de armazenamento dos valores da análise. Ou seja, por um lado o intervalo de análise define qual a frequência da análise em intervalos musicais (ex. semínima), isto é, qual a frequência a que os cálculos serão executados, e por outro lado a capacidade de armazenamento da memória define a distância máxima possível para observar o passado em termos do que aconteceu na performance musical, dado ser uma área de armazenamento com capacidade finita.

A principal finalidade desta análise consiste em retirar alguma informação relativamente ao que já ocorreu na performance ao longo do tempo, isto é, por outras palavras procura-se responder à questão: Qual a diferença relativamente ao que acontece no instante (ou estado) atual da performance em relação ao que já aconteceu no passado da mesma?

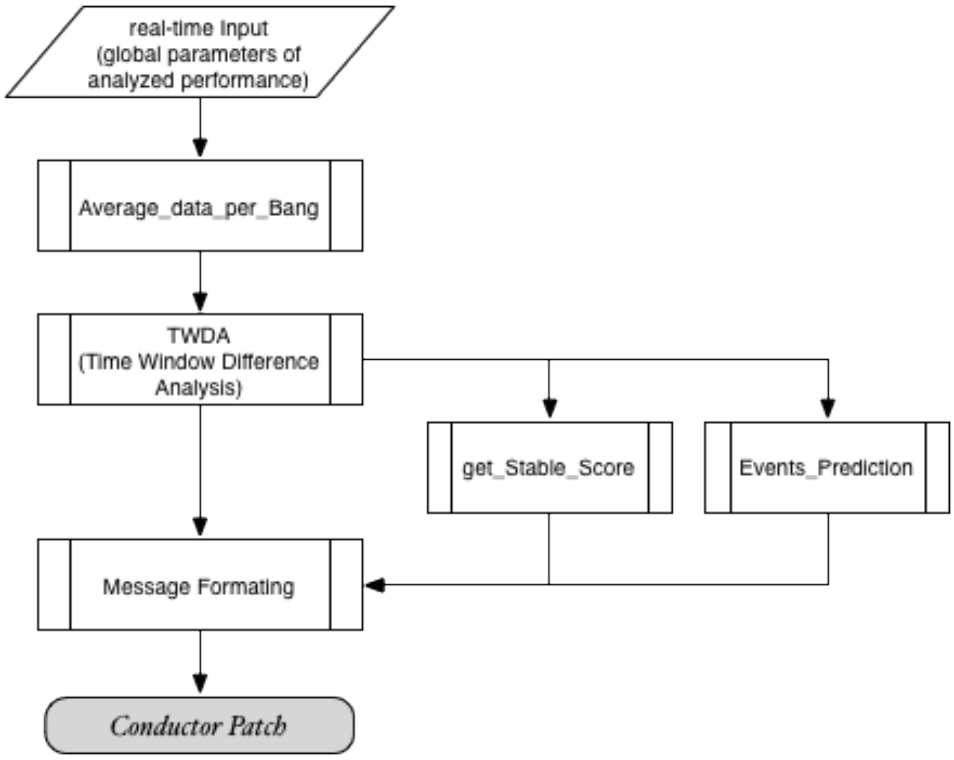


Figura 27 - Diagrama geral do módulo "Memory Analysis"

3.3.1.8.2. *AVERAGE DATA PER BANG*

A primeira etapa para responder à questão colocada anteriormente, consiste em retirar observações do estado atual da performance e armazená-las numa memória temporária. Falou-se anteriormente que a duração da janela temporal de análise depende de duas variáveis, o intervalo musical de observação dos valores recebidos (ou frequência) e a capacidade de armazenamento da memória temporária. É nesta abstração (patch de MaxMSP) que é definido o valor da primeira variável. Esta define-se em intervalos musicais pois depende do andamento musical da peça ou improvisação. Isto é, o valor em milissegundos de uma semínima (4n) numa música a 120 bpm é diferente se estiver a 90 bpm, logo, como o objeto [metro] está associado ao objeto [transport], sempre que ocorra uma mudança no andamento musical este mantém-se sempre sincronizado, graças à capacidade do objeto [transport] controlar remotamente o tempo relativo de todos os objetos associados a este. Assim, o argumento para instanciar o objeto [metro] é o valor de uma figura rítmica, e o intervalo de tempo é sempre relativo ao andamento musical.

A análise a curto e longo prazo diferem nesta variável, uma é inicializada com intervalo de semínima (4n) e a outra é inicializada com intervalo de 1 compasso (1n), respetivamente. O

valor mínimo ideal para a resolução de quantização situa-se perto da semicolcheia (16n). Como esta frequência é superior àquela definida para esta abstração (análise a curto e longo prazo), todos os valores intermédios recebidos são guardados temporariamente. Ao receber um «bang» esta abstração calcula a média de todos esses valores e envia o resultado final para a memória auxiliar de armazenamento.

Resumindo, esta abstração tem a função de capturar (à frequência predefinida para curto e longo prazo) todos os dados globais recebidos da primeira instância de análise, e calcula a média para enviar o valor final para o processo seguinte (TWDA).

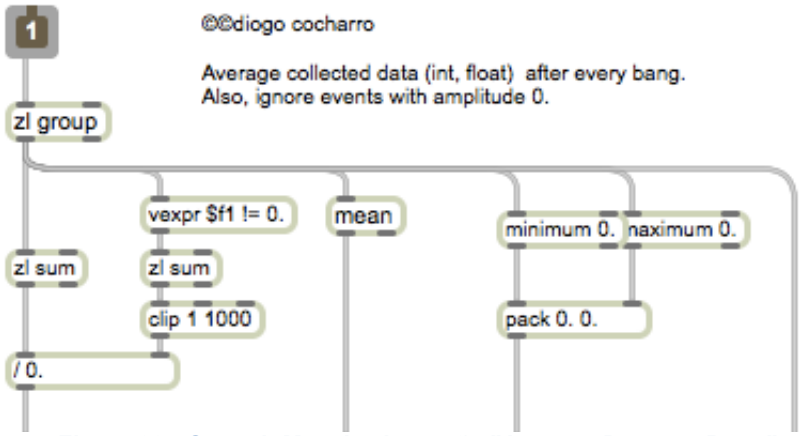


Figura 28 - O patch Max da abstração "Average Data per Bang"

A abstração tem várias funcionalidades apesar de não serem todas utilizadas na versão atual: o 4º «outlet» envia todos os valores capturados, o 3º envia o mínimo e o máximo entre os valores capturados, o 2º envia a média absoluta e o 1º «outlet» envia a média de todos os valores diferentes de zero.

3.3.1.8.3. *TIME WINDOW DIFFERENCE ANALYSIS (TWDA)*

Esta abstração tem como função armazenar os dados obtidos a partir da abstração descrita anteriormente e calcular as diferenças entre o estado atual da performance e o sucedido no passado. Logo, é nesta abstração que se encontra a segunda variável que influência a duração da janela temporal de análise, ou seja, é aqui que se define a capacidade de armazenamento da memória temporária.

O valor da capacidade predefinida para ambas as análises (curto e longo prazo) é de 8 valores, mas tendo em conta que a frequência definida na abstração "Average Data per Bang" é diferente em cada análise, por consequência, a relação entre estas duas variáveis torna a distância máxima de observação diferente. Isto significa que oito valores resultantes do cálculo da média a uma frequência de semínima (4n) equivalem a uma distância máxima de observação de 2 compassos, num contexto musical 4/4, enquanto na análise a longo prazo que está definida para calcular a média de compasso em compasso (1n), a distancia máxima de observação equivale a 8 compassos.

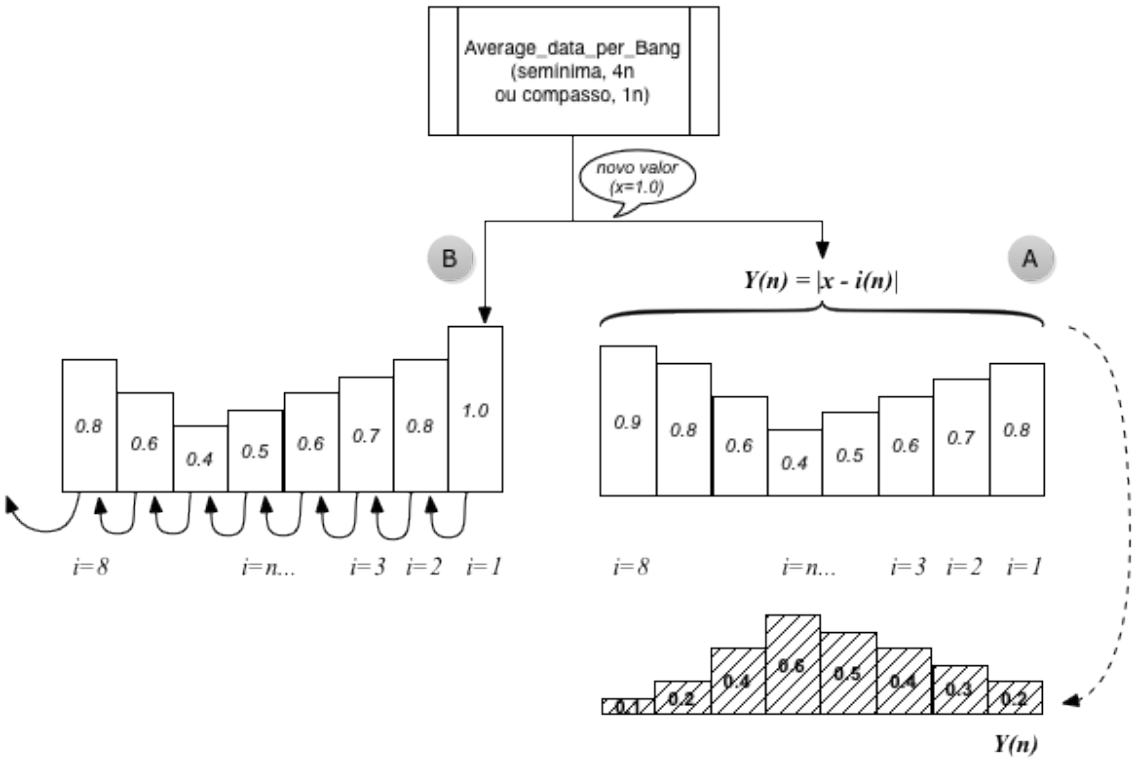


Figura 29 - «TWDA» (A) Cálculo das diferenças entre o novo valor recebido e os valores sucedidos; (B) Armazenamento do novo valor.

