UNIVERSIDADE FEDERAL DE MINAS GERAIS

Escola de Música

Programa de Pós-Graduação em Música

Fellipe Miranda Martins

Estudo exploratório de processos de transformação sonora a partir de Trevor Wishart

reinvenção e tradução para o ambiente SuperCollider

Belo Horizonte

2020

Fellipe Miranda Martins

Estudo exploratório de processos de transformação sonora a partir de Trevor Wishart

reinvenção e tradução para o ambiente SuperCollider

Versão final

Dissertação apresentada ao Programa de Pós-Graduação em Música da Escola de Música da Universidade Federal de Minas Gerais como requisito parcial à obtenção do titulo de Mestre em Música.

Linha de Pesquisa: Sonologia.

Orientador: José Henrique Padovani

Belo Horizonte

2020

M386e Martins, Fellipe Miranda.

Estudo exploratório de processos de transformação sonora a partir de Trevor Wishart [manuscrito]: reinvenção e tradução para o ambiente SuperCollider / Fellipe Miranda Martins. - 2020.
234 f., enc.; il. + 1 CD

Orientador: José Henrique Padovani.

Linha de pesquisa: Sonologia.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Música.

Inclui bibliografia.

1. Música - Teses. 2. Música e tecnologia. 3. Som - Registro e produção - Técnicas digitais. 4. Ondas sonoras. I. Padovani, José Henrique. II. Universidade Federal de Minas Gerais. Escola de Música. III. Título.

CDD: 789.96

Ficha elaborada pela Bibliotecária: Aline Pimenta Almeida CRB-6/3416

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE MÚSICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MÚSICA

FOLHA DE APROVAÇÃO

Dissertação defendida pelo mestrando **Fellipe Miranda Martins**, em 24 de setembro de 2020, e aprovada pela Banca Examinadora constituída pelos Professores:

_____

Prof. Dr. José Henrique Padovani Velloso
Universidade Federal de Minas Gerais
(orientador)

_____

Prof. Dr. Jônatas Manzolli
Universidade Estadual de Campinas

_____

Prof. Dr. Sérgio Freire Garcia
Universidade Federal de Minas Gerais

_____

Prof. Dr. Rogério Vasconcelos Barbosa
Universidade Federal de Minas Gerais

Documento assinado eletronicamente por **Jose Henrique Padovani Velloso**, **Professor do Magistério Superior**, em 24/09/2020, às 17:16, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rogerio Vasconcelos Barbosa**, **Membro**, em 24/09/2020, às 22:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Jônatas Manzolli**, **Usuário Externo**, em 25/09/2020, às 09:13, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Sergio Freire Garcia**, **Professor do Magistério Superior**, em 25/09/2020, às 09:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0272277** e o código CRC **CD0791BA**.

Referência: Processo nº 23072.225523/2020-71 SEI nº 0272277

Aos meus pais, Najla e José Hermano,
por todo carinho, afeto, amparo, incentivo, acolhimento
e por, de forma despretensiosa e sábia,
guiarem a criança curiosa a se tornar o adulto estudioso.

Agradecimentos

Qualquer texto seria pequeno para expressar minha profunda gratidão por José Padovani, seja pela sabedoria, zelo e imensa disponibilidade na orientação deste trabalho, pela força e incentivo nos meus pontos de insegurança, pelo comprometimento ético inabalável ou pelos impulsionamentos a sempre fazer o melhor. Meu sincero e eterno agradecimento por me ensinar a trilhar meu próprio caminho e não me desviar para o rumo que a outros pertence.

Todo este trabalho foi feito utilizado exclusivamente softwares livres e colaborativos, a saber, SuperCollider, Python e Spyder, LaTeX , abnTeX2 (além de dezenas de bibliotecas de terceiros) e TexMaker, Zotero, CDP e Linux, além de vários outros que foram utilizados indiretamente como o Pd, Sonic Visualizer e Praat. Além disso, inúmeras redes de compartilhamento de informações (listas de email, fóruns, sites pessoais, etc) foram utilizadas para solucionar dúvidas diariamente. Este trabalho só foi possível devido ao árduo esforço coletivo de inúmeros de programadores, desenvolvedores, usuários e mantenedores destes softwares livres e colaborativos. Meu agradecimento a todas essas pessoas, que me fazem ter a certeza que um mundo melhor e justo é possível. Em especial, ficam meus agradecimentos à generosa e incessantemente disponível comunidade do SuperCollider.

Agradeço a Trevor Wishart e James McCartney por tornarem o CDP e o SuperCollider softwares abertos.

Agradeço a Bárbara Ramalho, Davi Mota, Thiago Campolina e Rodrigo Borges, pois onde houve dúvida e insegurança sobre o caminho a seguir, eu encontrei a amizade, o exemplo e as orientações dedicadas.

Agradeço imensamente a Sérgio Freire, Maurício Loureiro e Hani Yehia por terem iniciado e consolidado a pesquisa acadêmica sobre o som e áudio digital, na UFMG e também no Brasil. Obrigado por poder aprender tanto sobre um tema que tanto me fascina. O esforço de vocês alicerça todo este trabalho.

Agradeço ao programa Ciência sem Fronteiras pela inacreditável experiência que transformou (chaveou) minha vida das mais variadas maneiras, academicamente, existencialmente, tecnicamente, etc. A pesquisa que resultou neste trabalho é fruto dessa oportunidade incrível.

A Tia Rosa e Tia Hessem, um enorme e caloroso agradecimento, por além de todo o suporte à minha trajetória de estudos, terem entregado diariamente o cuidado e carinho de verdadeiras mães.

Um enorme agradecimento a minha enorme família, gratidão impossível de ser

plenamente contemplada aqui. Obrigado Olavo e Maria Tereza por todo carinho e afeto e também por frequentemente terem sido cobaias e alvos de todos meus experimentos na infância.

Aos amigos e amigas, associados, eletricistas ou eletrizados, companheiros de bandas, de terras distantes, de longa ou curta data, a minha gratidão profunda por fazerem do mundo um lugar de afeto.

Um forte agradecimento a todo o povo de Araçuaí, que me ensinou a importância de uma vida alegre mesmo na adversidade e o valor de se desconhecer o anonimato.

Caroline Ferreira Rosa, se só o seu carinho me fosse dado por apenas um instante, toda minha vida já estaria completa. Como posso agradecer por ainda contar com suas revisões, conselhos e orientações diárias?

"O computador chegou ao estágio
em que resolve todos os problemas.
Exceto, claro, os dele."
(Millôr, A Bíblia do Caos)

Resumo

A prática criativa de realizar transformações em sons gravados tem origem nos trabalhos de Pierre Schaeffer e do *Groupe de Recherches Musicales* (GRM), contudo tal prática adquire uma nova dimensão a partir do desenvolvimento da suíte de programas computacionais Composers Desktop Project (CDP) – encabeçada pelo compositor, performer e desenvolvedor inglês Trevor Wishart. Apesar dos programas do CDP serem elaborados, a princípio, para execução em tempo diferido, o ambiente de programação de áudio SuperCollider – que é especialmente centrado na lógica do tempo real – apresenta flexibilidade para portá-los assim como permite e facilita a criação de novos processos correlatos. Dois tipos principais de transformações sonoras são analisados, aquelas nas quais os cálculos computacionais são realizados no domínio do tempo – utilizando a ideia de *conjuntos de onda* (*wavesets*) – e aquelas cujos cálculos se dão no domínio da frequência – valendo-se dos dados fornecidos pelo vocoder de fase. Neste trabalho, as transformações temporais e espectrais do CDP são revistas à luz das novas possibilidades trazidas pelo SuperCollider e avaliadas por meio da elucidação do mecanismo de transformação computacional e da escuta dos sons fonte e resultantes. Ao final, novas transformações e variantes expandidas são propostas utilizando o formato de pequenos estudos sonoros, mostrando como tais procedimentos vêm sendo incorporados e integrados às práticas criativas do autor deste texto.

Palavras-chave: Transformação sonora. Conjuntos de onda. Vocoder de fase. Trevor Wishart. CDP. SuperCollider.

Abstract

The creative practice of transforming recorded sounds has its origins on the works by Pierre Schaeffer and the *Groupe de Recherches Musicales* (GRM), however, this practice acquires a new dimension with the development of the computer programs suite Composers Desktop Project (CDP) – headed by the English composer, performer and developer Trevor Wishart. Although CDP's programs were primely designed for non-real-time execution, the audio programming environment SuperCollider – which is mainly centered at real time logic – features flexibility not only for porting these processes but also allows and facilitates the creation of new correlated ones. Two main sound transformation types were analyzed, those in which the computer calculations were realized on the time domain – using the *wavesets* idea – and those in which calculations were realized on the spectral domain – using the data from the phase vocoder. In this work, CDP's time and spectral transformations are revisited under the light of the new possibilities brought by the SuperCollider and the auditory evaluation of source and transformed sounds. At last, new transformations and expanded variants are proposed in the format of small sound studies, demonstrating how these procedures have been incorporated and integrated into the creative practices by the author of the present text.

Keywords: Sound Transformation. Wavesets. Phase Vocoder. Trevor Wishart. CDP. SuperCollider.

Lista de ilustrações

Lista de códigos

Lista de tabelas

Lista de arquivos de áudio

A lista completa de arquivos de áudio para escuta está disponível em <http://www.musica.ufmg.br/lapis/?p=1141> ou <https://archive.org/details/fmiramar_master_thesis/>

Lista de abreviaturas e siglas

CDP	*Composers Desktop Project* , ou Projeto Desktop dos Compositores
DAW	*Digital Audio Workstation* , ou Estação de Trabalho de Áudio Digital
DFT	*Discrete Fourier Transform* , ou Transformada Discreta de Fourier
DSP	*Digital Signal Processing*, ou Processamento digital de sinais
DTFT	*Discrete Time Fourier Transform* , ou Transformada Discreta de Fourier de Tempo Discreto
FFT	*Fast Fourrier Transform* , ou Transformada Rápida de Fourrier
FOF	*Fonction d'Onde Formantique* , ou Função de onda de formante
GUI	*Graphical User Interface* , ou Interface Gráfica do Usuário
IRCAM	*Institut de Recherche et Coordination Acoustique/Musique*, ou Instituto de Pesquisa e Coordenação de Música e Acústica
iSTFT	*inverse Short Time Fourrier Transform* , ou Transformada inversa de Fourrier de Tempo Rápido
LPC	*Linear Predictive Coding*, ou Codificação Linear Preditiva
NESS	*Next Generation Sound Synthesis*, ou Próxima Geração de Síntese de Som
SMS	*Spectral Modeling Synthesis*, ou Síntese de Modelagem Espectral
STFT	*Short Time Fourrier Transform* , ou Transformada de Fourrier de Tempo Rápido
OLA	*Overlap-Add*, ou Sobrepor-Somar
OSC	*Open Sound Control*, ou Controle Aberto de Som
TSM	*Time-Scale Modification*, ou Modificação da escala temporal

Sumário

Introdução

Ao longo do desenvolvimento da música eletroacústica, computação musical e das tantas formas de arte sonoras surgidas no século XX, várias técnicas e processos ficaram restritos a pequenos grupos de compositores, artistas e desenvolvedores. Longe de ser um fenômeno exclusivo destas áreas, diversas razões existem para tão recorrente acontecimento: custo de equipamentos e softwares, restrições relacionadas à propriedade intelectual, técnicas pouco usuais, procedimentos excessivamente trabalhosos, sigilo das técnicas como forma proteção econômica, ausência de preocupação relativa à preservação de memória, etc. A pesquisa aqui desenvolvida aborda um conjunto destas técnicas que, devido a um misto de alguns destes fatores, ainda encontram-se demasiadamente restritas.

No ano de 2013, entre aulas de eletromagnetismo e ensaios em uma banda *cover* de antigos sucessos do Clube da Esquina, me deparei com a capa de um livro intitulado *Audible Design: A Plain and Easy Introduction to Practical Sound Composition* (Figura 25) largado à sorte na confusão do estúdio de um amigo. Numa breve folheada, depreciei o conteúdo por parecer demasiadamente simplório diante da minha memória povoada pelos complexos diagramas de Bode e convoluções, porém a capa ocupou-se de um lugar reservado na minha memória. Não sabia eu que, dentro de um ano, estaria no *Koninklijk Conservatorium* em Haia, numa palestra do autor do referido livro, imediatamente rememorado por ser o responsável pela ousadia de ilustrar Joseph Fourier, símbolo sagrado de uma grande revolução na engenharia elétrica do século XX, como divisor entre a *aritmética* e a *música*. Tampouco imaginaria eu que numa palestra formal e protocolar sobre a disponibilização do código fonte do Composers Desktop Project (CDP), estaria meu futuro objeto de pesquisa durante dois anos sucessivos. Qual poderia ser a utilidade de antigas e exóticas funções de tempo diferido numa era de erupção da computação em tempo real, das redes velozes e das interfaces inteligentes?

No prosseguimento dos estudos, ainda fora do Brasil, bastou uma primeira experimentação com o vocoder de fase, com o objetivo de improvisar *loops* ao vivo, para que tanto os resultados quanto os mecanismos deste algoritmo tomassem toda minha atenção. A partir daí, depois de vários outros caminhos sinuosos, foi graças ao estranho acesso a este exemplar de *Audible Design* – fora de catálogo desde os anos 1990, inexistente em bibliotecas brasileiras e só disponibilizado digitalmente em 2019 – que pude compreender mais detalhadamente o funcionamento do CDP bem como as ideias técnicas, estéticas, computacionais e poéticas de Trevor Wishart. Ao longo do presente trabalho, espero que esse fio de conhecimentos ocultados por diferentes circunstâncias mundanas possa ser percorrido da maneira mais nítida o possível por qualquer leitor que se sinta igualmente instigado por um livro cuja a capa traz a provocação: *audere scientia est audire*.

* * *

O objetivo deste trabalho é o estudo, adaptação e implementação no ambiente SuperCollider de processos de transformação sonora do CDP, majoritariamente desenvolvidos por Trevor Wishart, buscando ao mesmo tempo compreender a dimensão poética e conceitual dessas ferramentas em sua práxis criativa originária e desenvolver novas abordagens e aproximações poéticas no contexto da pesquisa em questão. Ao fazer uma reinvenção e tradução de tais ferramentas, buscamos avaliar o uso que o compositor dá às mesmas e propor novas formas de utilização correlatas ou não. Portanto, não será feito aqui apenas o estudo de uma ferramenta, mas sim uma investigação que permita a criação de ferramentas correlatas para utilização em outros trabalhos artísticos, avaliando determinado aparato técnico já desenvolvido.

Na busca por uma estrutura e metodologia de trabalho acadêmico que consiga abarcar devidamente as especificidades do campo artístico – em especial o domínio musical e sonoro – sem realizar importações precipitadas de ferramentas e metodologias das ciências exatas, humanas e biológicas, procuramos aqui nos orientar pela metodologia schaefferiana de escuta como instrumento primeiro e preponderante de pesquisa. Desta maneira, os processos de transformação sonora são descritos no seu modo de funcionamento, sempre que possível ilustrados graficamente. A partir daí, arquivos de áudio são providos para que se possa ouvir, escutar, entender e compreender tais processos (SCHAEFFER, 2017).

Diferentemente de abordagens da computação musical, faremos uma abordagem mais conceitual e teórica do funcionamento dos algoritmos, focando, assim, menos nos detalhes de implementação e mais nos processos e suas consequências (sonoras). Assim, este trabalho concentrou-se em avaliar as descrições técnicas dos processos do CDP e, a partir daí, traduzi-los para o ambiente SuperCollider. Assim, ao invés de realizarmos uma avaliação do código fonte do CDP e portar tais processos para o núcleo do SuperCollider, avaliamos o cerne conceitual dos processos dos CDP, para, então, implementar ou indicar as formas efetivas de execução destes nas especificidades da arquitetura do SuperCollider. Em contraponto com trabalhos de extração da informação musical (MIR), faremos análises que privilegiem a escuta e a avaliação qualitativa dos procedimentos, o que permitirá, em pesquisas futuras, retomar a investigação destas transformações utilizando descritores e avaliações estatísticas.

Aproximaremos a compreensão do mecanismo de transformação computacional da compreensão auditiva dos sons fonte e resultantes, por meio de uma elucidação dos pontos proeminentes destes dois polos. Além disso, faremos, sempre que possível uma avaliação das possíveis sonoridades resultantes de cada transformação computacional. Optamos por referências à tipomorfologia schaefferiana, todavia não iremos enquadrar cada um desses conjuntos de fonte-transformação-sonoridade na mesma, pois isto seria pouco útil devido à idiossincrasia dos resultados sonoros para cada tipo de fonte e transformação.