A Figura 29 representa os principais processos que ocorrem na abstração «TWDA». O resultado das diferenças representa se houve muitas variações ao longo do período de tempo que está a ser observado, isto é, se os valores se manterem próximos de zero significa que não ocorreram muitas variações no parâmetro que estiver a ser analisado. Outra forma de interpretar, seria calcular a média de todos os resultados das diferenças, obtendo desta forma um valor que representa a quantidade de variação ao longo daquele intervalo de tempo.

O cálculo das diferenças pode ser feito escolhendo um de dois métodos, ou através das diferenças absolutas como está representado na figura, ou através da distância Euclidiana. Por

defeito está selecionado o primeiro método, mas o utilizador pode escolher o segundo método editando o patch de Max.

Em termos de interpretação da análise a curto e longo prazo, pode-se interpretar os seguintes casos:

- Se os resultados das diferenças a curto e longo prazo forem próximos de zero, significa que não houve alterações no parâmetro que está a ser analisado (manteve-se repetitivo).
- Se o resultado das diferenças a curto prazo for um valor próximo de 1, e o resultado a longo prazo se manter próximo de zero, significa que ocorreu uma pequena variação musical durante um curto espaço de tempo e que em seguida voltou, provavelmente, a executar o que estava a tocar antes.
- Se o resultado das diferenças a curto e longo prazo forem próximos de 1, significa que ocorreu uma transição ou modulação musical, isto é, ou a performance se alterou subitamente, ou ocorreu uma transição não repetitiva durante um longo período de tempo e, provavelmente, no final dessa transição não voltou a executar o que estava a tocar antes.

3.3.1.8.4. *GETSTABLESCORE*

Esta abstração de MaxMSP tem a finalidade de detetar se o estado atual da performance corresponde a uma parte estável, no sentido em que é repetitiva, ou se a performance consiste num momento em que ocorre muitas variações musicais.

Assim, este «patch» recebe o valor médio das diferenças calculadas na análise TWDA, e a partir do valor definido por dois limiares («thresholds») decide se o estado atual da performance corresponde a uma parte estável (que dura há algum tempo e se tem vindo a repetir) ou é instável (por exemplo, durante uma transição, ou quando há muitas variações na performance).

A utilização de dois limiares torna esta análise um pouco mais robusta, no sentido de filtrar algumas imprecisões. Existe um limiar superior e um inferior. Enquanto o estado atual da performance se mantiver abaixo do limiar superior, a performance é considerada em modo estável, ao ultrapassar esse limiar passa automaticamente para modo instável.

Quando a performance está em modo instável só passa a ser considerada modo estável quando o valor do estado atual da performance descer abaixo do limiar inferior. Normalmente o valor do limiar superior é sempre maior que o valor do limiar inferior.

Os limiares nunca são considerados ao mesmo tempo, isto é, o limiar que entra em vigor depende sempre da análise do estado atual da performance. Por exemplo, se a performance estiver em modo estável entra em vigor o limiar superior, quando o seu estado se alterar para modo instável, o limiar inferior para a estar em vigor e o superior é desativado.

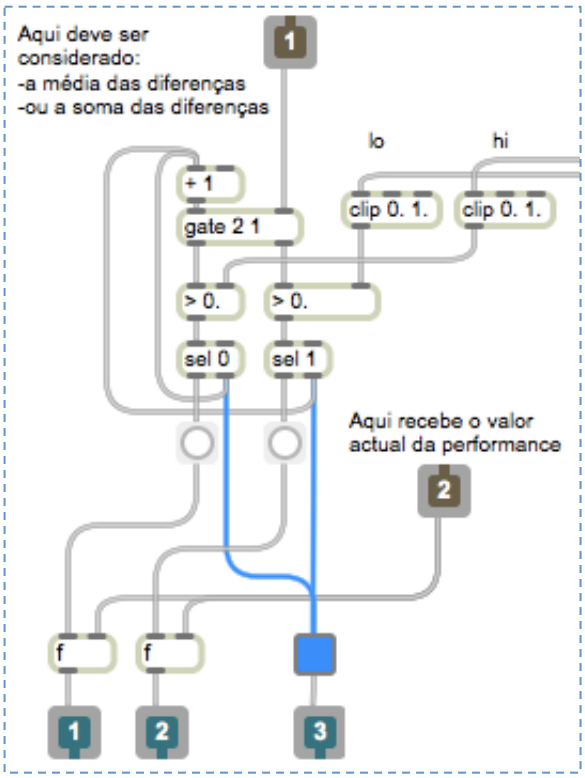


Figura 30 - Recorte do patch «getStableScore»

Como se pode verificar na figura anterior este «patch» tem três «outlets», quando a performance está em modo estável o valor do estado atual da performance é enviado pelo 1º «outlet», e em modo instável é enviado pelo 2º «outlet». O 3º «outlet» envia 0 ou 1 de um «toggle» como sinalizador, 1 para modo instável e 0 para modo estável.

3.3.1.8.5. *EVENTS PREDICTION*

Esta abstração tenta prever no futuro a direção que o comportamento da performance irá tomar, baseando-se no comportamento dos eventos passados e utilizando como ponto de partida o estado atual da performance. Por outras palavras, pretende-se extrapolar o comportamento da performance musical a partir do seu estado atual, utilizando como referência a memória da performance já sucedida. O método aplicado utiliza o valor do estado atual da performance, e o resultado das diferenças obtidas na abstração «TWDA», para efetuar os cálculos necessários à extrapolação.

A derivada representa a taxa de variação instantânea de uma função, e a lista de diferenças «TWDA» representa o comportamento da performance relativamente ao que já sucedeu.

Desta forma, partindo do cálculo da derivada dos itens da lista de diferenças, pretende-se aplicar a taxa de variação do comportamento da performance ao valor do estado atual da performance, isto é, a taxa de variação do comportamento da performance vai acumulando (somando ou subtraindo) a partir do valor do estado atual.

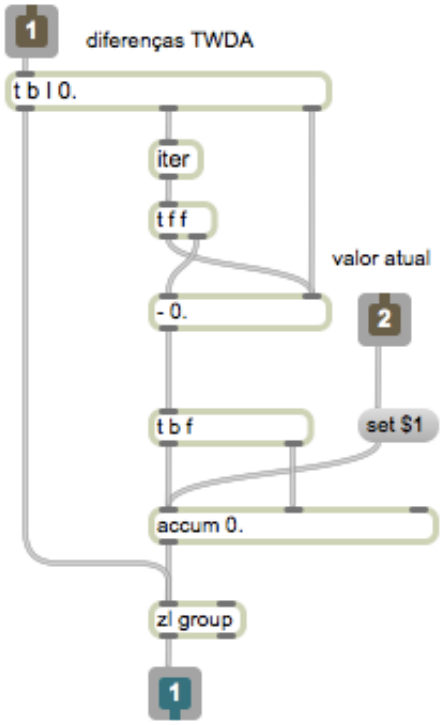


Figura 31 - Patch «EventsPrediction_v01»

A previsão de parâmetros é uma área de estudo da análise numérica. Entre outras, algumas técnicas populares são a interpolação, extrapolação, e regressão. Também as redes neuronais e algoritmos genéticos, correlação, filtros de *kalman*, etc. têm sido explorados na previsão de parâmetros.

Apesar de este método não ser o mais sofisticado, apresenta pelo menos um comportamento coerente, podendo ser utilizado para produzir comportamentos da performance não previstos pelo utilizador. Uma funcionalidade a considerar, seria medir em tempo-real a divergência entre a previsão e o estado atual da performance.

3.3.1.8.6. *MESSAGE FORMATING*

Esta abstração é fundamental, no sentido em que é a partir daqui que se preparam todas as mensagens enviadas para o módulo seguinte, o «Conductor», ou qualquer outro módulo que o utilizador deseje desenvolver. O método para formatação das mensagens é semelhante ao paradigma utilizado em OSC, as mensagens são catalogadas de acordo com o seu conteúdo, funcionando como endereço para aceder a estas.

De seguida são apresentadas as mensagens enviadas pelo módulo de análise em estrutura de árvore e descritas com o seu limite mínimo e máximo:

- 1ª instância de análise:
    - idle ([0 ou 1], 0 = estado de ócio, 1 = activo);
    - density ([0~1], densidade de notas dentro da janela de análise);
    - syncopation ([0~1], mediação da sincopação do padrão executado);
    - variance ([0~1], mediação das variações do padrão executado);
    - amplitude ([0~1], medição global da intensidade do padrão executado);
- Memory Analysis (análise executada a partir dos parâmetros medidos na 1ª instância):
    - amp / var / syn / den (abreviatura dos parâmetros anteriores);
        - sta / lta (short-term analysis / long-term analysis);
        - last (último valor usado no cálculo das diferenças, o estado atual);

- dif («array» ou lista do cálculo das diferenças);
- pred (previsão da evolução do parâmetro);
- transition («toggle» da transição entre modo estável e instável);
- unstable («float» dos valores em modo instável);
- stable («float» dos valores em modo estável);
- dif_sum (soma dos valores não duplicados da lista de diferenças);
- dif_avg (média dos valores diferentes de zero da lista de diferenças);

Os mesmos métodos de análise são aplicados nos parâmetros *amplitude*, *variance*, *syncopation* e *density*, e em ambos níveis, a curto e longo prazo.