Como trabalhos desenvolvidos nessa mesma direção metodológica podemos apontar o já tradicional Schaeffer et al. (1967), que não desvela os detalhes dos processos técnicos de produção sonora, porém faz importantíssimas reflexões sobre os mais variados aspectos do fenômeno sonoro. Numa direção mais similiar ao que será desenvolvido aqui temos Wishart (1994), e Wishart (2018) em menor grau, por seu modelo de alinhavar os procedimentos técnicos com as sonoridades resultantes e os tipos de sons fontes utilizados. Por último, é importante ressaltar também Endrich (2016a) e Endrich (2016b), cuja discussão sobre sonoridade é menor, porém fazem uma grande exploração dos tipos de fontes e dos procedimentos utilizados. Estas referências são uma motivação para este trabalho especialmente por mostrarem quão impraticável é a tarefa de se montar um guia exaustivo de "boas práticas" para transformações sonoras.

Wishart nomeia seus processos de manipulação de áudio digital de maneira bastante peculiar, frequentemente utilizando termos poéticos ou metafóricos para referir-se aos processos de modificação dos dados. De maneira geral, estes termos tendem a se referir mais à dinâmica de modificação dos dados computacionais do que às implicações sonoras que tais processos geram no áudio. Como não foram encontradas traduções para estes termos na língua portuguesa, todas as traduções feitas aqui foram elaborados por nós. Optamos por manter o sentido mais literal o possível, evitando traduções poéticas para termos que, na sua mencionada origem, já apresentavam conotações amplas.

No Capítulo 1 exploraremos algumas das posições estéticas e filosóficas de Trevor Wishart para compreender mais profundamente as motivações para o desenvolvimento do CDP, as maneiras de utilização do seu conjunto de funções bem como o processo de escolha dos materiais visando tal contexto. Tanto a literatura de Wishart quanto as publicações sobre o CDP apresentam pouca ou nenhuma tradução para o português, o que restringiu consideravelmente a circulação de tais materiais e ideias nos círculos lusófonos. Assim, decidimos acrescentar as investigações sobre a obra de Wishart no Apêndice A, permitindo ao leitor uma compreensão mais aprofundada e contextualizada dos temas aqui abarcados.

No Capítulo 2 iremos apresentar os conceitos, princípios e os embasamentos teóricos que alicerçam as transformações propostas por Wishart. Apresentaremos também os conceitos fundamentais do SuperCollider que suportarão a realização prática de tais transformações, bem como uma contextualização de como os procedimentos computacionais utilizados no CDP estão inseridos no universo da música com computadores. Numa divisão que é mais operacional do que conceitual ou fenomenológica, mantemos a ideia do CDP de agrupar como transformações espectrais os processos que utilizam o vocoder de fase e, de outro lado, agrupar nas transformações temporais os processos de envelopamento, substituição de ataque, estratégias de reprodução, granulação, processos microssonoros e transformações de conjuntos de onda. Ao final deste capítulo, fazemos uma discussão sobre os procedimentos focados na difícil tarefa de realizar modificações na altura e escala

de tempo-frequência à luz dos demais procedimentos discutidos no capítulo.

No Capítulo 3 iremos mostrar como os processos de transformação de Wishart podem ser executados e expandidos por meio da arquitetura do SuperCollider. A partir de uma seleção das funções mais generalizáveis do CDP, fazemos, para cada uma, uma breve descrição, avaliação da sua possível sonoridade e exemplificamos seu funcionamento com exemplos no SuperCollider. As discussões e o exemplos realizados neste capítulo servirão de base para os estudos sonoros desenvolvidos no próximo capítulo.

A investigação dos processos técnicos e estéticos de Trevor Wishart realizada ao longo da elaboração deste trabalho, em conjunto com o estudo das possibilidades de adaptação das funções do CDP para o ambiente SuperCollider, nos permitiram vislumbrar diversos outros processos computacionais criativos, sendo eles processos derivados ou inteiramente novos. Além disso, como ferramenta de pesquisa e motivação para nossa prática criativa, refizemos, por outras vias, percussos estéticos trilhados e propostos por Wishart. Ultrapassa a dimensão deste trabalho implementar, descrever e avaliar rigorosamente todos estes novos procedimentos, portanto nos propomos, no Capítulo 4, a mostrar como tais procedimentos vêm sendo incorporados na nossa prática criativa. Pretendemos, assim, por meio da elaboração e avaliação de pequenos estudos sonoros fornecer algumas das explicações analíticas gerais dos novos procedimentos, desvelar a motivação para sua criação, avaliar brevemente a relação com as fontes sonoras utilizadas, sintetizar algumas das características sonoras resultantes e expor algumas das ideias poéticas que embasaram tais criações.

1 Ideias e ferramentas de Trevor Wishart

> *Nós devemos, de fato, reconhecer que os os textos verdadeiramente potentes de nossos tempos certamente não são os textos como este livro, ou mesmo as genuínas teorias científicas, mas sim os programas de computador por si mesmos*
>
> —Trevor Wishart, *Audible Design*

Neste capítulo mostraremos como as diversas frentes de atuação de Trevor Wishart – composição, performance, desenvolvimento de software e elaboração teórica – convergiram na arquitetura do Composers Desktop Project (CDP) e nas suas formas de utilização, propostas pelos desenvolvedores deste programa.

Num primeiro momento avaliaremos várias das ideias estéticas e filosóficas de Wishart diante da arte sonora, em especial a crítica à formalização da música do século XX e ao uso irrestrito e irrefletido da tecnologia na música, o contínuo sonoro, a metamorfose sonora e a transformação sonora. Em seguida faremos uma descrição geral do funcionamento do CDP e das formas de utilização do mesmo. Ao final faremos uma reflexão sobre o papel das linguagens de programação, interface primeira e fundacional dos recursos computacionais, na mediação entre os papéis de compositor, intérprete, performer, *luthier* e desenvolvedor e entre sua consequência última, as realizações sonoras com computadores.

* * *

Trevor Wishart (1946) é um compositor, desenvolvedor de software e performer inglês, nascido e radicado em York. Wishart desenvolveu e sistematizou um grande conjunto de técnicas vocais estendidas, elaborou significativa produção textual sobre arte sonora, transformação e espacialização de sons, e foi um dos principais idealizadores e desenvolvedores do Composers Desktop Project (CDP) – suíte de programas dedicada a realizar transformações de sons gravados.

Nenhuma das produções de Wishart apresenta tradução para o português, o que, em conjunto com sua maneira quase artesanal e radicalmente independente de publicar e distribuir, restringiu a circulação de suas ideias no cenário brasileiro. Como o autor aborda uma grande variedade de temas, sempre que necessário faremos pequenas apresentações das ideias centrais para os temas do presente estudo. Assim, para possibilitar uma melhor contextualização da obra de Wishart em relação a este trabalho, incluímos o apêndice *Trevor Wishart: biografia, peças sonoras e textos* ao final desse texto.

Na sua produção textual, Wishart apresenta uma defesa reiteradamente aguerrida de suas ideias e se posiciona acaloradamente acerca dos diversos temas tocantes à música dos séculos XX e XXI. Além disso, toda a sua produção é frequentemente marcada

Figura 1 – Trevor Whisart durante improvisação vocal

Fonte: Cemm

por traços fortemente pessoais: livros e artigos geralmente escritos em primeira pessoa; ausência de rigor das citações no estilo acadêmico; constante tom de humor, ironia e muitas vezes deboche; suposições e especulações diversas, etc. Essa conjunção de fatores dá à produção de Wishart um caráter de manifesto político-estético idiossincrático, na qual técnica, análise, política, filosofia, cultura e estética se misturam do início ao fim das obras. Apesar do compositor afirmar recentemente que já mudou várias das suas fortes opiniões (WISHART, 2012b), faremos a seguir uma breve exposição de algumas dessas ideias para compreender, mais especialmente, como se dão as relações entre procedimentos individuais de cada ferramenta, o processo de criação de ferramentas e a utilização destas na composição.

> (...) boas composições de sons sempre incluem um processo de descoberta, consequentemente chegar a um acordo com o inesperado e o involuntário, um processo crescentemente informado pela experiência assim que a compositora ou o compositor se engaja no artesanato, mas mesmo assim sempre aberto tanto para a descoberta surpreendente quanto para os erros de julgamento. Humildade perante à experiência é um essencial traço de caráter! (WISHART, 1994, p. 9, tradução nossa).

1.1 Crítica à formalização da música do século XX e ao uso irrestrito e irrefletido da tecnologia na música

On Sonic Art (1986) – a mais popular obra do autor – apresenta uma forte crítica à excessiva importância da notação para a música de concerto ocidental, expondo como a dissociação entre a escrita e a prática levou a música do século XX, sobretudo a música serialista, a um excesso de formalização. Num denso emaranhado de posições sociológicas, políticas e filosóficas, Wishart argumenta que as prioridades da notação simplesmente não refletem as prioridades musicais, na verdade elas as criam. Três pontos apoiam este argumento: (1) que a notação é fundamentalmente baseada na lógica de treliça [1] (alturas, durações e dinâmicas serem tratados como pontos discretos localizados em um reticulado uniforme), o que reduz ou muitas vezes elimina a importância de diversos outros aspectos essenciais ao fenômeno sonoro; (2) que é desnecessária a primazia da altura e duração como parâmetros essenciais da organização musical; (3) que a percepção e concepção da música focada na notação teria levado a música do século XX à uma abordagem formalista abstrata, engendrando assim peças cuja estruturas e sentido somente podem ser percebidas na leitura da partitura e não na experiência sensorial. (WISHART, 1996, p.11- 43, tradução nossa).

Wishart também aponta que a música instrumental da segunda metade do século XX frequentemente super-valorizava teorias, concepções técnicas, aspectos visuais da partitura e formalizações racionais de sistemas de organização em detrimento à experiência básica de percepção de tais proposições. Apesar de não se alinhar a nenhum movimento estético declaradamente, Wishart, nesse aspecto, tem uma clara aproximação com as ideias centrais de Pierre Schaeffer e o *Groupe de Recherches Musicales* (GRM), priorizando a importância da experiência sensorial para a avaliação e criação de seus procedimentos sonoros.

> O que eu procuro são critérios, que possam ser verificados pela experiência, para o fazer musical. A preocupação com a notação convencional pode nos levar ao formalismo, uma situação na qual não mais existe qualquer verificação experiencial das nossas teorias sobre como compor música. (WISHART, 1996, p. 11, tradução nossa).
>
> (...) para mim, por outro lado, uma experiência musical que pareça aleatório é aleatória. A experiência que o ouvinte têm *é* a música; a metodologia do compositor, não importa o quão racional ela seja, é um dispositivo heurístico percebendo a fonte daquela experiência no som. (WISHART, 1996, p. 43, tradução nossa).

[1] Tradução nossa para o termo *lattice*. A palavra mais adequada que encontramos para traduzir esse termo é *treliça*, em especial referência à rede reticulada utilizada como ornamento e decoração na arquitetura e não uma referência à estrutura homônima da construção civil, constituída geralmente de uma rede triangular para suporte e equilíbrio de construções. O termo *lattice* em inglês também apresenta comumente o primeiro sentido mencionado, sendo que o objeto da construção civil mencionado é denominado *truss*.

Ainda em concordância com as ideias de Schaeffer, Wishart ressalta que tanto a tecnologia quanto o conceito ocidental de instrumento musical - dispositivo de timbre relativamente estável dentro de uma dada faixa de alturas discretas - reforçam a ideia de música como uma restrição de poucos parâmetros sonoros à uma organização em *treliça* (WISHART, 1996, p.23). Em consonância com tal crítica, Wishart também aponta um desdobramento similar que ocorreu ao longo da integração dos diversos aparatos eletrônicos na música: apesar da imensa possibilidade de geração de sons dos sintetizadores, as demandas da indústria pressionaram seu desenvolvimento para a lógica da organização em *treliça*, como é nítido no amplamente utilizado protocolo MIDI.

Wishart ressalta ainda que devido aos fortes avanços computacionais dos anos 80 e 90 que levaram à possibilidade de criação de *meta-instrumentos* - softwares, computadores embarcados ou sintetizadores capazes de gerar uma colossal quantidade de sons - ainda mais critério e seleção seriam necessários aos compositores. Uma vez que se dispõe de uma ferramenta tão poderosamente generalista, para que se consiga atingir alguma dada proposta musical faz-se necessário ainda mais rigor de testes de escuta e seleção, pois qualquer proposta técnica é facilmente atingida (WISHART, 1992). Em Audible Design (1994), o compositor menciona ainda que vários procedimentos de transformação de som vem sendo generalizados sob a forma de "efeitos" ou "*plugins*", o que gera uma justificação do uso da tecnologia por si mesma, quase sempre desvalorizando a experiência do resultado sonoro. Na visão estética do compositor, boas transformações sonoras só podem ser atingidas quando se seleciona minuciosamente os materiais e os processos, gerando assim os resultados específicos que satisfazem suas propostas de experiência sonora.

> Construir, testar pela escuta, reconstruir. Essa habilidade irá, espera-se, reduzir a tentação de produzir posturas ideologicamente impregnadas para defender nossos procedimentos composicionais. (WISHART, 1992, p. 586, tradução nossa).

É possível que essa refração à hipertrofia da racionalização e à supervalorização da escrita tenham engendrado o posicionamento mais estrito de Trevor Wishart de sempre propor a cada peça uma motivação claramente estética e outra técnica. Nos seus trabalhos mais recentes, de 2011 em diante, o compositor, entretanto, afirma que não tem mais essa noção rígida de separação entre técnica e estética, mas que a poética geralmente provê um arco geral para os seus processos, sendo que a relação entre técnica e estética geralmente é densa e complexa (WISHART, 2011).

1.2 Contínuo sonoro

> Música, entretanto, não pode ser divorciada do meio do som e entra na nossa experiência como parte de uma realidade imediatamente concreta; ela impinge em nós e fazendo isso afeta nosso estado. Além disso, como

> Susanne Langer relembra em Sentimento e Forma, em sua articulação do *contínuo temporal* da experiência concreta, isso corresponde diretamente ao contínuo da nossa experiência, o contínuo fluxo de nosso estado-de-resposta. (WISHART, 1996, p.16, tradução nossa).

Influenciado tanto pelas ideias de Pierre Schaeffer quanto por suas pesquisas sobre as possibilidades do aparelho vocal, Wishart propõe – sem o rigor científico e estruturalista típico dos textos schaefferianos – que o fenômeno sonoro seja compreendido como contínuo e localizado em um espaço igualmente ininterrupto: o chamado *contínuo sonoro*. Por espaço, entenda-se tanto o meio de propagação sonora para ondas acústicas quanto para um espaço de representação de características sonoras (por exemplo uma representação tridimensional com os eixos de altura, duração e timbre). Numa metáfora que brinca com sua formação acadêmica inicial (graduação em química), Wishart sugere que a *arte sonora* deve se livrar das suas concepções arquitetônicas – nas quais os discretos blocos de sustentação do sistema são empilhados e organizados rigidamente em estruturas tipo *treliça* – e adotar uma formulação mais próxima da química – na qual os elementos constitutivos, nas devidas condições, podem apresentar estados diversos e misturam-se e reagem entre si num contínuo, originando novos compostos com novas propriedades (WISHART, 1993).

Em Wishart (1996) várias especulações sobre a estrutura desse espaço são propostas, tais como se tal espaço disporia de alguma métrica natural, se tal espaço poderia ter alguma estrutura matematicamente descritível, etc. Além disso, Wishart também propõe diversas possibilidades de comportamento das estruturas sonoras dentro desse contínuo, numa racionalização que dialoga com o trabalho de Denis Smalley; propõe processos de notação, especialmente para as peças vocais; proposições sobre a ideia de *gesto* para a tipo-morfologia dos sons; contraponto entre tais gestos de sons; além de uma densa seção sobre espacialização no *contínuo sonoro*.

Já em Wishart (1994) elenca-se os parâmetros que podem ser mais facilmente controlados e percebidos dentro do *contínuo sonoro*, nos termos do compositor: altura ou faixa-de-altura; movimentos de altura; harmonicidade, inarmonicidade e suas evoluções; envelope espectral, formantes e sua evolução; estabilidade espectral; *ruidosidade* [2] (e.g. comparação entre fala vocal, não-vocal, semi-vocal); ataque[3] e seu envelope; continuação dispersiva, ondulatória e forçada; envelope de intensidade (sustentação, ataque-ressonância, crescimento, etc), dentre outros. Ressalta-se que ao longo dos dois primeiros livros de Wishart não é de seu interesse refazer a tipo-morfologia schaefferiana, mas sim fornecer acréscimos e também propor experimentos e questionamentos auditivos. A importância da experiência sensorial é reforçada ainda mais em Wishart (1994) quando o compositor se propõe a testar essas diversas estruturas da organização musical por meio de processos de transformação computacional ou não.

[2] tradução nossa para o termo *noisiness*.
[3] Wishart utiliza o termo *onset* para referir-se à duração do ataque de um som.

> As dimensões da altura e duração têm uma propriedade especial – serem cíclicas. Por meio disso quero dizer que podemos mover continuamente (sem passos) até a dimensão da altura, mas eventualmente chegaremos a um lugar que, perceptualmente, assemelha-se com nosso ponto de partida, a oitava acima da altura, que nós percebemos, em algum sentido, como sendo a mesma altura.
>
> A solução que eu propus para esse paradoxo é reintegrar todas essas propriedades definíveis (incluindo altura e duração) no som em si mesmo e adotar uma visão holística com o som. Nós poderemos então fazer qualquer transformação desses sons em um espaço multidimensional, não só ao longo dos eixos das alturas e durações, e também não somente ao longo de uma das muitas dimensões do timbre, mas em direção qualquer que seja, mudando, por exemplo, altura, envelope espectral e inarmonicidade todas uma só vez. Mais importante, nós devemos considerar a forma musical como a organização dos sons por meio dessa ideia generalizada de transformação dos sons, e não simplesmente como a organização de propriedades dos sons particularmente abstratas e idealizadas (altura e duração). (WISHART, 2011, p. 152, tradução nossa).

Uma ideia trazida de Schaeffer e do GRM explorada com mais detalhes por Wishart é a conjectura de que a a categoria timbre poderia fazer mais sentido perceptualmente se a altura fosse considerada como um caso específico dessa grande categoria, e não como uma dimensão perceptiva separada. *On Sonic Art* apresenta, ainda, especulações sobre como a teoria matemática da catástrofe poderia auxiliar nas explicações e fundamentações das *metamorfoses sonoras* relacionando as possíveis alterações no espaço do *contínuo sonoro* com as formas espectrais dos sons.

1.3 Metamorfose sonora e transformação sonora

Os conceitos mais explorados na obra de Wishart, tanto na sua produção bibliográfica quanto artística, são os de *metamorfose sonora* e *transformação sonora*. Apesar do compositor não apresentar uma definição clara e delimitada destes conceitos – além do seu significado variar ao longo da produção bibliográfica – faremos aqui uma redefinição própria destas ideias, bem como uma demarcação entre elas, clarificando os sentidos destes conceitos para os fins deste trabalho.

- ***transformação sonora*** - é todo o processo que altera um dado som em um dado espaço de tempo.
  - pode ser um processo extremo, e.g. transformação de uma senoide em um ruído branco.
  - os materiais de origem e destino podem ter ou não correlação auditiva.
  - pode ocorrer suavemente de um ponto ao outro (*transformação contínua*) – e.g. modulação em frequência de uma senoide por uma onda moduladora de frequência lentamente crescente; filtragem variável e suave de uma entrada

de áudio (microfone, captador, etc); aplicação de um processo de esticamento temporal gradativamente mais longo em um som gravado.

  - pode ocorrer em uma sequência discreta de passos (*transformação sequencial*) – e.g. um curto *sample* de percussão que é repetido e a cada vez distorce-se um pouco mais.
  - pode durar indefinidamente e transitar entre diversos materiais sonoros.
  - admite a alteração de quaisquer parâmetros sonoros, sejam aqueles mais operacionais, como velocidade de reprodução, taxa de amostragem, amplitude, etc, ou mudanças em parâmetros computacionais que visem alterar diretamente atributos perceptivos, *e.g.* filtragens que visem alterar a altura percebida.

- ***metamorfose sonora*** - processo que altera um som, em um dado espaço de tempo, gerando um outro som correlato auditivamente.

  - é um caso particular de *transformação sonora*.
  - inicia-se em um som específico (fonte) e alcança outro som específico (destino), logo apresenta duração finita.
  - deve necessariamente apresentar correlação auditiva, portanto é um conceito subjetivo.
  - busca manter durante o processo de transição "a unidade essencial do todo" [4], como forma de manter a contínua coerência auditiva do processo.
  - como se almeja transições específicas entre dois sons, apenas alguns tipos de parâmetros poderão ser alterados, *e.g.* em uma metamorfose entre dois sons de mesma altura mas executados em instrumentos distintos, não é uma estratégia viável reduzir a resolução temporal para buscar tal metamorfose.

Em primeiro lugar, destacamos que ambas as definições estão fortemente associadas à ideia de forma musical, ou seja, como as médias e curtas estruturas sonoras desenvolvem-se e organizam-se numa escala temporal mais longa. Wishart afirma que essas suas noções de forma da *metamorfose sonora* e *transformação sonora* podem ser vistas como uma generalização das noções de variação e desenvolvimento da música instrumental, mas aplicadas ao *som como um todo* (WISHART, 2012b). Esses conceitos tem origem também nas avaliações que Wishart faz das abordagens que as diversas correntes da *arte sonora* deram à ideia de forma musical.

> Duas tradições emergiram no mundo da *arte sonora* (e essas não são mutuamente exclusivas, como eu espero demonstrar em algumas das minhas peças). A primeira deriva do trabalho de John Cage e é mais

[4] Dostoiévski, F. *Os Irmãos Karamázov*, trad. Paulo Bezerra.

> prevalente entre artistas visuais e de mídia. Aqui, as noções tradicionais de "linguagem"musical são abandonadas em favor da montagem direta dos materiais. Esses materiais são geralmente escolhidos pelo que eles representam (o que eles são enquanto gravação de algo, o que a gravação revela sobre suas origens) e com uma inclinação estética que defronta a *ideia por trás* de usar um som em particular, ou uma justaposição de sons.
>
> A segunda abordagem, originalmente nascida do trabalho do GRM em Paris na metade do século XX, relaciona-se mais intrinsecamente com as *relações audíveis* entre os sons, e, portanto, está mais inclinada à tradição da música instrumental. Entretanto, essa abordagem originalmente cresceu em um contexto de alto modernismo da música da metade do século XX, com suas complexas estruturas musicais. A resultante sofisticação da forma, combinada com a ausência de notação escrita, significou que os trabalhos poderiam ser difíceis de serem penetrados analiticamente e para propósitos didáticos. Isso significou que a linguagem comum para abordar sons *como sons* cresceu mais por um processos de osmose, ou pela aprendizagem das técnicas comuns de estúdio, do que por uma análise detalhada ou um estudo de trabalho existentes. No final do século XX, a chegada da gravação digital e da análise computacional de sons fez crescer imensamente nosso conhecimento e habilidade de manipular estes sons. O conhecimento e os *insights* que tinham crescido fora da prática de estúdio podiam agora ser amplamente estendidas de forma racional pela aplicação da potência computacional ao som. (WISHART, 2012b, p. 4, Tradução nossa).

As peças de Wishart tais como *Red Bird* e *Journey into Space* [5], e em especial suas ideias sobre forma, são muitas vezes associadas à ideia de narrativa – ou de uma possibilidade da existência de tal tipo organização relacional na *arte sonora* – e também à possível ideia de *metáfora musical*, no qual o mundo sonoro acusmático permitiria que algum sentido conotativo que seja compartilhado e compreendido por ouvintes em geral. Wishart inicia alguma exploração nessa área em *On Sonic Art*, na direção de questionar se ouvimos as metamorfoses sonoras somente porque estamos num contexto acusmático ou porque existe um paralelo físico em tais transformações. Todavia sua produção bibliográfica não avança muito na direção de investigações mais precisas e científicas que justifiquem os fenômenos apresentados. Tais concepções não serão exploradas extensivamente neste trabalho, mas corroboram a ideia da centralidade da escuta acusmática para a plena realização das ideias técnicas e estéticas de Wishart (AMELIDES, 2016; LANDY, 1991).

> Ele fala sobre aplicar metamorfose a sons para criar “metáforas musicais”. Ele é interessado em narrativa (o que pode ser difícil de apreender ou, em algumas palavras, ambíguo) em oposição à abstração da arte pela arte. Existiram vários tratados escritos sobre metáfora na tradição europeia de música instrumental, todavia essas metáforas estão enraizadas na música, que é essencialmente abstrata. É por meio da natureza concreta da maioria das transformações sonoras eletroacústicas (por elas consistirem de dois ou mais sons representativos) que o antigo conceito de metáfora

[5] Uma das primeiras peças acusmáticas de Wishart, baseada essencialmente na ideia de (re)construir paisagens sonoras de três tipos de viagem de um personagem principal: uma jornada ao mundo real, uma jornada ao espaço e uma jornada ao mundo dos sonhos.

> dentro de um arcabouço musical pode ser modernizado. (LANDY, 1991, tradução nossa).
>
> Por fim, uma palavra especial sobre música eletroacústica. No mundo físico real nos somos capazes de dizer claramente que um som de uma barra metálica caindo no chão não é um enunciado, enquanto um som produzido por um ser é. No espaço virtual dos alto-falantes esse tipo distinção se torna difícil de ser feita. Um objeto sonoro criado artificialmente pode ser articulado de tal maneira que nós apreendemos pistas de um enunciado ou não. Nós brincamos com essa interpretação da paisagem. Esse aspecto do meio eletroacústico é outra característica contribuindo para a sua potencial qualidade onírica, a criação de um universo artificial no qual nossas pressuposições são colocadas em questionamento e onde nós somos trazidos a ver o mundo de uma perspectiva inteiramente diferente. (WISHART, 1994, p. 262, tradução nossa).

Na aplicações comerciais de algoritmos de áudio, o termo morfagem (tradução nossa para *morphing*) é utilizado sempre que se deseja transformar um dado som fonte em um dado som destino, em especial, a transformação entre sons de diferentes instrumentos musicais. Em Caetano (2011) há estudo aprofundado destas técnicas em conjunto com uma formalização dos princípios que validariam a ideia de morfagem, como por exemplo, a correspondência entre os objetos, a suavidade da transição, a distância conceitual, dentre outros. No presente trabalho, faremos uma distinção conceitual entre *morfagem* e *metamorfose sonora*: a primeira representa os algoritmos e técnicas para a transformação de um som fonte em um som destino (interpolação espectral, síntese de modelagem espectral, *cross-fade*, etc), geralmente focados na transformação de objetos sonoros facilmente delimitáveis, como as notas instrumentais; a *metamorfose sonora*, por sua vez, é a utilização dos algoritmos e técnicas de *morfagem* para a realização de tarefas criativas mais complexas, por exemplo transformar um enunciando de uma sílaba em um som de enxame de abelhas mantendo a coerência auditiva dessa transformação, ou seja, transformar um objeto sonoro claramente delimitável em uma nuvem de objetos que compõem uma textura. Neste cenário, o trabalho de Wishart fornece a contribuição de mostrar que as soluções para esse problema passam por várias relações de compromisso: escolhas das fontes e dos materiais alvos, procedimentos específicos para cada tipo de material, necessidade de geração de materiais sonoros intermediários pouco redundantes, uso da espacialização de maneira criativa, sincronização entre temporalidades dos objetos e processos, dentre outros (WISHART, 1994).

1.4 Transformações sonoras: um problema de abordagem

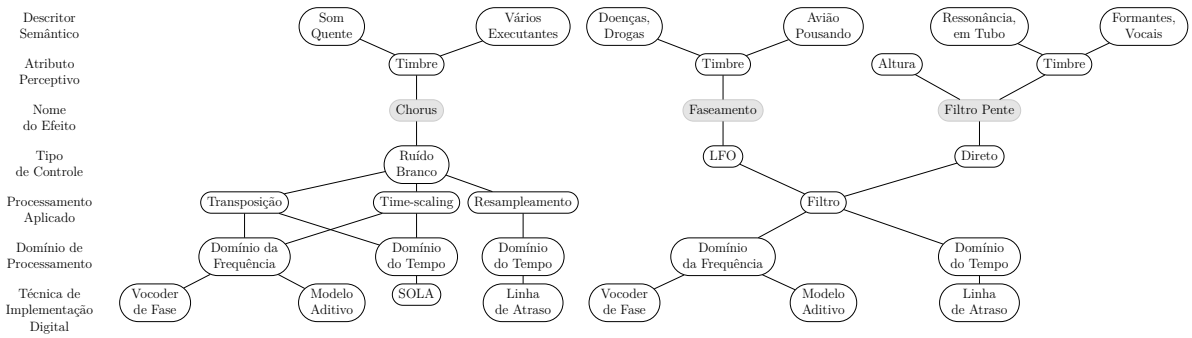
As tentativas de categorização sistemática dos processos de transformação sonora costumam levar a dois paradigmas igualmente problemáticos: de um lado modelos que optam por uma excessiva simplificação para garantir a delimitação das classes; de outro

lado modelos que admitem classes mais amplas contudo geram grandes, turvas e complexas intersecções e interconexões entre as classes.

O fenômeno sonoro é multifacetado: um processo de transformação (distorção, filtragem, reverberação, etc) sempre produz ou vem acoplado a vários outros efeitos sonoros. Uma vez que é comum a predominância de um dado efeito sobre os outros, é frequente as literaturas da área optarem por uma classificação perceptiva que visa a predominância perceptiva de um dado efeito sonoro. Isso restringe os algoritmos a modelos que se comportam bem diante da teoria, mas que, do ponto de vista criativo, reduzem suas possibilidades técnicas e sonoras.

Diante desse cenário, devemos posicionar as transformações propostas por Wishart de acordo com as tradicionais categorias existentes de tipos de efeitos sonoros? Como fazê-lo? As propostas de modelos mais rigorosos (ZOLZER, 2011), como na Figura 2, são esclarecedoras da maneira como as implementações e nuances computacionais se dão. Todavia, estes modelos forneceriam elementos suficientes para os trabalhos de arte sonora? Por outro lado, as descrições propostas por Wishart, frequentemente imbuídas de regionalismos, personalismos e outros diversos aspectos pouco formais e científicos forneceriam parâmetros esclarecedores sobre os processos, permitindo uma compreensão ampla de tais descrições?

Figura 2 – Classificação de três efeitos sonoros digitais – *chorus*, faseamento e filtro pente – de acordo com suas categorias semânticas, perceptivas e computacionais



Fonte: traduzida e adaptada de Zolzer (2011, p. 244)

Uma vez que as possibilidades de transformações sonoras às quais se pode submeter um dado som são infinitas, faz-se necessário determinar um critério de demarcação (técnico, estético, filosófico, poético, etc), possibilitando, assim, a exploração sistemática de uma dada transformação. Wishart – partindo de sua motivação estética de realizar *metamorfoses sonoras* – opta pelo critério de delimitar como transformações sonoras os procedimentos que mantenham uma linha de percepção auditiva coesa. Sob outra perspectiva, a literatura

tradicionalmente mais próxima da engenharia e da computação musical [6] limita-se a procedimentos que podem ser considerados canônicos no campo do processamento de sinais digitais (distorção, filtragem, *chorus*, etc) como em Zolzer (2011), Reiss e McPherson (2014), Park (2010), cujas predições tendem a corroborar modelos teóricos tradicionais no campo do DSP e cujos resultados tendem a ser mais controlados[7].

> O poder de processamento de sinais dos computadores significa que, tecnicamente falando, nosso som inicial pode ser infinitamente maleável. Nestas circunstâncias, nós precisamos fazer uma distinção entre o que é *tecnicamente* uma transformação e o que é *perceptivamente* uma transformação. Em um caso extremo, qualquer som inicial pode ser transformado em em ruído por meio da multiplicação sucessiva de cada amostra por um número aleatório diferente. Tecnicamente falando, isso pode ser considerado uma transformação (o algoritmo envolvido é bastante direto), mas perceptivamente não há relação entre o som inicial e o som resultante (de fato a mesma resultante, ou uma muita similar, pode surgir de qualquer som que escolhamos). Algumas mudanças sutis na natureza do som (por exemplo mudanças nas características do ataque) podem alterar radicalmente nossa atribuição de fontes (o que nós imaginamos que a fonte sonora seja), portanto nós devemos estar bastante cientes se desejamos preservar algum senso de conexão entre a fonte e o resultado. (WISHART, 2012b, p. 72, tradução nossa).