3.3.2. Análise da Performance a Partir do Controlador Gestual Wiimote

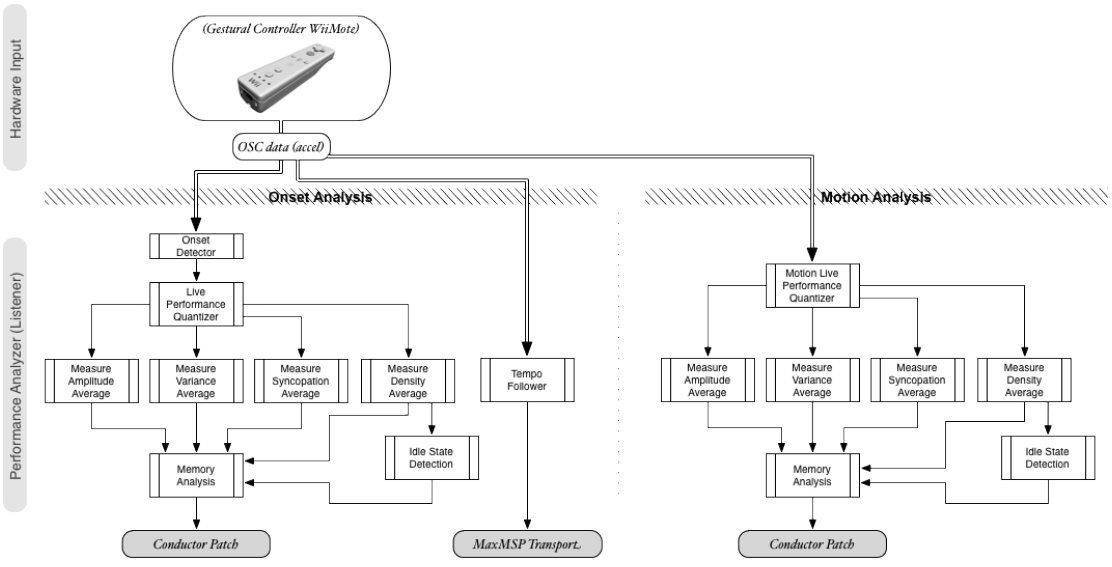


Figura 32 - Diagrama geral da análise da performance a partir do controlador Wiimote.

Este módulo assemelha-se com o módulo descrito anteriormente sobre a análise da performance com instrumentos MIDI. No entanto, o tipo de dados recebido a partir do *Wiimote* por OSC não tem o mesmo formato que os dados MIDI, do qual facilmente podemos extrair «onsets», «offsets», duração, tempos delta, etc.

O *wiimote* envia os dados dos seus sensores a partir do software *Osculator*, que permite reconhecer diversos sensores/controladores e enviar os seus dados para qualquer destinatário formatados como MIDI ou OSC. Tendo em conta que o *wiimote* é constituído por acelerómetros e giroscópios, os dados recebidos em MaxMSP a partir do *Osculator* aparecem como um fluxo constante de valores lidos dos sensores do *wiimote*, mesmo que este se encontre em repouso.

A base de métodos para análise da performance com o *wiimote* é a mesma utilizada com instrumentos MIDI. A diferença reside no tratamento prévio aos dados recebidos por OSC em MaxMSP, enviados pelo *Osculator*, por isso faz-se duas abordagens ao tratamento dos dados do *wiimote:* uma abordagem rítmica e outra gestual. Isto é, na abordagem rítmica, o *wiimote* é tratado como se fosse um instrumento do tipo *shaker*, onde os movimentos são cíclicos e rítmicos, e por isso procura-se extrair «onsets» do gesto executado para formar padrões rítmicos e por sua vez analisados. Na abordagem gestual, os dados gestuais são filtrados e quantizados, e por fim analisados utilizando os mesmo métodos já descritos.

3.3.2.1. GET_WII_MOTION_ANALYSIS_(V05)

Esta abstração tem duas finalidades, filtrar os dados gestuais do *wiimote* e detetar «onsets» nesses mesmos dados. Posteriormente os gestos filtrados são encaminhados para a análise gestual e os «onsets» detetados são encaminhados para a análise rítmica.

Durante as experiências iniciais com esta abstração, chegou-se a converter os dados gestuais em sinal áudio para tirar proveito de alguns objetos nativos do MaxMSP que auxiliam na deteção de «onsets», como o [change]~, [spike~], [edge~], [sync~]. Mas, com a intenção de que esta abstração não dependesse de DSP[24] e com o intuito de poupar recursos do computador procurou-se outras soluções.

A abordagem seguinte surgiu ao consultar um artigo sobre o projeto «Wiigee» (Schlöomer, Poppinga, Henze, & Boll, 2008) que descreve uma biblioteca em *Java* para reconhecimento gestual utilizando o *wiimote*. Neste artigo é descrito alguns dos métodos utilizados para

[24] Digital Signal Processing

filtragem dos dados recebidos do *wiimote*, nomeadamente o «idle state filter» e o «directional equivalence filter», um que filtra dados que não contribuem para o gesto a partir de um «threshold» e outro que filtra dados que são redundantes.

Alguns desses métodos foram transcritos como patches de Max, mas exigiam alguma afinação devido ao atraso temporal que introduziam no fluxo de dados.

A finalidade na aplicação destes filtros surge da necessidade de reduzir e simplificar os dados provenientes dos acelerómetros do *wiimote*, para otimizar outros processos que dependam desses dados. Também no intuito de reduzir ao mínimo a utilização de objetos externos ao Max a abstração apresenta atualmente a estrutura observada na Figura 33.

O primeiro filtro baseia-se no objeto [slide] que é nativo do Max. O segundo filtro consiste numa abstração baseada na fórmula de filtro «lowpass» sugerida por Curtis Roads (Roads, 1996, p. 403). O terceiro filtro consiste numa combinação entre os objetos [slide], [function] e [pow] em que o filtro se ajusta dinamicamente, isto é, quando a amplitude dos dados dos acelerómetros é baixa a sensibilidade do filtro aumenta, à medida que a amplitude se aproxima do máximo (1) a sensibilidade do filtro diminui. Isto permite preservar os picos mais importantes dos gestos e filtrar com mais eficácia o ruído de baixa amplitude.

A partir daqui os dados filtrados são re-encaminhados para a abstração de análise gestual e para a filtragem final antes da deteção de «onsets». O último filtro «directional equivalence» já foi referido anteriormente.

Com os dados gestuais devidamente simplificados e reduzidos, a deteção de «onsets» baseia-se na derivada dos dados recebidos. Um gesto a partir do acelerómetro, consiste num sinal que tem um ataque, atinge um pico e decresce. No momento do ataque a derivada dá um valor negativo, quando atinge o pico o declive é zero, logo a derivada dá zero e a partir do momento que decresce a derivada dá um valor positivo. É neste comportamento, a transição de valores negativos para valores positivos, que se baseia a deteção de «onsets». Quando é detetado um «onset», a abstração envia o valor do pico de amplitude e um «bang».

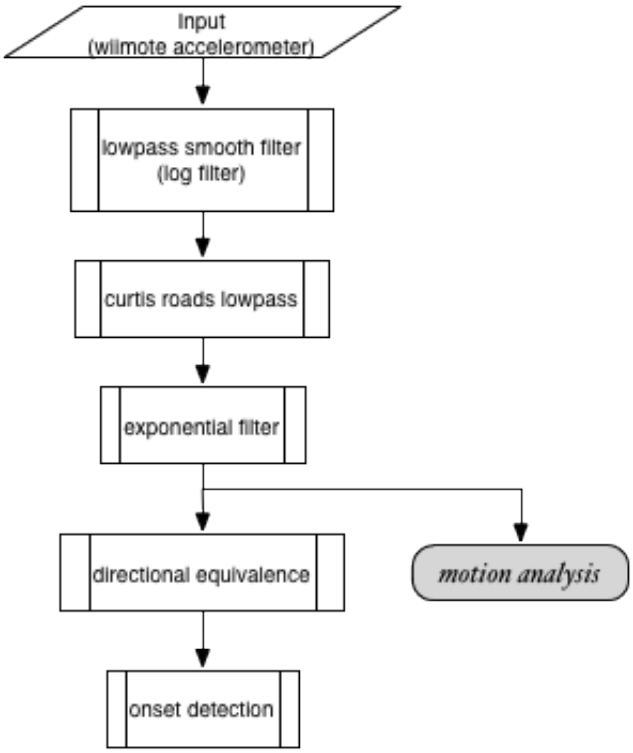


Figura 33 - «get_wii_motion_analysis», a abstração que filtra e deteta «onsets» a partir dos dados do wiimote.

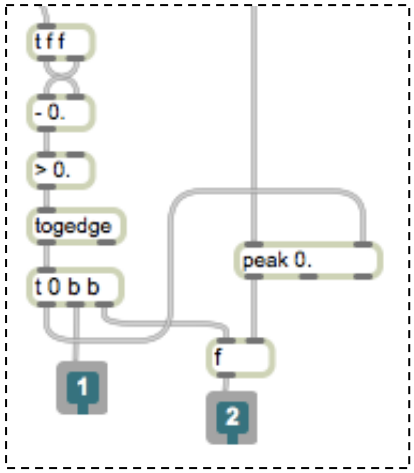


Figura 34 - Patch para deteção de «onsets»

3.3.3. "CONDUCTOR" – CONDUÇÃO DO KIN.RHYTHMICATOR

Este módulo é o responsável pela interação entre a análise da performance do utilizador e o *Kin.Rhythmicator*. É aqui que o utilizador define todos os parâmetros de mapeamento. A conceção deste módulo realizou-se com dois aspetos em mente, o tipo de relação dos mapeamentos, e o papel musical de cada mapeamento.