Dessa maneira, é mais frequente observarmos os escritos de Wishart abordando as transformações sonoras sob o prisma dos materiais e seus derivados, especialmente à luz de uma escuta de inspeção minuciosa que disseca os objetos sonoros para utilizar seus elementos como pontos de inflexão das *metamorfoses sonoras*. O procedimentos descritos em sua literatura são, portanto, experimentos pouco controlados, cujo leitor tem acesso a um guia geral dos métodos e algumas ferramentas básicas (programas do CDP), mas não acessa os insumos. A perspectiva da engenharia e computação, por sua vez, concentra-se frequentemente em resultados controláveis, buscando processos que sejam generalizáveis a vários tipos de fontes sonoras e que possibilitem a avaliação por meio de parâmetros quantitativos (eficiência computacional, relação sinal-ruído, potência e largura de banda, etc).

> (...) a questão que um compositor se pergunta é: este processo é esteticamente interessante? O som resultante deste processo se relaciona perceptivamente e musicalmente de uma maneira útil ao som que começamos ? O que nós estamos procurando é uma maneira de transformar

[6] No Brasil, a demarcação das áreas de estudos envolvendo som e tecnologia ainda é turva e recente. Adotamos aqui os termos que são mais utilizados internacionalmente: música com computadores, tradução de *computer music*, como a área que se ocupa, na prática, das possibilidades e limites técnicos criativos de produção sonora utilizando computadores; sonologia como um termo mais geral de estudos do som, frequentemente mais ligado à área de humanidades; engenharia de áudio e computação musical, como abordagens específicas, das áreas da computação e engenharia, ao fenômeno sonoro

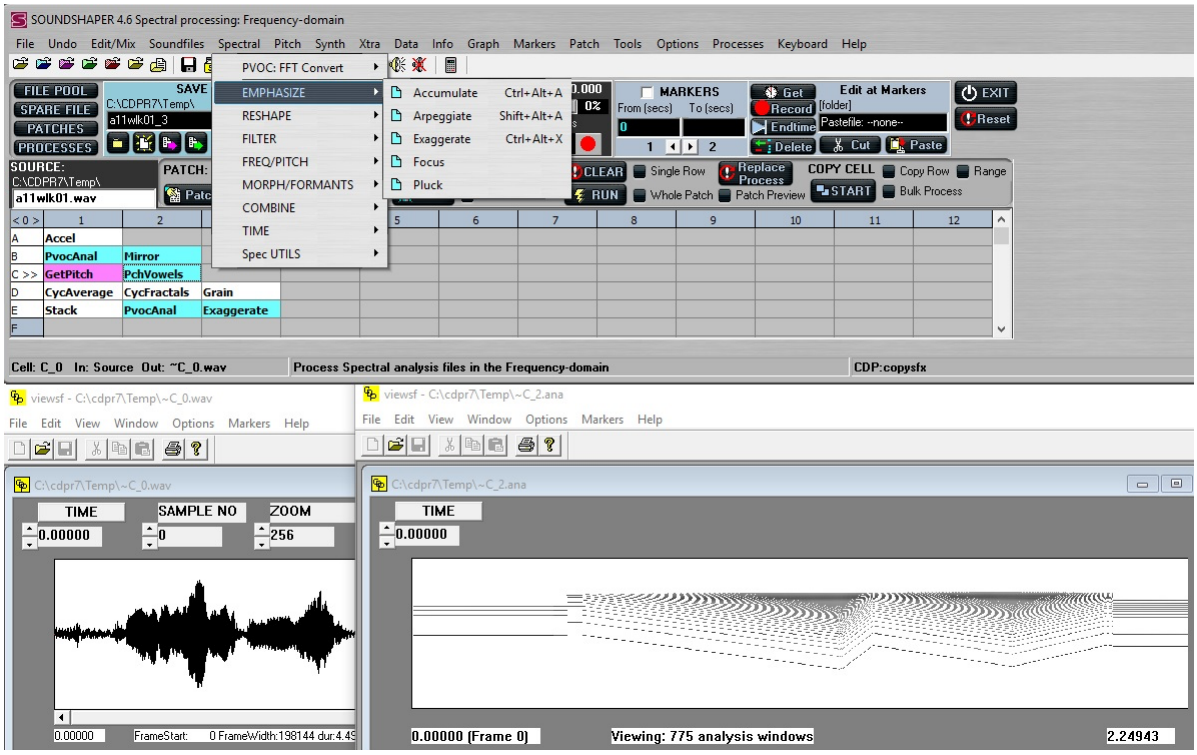
[7] A avaliação das possibilidades, efeitos e premissas de transformação sonora ainda é um campo de estudos recente, sendo que as referências encontradas vão pouco além das anteriores e dos escritos de Wishart

> um material sonoro para gerar sons resultantes que, claramente, sejam intimamente próximos da fonte, mas também claramente diferentes. (WISHART, 1994, p.4, tradução nossa).

Buscamos neste texto aproximar essas duas abordagens apresentando, sempre que possível, tanto as práticas comuns de utilização dos programas do CDP quanto esclarecer a relação entre o material sonoro escolhido, o procedimento computacional e seus efeitos perceptivos.

1.5 Composers Desktop Project (CDP)

Figura 3 – SoundShaper - Interface gráfica do CDP para Windows



Fonte: próprio autor

O CDP tem suas origens em 1986 a partir da iniciativa colaborativa de vários compositores e desenvolvedores ingleses: Andrew Bentley, Archer (Tom) Endrich, Richard Orton, Trevor Wishart, David Malham e Martin Atkins. O projeto nasceu como um estudo de viabilidade da portabilidade do programas desenvolvidos no Computer Audio Research Laboratory (CARL)[8] da suas plataformas originais (computadores *mainframe*) para os computadores *desktop* (neste caso o Atari ST). O CDP teve um importante salto a partir da residência artística de Wishart no Institut de Recherche et Coordination Acoustique/Musique (IRCAM), momento em que ocorreu o início da implementação das

[8] Em especial o *cmusic* e o vocoder de fase.

transformações espectrais via vocoder de fase e transformações temporais envolvendo grãos e conjuntos de onda (ENDRICH, 2016c; WISHART, 2000) (ENDRICH, 1997, p.33).

A arquitetura do CDP constitui-se de uma suíte de vários pequenos programas modulares. O seu funcionamento geral consiste em carregar um arquivo de áudio, aplicar uma transformação singular, ou múltiplas transformações sucessivas, e então gravar um novo arquivo de áudio resultante desse processo. Wishart refere-se a cada rotina específica composta por estas três etapas como um *instrumento* e a esse procedimento de transformação de sons como composição de sons.

O CDP executa transformações em dois tipos básicos de arquivos: arquivos de áudio (.wav e .aiff), que servirão de base para os processos temporais; e arquivos de análise de frequência (.ana), base para os processos espectrais e obtidos por meio de FFT. Além disso, existem arquivos auxiliares que fornecem instruções de controle para os processos: envelope espectral (.for), que provê as formantes; a curva de altura (.fqr); e arquivos gerais com instruções de controle (.txt e .bkr) (CDP, 2016b). O programa, além de apresentar suas intrínsecas funções de transformação, apresenta várias funções clássicas de estações de trabalho de áudio digital (DAWs), tais como mixagem, cross-fade, filtragem, reverberação etc. Tais funções não estão no escopo deste trabalho, pois já apresentam um infinidade de implementações em variados softwares e hardwares, e.g. Reaper, Pro Tools, Logic, Cubase.

A principal documentação do CDP organiza seus diversos procedimentos em dois grandes grupos: procedimentos temporais e espectrais (CDP, 2016a). Depois desta divisão, faz-se uma subdivisão em grupos de funções de acordo com a forma como os dados serão manipulados, conforme vemos abaixo:

Descrição geral dos grupos de funções

- **Blur** – borra os dados espectrais tornando difusa a clareza
- **Combine** – combina os dados da análise de frequência de dois ou mais arquivos
- **Distort** – distorção baseada em conjuntos de onda (pseudo ciclos de onda)
- **Envel** – alterações na amplitude do envelope
- **Envnu** – operações especializadas em envelopes
- **Extend** – processos que estendem e/ou segmentam um som no tempo
- **Filter** – técnicas para filtrar ou focar certas bandas de frequência
- **Focus** – foca ou sustenta dados de magnitude espectral
- **Formants** – extrai e impõe envelope espectral (formantes)
- **Grain** – granula e manipula grãos
- **Hilite** – enfatiza a magnitude espectral, incluindo filtragem espectral
- **Housekeep** – utilidades básicas de organização de arquivos de áudio
- **Modify** – grande variedade de efeitos incluindo a brassagem[9], transposição, intensidade, pan, atraso e modulação em anel
- **Morph** – cria transições suaves entre os sons, utilizando dados de magnitude e frequência

[9] ver seção 2.2

- **Multichannel** – difusão e processamento multicanal
- **Multi-channel Toolkit** – funções de manipulação de arquivos multicanal
- **Oneform** – operações em uma única formante
- **Pitch (& Harmony/Freq)** – operações baseadas em parciais
- **Pitchinfo** – informações sobre parciais ou extraídas de traços de alturas
- **Psow** – manipulação de grãos de altura síncrona (FOFs)
- **Pvoc** – FFT análise e ressíntese
- **Repitch** – extrai e altera dados da análise de frequência relacionados com a altura
- **Reverb** – funções para reverberação de arquivos de som
- **Rhythm** – rearranja e reorganiza temporalmente eventos em um arquivo de som
- **Sfedit** – edição básica de arquivos de som
- **Sndinfo** – informações básicas dos arquivos de som
- **Spec** – ganho e utilitários de edição no domínio espectral
- **Specinfo** – informações sobre os dados espectrais
- **Specnu** – várias funções, especialmente para despoluir os arquivos de análise de frequência
- **Strange** – alguns efeitos pouco usuais e imprevisíveis
- **Stretch** – estica (ou contrai) os dados de tempo ou frequência
- **Submix** – mixagem e processos relacionados
- **Synth** – funções para sintetizar sons
- **Sysutils** – utilitários do sistema
- **Texture** – cria texturas de sons

CDP (2016a, tradução e adaptação nossa)

Historicamente, os programas do CDP eram invocados por linhas de comando e processados em lote (arquivos *batch*) ou por meio de scripts Shell, conforme mostra o Código 1.1, sendo que as operações principais eram de leitura e escrita em disco. Assim, o CDP ocupou-se majoritariamente em criar processos de transformação de áudio digital (DSP), deixando as maneiras de acessar tais processos (interfaces gráficas, linguagens de *scripting*, hardwares de controles, etc) para outras plataformas.

```
1  # Superpõe miniaturas dos conjuntos de onda fonte em si mesmos
2  distort fractal /Users/nomedeusuario/Desktop/drone.wav /Users/nomedeusuario
       /Desktop/.wav 3 0.9
3
4  # Análise espectral com janela de 512 pontos
5  pvoc anal 1 /Users/nomedeusuario/Desktop/drone.wav /Users/nomedeusuario/
       Desktop/C3-piano-mono.ana -c512
6
7  # Processamento - exagera o contorno espectral
8  focus exag /Users/nomedeusuario/Desktop/C3-piano-mono.ana /Users/
       nomedeusuario/Desktop/C3-piano-mono-exag.ana 0.5
9
10 # Ressíntese
```

```
11 pvoc synth /Users/nomedeusuario/Desktop/trumpet_glis.ana /Users/
       nomedeusuario/Desktop/C3—piano—mono—exag.wav
```

Código 1.1 – Executando o CDP por meio de comandos Bash no Terminal

Atualmente, existem duas interfaces gráficas (GUI) desenvolvidas exclusivamente para operações dos programas do CDP, Sound Loom (Mac) e SoundShaper (PC) – ver Figura 3 – que funcionam de forma similar às DAW tradicionais. Tais interfaces são pouco amigáveis, pouco transparentes e muitas vezes confusas. Acrescentando-se isso ao fato de que a documentação do CDP é restrita e concisa, é comum que a curva de aprendizado de tal ferramenta seja bastante inclinada.

Desde a sua criação, os idealizadores do CDP sempre ressaltaram que suas posições estéticas fundamentaram a organização e as diretrizes de elaboração dessa suíte de programas, sendo, portanto, um software *de compositores para compositores*. Há uma menção feita por Endrich (2016c) de cinco metas almejadas por este grupo:

- foco na música concreta;
- deixar a síntese sonora para outros softwares;
- utilização de linha de comando;
- foco na escuta ao invés de foco na representação gráfica;
- máxima funcionalidade.

Aparentemente todas essas ideias não são um consenso estrito no grupo, porém, dois pontos básicos sobressaem: (1) que é uma ferramenta específica para o trabalho com sons gravados, em oposição aos recursos de síntese, estando, por isso, mais próxima da forma de trabalho do GRM e da *música concreta*; (2) que apesar de ser uma ferramenta de trabalho em tempo diferido (*offline*) seria muito provável que no futuro fosse possível uma integração das ideias desenvolvidas neste com DAWs e ambientes de programação de áudio para tempo real. Isso se torna ainda mais claro tanto na decisão do grupo de desenvolvedores de publicar um manifesto sobre o uso desta ferramenta quanto nas ideias em si expressas neste texto:

> **Sobre Nós**
>
> **Um ponto de vista de compositores**
>
> Os membros do CDP são profundamente envolvidos nas grandes mudanças do fazer musical atuais. Como uma organização, nós colocamos ênfase na transformação do sons em contextos musicais criativos.
>
> Nosso foco é prover novas ferramentas computacionais para o design sonoro, nas quais a ênfase está num detalhado e flexível acesso às características internas dos sons. Nossa meta é ajudar os compositores a trabalharem com sons. Para esse fim, numerosos programas são semi-algorítmicos em sua natureza, como as rotinas de segmentação e o Conjunto de Texturas, que modelam inteiramente passagens musicais

> multi-evento – "semi-algorítmico nesse contexto significa formas de ajustar tanto os parâmetros individuais quanto os múltiplos eventos ao longo do tempo.
>
> Em outras palavras, CDP não é somente focado em funções "DSP" (processamento digital de sinais) nem em sequenciamento ou edição de áudio, e de forma alguma em síntese sonora: outros softwares lidam com essas coisas satisfatoriamente. CDP foca em transformar sons existentes (*musique concrète*) e em como estes podem ser modelados ao longo do tempo para criar qualquer coisa, desde um único som ou gesto complexo até uma paisagem sonora estendida ou composição.
>
> Ao fazer isso, nós principalmente adotamos um ponto de vista de compositores e procuramos criar um ambiente mutuamente apoiador, no qual, praticando, compositores podem se ajudar, tanto no design de ferramentas de software quanto no uso efetivo destas. Nós acreditamos e dependemos na crucial importância da habilidade dos compositores para escutar música, não só antes dela ter sido escrita, mas antes mesmo das formas de produção de música estarem disponíveis.
>
> **Nossa Visão de Longo Termo**
>
> Numa escala de muito longo termo, nós entrevemos os sons integrados em completude no todo da fabricação da música, velocidades de processamento e dispositivos de interface que irão possibilitar processamento em tempo real e uma nova geração de instrumentos para performance, o contínuo uso de operações *off-line* conjuntamente às de tempo real e a flexibilidade de design de *scripts* algorítmicos e semi-algorítmicos. Nós encorajamos desenvolvedores terceiros de ferramentas de softwares a levar essas ideias além, tanto no envolvimento do software CDP em novas dimensões quanto complementado-o com ferramentas de pensamento composicional. CDP (2018a, tradução nossa).

A postura de Wishart, compartilhada também pelos demais compositores do grupo do CDP, é crítica em relação ao uso indiscriminado dos chamados "efeitos sonoros" (pedais de guitarra, reverberadores, plugins, etc), sendo a ideia de que um dado efeito seria universalmente utilizável em qualquer tipo de fonte sonora, independentemente das características espectromorfológicas da mesma (WISHART, 1992; WISHART, 1994). Esse fator fica nítido tanto na documentação quanto nas funções do CDP, que são frequentemente orientadas ao trabalho com sons tipomorfologicamente específicos, por exemplo, transformações cujo resultado esperado só ocorre com sons percussivos, ou sons polifônicos, dentre outros tipos referidos.

Tradicionalmente, este software foi lançado de forma comercial e apenas recentemente, em 2014, o código fonte do CDP (cujo o cerne é escrito em C) foi publicado sob a licença pública geral menor GNU (LGPL - GNU *Lesser General Public License*). Dessa maneira, o padrão de desenvolvimento do CDP ainda não é tipicamente colaborativo e aberto, como por exemplo em ambientes como o Pd e o SuperCollider, sendo que o histórico das discussões de desenvolvimento não está acessível à comunidade. Por esse conjunto de fatores, as principais funcionalidades do CDP ainda não estão plenamente disponíveis nem em plataformas de áudio comerciais, científicas ou de desenvolvimento.

No campos dos trabalhos envolvendo áudio digital, Wishart adota o CDP como sua principal ferramenta e vale-se das possibilidades de trabalho em tempo diferido como principal metodologia de trabalho e apresentação de suas obras. O compositor aponta que seus objetivos técnicos e estéticos tendem a ser independentes do uso de alguma ferramenta específica (WISHART, 1994; WISHART, 2012b). É importante frisar que esta escolha de ferramentas não é orientada por um desejo estético de exploração das possibilidades da ferramenta em si, mas especialmente por fatores econômicos, como a impossibilidade de aquisição de hardwares de áudio de tempo real nos anos 1980; práticos, *e.g.* a dificuldade de realização de transformações sonoras nos estúdios analógicos; e históricos, pois quando Wishart propôs sua residência no IRCAM em 1986, exista uma pequena quantidade de linguagens de programação de propósito geral, e as linguagens destinadas à música ainda estavam em estágio de desenvolvimento.

Entretanto, não se pode dizer que tal escolha não se reflita em características estéticas de sua obra. Primeiramente, Wishart desenvolve as ferramentas primárias de trabalho (DSP) que em seguida utilizará no desenvolvimento das suas ferramentas secundárias de trabalho (processos de esticamento, granulação, vibrato, combinações entre eles, etc). Em segundo, o compositor-desenvolvedor escolhe tanto uma metodologia de trabalho (processamento individual de amostras gravadas) quanto as ferramentas primárias de desenvolvimento de suas ferramentas secundárias (linguagem C para desenvolver os programas do CDP) que sejam o mais próximo o possível do material, refreando as possibilidades do uso de abstrações e camadas que mapeiem seus processos à outros de alto nível ou automatização de procedimentos. Apesar das possibilidades de uso dos processos do CDP por meio de *scripts* via Bash ou linguagem Csound, Wishart parece evitar estes processos, especialmente pelo fato dos mecanismos de controle algorítmicos frequentemente impingirem uma marca sonora característica das suas variações e modulações nos materiais aos quais são aplicados, acentuando a percepção de uma fonte ou alteração mecânica em conjunto com uma redução da percepção de uma transformação mais natural e fluida.

> Trabalho *offline* e em tempo real são diferentes tanto de uma perspectiva musical quanto computacional. Em um trabalho *offline*, a principal coisa que você não precisa lidar é com ter o processo concluído em tempo específico. Um programa deve ser eficiente e rápido, eu compreendo as demandas temporais. Quando *offline*, tudo isso é simplesmente irrelevante. Além disso, quando *offline* você utiliza o tempo que achar necessário para produzir um resultado sonoro, por exemplo, um processo pode consultar um som inteiro e tomar decisões sobre o que fazer, rodar um segundo processo, e baseado neste último rodar um terceiro e assim por diante. À medida em que as máquinas se tornam rápidas e os programadores mais inteligentes (e se compositores estão felizes com sons a serem processados *offline* e reinjetados depois no sistema) então você provavelmente pode se livrar da maioria desses problemas.
>
> Entretanto, a principal diferença é estética. Se você estiver processando um evento que acontece ao vivo, você tem de aceitar qualquer coisa que vier ao microfone ou dispositivo de entrada. Independente de quão

> precisa seja a partitura, os sons que chegam serão sempre sutilmente diferentes a cada performance. Então os processos que você usa terão de trabalhar com uma grande faixa de potenciais entradas. O principal problema com música ao vivo é que você tem de ser o tipo de pessoa que gosta de ir a concertos, ou se contenta com o meio social de um concerto; e muitas pessoas não se satisfazem com isso. Isso pode ser mudado, por um lado, por educação, mas, mais significativamente, por outro lado fazendo o o mundo dos concertos mais amigável a mais grupos de pessoas e, por exemplo, realizando performances em lugares incomuns(eu já me apresentei em clubes de homens trabalhadores, escolas, etc).
>
> Trabalhando *offline* você pode trabalhar com características únicas de um som particular – podendo ser um multifônico que você gerou em uma sessão de improvisação mas que não é simplesmente "reproduzível" no futuro em alguma performance. Além disso, a ausência de *performers* em um palco pode parecer uma fraqueza, mas isso tem suas recompensas. Compare um teatro e um filme. Obviamente, uma performance ao vivo tem seus pontos fortes – um contato vívido com músicos, o teatro em um palco etc, mas você definitivamente sempre estará em um espaço de concertos. Em um evento puramente eletroacústico nós podemos criar um mundo onírico que pode ser realista, abstrato, surreal ou todas essas coisas em tempos diferentes – um teatro dos ouvidos que pode ser transportado do ambiente do aqui e agora para um mundo de sonhos. A música eletroacústica não é diferente do cinema no que se refere à sua repetibilidade, exceto que a difusão sonora, nas mãos de um habilidoso difusor, pode fazer cada performance ser única, uma interação com o público, a situação de concerto e a acústica do local. Divorciar a música de sua imediatez com o palco de concertos nos permite explorar mundos imaginários, encantados de sons, além das convenções sociais da sala de concerto. (MILANI, 2009, *apud* WISHART, tradução nossa).

Wishart ainda faz uma notável ressalva: a importância mais essencial da utilização de processos de tempo real não é para acelerar ou tornar mais responsivas algumas práticas de trabalho em estúdio – o que modernamente é plenamente satisfeito pelo aumento do *throughput* (taxa de transferência média) – mas sim para integração realmente *ao vivo* de processos computacionais e humanos.

> Eu vou fazer uma distinção entre processos de tempo real e peças de tempo real: se alguém está fazendo uma peça de mídia fixa, então a vantagem de usar processos de tempo real é que você pode explorar as saídas de um processo por meio dos gestos físicos antes de decidir o que você quer usar. Eu não tenho trabalhado assim simplesmente porque eu nunca me envolvi com programação de tempo real. A maioria dos processos que eu uso trabalha mais rápido do que o tempo real e eu não tenho nenhum problema em desenhar gráficos ou digitar tabelas, portanto eu não sinto falta disso [tempo real]. Entretanto, para uma peça em tempo real, deve-se utilizar processos em tempo real, e eu tenho trabalhado numa peça de tempo real para vozes ao longo dos últimos 2-3 anos, com variados graus de sucesso.
>
> Uma questão que você tem de perguntar nessa caso é "Porque as vozes estão sendo transformadas?". Em uma peça de estúdio, nós já estamos em um mundo imaginário onde qualquer coisa é possível. Em uma situação de tempo real, nós temos o contraste entre a presença física dos *performers* no palco e a possibilidade de desincorporação das suas vozes. Qual é o a significância teatral ou simbólica disso? Eu já tentei lidar com essa

> questão algumas vezes. (VASSILANDONAKIS, 2009, *apud* WISHART, p.19, Tradução nossa)

1.6 Linguagem de programação: instrumento musical, sistema de notação ou empecilho artístico?

No campo da arte sonora, especialmente quando se trabalha com dispositivos tecnológicos, ocorre uma simbiose entre ferramenta, objetivo estético e metodologia de trabalho. No caso de Wishart – cujo objetivo estético principal aparenta ser o de criar metamorfoses sonoras validadas pelo crivo da escuta – é de se esperar a adoção de uma metodologia de trabalho que prioriza cuidado com os pequenos detalhes – deixando de lado as automações, as indeterminações e as probabilidades – em conjunto com uma ferramenta capaz de realizar pequenos recortes, variações precisas e saídas estáveis. Consequentemente, Wishart, para suas composições eletroacústicas, costuma testar materiais e transformações durante longos períodos e, valendo-se de algoritmos de tempo diferido, opta pelo modelo de peças de mídia fixa.

À época na qual iniciava-se o desenvolvimento do CDP havia uma forte demanda para que as ferramentas de música computacional pudessem operar em tempo real, o que, em conjunto com as variadas tecnologias que surgiam, contribuía para a crescente dissipação das fronteiras entre as definições de improvisador/performer, construtor de instrumentos e compositor (WISHART, 1994, p. 7). A perspectiva de Wishart contribui para o entendimento de que as ferramentas de tempo real são essenciais para possibilitar três dimensões: (1) atividade de improvisador/performer no estúdio eletrônico ou computador; (2) possibilidade de construção de instrumentos eletrônicos ou algorítmicos ao vivo; (3) para agilizar e dar fluidez às tarefas de composição, estas últimas tradicionalmente consideradas como tempo diferido.

Atualmente, nota-se ainda na indústria do áudio uma preponderante demanda por sistemas que operem exclusivamente em tempo real. Faz-se necessária a consideração que essa preponderância reduz os potenciais de integração das três dimensões apontadas anteriormente. Determinados tipos de manipulações de dados são realizáveis somente em lógicas de tempo diferido, especialmente aquelas em que se faz uma análise do todo para então realizar os processos nas partes. A lógica de tempo diferido também permite inúmeros refinamentos cujos resultados sonoros costumam apresentam variadas qualidades em relação aos de tempo real. Desse modo, a perspectiva de Wishart mostra que ainda é necessário desenvolver integrações mais fluidas entre os sistemas de tempo real e diferido de modo a favorecer as possibilidades oriundas da hibridização das atividades de improvisação/performance, construção de instrumentos e composição.

Nesse ótica, o ambiente SuperCollider oferece diversas novas possibilidades da

referida hibridização, sobretudo devido à sua separação entre linguagem de controle (cliente) e motor de síntese (servidor). Devido à sua arquitetura de comunicação via mensagens OSC, o servidor *scsynth* pode ser controlado por diversas *linguagem de programação de propósito geral*, desde linguagens funcionais e procedurais tais como Haskell, Scheme e CommonLisp até linguagens orientadas a objeto e multiparadigma tais como Python, Java, Processing, JavaScript, Scala, Lua, Ruby, dentre outras – e até mesmo paradigmas de programação visual como Pd, Max e vvvv. Além disso, existem variações e bibliotecas dessas linguagens voltadas especificamente para o *live coding* tais como o Sonic Pi, TidalCycles, FoxDot etc. Assim, o ambiente SuperCollider, pela sua concepção mais generalista e integração com outros sistemas de áudio, permite que a fronteira entre tempo real e diferido se torne mais maleável pelo usuário.

Outro ponto a se ressaltar é que a linguagem interna ao SuperCollider (*sclang*) consegue apropriar-se seletivamente de construções presentes em linguagens de programação de uso geral (condicionais, laços, lamba, compreensão de listas) tanto para permitir a manipulação de dados mais usuais (vetores, matrizes, números complexos, *strings*, etc) quanto dos dados musicais e de áudio (MIDI, escalas, temperamento, notação rítmica proporcional, etc) – essencialmente adaptando essas construções e também propondo novas estruturas. Podemos lembrar aqui, que a *sclang* apresenta variadas possibilidades de trabalho relativas à MIDI, áudio multicanal, padrões temporais, barramentos, envelopes, escalas, padrões temporais, dentre outras, que são exclusivas desta linguagem. Diferentemente de outras linguagens de programação musicais desenvolvidas anteriormente – como o Csound, Nyquist e Common Lisp Music – a *sclang* propõe-se a difícil tarefa de contemplar bem tanto as estruturas da música tradicional quanto as das música contemporânea e eletroacústica. Se a programação pode ser vista como um processo de traduzir uma linguagem conveniente a seres humanos para uma linguagem conveniente ao computador, a *sclang* enquanto linguagem de programação musical também necessita de uma etapa extra de traduzir uma linguagem conveniente a músicos e artistas sonoros para uma linguagem de programação de propósito geral Blackwell e Collins (2005). Nesse sentido, as linguagens de programação musicais funcionam também como um sistema de notação, oferendo a interface entre a elaboração/organização das estruturas sonoras (forma, instrumentação, automações, partituras e seguidores de partituras, etc) e sua realização final (peça de mídia fixa, *live eletronics*, composição algorítmica, etc).

A linguagem nativa ao SuperCollider, *sclang*, em especial, pode assumir facetas múltiplas como sendo uma espécie de instrumento musical, uma oficina de construção de instrumentos digitais, um ambiente de composição, uma linguagem para gerar notação musical tradicional ou estendida, dentre inúmeras outras possibilidades. Há que se entender aqui que a linguagem de programação é, neste caso, estruturante do pensamento musical e sonoro, não sendo uma somente uma ponte entre as ideias do compositor e a realização

sonora final, numa associação mais livre com as ideias de Pierre Bourdieu[10], poeticamente podemos pensar as linguagens de programação musical como *estruturas estruturadas predispostas a funcionarem como estruturas estruturantes* dos objetos musicais e sonoros. Num caso tipicamente limítrofe das possibilidades computacionais, a *sclang* pode funcionar como um instrumento de improvisação sobre si mesma, controlando sons no servidor de áudio *scsynth* e outros aplicativos, prática conhecida como *live coding*.

> É impossível escrever um programa enquanto ele roda. Um programa descreve e determina um processo, portanto mudar a descrição implica um novo começo, um novo processo. É possível ainda estruturar um programa de tal forma que as partes possam ser intercambiadas dinamicamente e sua forma textual possa ser rearranjada para permitir a reescrita desses partes enquanto o processo inteiro continua. Assim, ao invés primeiramente projetar uma aplicação que tem pontos fixos de interação (parâmetros), o texto do programa em si mesmo se torna a interface principal. Ao invés de planejar adiante e prover um grande número de parâmetros, nós podemos modificar o programa a qualquer ponto. Embora em muitos campos o programa é somente de interesse da aplicação desejada, aqui o programa é um reflexo do pensamento e uma parte integral do processo de raciocínio. (WILSON; COTTLE; COLLINS, 2011, p.207, tradução nossa).

Esse tipo de linguagem mista adiciona uma camada de abstração para facilitar a aproximação das construções computacionais às estruturas musicais e sonoras, mas no seu âmago executa blocos de linguagem de propósito geral. Todavia, não se pode dizer que este tipo de abstração seja um facilitador para artistas e desenvolvedores, tampouco que as abstrações específicas do SuperCollider sejam, em geral, facilitadoras artísticas no campo da programação. Especialmente devido ao fato da metodologia de trabalho dos artistas ser fortemente relativa ao indivíduo, o tipo de ferramenta escolhida também o será.

As escolhas projetivas de um dado software irão naturalmente implicar em suas possibilidades e limitações, tais escolhas são parte fundante do mesmo pois não é possível elaborar um software totalmente abrangente e universalista. Fica mais nítido que a opção de Wishart por desenvolver seus processos em linguagem C, por um lado torna-os eficientes para execução na máquina, mas por outro dificulta a manipulação de tais funções numa lógica mais próxima da lógica musical, todavia é essa escolha que permitiu a produção da sua obra. Além disso, pela arquitetura do CDP, não é possível contemplar procedimentos em tempo real. Há também neste caso o referido processo de simbiose entre ferramenta, objetivo estético e metodologia de trabalho. Não parece o caso que Wishart tenha optado pela arquitetura do CDP porque seu objetivo estético envolvia processos de tempo diferido, mas sim que a conjunção de fatores como a ideia de metamorfose sonora, a preferência por sons gravados, a arquitetura de software em tempo diferido, o método de teste e

[10] BOURDIEU, Pierre. Esboço de uma Teoria da Prática. In: ORTIZ, Renato (Org.). A sociologia de Pierre Bourdieu, São Paulo: Editora Ática, 1994, n. 39, p. 60-61. Coleção Grandes Cientistas Sociais.

avaliação dos procedimentos sonoros, dentre outros, resultaram, em conjunto e simbiose, na produção artística e técnica de Trevor Wishart.

2 Processos de transformação sonora

> *(1) Qualquer som, seja ele qual for, pode ser o material de partida para uma composição musical; (2) As formas pelas quais esse som pode ser transformado estão limitadas somente pela imaginação do compositor; (3) A estrutura musical depende do estabelecimento de relações audíveis entre os materiais sonoros.*
>
> —Trevor Wishart, *Audible Design*

Neste capítulo apresentaremos os conceitos, princípios e as lógicas de funcionamento das transformações sonoras propostas por Wishart bem como uma contextualização de tais ideias nos cenários da computação musical e da música com computadores.

Na primeira seção fazemos uma apresentação geral do ambiente SuperCollider e seus quatro componentes básicos – *scsynth*, *supernova*, *sclang* e *scide* – bem como uma breve comparação entre sua forma de utilização e o modo de trabalho proposto pelo CDP.