O estudo de mapeamentos no seio da música, tem sido explorado sobretudo a dois níveis, o da composição, e no design de instrumentos digitais e motores de síntese sonora. Daqui se deduz o papel dos mapeamentos, segundo Hunt et al. (2000):

- Mapeamento pode ser uma característica especifica de uma composição;
- Ou o mapeamento pode ser parte integrante de um instrumento;

A palavra mapeamento, no contexto deste módulo refere-se à ligação ou correspondência entre os parâmetros analisados da performance musical do utilizador e os parâmetros do algoritmo generativo *kin.rhythmicator*, logo tem sobretudo um papel ao nível da composição.

Segundo Hunt et al. (2000) existem duas categorias de mapeamentos:

- Mapeamentos efetuados a partir de mecanismos generativos (ex. redes neuronais, algoritmos genéticos).
- Mapeamentos que utilizam estratégias explicitas de relação.

A diferença entre eles consiste no facto de que a primeira abordagem pode necessitar de treino ou aprendizagem para se adaptar internamente, enquanto que a outra exige que o papel da relação seja mais explícito. A segunda categoria é frequentemente utilizada ao nível do mapeamento de parâmetros de síntese sonora a instrumentos.

Os mapeamentos ao nível do módulo «conductor» assentam na segunda categoria, mapeamentos explícitos, no entanto o utilizador pode definir o papel de cada um desses mapeamentos em termos do espaço de modulação, sobrepondo a relação direta e levando o mapeamento para outro nível de abstração onde poderão surgir resultados não previstos ao combinar diferentes tipos de relações.

Hunt et al. (2000) Identificam dentro dos mapeamentos explícitos diversas estratégias para relacionar parâmetros, frequentemente aplicadas no design de instrumentos e síntese sonora:

- Um para um;
- Um para muitos;
- Muitos para um;
- Muitos para Muitos (combinação entre as relações anteriores);

O módulo [Conductor] ao nível da estratégia de relacionamento de parâmetros, apresenta ao utilizador uma grelha (como uma matriz ou tabela) dentro da qual o utilizador pode ativar mapeamentos entre os parâmetros da análise e os parâmetros do *kin.rhythmicator* (Figura 35).

Figura 35 - Interface do «Conductor» versão para instrumento MIDI.

3.3.3.1. O ACESSO AOS PARÂMETROS DO KIN.RHYTHMICATOR

No topo da Figura 35 estão identificados os parâmetros da análise associados à grelha de mapeamento, «amplitude», «variance», «syncopation», e «density» os quais já foram descritos anteriormente.

À direita da grelha encontram-se identificados os principais parâmetros do *kin.rhythmicator* de controlo da interpretação/geração rítmica. A ativação de um mapeamento faz-se clicando na célula onde se cruza a coluna com a linha dos respetivos parâmetros pretendidos.

Os parâmetros correspondentes às primeiras três linhas «+ variations <> - variations», «syncopated <> non-metrical», e «radius» correspondem ao interface circular presente no «kin.rhythmicator». Em termos de movimento no círculo, o primeiro parâmetro corresponde ao movimento horizontal dentro do circulo, o segundo ao movimento vertical, e o terceiro, por último, define o raio. De certa forma todos estes diferentes parâmetros fazem parte de um só (o círculo e as suas coordenadas polares), no entanto o interface circular do *kin.rhythmicator* já é por si um sistema que permite ao utilizador controlar vários parâmetros em simultâneo com um só gesto.

O parâmetro «density» diz respeito à densidade de eventos, e o parâmetro «dynamic range» corresponde a uma lista de dois valores, um mínimo e um máximo, que corresponde respetivamente ao espaço de amplitude dos eventos gerados.

O parâmetro «morph» não faz parte da versão publicada do *kin.rhythmicator*, esta modificação foi introduzida no âmbito do projeto desta dissertação. A ideia surgiu no sentido de tirar proveito do sequenciador interno do *kin.rhythmicator* para introduzir uma funcionalidade de imitação da performance do utilizador, mas que pudesse ser manipulada também, de forma semelhante à manipulação das probabilidades métricas.

Basicamente, o utilizador envia uma lista com o padrão rítmico para o *kin.rhythmicator*, e a partir do parâmetro «morph» faz a interpolação gradual entre a lista de probabilidades e o padrão rítmico que enviou, e vice-versa. Como exemplo de mapeamento, este parâmetro pode ser usado para criar imitação sempre que a performance tenha poucas variações. Por defeito, o módulo «conductor» atualiza automaticamente o padrão armazenado no *kin.rhythmicator* sempre que a performance é detetada em modo estável. O utilizador pode também alterar esta funcionalidade editando o «patch».

O método utilizado para «morphing» consiste em dar mais prioridade às pulsações rítmicas de baixo nível hierárquico, isto é, aquelas que em geral são consideradas os tempos

fortes do compasso musical e que têm prioridade durante a interpolação dos dois padrões rítmicos. A deteção do nível hierárquico de cada pulsação é uma funcionalidade já utilizada no módulo para medir sincopação, calculada a partir da distribuição de pesos do compasso. A interpolação é baseada numa função exponencial, onde o expoente estabelece o grau de importância de cada nível hierárquico das divisões métricas, sendo por sua vez, atribuído previamente para cada nível hierárquico um valor próprio para o expoente.

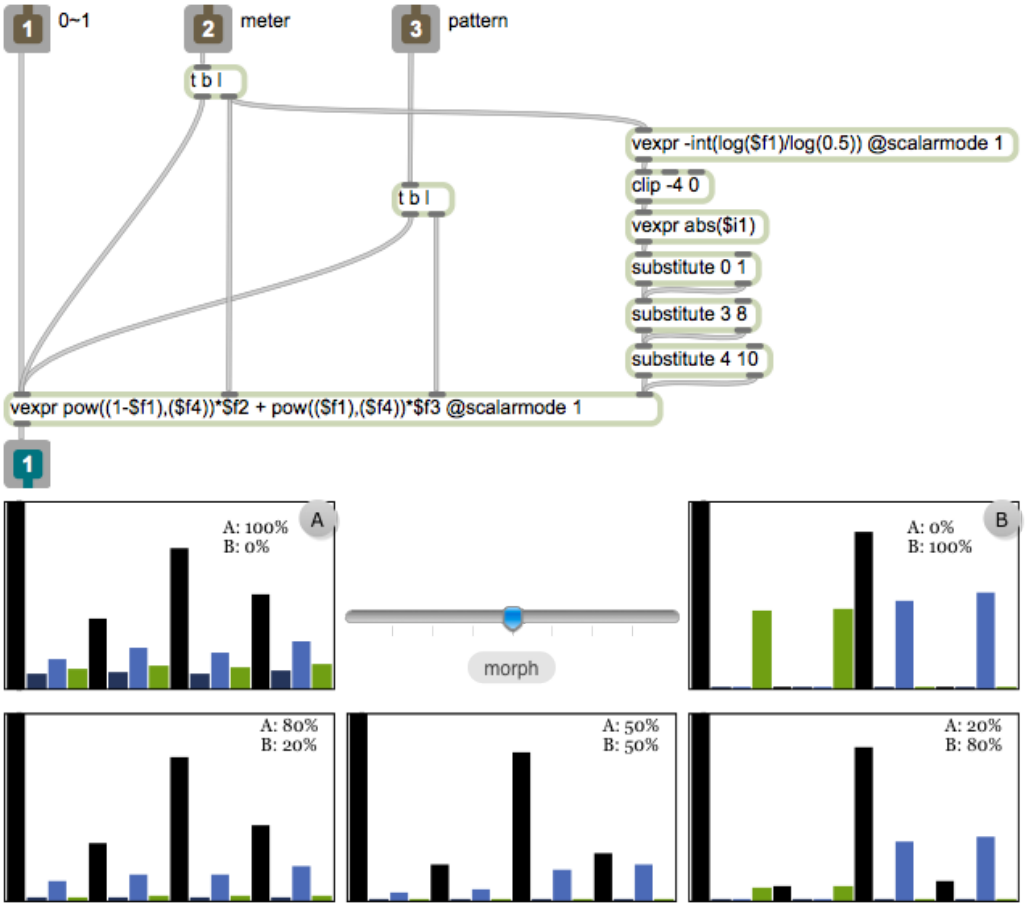


Figura 36 – (Acima) «Patch» que calcula a interpolação entre 2 padrões rítmicos; (Abaixo) Exemplo de interpolação entre o padrão rítmico A e B.

Por fim, o *kin.rhythmicator* possui também as funcionalidades «hold» e «unstable», a primeira permite tocar o último padrão gerado em «loop», que é guardado temporariamente e repetido até que se volte ao modo normal. Durante este período o utilizador pode manipular parâmetros do ritmo, mas estes só serão escutados quando o utilizador desativar a funcionalidade «hold».

A manipulação rítmica em modo «stable» é feita a partir do padrão gerado no ínicio de cada ciclo. Em modo «unstable» o ritmo evolui constantemente para novos padrões. A partir da deteção de modo instável, disponível para cada parâmetro analisado da performance, o utilizador pode manipular estas funcionalidades automaticamente, apenas uma de cada vez.

3.3.3.2. MANIPULAÇÃO DE PARÂMETROS E O PAPEL MUSICAL DE CADA MAPEAMENTO

Quando o utilizador ativa um mapeamento numa célula da grelha, este não está só a criar uma relação entre o parâmetro da análise selecionado e o parâmetro do *kin.rhythmicator*, está também a definir o papel musical dessa relação em termos de modulação, como será explicado mais adiante.

Alguns dos princípios de implementação propostos neste projeto encontram-se de semelhante forma implementados no software de sequenciação *Ableton Live* (Ableton, 2012), no sentido em que a modulação do parâmetro assume um certo papel. Para além da funcionalidade de automatizar parâmetros, este oferece outra funcionalidade dentro dos «clips» MIDI ou áudio que permite modular esses mesmos parâmetros a partir do valor estabelecido. No entanto o tipo de relação é predefinido pelo autor do software ou plugin de acordo com o tipo de parâmetro, e só está acessível para quem programa em «MaxForLive».

A forma como ocorre a manipulação dos parâmetros pelo «conductor» consiste numa combinação entre o estado atual do *kin.rhythmicator* (o estado inicial desses parâmetros que é dado pelo utilizador), e o tipo de relação selecionado no mapeamento estabelecido.

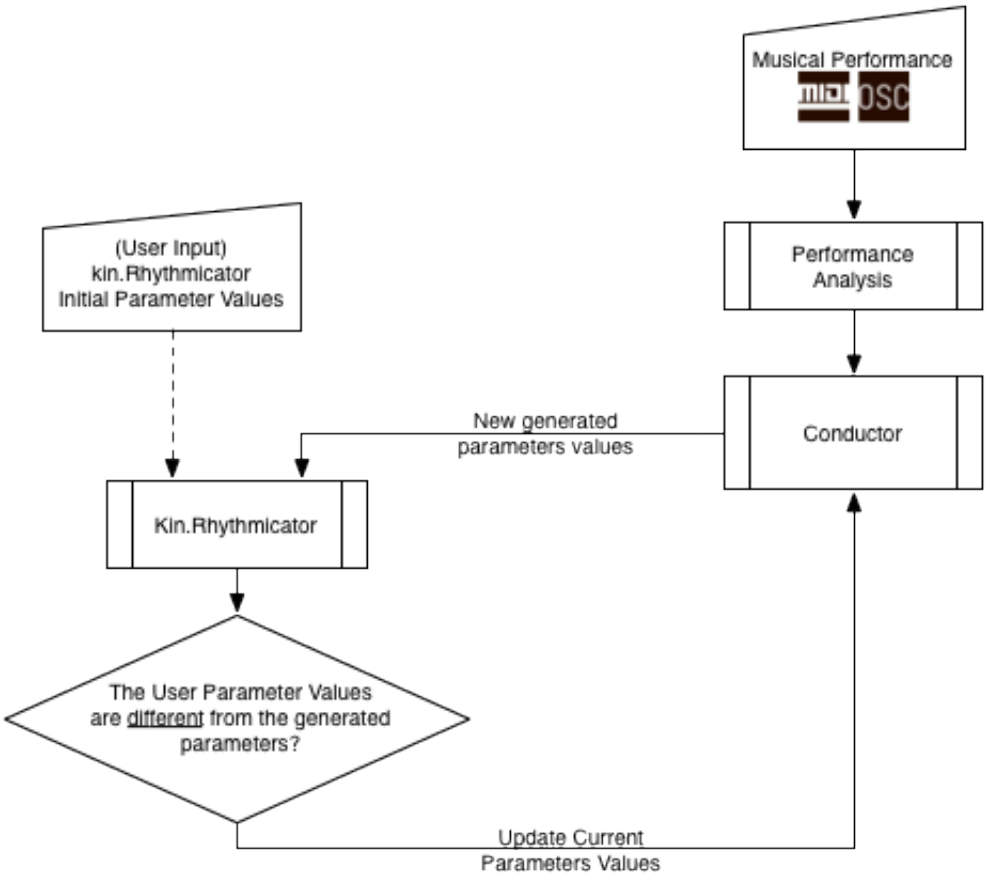


Figura 37 - Diagrama geral do modo de funcionamento do módulo «Conductor»

O que acontece inicialmente é que o [conductor] capta os valores do estado inicial do *kin.rhythmicator* configurado pelo utilizador. Esses valores servirão para o cálculo do espaço de modulação.

O espaço de modulação consiste numa área selecionada automaticamente dentro do âmbito (ou espaço) total do parâmetro. Por exemplo, podemos imaginar o parâmetro densidade que tem um âmbito total de [0~100], e que conforme o valor inicial dado pelo utilizador e o tipo de relação selecionado, obteríamos no cálculo um espaço de modulação de [10~50]. O espaço de modulação representa assim o âmbito selecionado dentro do espaço total do parâmetro para o [conductor] gerar novos valores a partir da análise da performance.

Assim sendo a abstração que manipula (no patch chama-se [modulation mapping slot] ou [mms]) e gera novos valores de parâmetros necessita de conhecer algumas informações à partida para realizar os seus cálculos, nomeadamente, o âmbito (espaço) total e valor (estado) inicial do parâmetro associado ao *kin.rhythmicator*, o valor do âmbito de modulação e também qual o tipo de relação de modulação selecionada. Esses atributos serão descritos mais adiante.

Em termos de relação de modulação (ou de mapeamento), o utilizador tem 6 opções à disposição, cada um destes modos define assim qual o papel musical da relação estabelecida. Os modos (apresentados na Figura 38) são:

Figura 38 - Interface dos 6 modos de modulação

- **X**: não existe nenhuma relação estabelecida;
- **Dir**: o primeiro modo representa o mapeamento direto dos parâmetros da análise com o *kin.rhythmicator*. Aqui o espaço de modulação corresponde ao âmbito total do parâmetro.
- **Uni**: modo «unipolar», o valor do parâmetro é modulado entre o mínimo do âmbito total do parâmetro (definido pelo âmbito de modulação) e o seu valor atual.
- **Bip**: modo «bipolar», o âmbito total de modulação equivale a duas vezes a distância entre o valor atual do parâmetro, e o limite mais próximo desse mesmo parâmetro, definido utilizando o atributo que define o espaço de modulação. Se o valor atual do parâmetro corresponder exatamente ao centro do limite inferior e superior do espaço total do parâmetro, então o espaço de modulação irá corresponder ao espaço total do parâmetro.
- **Add**: modo «additive», o valor atual situa-se ao centro do espaço de modulação. Os limites do espaço de modulação (definido pelo atributo âmbito de modulação) são truncados se ultrapassarem os limites totais do parâmetro.
- **Abs**: modo «absolute», o espaço de modulação corresponde à metade superior ou à metade inferior do espaço total do parâmetro. Se o valor atual se situar na metade inferior do espaço total do parâmetro, o espaço de modulação assume a metade inferior a partir do valor atual subtraindo o âmbito de modulação definido. Se o valor se situar na metade superior do espaço total do parâmetro, o espaço de modulação assume a metade superior a partir do valor atual somando o âmbito de modulação definido pelo utilizador.
- **Gau**: modo «gaussian», consiste na exploração da aleatoriedade no seio da relação de mapeamento. Este modo explora uma distribuição *gaussiana* como probabilidades para

gerar valores aleatórios para o parâmetro, no entanto, alguns atributos dessa distribuição são definidos por parâmetros da análise. O pico da curva corresponde ao valor atual enquanto que o desvio do pico é definido pelos valores recebidos da análise. A curva de probabilidades é gerada em tempo-real sempre que é recebido um novo valor da análise, e em seguida é gerado um novo valor para o parâmetro.