Em uma segunda seção, avaliaremos as técnicas de manipulação computacional dos sons no domínio do tempo mais utilizadas no CDP: envelopamento, substituição de ataque, estratégias de reprodução, granulação, processos microssonoros e transformações de conjuntos de onda. Em uma seção seguinte, analisaremos os principais mecanismos utilizados nas transformações espectrais do CDP: a transformada de Fourier de tempo curto (STFT) e o vocoder de fase. Em ambos as seções, abordaremos os problemas, as soluções técnicas e os consequentes artifícios sonoros gerados por tais relações de compromisso.

Ao final do capítulo, faremos uma discussão conceitual e metodológica sobre os procedimentos concentrados na árdua tarefa de realizar modificações na altura e escala de tempo-frequência, tarefa essa que é abordada ampla e transversalmente nas funções do CDP bem como nas propostas estéticas e conceituais de Wishart.

* * *

2.1 Apresentação do SuperCollider

> *Para escrever uma carta para o servidor, você precisa de um papel e de um carteiro que a entregue: sclang faz ambas. (...) O poema que você escreve na linguagem SC é parafraseado pelo interpretador em prosa OSC, que o servidor mais mundano é capaz de compreender.*
>
> —Andrea Valle, *Introduction to SuperCollider*

> *As motivações para projetar o SuperCollider foram a habilidade de realizar processos sonoros que fossem diferentes a cada vez que fossem tocados, escrever peças de uma maneira que descrevesse uma faixa de possibilidades ao invés de uma entidade fixa e facilitar a improvisação ao vivo do compositor/performer.*
> —James McCartney, *Rethinking the Computer Music Language: SuperCollider*

Adaptando uma definição sucinta dada pela comunidade de usuários e desenvolvedores na sua página principal, o ambiente SuperCollider pode ser entendido como o conjunto dos seguintes elementos:

- O *SuperCollider* é um ambiente computacional para síntese de áudio e composição algorítmica; gratuito e de código aberto – originalmente desenvolvido por James McCartney em 1996 e atualmente em desenvolvimento colaborativo. Este ambiente é composto por três principais componentes: um servidor de áudio em tempo real (*scsynth*); uma linguagem de programação musical (*sclang*) e um editor para esta linguagem (*scide*) (SUPERCOLLIDER, 2019; MCCARTNEY, 2002, Tradução e adaptação nossa).

Cada um dos três componentes principais do SuperCollider pode executar uma infinidade de tarefas, sendo que cada um destes pode ser utilizado em conjunto ou separadamente. As definições gerais dadas pela comunidade ajudam a esclarecer um pouco suas funções e utilizações.

- *scsynth* – um servidor de áudio de tempo real, desenvolvido em C++, forma o cerne da plataforma. Os blocos constitutivos mais fundamentais do servidor são os *Ugens*, que representam, do lado da linguagem, os cálculos com sinais de áudio ou de controle, e são chamados pelos *Synths*: um bloco unitário de produção de áudio. O *scsynth* permite a fluida combinação de muitas conhecidas e desconhecidas técnicas de áudio, passando pelas sínteses aditiva e subtrativa, modulação em frequência (FM), síntese granular, análise-resíntese, síntese contatenativa e modelamento físico.

- *supernova* – um servidor de áudio de tempo real com paralelização automática em multiprocessadores. Como não é possível realizar paralelização automática do grafo de síntese sem realizar mudanças na semântica da tradicional biblioteca de classes do SuperCollider, dois novos conceitos são introduzidos: *grupos paralelos* e *nós-satélite*, que expõem o usuário explicitamente ao paralelismo.

- *sclang* – uma linguagem de programação interpretada, focada no trabalho com sons – igualmente capaz das variadas possibilidades oferecidas por linguagem comerciais – e também o interpretador dessa linguagem. Os códigos escritos na linguagem *sclang*

são interpretados em comandos OSC e enviados ao servidor *scsynth*(ou *supernova*), mas também há a possibilidade de se escrever os códigos diretamente na forma de mensagem OSC, com o custo da perda do alto nível das estruturas de *sclang*. Dentre os usos comuns desta linguagem estão a composição algorítmica, o sequenciamento, a elaboração de novos métodos de síntese, conexão do aplicativo com *hardware* externos (controladores MIDI, tablets, outros computadores, etc), desenvolvimento de GUIs, comunicação entre softwares (interconexão do SC com plataformas de renderização de vídeo, análise de dados, WEB, etc).

- *scide* – um ambiente de desenvolvimento integrado com interfaces gráficas para visualização de formas de onda, espectroscópio, árvore de nós, menu de ajuda e muitos outros.

  SuperCollider (2019), Valle (2016), Blechmann (2011, Tradução e adaptação nossa).

A característica mais fundamental do SuperCollider provavelmente é sua versatilidade como ambiente de trabalho de áudio digital, seja por permitir equilibradamente o uso de estruturas de tempo real e tempo diferido, seja por sua facilidade de interconexão com outros softwares e ambientes de programação, seja pela sua variedade de níveis de abstração; ainda, sua comunidade é bastante ativa e o sistema apresenta aplicações de ponta no setor de áudio. Além disso, poucas linguagem de programação destinadas a lidar com estruturas musicais e de áudio – campos que podem ser radicalmente diferentes – conseguem apresentar as mesmas flexibilidade, estabilidade e equilíbrio em ambos os campos.

A linguagem *sclang* apresenta uma abstração extremamente útil para o trabalho com arte sonora: *Patterns*, que são um modelo de *streams* na qual um dado evento sonoro definível (evento glissando, evento explosão curta de mensagens numéricas, etc) ou padrão (e.g. evento nota) é executado toda vez que um controlador chama o próximo evento. Os *Patterns* são especialmente úteis pois podem apresentam uma série de comportamentos tipicamente musicais (e.g. sequências, movimentos brownianos, repetições, etc) e também porque sua escrita é profundamente concisa (os eventos são escritos por meio de pares de chaves e valores), conforme podemos analisar no padrão abaixo.

```
1 SynthDef(\senoide, { |out, freq = 800, sustain = 1, amp = 0.1|
2         var envelope, sinal;
3         sinal = SinOsc.ar(freq); // Define a senoide como um sinal
4         envelope = EnvGen.kr(Env.perc, levelScale: amp, doneAction: Done.
              freeSelf); //gera um envelope percussivo
5         sinal = sinal * envelope;
6     Out.ar(out, sinal)
7 }).add;
8
```

```
9  //Pbind é um padrão que une os pares chave–valor ( \instrument <–> \senoide
       , \freq <–> Pwhite(400, 700, inf) )
10 Pbind(
11         \instrument, \senoide,
12         // Frequências aleatórias entre 400Hz e 700Hz
13         \freq, Pwhite(400, 700, inf),
14         //Durações sequenciais de 1/18 s, 5/18 s, 3/18 s ...
15         \dur, Pseq([1,5,3,7,8,2,9,2,3,4]/18, inf)
16 ).play;
```

Código 2.1 – Sons senoidais acionados por Patterns no SuperCollider

Como o SuperCollider apresenta integração entre linguagem de controle de estruturas musicais com ferramentas tradicionais das linguagem de programação comercial, é possível re-implementar várias das funções do CDP na própria *sclang*, sem ser preciso adentrar em níveis mais baixos (como é o caso dos *UGens* que são programados em C++ juntamente com a arquitetura do servidor). A grande vantagem nesse é ponto é que é possível integrar diversas funções do CDP em um só *Pattern* do SuperCollider, além do fato de que a criação de novas funções correlatas, quando necessário, também se torna proporcionalmente mais simples.

A arquitetura cliente-servidor separa, de um lado, os processos de controle e, de outro, os cálculos de áudio. Isso permite estabilidade – caso o interpretador trave, o servidor continuará funcionando, e vice-versa; modularidade – diferentes softwares ou diferentes usuários(cada qual com seu *sclang*) podem controlar um mesmo servidor de áudio *scsynth*; e flexibilidade – inúmeras formas de enviar mensagens OSC ao servidor podem ser utilizadas, algumas das existentes são via linguagens tipicamente comerciais como Python, Haskell, Processing, Lua, Ruby (via a linguagem Sonic Pi focada em *live coding*), dentre outras, e também o uso da linguagem *sclang* via editores de texto como o Emacs e VIM.

Por ser prioritariamente um ambiente de tempo real e por apresentar uma arquitetura cliente-servidor, a renderização de áudio em tempo diferido no SuperCollider não apresenta todas as funcionalidade dos procedimentos de tempo real. Além disso, procedimentos que apresentam custo computacional mais alto, como é o caso da FFT e do vocoder de fase, tendem a serem implementados sob a forma de UGens mais fechados, cuja interação com os mecanismos do servidor é mais limitada.

A linguagem *sclang* nos permite executar comandos Bash em um terminal, o que pode ser utilizado para acessar o CDP em tempo diferido, conforme Código 2.2, ou mesmo podemos utilizar uma linguagem intermediária como o Python para criar *scripts* que executam longas e iteradas ações no Terminal, por exemplo controlando CDP. Uma vez que os processos do CDP podem ser acessados pelo SuperCollider por meio de *scripting* (tanto da *sclang* quando de outras linguagens) e/ou implementações transpostas, quando

a arquitetura do SuperCollider não permitir uma transposição de processos idênticos aos do CDP ou quando não tivermos restrições de tempo real, optaremos pelo *scripting* ou processos mistos; quando a arquitetura do CDP desfavorecer processos algorítmicos de transformação sonora ou quando o trabalho em tempo real for imprescindível, optaremos pelas transposições ou adaptações dos programas do CDP no ambiente SuperCollider.

```
1  // Omite um em cada quatro conjuntos de onda
2  "distort omit /Users/nomedeusuario/Desktop/drone.wav /Users/nomedeusuario/
       Desktop/mdf_rad.wav 1 4".runInTerminal;
3
4  // Análise espectral com janela de 1024 pontos
5  "pvoc anal 1 /Users/nomedeusuario/Desktop/drone.wav /Users/nomedeusuario/
       Desktop/trumpet.ana -c1024".runInTerminal;
6
7  // Processamento - randomização dos canais espectrais
8  "glisten glisten /Users/nomedeusuario/Desktop/trumpet.ana /Users/
       nomedeusuario/Desktop/trumpet_glis.ana 4 250 -p3 -d0.7 -v0.25".
       runInTerminal;
9
10 // Ressíntese
11 "pvoc synth /Users/nomedeusuario/Desktop/trumpet_glis.ana /Users/
       nomedeusuario/Desktop/trumpet_glis.wav".runInTerminal;
```

Código 2.2 – Executando o CDP a partir do SuperCollider por meio de comandos Bash no Terminal

Faz-se necessário ressaltar como a perspectiva generalista do SuperCollider – que tenta absorver a maior quantidade de pontos das linguagens de programação de propósito geral em direção à criação de uma linguagem de programação musical generalista (MCCARTNEY, 2002) – admite utilizações heterodoxas e divergentes. Wilson, Cottle e Collins (2011) evidenciam como essa relação entre aumento de graus de liberdade e necessidades computacionais de restrição é benéfica no campo da música computacional.

> O SuperCollider, assim como as camadas de uma boneca russa matriosca, é um sistema de sistemas, contendo um número sempre crescente de subsistemas especializados que variam de sistemas de propósito geral a sistemas altamente idiossincráticos. Extensões a uma linguagem acrescentam tanto novas possibilidades quanto novas restrições. (...) De fato, uma composição no SuperCollider pode ser entendida com uma obra de arte dentro de uma obra de arte, ou como uma linguagem dentro de uma linguagem.
>
> Os benefícios que esses subsistemas trazem superam as dificuldades iniciais de compreensão da suas maneiras de operação. Em programação (assim como em atividades artísticas no geral), a necessidade de restrições é ao menos tão importante quanto o desejo por recursos – limitações são em si mesmas recursos que requerem implementações. Essa consideração é revelada especialmente na turva distinção entre uma ferramenta e seu resultado , um aplicativo e uma obra de arte ou um modelo. Quanto mais geral, generativa ou aleatória uma obra se torna, mais ela torna-se

> um ambiente, um sistema de axiomas ou simplesmente um material para novas obras de arte dentro do seu próprio mundo. (WILSON; COTTLE; COLLINS, 2011, p. 635, tradução nossa).

Essa capacidade de interface e aninhamento de sistemas dentro de sistemas por si só constitui um grande e profícuo tema de estudos, especialmente em vista de quão fértil o SuperCollider é neste campo. Esta dissertação de mestrado é, portanto, um dos possíveis recortes de utilização do SuperCollider: concentrando-se na realização de transformações sonoras por meio dos diversos subsistemas existentes (*Quarks* de conjuntos-de-onda, *plugins* externos de vocoder de fase, execução de *scripts*, etc), utilizamos um dialeto da linguagem SuperCollider – os *Patterns* – para elaborar uma série de pequenos estudos relativos às proposições técnicas-estéticas abordadas por Trevor Wishart.

2.2 Transformações temporais

2.2.1 Envelopamento e estratégias de ataque-continuação

O envelope de um som é comumente caracterizado pela envoltória da amplitude, ou seja, pela curva de variação lenta que envolve uma dada forma de onda. Para se ter referências de análise ou para facilitar a produção de tais curvas computacionalmente, divide-se, em geral, esse envelope em quatro seções básicas: ataque, decaimento, sustentação e relaxamento.

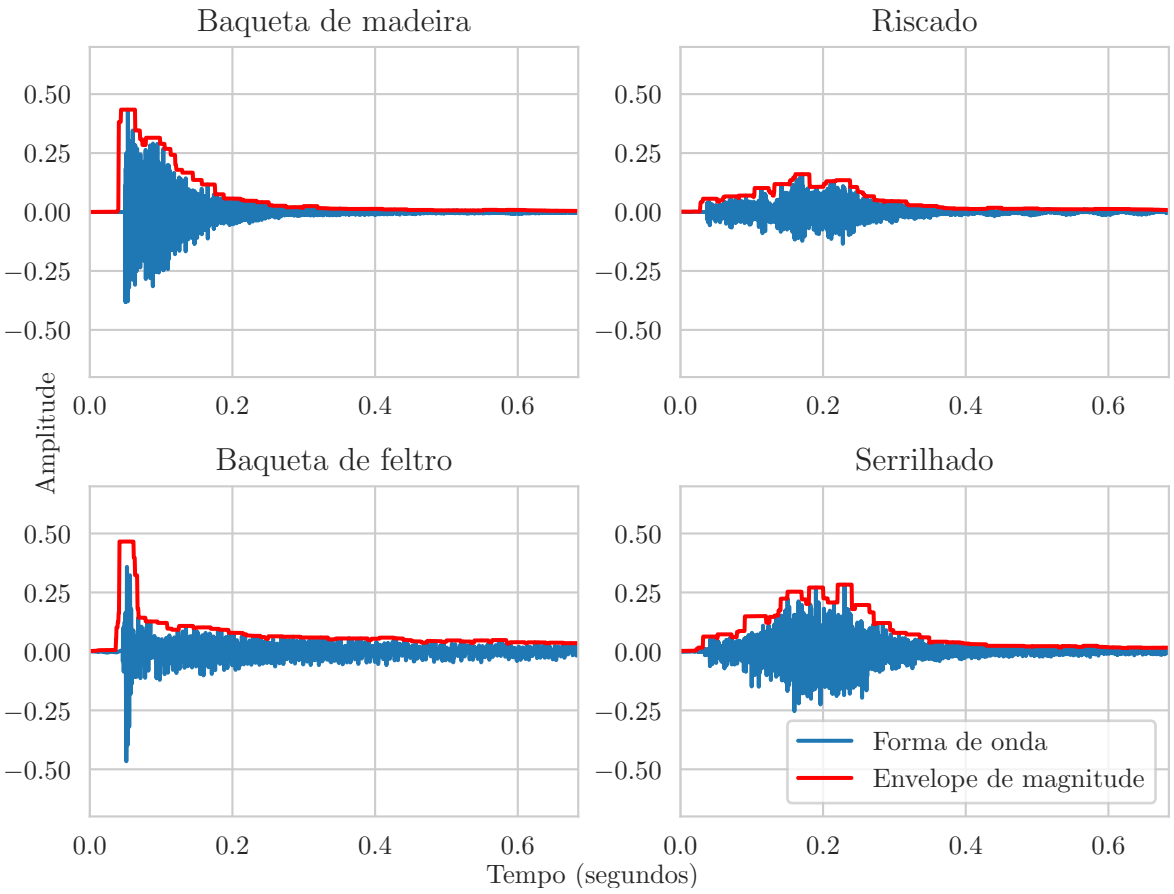
Existem inúmeras formas de se extrair o envelope temporal de um dado sinal: utilização de filtros passa-baixa, aproximando-se pelo cálculo de energia do sinal ou da raiz média quadrática (RMS), por meio da retificação de onda, através da transformada de Hilbert, dentre outros. A maneira mais simples e intuitiva aplica uma janela deslizante ao longo do sinal e extrai um valor único desta janela, geralmente o ponto de máximo. Assim, o tamanho da janela irá determinar quão suave será a curva do envelope extraído: janelas de tamanho pequeno avaliarão o sinal de maneira mais precisa e serão sensíveis à oscilações mais rápidas no sinal, podendo capturar as oscilações responsáveis pela altura. Contrariamente, as janelas de maior tamanho avaliarão o sinal de maneira mais grosseira e tenderão a capturar somente oscilações lentas, como as relativas à dinâmica e *allure*[1] do objeto sonoro.