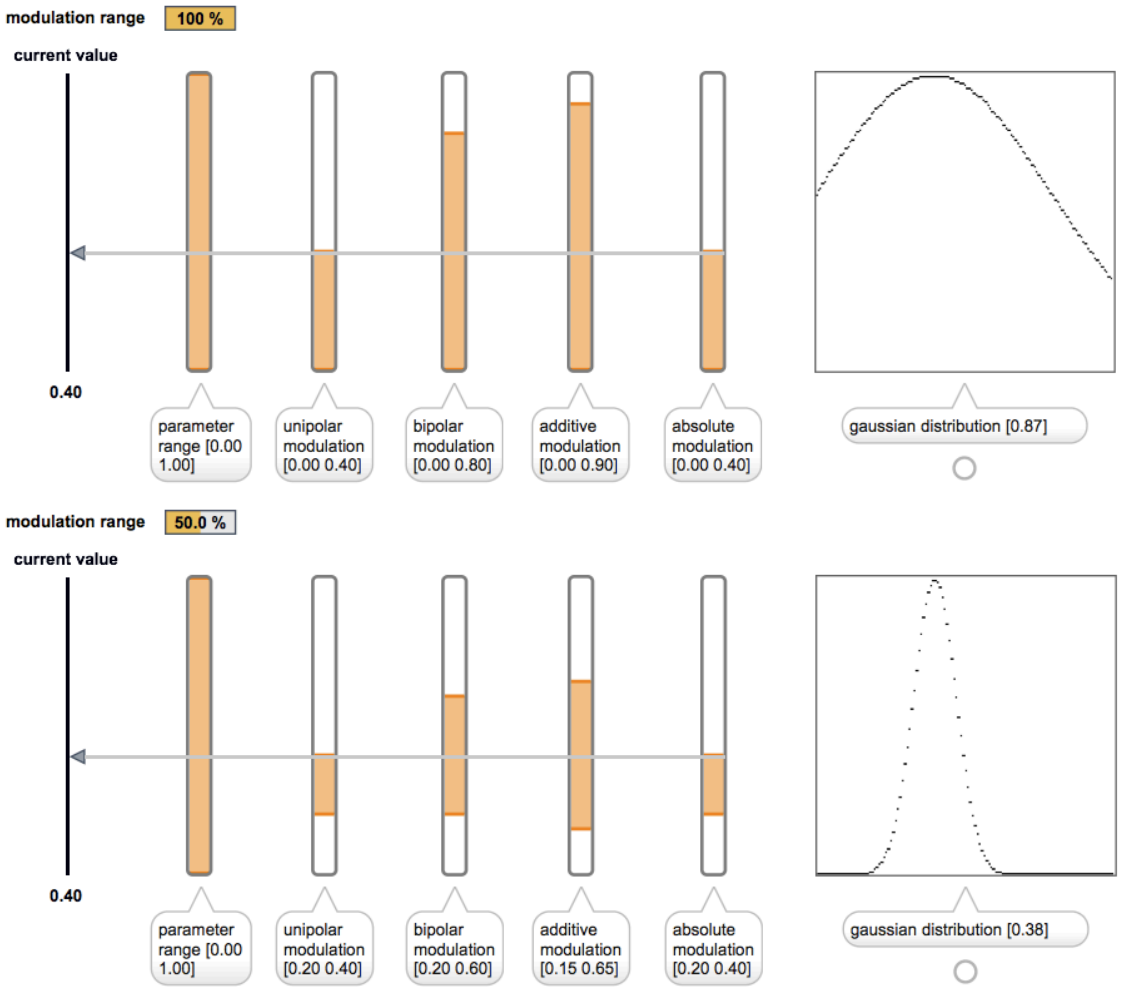


Figura 39 - Exemplo dos diferentes modos de modulação, acima com o âmbito de modulação definido a 100% e em baixo com o mesmo definido a 50%.

Na Figura 39 observa-se um exemplo de cada tipo de relação que se pode estabelecer e qual o papel musical que esta pode ter em termos de variação dos valores dos parâmetros. Aqui o valor atual do parâmetro («current value») tem um papel crucial, este é definido à partida pelo utilizador, representando de certa forma o ponto de partida na performance rítmica do *kin.rhythmicator*. É também a partir deste valor que se estabelece o tipo de relação do mapeamento.

3.3.3.3. Interface de Utilizador e Outros Módulos

Por fim, falta efetuar uma descrição geral do interface de utilizador dos módulos desenvolvidos. As funcionalidades mais importantes já foram descritas nos módulos anteriores, nesta secção irá apresentar-se as restantes. Uma característica comum a todos os módulos em termos de ajuda ao utilizador, é que se este repousar o ponteiro do rato sobre um qualquer objeto do interface, surgirá uma janela com uma descrição da funcionalidade.

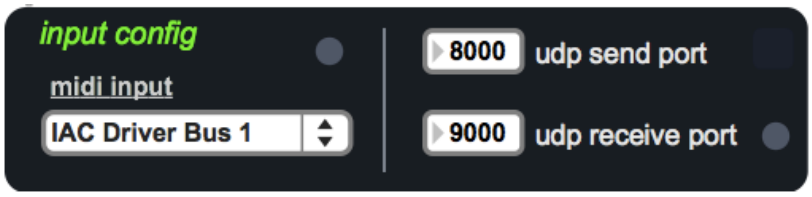


Figura 40 - Interface do módulo de configuração do instrumento MIDI e gestual

O interface representado na Figura 40 corresponde ao módulo responsável por configurar os instrumentos que irão enviar os dados da performance para o respetivo módulo de análise. Este módulo consiste num dois em um, no sentido em que permite configurar a porta para um instrumento MIDI e para o instrumento gestual como o *wiimote*. Os controlos gestuais são interligados através do software *Osculator*, e os seus dados são enviados através do protocolo UDP[25].

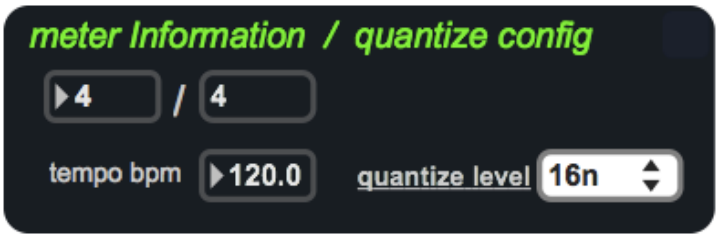


Figura 41 - Interface do módulo de configuração da resolução de quantização.

[25] User Datagram Protocol

A Figura 41 representa o módulo que permite configurar a resolução de quantização. Este é comum a todas as instâncias do módulo de análise, isto é, comunica com todas em geral, sendo que para tal basta instanciar apenas uma vez o [bpatcher].

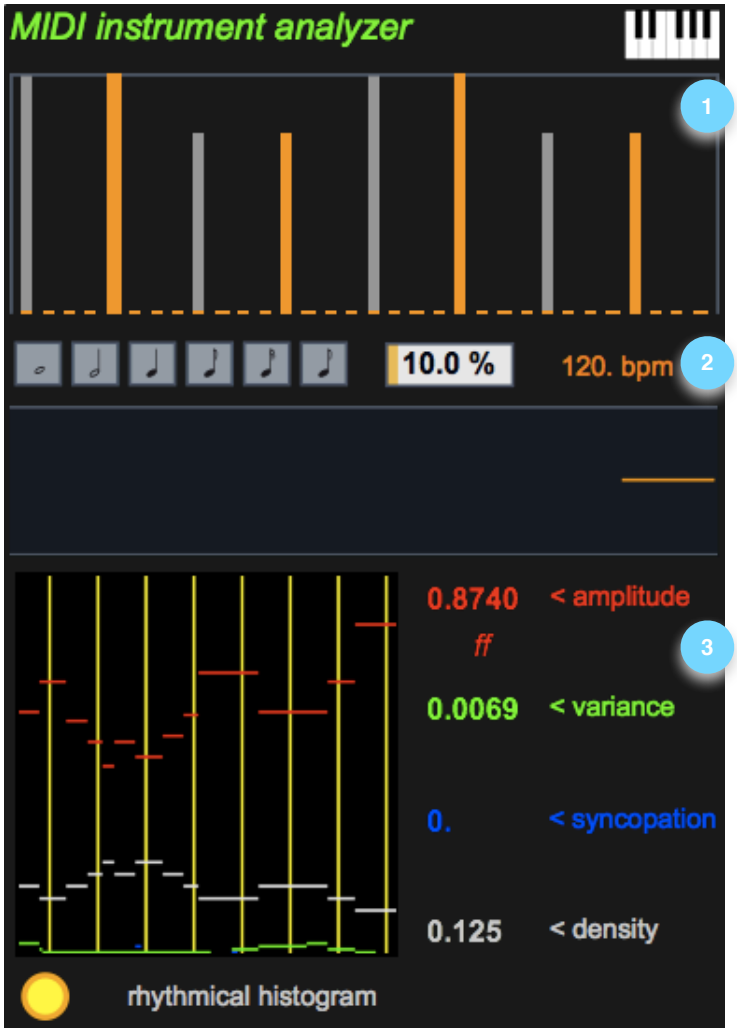


Figura 42 - Interface do módulo de análise.

Existem duas versões do módulo de análise, uma dedicada a instrumentos MIDI e outra dedicada a controladores gestuais, mas em termos de interface são muito semelhantes. Esta divide-se em várias secções conforme numeradas na Figura 42.

1. Aqui apresentado o padrão da performance rítmica após a quantização. Os eventos a cor de laranja representam a performance em tempo-real e os eventos a cinzento representam o ultimo padrão rítmico executado.

2. Esta secção é dedicada ao módulo [tempo follower], que deteta os desvios temporais no andamento da performance. Esta mostra qual a última figura rítmica detetada na

performance, contém um [slider] que permite configurar qual o desvio máximo permitido e mostra qual o andamento atual da performance. Abaixo aparece um registo histórico da evolução do andamento ao longo do tempo.

3. Nesta secção é apresentado ao utilizador os resultados dos parâmetros analisados da performance, informando qual o valor atual e apresentando também um registo da evolução deste ao longo da performance. O botão no canto inferior esquerdo informa qual o estado da deteção de ócio.

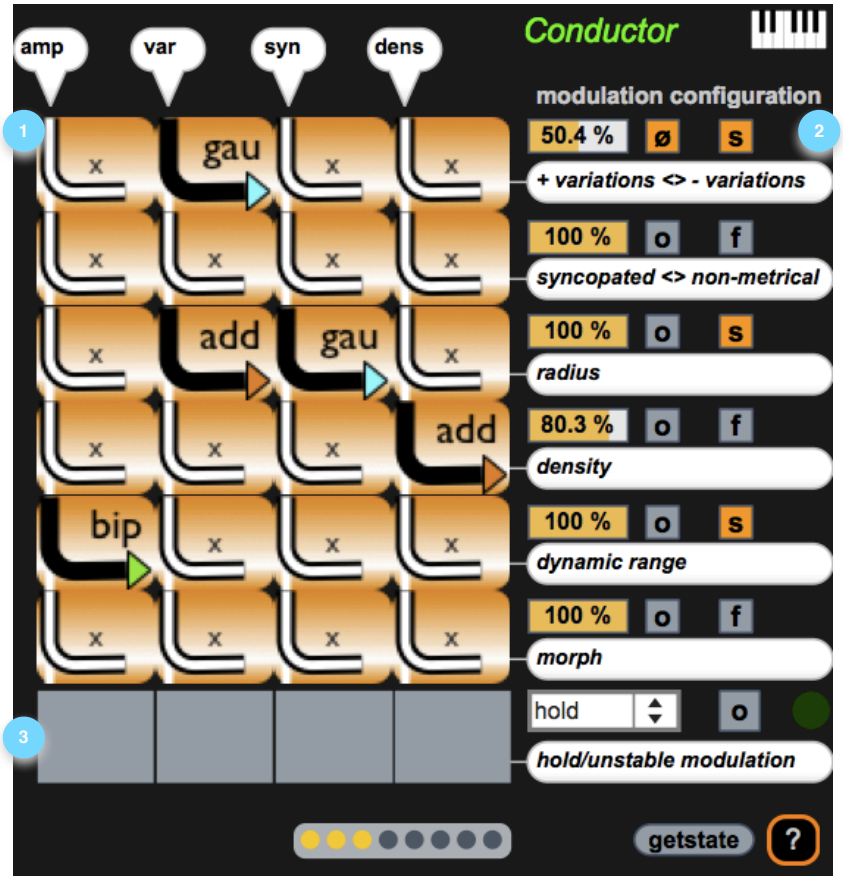


Figura 43 - Interface do módulo «Conductor»

A Figura 43 representa o interface de utilizador do módulo [conductor]. Tal como acontece com o módulo de análise, também existem duas versões, uma dedicada a instrumentos MIDI e outra dedicada a controladores gestuais. Este interface está também legendado por secções:

1. Corresponde à grelha onde o utilizador ativa os mapeamentos e define qual o seu papel, isto é, que tipo de modulação irá exercer.