O ataque[2] – primeira seção do envelope temporal – em particular é importante

[1] *Allure*, algumas vezes traduzida como *andadura* ou *marcha*, refere-se às oscilações nas características de um som sustentado, como, por exemplo as flutuações do tremolo e vibrato (SCHAEFFER, 2017; CHION, 2009)[p. 443-446; p. 178-183]. Como essa definição não é operacional, será utilizado o termo modulação quando forem abordados os parâmetros computacionais que geram tais flutuações, mantendo, assim, o termo *allure* para se referir à percepção auditiva de tais flutuações.

[2] Em Wishart (1994) o ataque é chamado de *onset*. Seguindo a nomeclatura mais utilizada atualmente, iremos utilizar a palavra *onset* para referir-se ao instante de início de um som, e não à sua porção inicial (ataque).

Figura 4 – Envelopes temporais de quatro sons de sinos



Fonte: elaborada pelo autor, código baseado em Müller e Zalkow (2019)

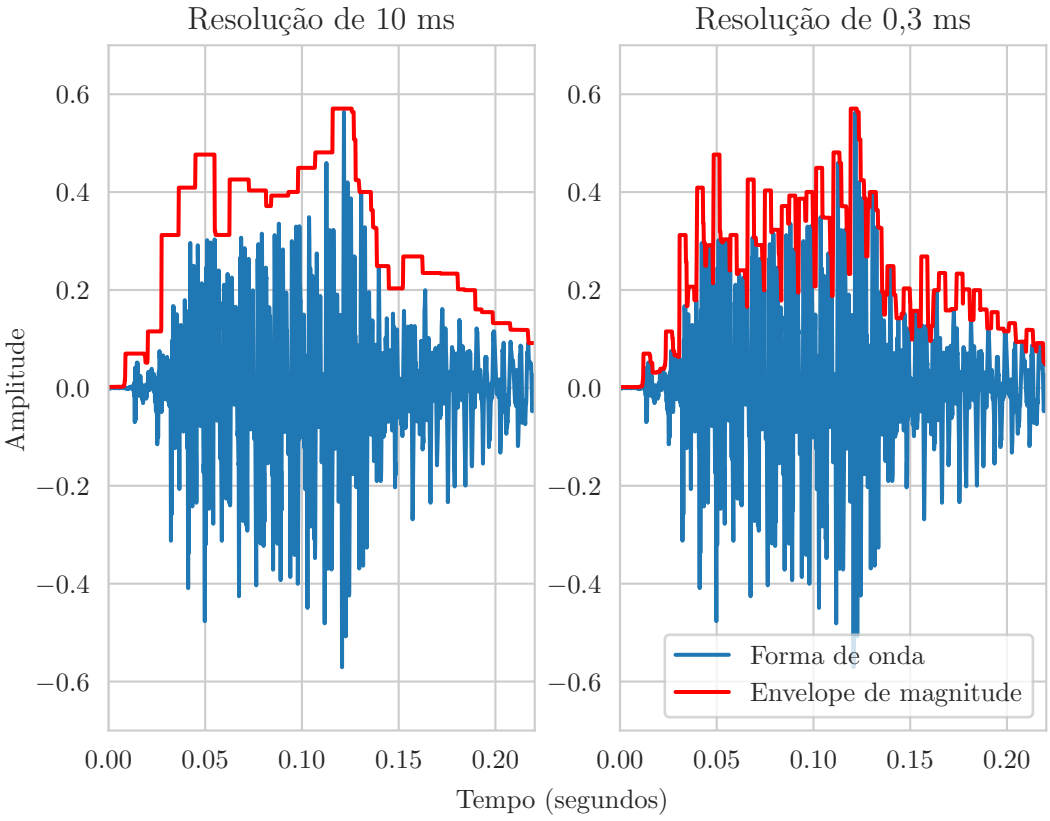
definidor e caracterizador do timbre. Os trabalhos de Pierre Schaeffer e do GRM exploraram variadas possibilidades criativas de modificação do ataque de sons gravados mostrando como alterações no ataque de um som gravado podem levar a consequências extremas, tais como dificultar e confundir a atribuição de fonte sonora à uma dada gravação (SCHAEFFER et al., 1967; SCHAEFFER, 2017). Um sino, como na Figura 4, pode soar de formas bastantes distintas alterando-se a maneira como é percutido – o que transforma principalmente as características do seu ataque.

Áudio 2.1 (2-1_4-sinos-stereo.wav). *Arquivo de áudio disponível em <http://www.musica.ufmg.br/lapis/?p=1141>.* Quatro modos de produção sonora de um mesmo sino, encadeadas na seguinte ordem:

1. percutido com baqueta rígida;
2. percutido com baqueta de feltro;
3. riscado com metal;
4. riscado com metal serrilhado.

Como o envelope está correlacionado com as oscilações mais lentas da forma de onda, a definição da resolução de análise de extração de envelope expõe as fronteiras entre as oscilações percebidas como altura e timbre, as oscilações percebidas como grãos e as oscilações percebidas como eventos temporalmente distintos. A Figura 5 ilustra tais situações limítrofes.

Figura 5 – Comparação entre duas diferentes resoluções de extração de envelope



Fonte: elaborada pelo autor, código baseado em Müller e Zalkow (2019)

Envelopamento é o processo de extração do envelope temporal de um dado som para, em seguida, impor tal curva ao sinal de outro som. Esse processo não é trivial de ser realizado analogicamente e, tradicionalmente, o custo computacional de se gerar envelopes digitais de alta resolução sempre foi alto. Atualmente, a facilidade de implementação desse procedimento permite, por exemplo, impor envelopes complexos – extraídos de sons gravados ou gerados algoritmicamente – em sons sintetizados.

Wishart (1994) apresenta diversas estratégias de alteração da *continuação* (decaimento, sustentação e relaxamento, ou seja, todas as porções após o ataque) dos objetos sonoros, que chamamos aqui de estratégias de ataque-continuação. Estas técnicas estão ligadas às técnicas desenvolvidas no GRM de *substituição de ataque* (GORNE, 2017). Basicamente, as estratégias de ataque-continuação são uma forma de montagem sonora

consistindo em emendar o ataque de um som à continuação de outro, valendo-se, geralmente, de tempos rápidos de ataque e transição entre os sons para promover a percepção da ocorrência de um único objeto sonoro.

Uma subcategoria de tais técnicas é denominada de *continuação construída*, nas quais o ataque de um dado som é executado normalmente, mas a sua continuação é transformada – como por exemplo aplicação de forte reverberação, modificação da escala temporal das porções finais do som e modulação de tais porções (tremolo, vibrato, filtragem variável, etc) (WISHART, 1994, p. 48-54). Esta técnica tende funcionar quando o ataque é rápido e a *continuação* é mais lenta – o que colabora para a distinção perceptiva entre as duas porções, evitando que alterações nas continuação sejam percebidas como *fade in* ou objetos sonoros segregados.

2.2.2 Estratégias de reprodução

A modificação da velocidade de reprodução de uma dada gravação é uma técnica clássica de transformação sonora que, quando realizada de maneira pura e direta, conjuga, ao menos, quatro fortes efeitos: transposição de altura e de frequências, alteração da duração dos eventos, alteração da largura de faixa do espectro e alteração das formantes e do envelope espectral. Isto significa que sempre desejarmos, por exemplo, alterar a altura utilizando este método, os outros três efeitos irão ocorrer conjuntamente. O desacomplamento de tais efeitos só é possível utilizando-se algum tipo de processamento do fluxo de dados da reprodução, como será discutido ao longo deste texto.

Sob a ótica das possibilidades criativas da música com computadores, um arquivo de áudio pode ser reproduzido de inúmeras maneiras: continuamente, intermitentemente, aceleradamente, com movimentos de ziz-zag, brownianos, por meio de pequenos ou grandes saltos, com acelerações e desacelerações fractais, senoidalmente, etc. As taxas de modificação e o grau de continuidade destas leituras, em especial, irão determinar quais serão os efeitos perceptivos de tais operações.

Computacionalmente, devido à ilimitada quantidade de estratégias de reprodução possíveis, bem como de seus efeitos, torna-se impraticável realizar uma classificação e categorização exaustiva. Portanto, na subseção 3.1.2 iremos nos concentrar em algumas possibilidades algorítmicas do SuperCollider que possam estender as possibilidades já implementadas no CDP.

2.2.3 Granulação e microssom

Sempre que se trabalha com pequenos segmentos de áudio, de grandeza menor que um objeto sonoro claramente definido, adentramos no campo dos chamados microsons. Isto posto, qualquer operação de áudio digital pode ser enquadrada nesta categoria, uma

vez que trabalhamos sempre com amostras discretas, quase sempre processadas em blocos. Roads (2001) apresenta uma exaustiva exploração das teorias, possibilidades criativas, classificações e limitações das abordagens de microssom.

> Um grão de som é um curto evento microacústico, com a duração próxima do limiar humano da percepção auditiva, tipicamente entre um milésimo de segundo e um décimo de segundo (de 1 ms a 100 ms). Cada grão contém uma forma de onda moldada por um envelope de amplitude. (ROADS, 2001, p. 86, tradução nossa).
>
> Em geral, a síntese inicia-se com um material cru na escala de tempo micro: um único período de forma de onda, um impulso, um estampido de ruído uniformemente distribuído. A transformação, ao contrário, usualmente começa com um sinal relativamente longo de samples de áudio (maior do que 100 ms), na escala dos objetos sonoros. (ROADS, 2001, p. 86, tradução nossa).

Um som formado por grãos – sejam eles sobrepostos ou espaçados – tende a implicar em algum tipo de descontinuidade recorrente. Em dado limite, estes grãos podem se tornar tão espaçados que passam a ser percebidos como eventos distintos; no limite oposto, se os grãos apresentarem um espaçamento muito pequeno, o ouvido humano pode perceber este som como um ruído constante ou como um som harmônico de altura constante[3]. Além disso, emerge em tais sons uma sensação auditiva específica, parecida com as flutuações de amplitude porém ligeiramente diferente, que é chamada, de forma generalizada, de *granularidade* ou *rugosidade*. No mundo físico existem duas formas básicas de produção de sons granulares: a iteração, ou seja, repetição rápida de perturbações de curta duração (percussões rufando, língua batendo rapidamente no palato, um carro atravessando uma pista com pequenas lombadas igualmente espaçadas); e o atrito entre materiais (fricção do arco do violino produzindo uma nota, borrachas esfregadas entre si, unhas arranhado uma superfície, etc), que também é uma forma de iteração, porém microscópica. Assim, é possível que um som naturalmente apresente características granulares, mas também é possível granular um som contínuo por meio de recursos computacionais ou mesmo sintetizar um som a partir da repetição rápida de grãos constituídos por formas de onda sintéticas.

Como o sinal de áudio tende a apresentar um fluxo oscilatório perene e de alto nível de redundância, a divisão deste em pequenos grãos tende a alterar esta característica básica de continuidade, introduzindo artefatos. As principais estratégias adotadas para manejar este problema são: (1) admitir estas quebras e separar temporalmente os grãos; (2) encontrar pontos de descontinuidade e/ou estabilidade que possam funcionar como elementos naturais de transição; (3) selecionar grãos que se sobrepõem, visando preservar minimamente o nível de redundância e continuidade entre estes. Estas três estratégias

[3] Uma avaliação minuciosa dos aspectos auditivos, físicos e conceituais das características iterativas e contínuas dos sons é feita em Schaeffer (2017)

para trabalhar ondas contínuas como sendo constituídas de partículas discretas perpassam os diversos programas do CDP, curiosamente apontando como as aspirações de *contínuo sonoro* de Wishart para os parâmetros tradicionais da música (altura, duração, intensidade e timbre) são, em termos práticos, realizáveis por sistemas discretos e quantizados em treliças.

> Variações descontínuas na escala microtemporal da música derreteram as congeladas abstrações da teoria musical tradicional, tais como altura, timbre instrumental e marcação dinâmica. Mesmo as noções sagradas de continuidade de uma nota e simultaneidade se revelaram ilusórias. A escala microtemporal derrete essas categorias congeladas em morfologias de constante evolução. (ROADS, 2001, p. 330, tradução nossa) .

Em especial, três técnicas desenvolvidas por Wishart no CDP produzem nos sons características claramente iterativas: *granulação*, *brassagem* [4] e transformações de *conjuntos de onda*. Wishart – diferentemente da nomeclatura mais amplamente adotada, que é a proposta por Roads (2001) – denomina por granulação somente aqueles processos de transformação que escolhem como grãos fragmentos de ondas localizados entre silêncios[5]. Os processos microsonoros que permitem a seleção de grãos em pontos quaisquer da forma de onda, com possibilidade de sobreposição e aplicação de função de janela, são denominados *brassagem* e *sausage*[6], quando se executa uma *brassagem* de vários arquivos de áudio simultaneamente. As transformações de *conjuntos-de-onda* são o caso específico de seleção de grãos nos pontos da onda que começam e terminam em zero, explorados mais detalhadamente na próxima seção.

Há ainda no CDP um grupo de funções para analisar e transformar grãos de altura síncrona (*psow*), em especial a função de onda de formante (FOF)[7] (CDP, 2019a), e um grupo de funções para criar texturas (*texture*). Essas funções fogem do escopo deste trabalho, portanto não serão abordadas nos próximos capítulos.

A premissa básica dos algoritmos microsonoros permite extrair retratos fixos de um filme de movimentos redundantemente oscilatórios e continuamente modulados. Esse fator, por si só, permite não somente a possibilidade de se realizar análises e ressínteses arbitrárias nestes fragmentos, mas permite transformações sonoras que vão muito além da consequência direta de moldar os aspectos iterativos dos sons. É possível introduzir características

[4] A palavra brassagem refere-se ao processo de mistura de grãos para a fabricação de cerveja. É provável que o CDP tenha adotado tal termo devido à utilização do mesmo pelo compositor Bernard Parmegiani, membro do GRM, na experimentação de processos de micromontagem com fita magnética no final dos anos 1970 (ZOLZER, 2011, p. 210).

[5] No CDP, estes pontos de silêncios são momentos em que a onda fica estacionada em zero ou abaixo de um dado limiar

[6] Referência ao processo de fabricação de salsicha.

[7] Função de onda de formante (FOF) são grãos senoidais com ataques bruscos ou suaves e decaimento quase exponencial, desenvolvidos do Xavier Rodet com propósito de criar um sistema de síntese vocal baseada em formantes.

idiossincráticas tais como distorção, modificações da escala temporal (esticamento e encurtamento temporal ou de altura), reverberação, filtragem, imposição de formantes, etc que outros métodos de manipulação não conseguem alcançar.

> Um único grão serve como bloco de construção de objetos sonoros. Por meio da combinação de milhares de grãos ao longo do tempo, nós podemos criar atmosferas sônicas animadas. O grão é uma apta representação do som musical porque ele captura duas dimensões percetivas: informações do domínio temporal (tempo de início, duração, forma do envelope) e informações do domínio da frequência (a altura de uma forma de onda dentro de um grão e o espectro de um grão). Isso se opõe às representações baseadas em amostragem que não capturam informação do domínio da frequência, e métodos abstrados de Fourier, que somente levam em conta o domínio da frequência. (ROADS, 2001, p. 87, tradução nossa) .

2.2.4 Conjuntos de onda

Os *conjuntos de ondas* (*wavesets*)[8] foram propostos em Wishart (1994) e são definidos como os fragmentos de onda que se encontram dentro de três cruzamentos de zero. Para casos simples, como senoides, os conjuntos de onda equivalem à forma de onda periódica em si. Todavia, para sons complexos a forma de onda contida em tais conjuntos costuma abarcar diversas outras oscilações e/ou componentes contínuas, conforme pode ser visto na Figura 6.