2. Nesta secção aparecem identificados os parâmetros disponíveis do *kin.rhythmicator*, e alguns controlos da interação, nomeadamente:

    a. [slider] para definir o espaço de modulação máximo;

    b. botão de fase, que este inverte os sinais da análise, logo invertendo também o tipo de resposta gerado;

    c. botão para definir a frequência da geração de parâmetros, em que «f» corresponde a resposta rápida e «s» corresponde a uma resposta lenta. No modo resposta rápida é gerado um novo parâmetro sempre que é recebido uma mensagem da análise, e no modo resposta lenta os parâmetros são gerados no inicio de cada compasso.

3. Esta secção contém junto à grelha os botões que ativam a modulação da função «hold» ou «unstable» do *kin.rhythmicator*. Além destes existem também os botões para gerir «presets», que armazenam um «preset» pressionando a tecla «shift» e clicando em simultâneo; e que chamam o «preset» apenas ao clicar. O botão «getstate» serve apenas para o [conductor] ler o estado atual dos parâmetros do *kin.rhythmicator*.

CAPÍTULO 4

4.1. CONCLUSÃO E TRABALHO FUTURO

Há uma grande comunidade de músicos e compositores que procura cada vez mais introduzir nos seus trabalhos elementos que permitam alargar a criatividade, originalidade e interatividade das suas obras.

Verifica-se também que cada mais compositores e músicos desenvolvem competências de programação informática para permitir a introdução de novos elementos nas suas obras, indicando que talvez esta seja, provavelmente, uma competência do futuro, da mesma forma que hoje em dia é essencial saber ler e escrever.

Com a propagação e acessibilidade de ferramentas úteis no design de sistemas interativos torna-se também necessário desenvolver relações entre estas. Os músicos e compositores têm explorado nas suas obras uma grande diversidade de tecnologias, técnicas de manipulação e síntese sonora, instrumentos digitais, e algoritmos generativos para o desenvolvimento de sistemas musicais interativos. É crucial que ao combinar diversas tecnologias e meios se desenvolva também as relações entre estes, o que por sua vez resultará na narrativa que mantém o interesse e expectativa de todos os intervenientes, músico, compositor, artista, público, etc.

4.1.1. RESUMO DO TRABALHO

O projeto desta dissertação surgiu com várias premissas em mente, a introdução da performance na própria performance para o controlo de algoritmos generativos, a utilização da performance como meio de reduzir a necessidade do utilizador manipular parâmetros durante a mesma, e o desejo de desenvolver relações mais orgânicas e humanas no desenvolvimento de sistemas musicais interativos e sobretudo na manipulação dos mesmos em tempo-real.

Com isto em mente, considera-se que uma componente importantíssima para a qualidade da interação entre músicos é a capacidade que estes têm de ouvir e de entender o que está a acontecer, para que a própria capacidade de resposta seja de igual qualidade. Não é por acaso que os músicos e compositores passam uma vasta quantidade do tempo durante os seus estudos a desenvolver estas capacidades.

Daí, o projeto desta dissertação intitulado «kinductor», se basear no desenvolvimento de dois módulos, que interligados entre si se complementam para permitir interação com o *kin.rhythmicator* a partir da performance musical ou gestual. Desenvolveu-se um módulo de "ouvinte" e outro de "resposta", em que um analisa a performance musical em tempo-real e o outro a partir daquilo que foi percecionado cria uma resposta.

Existem duas versões do módulo de análise, uma dedicada a instrumentos MIDI e outra dedicada a controladores gestuais como o *wiimote*.

Na análise procura-se extrair informação da performance de forma semelhante à análise abstrata que a perceção humana efetua da música. Isto significa que em vez de se partir para uma análise do ponto de vista musical, que necessita e manipula o conhecimento disponível de teoria musical, se utilizam certos atributos gerais e abstratos que permitem descrever a performance musical, nomeadamente, densidade de eventos, quantidade de variações, sincopação e amplitude geral dos eventos.

Posteriormente, estes atributos são encaminhados para outro processo de análise, o qual tenta extrair informação relativa à memória que o sistema tem da performance, ou seja, procura observar o que sucedeu no passado em comparação com o que sucede no presente e inferir outro nível superior de informação relativamente à performance. Esta análise sucede a dois níveis de memória, intitulados a curto prazo e a longo prazo, que permite medir respetivamente a divergência dos parâmetros da performance atual com o que sucedeu até 8 pulsações atrás ou com o que sucedeu até 8 compassos atrás, permitindo distinguir por exemplo variações a curto prazo e grandes transições musicais a longo prazo.

O módulo de análise complementa-se ligado ao módulo interação, o [conductor]. Este módulo é responsável por mapear os dados da análise com os parâmetros do *kin.rhytmicator*. Para tal o utilizador define os parâmetros iniciais deste e também quais e que tipo de relações serão estabelecidas entre a análise e o *kin.rhythmicator*. O utilizador pode decidir qual a estratégia de mapeamento a utilizar, um para um, um para muitos, muitos para um, ou muitos para muitos. E dispõe ainda de 6 modos de relacionar os parâmetros, de forma a atribuir a cada mapeamento um certo papel ou comportamento musical.

4.1.2. RESUMO DAS CONTRIBUIÇÕES E LIMITAÇÕES

Um objetivo que se considera atingido consistiu em desenvolver este projeto sem depender de objetos externos ao Max, excetuando aqueles que fazem parte do próprio objeto de estudo, a interação com o *kin.rhythmicator*.

Todos os algoritmos e abstrações de análise e manipulação/geração foram desenvolvidos com base em objetos nativos do Max. Isto torna possível o prolongamento e desenvolvimento deste projeto para o futuro, facilitando o acompanhamento da evolução do próprio MaxMSP e dos sistemas operativos. Ao desenvolver estes processos de raiz, também permitiu tomar conhecimento de conceitos e técnicas pouco familiares, como deteção de «onsets», gestão de listas, análise temporal, extrapolação, processos de modulação, quantização, medição de densidade, sincopação, amplitudes e variações, etc.

Os testes efetuados com o software desenvolvido, foram realizados sobretudo durante a fase de desenvolvimento. A avaliação da robustez dos diversos algoritmos, e a afinação de certos parâmetros foi também efetuada pelo próprio autor, algumas vezes de forma empírica e por vezes também recorrendo a sugestões e debates externos.

Um dos objetivos propostos, foi desenvolver uma versão do software codificado em MaxForLive, que permitisse a divulgação pela vasta comunidade de utilizadores do *Ableton Live*, mas que não foi finalizado a tempo desta apresentação, ficando assim em falta também submeter este software ao ambiente exigente da composição ou improvisação.

Tendo em conta que o código fonte do software será disponibilizado na internet[26], considera-se que este oferece um contributo positivo para aqueles que desenvolvem sistemas musicais interativos em MaxMSP. Pela pesquisa efetuada nos *forums* e *sites* relacionados com

[26] http://code.google.com/p/kinductor/

o MaxMSP não foi encontrado nada semelhante e acessível de forma gratuita aos utilizadores desta linguagem. Este projeto oferece uma grande variedade de «patches» que podem ser reutilizados pelos utilizadores nos seus projetos. Para além disso o utilizador também pode adaptar livremente estes módulos a outros algoritmos musicais em vez do *kin.rhythmicator*. O projeto incidiu sobre o estudo da condução de algoritmos musicais, mas este pode também ser aplicado na manipulação de efeitos ou de algoritmos de síntese sonora.

O *kin.rhythmicator* é um gerador de ritmo que por si só acrescenta um grau de desafio extra. Este é na sua natureza um gerador de ritmo estocástico, logo daí depreende-se que mesmo especificando os parâmetros exatos do ritmo desejado, é bastante possível que este surpreenda o utilizador, pois o que corre no seu sangue são um conjunto de probabilidades genéricas relativas ao compasso musical, descartando conhecimento do estilo musical ou do que o utilizador acabou de executar. Desta forma, uma das sugestões seria elaborar uma versão do *kin.rhythmicator* integrada com os algoritmos deste projeto que pudesse internamente e de forma sofisticada manipular informação recolhida do utilizador (a sua própria performance por exemplo) ou informação relacionada com estilo musical.

Por fim, apesar de na data em que esta dissertação foi redigida o *Kinductor* ser apenas um protótipo observou-se que o caráter multidimensional de mapeamentos permite relações mais musicais entre parâmetros, isto é, o utilizador pode prever o comportamento do mapeamento e ser surpreendido em simultâneo, graças à multidimensionalidade que mapeamentos que permite combinar vários parâmetros da análise com um parâmetros de controlo. Por outro lado, a descrição abstrata da performance rítmica, isto é, utilizando termos semânticos como variações, amplitude, densidade e sincopação revela-se útil neste tipo de mapeamentos a software generativo como o *kin.Rhythmicator*.

4.1.3. TRABALHO FUTURO

Na secções anteriores já foram dadas algumas sugestões, no entanto decidiu-se resumir todas as ideias numa secção apenas.

O desenvolvimento de sistemas interativos tem muito a ganhar com o cruzamento de variadas áreas do conhecimento e técnicas, esta é também a base do desenvolvimento de HCI. Tendo em conta que cada técnica tem os seus pontos fortes e fracos, ao combinar diferentes métodos cria-se um sistema mais robusto e versátil.

Neste momento o projeto *Kinductor* precisa de continuar a ser avaliado e testado de forma exaustiva. Verificou-se também que o desenvolvimento de sistemas interativos é processo

contínuo de evolução, onde é necessário manter todas as componentes atualizadas, testar constantemente o software, procurar «bugs», e procurar melhorar sempre os algoritmos. Durante o desenvolvimento do *kinductor* houve muitas notas, perguntas e ideias que ficaram no papel, passando a listar:

- Codificar o projeto «kinductor» em *MaxForLive* para este poder ser utilizado dentro do sequenciador *Live*;
- Desenvolver uma versão híbrida do *kin.rhythmicator* com o *kinductor* embutido;
- Como implementar um algoritmo genético interativo ou uma rede neuronal no *kinductor* para gerar parâmetros?
    - Este tomaria como código genético a análise da performance?
- Como melhorar a extrapolação de parâmetros?
- Que tipos de análise de memória se podem efetuar?
- Que outros parâmetros se podem analisar da performance? Estes têm de depender do tempo musical e da quantização?
- Melhorar a análise gestual e a deteção de «onsets»;

BIBLIOGRAFIA

Ableton. (2012). *Ableton*. Obtido em Jun de 2012, de Ableton Live: www.ableton.com

Biles A., J. (2003). GenJam in Perspective: A Tentative Taxonomy for GA Music and Art Systems. *Leonardo Music Journal , 36* (1), 43-45.

Biles A., J. (1994). GenJam: A genetic algorithm for generating jazz solos. In Citeseer (Ed.), *Proceedings of the International Computer Music Conference*, (p. 131). Aarhus, Denmark.

Bongers, B. (2000). Physical interfaces in the electronic arts. Interaction theory and interfacing techniques for real-time performance. In M. Wanderley, & M. Batiters (Ed.), *Trends in Gestural Control of Music* (p. 30). Cambridge, Paris: IRCAM.

Chadabe, J. (1 de 12 de 2001). *A Statement...* Obtido em 18 de Jan de 2012, de Joel Chadabe: http://www.chadabe.com/statement.html

Collins, N. M. (2006). Towards Autonomous Agents for Live Computer Music : Realtime Machine Listening and Interactive Music Systems. *Centre for Music and Science - Faculty of Music* (p. 245). Cambridge: University of Cambridge.

Cook, P. R. (2001). *Music Cognition and Computarized Sound.* Cambridge, Massachusetts: MIT Press.

Cycling74. (1 de 03 de 2012). *Cycling74*. Obtido em 1 de 03 de 2012, de Cycling74: http://cycling74.com/whatismax/

Cycling74. (1 de 03 de 2012). *Max6 Documentation.* Obtido em 26 de 03 de 2012, de Cycling74: http://www.cycling74.com/docs/max6/dynamic/c74_docs.html

Hunt, A., Wanderley, M. M., & Kirk, R. (2000). Towards a model for instrumental mapping in expert musical interaction. In U. o. Library (Ed.), *2000 International Computer Music.* Michigan: MPublishing.

IRCAM - imtr. (22 de Ago de 2011). *Antescofo*. Obtido em 1 de Sep de 2011, de imtr (Real-Time Musical Interactions): http://repmus.ircam.fr/antescofo

IRCAM - imtr. (15 de Fev de 2011). *Score Following History*. Obtido em 20 de Ago de 2011, de imtr (Real-Time Musical Interactions): http://imtr.ircam.fr/imtr/Score_Following_History

IRCAM - imtr. (16 de Mar de 2010). *Synchronous Programming*. Obtido em 1 de Ago de 2011, de imtr (Real-Time Musical Interactions): http://imtr.ircam.fr/imtr/Synchronous_Programming

IRCAM. (2011). *Who are we?* Obtido em 1 de Ago de 2011, de IRCAM: http://www.ircam.fr/ircam.html?&L=1

Mcgilvray, D. (2008). On The Analysis of Musical Performance by Computer. *Department of Electronics & Electrical Engineering* (p. 176). Glasgow: University of Glasgow.

NYU Center for Advanced Technology in Multimedia. (s.d.). *Technologies and Projects - Projects - Cypher*. Obtido em 07 de 07 de 2012, de NYU Center for Advanced Technology (CAT): http://cat.nyu.edu/current/technologies/cypher_txt.html

Pachet, F. (2004). Beyond the cybernetic jam fantasy: The continuator. *Computer Graphics and Applications, IEEE , 24* (1), 31-5.

Pachet, F. (2002). The Continuator: Musical Interaction With Style. *Journal of New Music Research , 31* (1), 333-341.

Priberam. (1 de Setembro de 2011). *Dicionário Priberam da Língua Portuguesa*. Obtido em 1 de Setembro de 2011, de Priberam: http://www.priberam.pt/dlpo/dlpo.aspx

Ribeiro, N. (2007). *Multimédia e Tecnologias Interactivas* (3ª edição ed.). Lisboa: FCA.

Roads, C. (1996). *The Computer Music Tutorial.* Massachusetts: The MIT Press.

Rowe, R. (1992). Machine Listening and Composing with Cypher. In M. Press (Ed.), *Computer Music Journal . 16-1*, pp. 43-63. Spring: MIT Press.

Rowe, R. (2001). *Machine Musicianship* (Vol. I). Cambridge: The MIT Press.

Schlöomer, T., Poppinga, B., Henze, N., & Boll, S. (2008). Gesture recognition with a Wii controller. *2nd International Conference on Tangible and Embedded Interaction.* Bonn, Germany.

Schutterhoef, A. v. (13 de Nov de 2007). *ISPW*. Obtido em 1 de Ago de 2011, de wikiC/2.06 Utrecht School of the Arts: http://knorretje.hku.nl/wiki/ISPW

Schutterhoef, A. v. (13 de Nov de 2007). *Sogitec 4X*. Obtido em 1 de Ago de 2011, de wikiC/2.06 Utrecht School of the Arts: http://knorretje.hku.nl/wiki/Sogitec_4X

Sioros, G., & Guedes, C. (2011). Automatic Rhythmic Performance in Max/MSP: the kin. rhythmicator. In NIME (Ed.), *NIME - New Interfaces for Musical Expression* (pp. 88-91). Oslo: NIME.

Sioros, G., & Guedes, C. (2011). COMPLEXITY DRIVEN RECOMBINATION OF MIDI LOOPS. *12th International Society for Music Information Retrieval Conference (ISMIR 2011)* (pp. 381-386). ISMIR.

Weinberg, G., & et al. (2012). *Robotic Musicianship Group - Projects - Haile*. (G. Weinberg, Produtor, & Georgia Tech) Obtido em 1 de Jul de 2012, de Georgia Tech - Center for Music Technology: http://www.gtcmt.gatech.edu/research-projects/haile

Wetzel, D. B. (2004). Overview of Interactive Electroacoustic Music Performance. In D. B. Wetzel, *Analysis, Reconstruction, and Performance of Interactive Electroacoustic Music* (pp. 22-45). Arizona.

Wikipedia. (19 de 03 de 2012). *Ableton Live*. Obtido em 22 de 03 de 2012, de Wikipedia: http://en.wikipedia.org/wiki/Ableton_Live

Wikipédia. (19 de Agosto de 2011). *Human Computer Interaction*. Obtido em 13 de Julho de 2011, de Wikipédia: http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction

Wikipédia. (23 de Ago de 2011). *IRCAM*. Obtido em 01 de Set de 2011, de Wikipédia: http://en.wikipedia.org/wiki/IRCAM

Wikipédia. (22 de Jun de 2011). *ISPW*. Obtido em 1 de Ago de 2011, de Wikipédia: http://en.wikipedia.org/wiki/ISPW

Wikipedia. (16 de 03 de 2012). *Max (software)*. Obtido em 20 de 03 de 2012, de Wikipedia: http://en.wikipedia.org/wiki/Max_(software)

Wikipedia. (22 de 03 de 2012). *MIDI*. Obtido em 22 de 03 de 2012, de Wikipedia: http://en.wikipedia.org/wiki/MIDI

Wikipedia. (10 de 03 de 2012). *Open Sound Control*. Obtido em 22 de 03 de 2012, de Open Sound Control: http://en.wikipedia.org/wiki/Open_Sound_Control

Wikipédia. (28 de Jan de 2011). *Score Following*. Obtido em 01 de 08 de 2011, de Wikipédia: http://en.wikipedia.org/wiki/Score_following

Wikipédia. (17 de Fev de 2011). *Sogitec 4X*. Obtido em 1 de Ago de 2011, de Wikipédia: http://en.wikipedia.org/wiki/Sogitec_4X

Wikipedia. (15 de 03 de 2012). *Wii Remote*. Obtido em 22 de 03 de 2012, de Wikipedia: http://en.wikipedia.org/wiki/Wii_Remote

Winkler, T. (2001). *Composing Interactive Music: Techniques and Ideas Using Max.* Massachusetts: The MIT Press.

Página em branco