A definição dos conjuntos de ondas é estratégica: como são escolhidas duas porções de uma onda entre cruzamentos de zeros, existe uma forte tendência a escolher ciclos de sinais opostos e evita-se operar em pontos de transições bruscas. Isso evita transformações que gerem sinais contínuos e também minimiza o aparecimento de cliques em transformações.

A taxa de cruzamentos de zero de um sinal é também um indicativo de estabilização de altura no sinal: sinais ruidosos tendem a apresentam oscilações muito rápidas, logo, alta taxa de cruzamento de zero, e sinais tônicos tendem a apresentam taxas de cruzamento de zero com valores menores e mais estáveis. Assim, é comum definir um limiar de cruzamento de zero para evitar que pequenos ruídos de fundo, que são pouco perceptíveis auditivamente em gravações, influenciem excessivamente nas operações de *conjuntos-de-onda* .

Nas literaturas tradicionais de processamento de sinais de áudio e música com computadores, tais como Reiss e McPherson (2014), Zolzer (2011), Park (2010), sempre que uma operação processa um sinal em grandes blocos (passos de processamento que não sejam de amostra por amostra de áudio) ou sempre que o processamento atue em pontos específicos da forma de onda, as chamadas operações não-lineares, é dito que o sinal sofreu *distorção* – ou mais especificamente *distorção harmônica* [9]. Essa concepção

[8] Ao longo da documentação do CDP estes também são chamados de *pseudo ciclos de onda*

[9] A não-linearidade, em processamento de sinais, introduz novos componentes espectrais no sinal e gera sons perceptivamente caracterizados como ásperos, ruidosos, quentes, duros, etc.

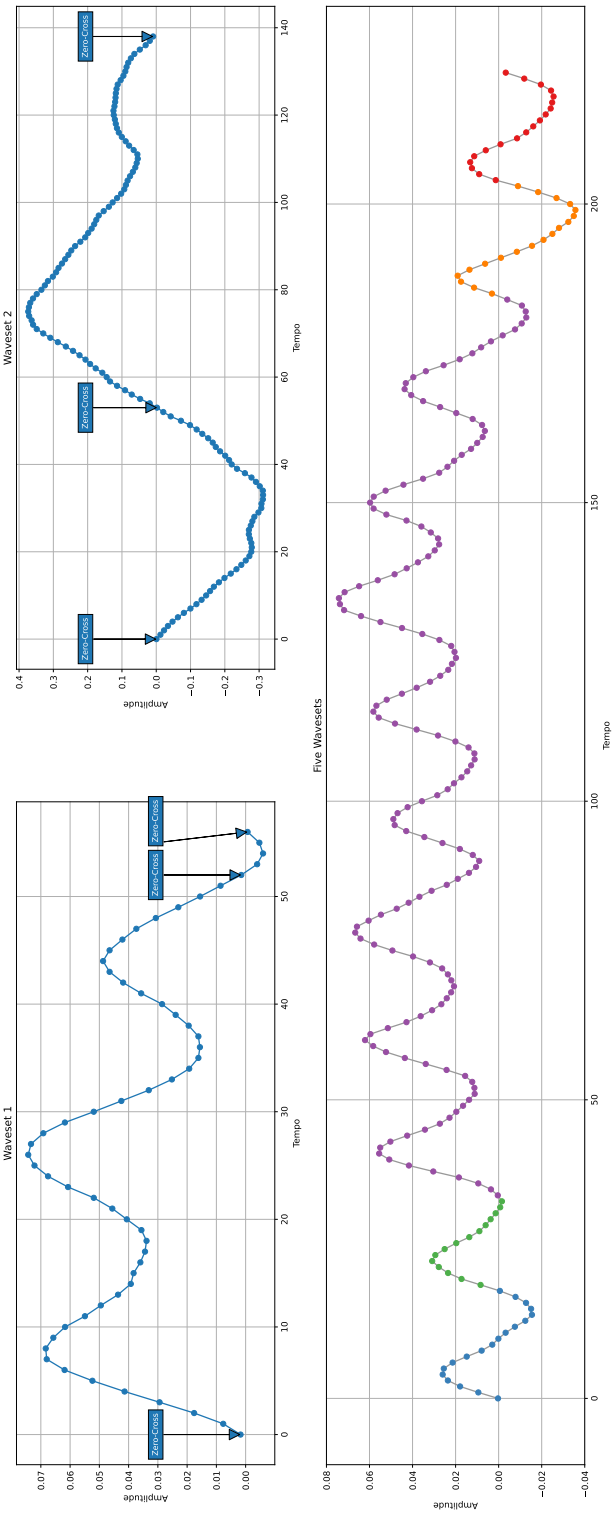


Figura 6 – Conjutos-de-onda – conceitos básicos

Fonte: próprio autor

parte de uma premissa de construção de dispositivos eletrônicos nos quais a distorção ou era um problema a ser evitado – como é o caso de projeto de componentes de áudio de alta fidelidade – ou era um efeito a ser mantido em moldes bastante delimitados – como é o caso das distorções *fuzz* e *overdrive* de guitarra.

No CDP, as transformações de *conjunto-de-onda* são abrangentemente enquadradas na categoria de *distorção*, possivelmente pelo fato de se considerar que transformações em *conjuntos-de-ondas* inevitavelmente introduzirão não-linearidades (ENDRICH; FRASER; DOBSON, 2015). Nas transformações mais radicais desse grupo de fato sobressaltam características sonoras de distorção harmônica, todavia diversas outras características frequentemente predominam, como granulação, modificação de reprodução, modificação de escala temporal, etc. Além disso, sonoridades similares às produzidas pela distorção harmônica – por exemplo redução da taxa de amostragem e/ou quantização, compressão de dados, perdas totais de sinal (*dropouts*), etc – que também são consequências comuns das transformações de conjuntos de onda, costumam ser classificadas tão somente como *distorção*.

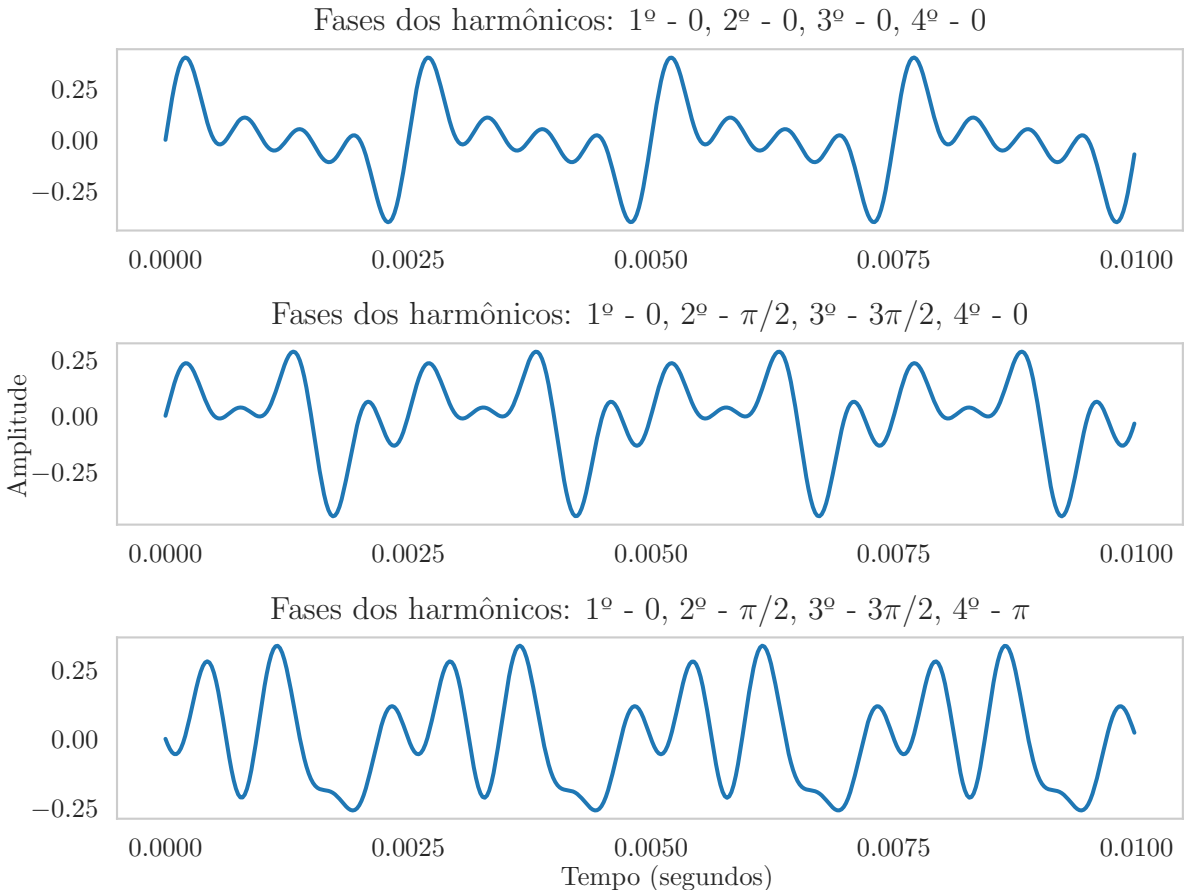
É importante ressaltar ainda que as transformações de *conjunto-de-onda* estão mais fortemente ligadas a aspectos visuais e gráficos da forma de onda do que a aspectos da física ou processamento de sinais sonoros por si só. Este frequentemente é o caso quando operamos com sons cujo timbre é idêntico, mas as formas de onda são diferentes. Como exemplo, sons que apresentam a mesma quantidade, ordem e magnitude de harmônicos apresentarão, auditivamente um mesmo timbre. Caso a relação de fase entre estes harmônicos for alterada, o timbre permanence idêntico, mas a forma gráfica da onda se altera, podendo também alterar os *conjuntos-de-onda*, conforme Figura 7, Áudio 2.2 e Áudio 2.3.

Áudio 2.2 (400hz-harm-fora-fase.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.* Sequência de sons compostos por uma fundamental de 400Hz e seus quatro primeiros harmônicos(800Hz, 1200Hz e 1600Hz). No primeiro as fases dos harmônicos, em relação à fundamental, são, respectivamente, 0, 0, 0 e 0; 0, no segundo som 0, $\pi/2$, $3\pi/2$ e 0 ; no terceiro som $\pi/2$, $3\pi/2$ e π.

Áudio 2.3 (2-3_400hz-harm-fora-fase-distort-fractal.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.* Cada som do Áudio 2.2 é apresentado seguido de sua transformação fractal de conjunto-de-onda (ver Tabela 3.1.3) com parâmetro de escala igual 4 (quatro fractais dentro de um conjunto-de-onda).

A presença de inarmonicidade também altera a forma de onda de um dado som, especialmente se os desvios em relação ao múltiplos inteiros da fundamental forem pequenos, o que causa batimentos, logo teremos conjuntos de onda que evoluem temporalmente

Figura 7 – Sons com mesma fundamental e harmônicos defasados do Áudio 2.2

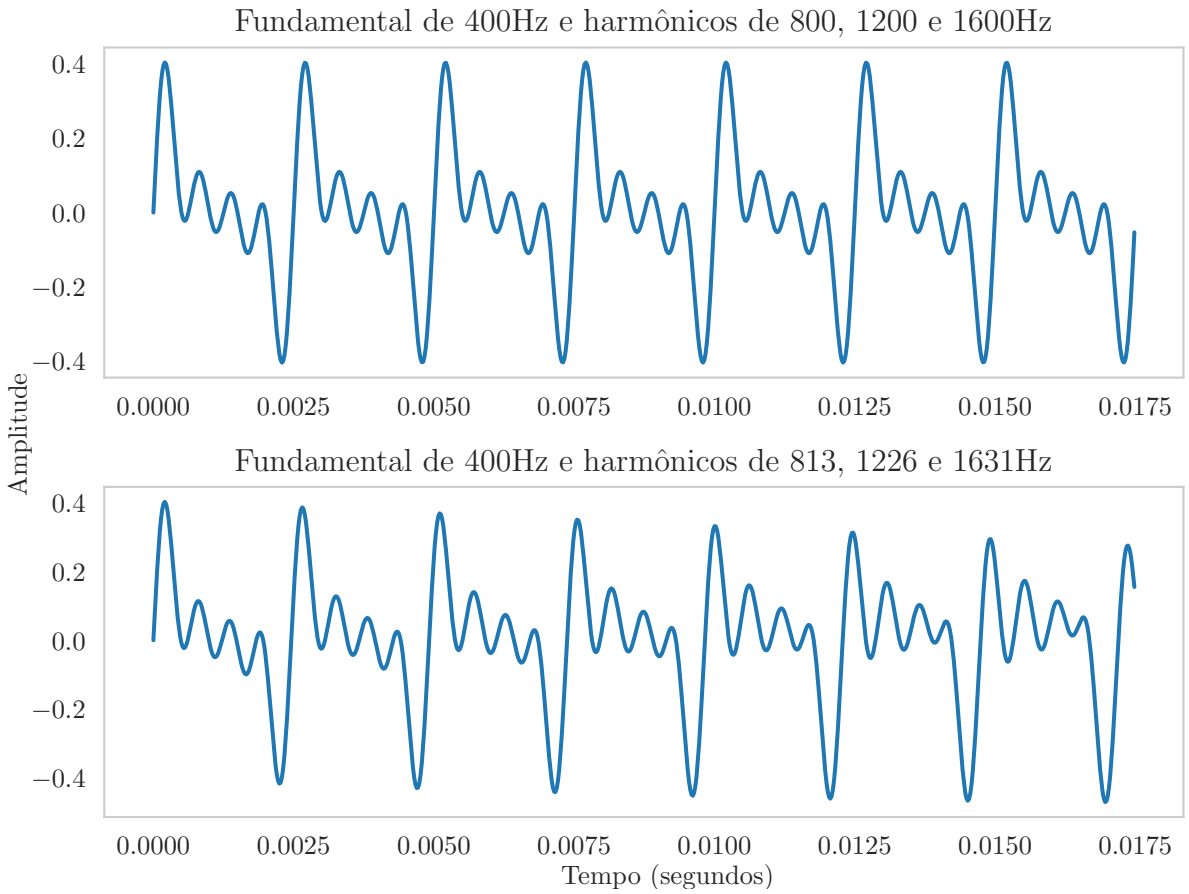


Fonte: elaborada pelo autor

em relação à forma de onda, conforme é possível visualizar na Figura 8. A presença da inarmonicidade costuma levar a diferenças na percepção de altura e afinação, conforme é possível perceber no Áudio 2.4, assim, transformações de conjuntos de onda em sons inarmônicos pode levar a efeitos combinados (altura, batimentos e *allure*), conforme Áudio 2.5. Neste casos, quando temos diferenças de inarmonicidades que são pouco nítidas aos ouvido humano, a aplicação de transformações de conjuntos de onda pode gerar resultados nitidamente distintos.

Áudio 2.4 (2-4_harm-inharm-400.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.* Sequência de sons para comparação entre harmonicidade completa e com a presença de inarmonicidade. O primeiro composto de uma fundamental de 400Hz e seus quatro primeiros harmônicos(800Hz, 1200Hz e 1600Hz), todos em fase, e o segundo composto de uma fundamental de 400Hz e seus quatro primeiros inarmônicos (813Hz, 1226Hz e 1631Hz), também em fase.

Figura 8 – Segmentos da forma de onda de um som harmônico e um som inarmônico, presentes no Áudio 2.4



Fonte: elaborada pelo autor

Áudio 2.5 (2-5_harm-inharm-distort-multiply.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.* Cada do som do Áudio 2.4 seguido de sua transformação do tipo multiplicação de conjunto-de-onda (*Distort Multiply*).

Um dos artefatos mais frequentemente encontrados na transformações de conjuntos de onda avaliadas neste trabalho é uma sonoridade similar à de objetos de borracha sendo atritados ou de sons de líquidos. Ainda não encontramos explicação na literatura para este fenômeno, tampouco podemos precisar qual operação computacional causa precisamente este artefato. Todavia, nossa principal hipótese é de que as oscilações de sons ruidosos, quando repetidas de formas específicas, tendem manifestar tal tipo de efeito. No Áudio 2.6 mostramos como aparentemente a repetição de conjuntos de onda de um ruído rosa pode levar a tal tipo de sonoridade, porém é difícil afirmar porque tal sonoridade aparece em diversos outros procedimentos, como, por exemplo, na inversão de conjuntos de onda.

Áudio 2.6 (2-6_artefatos_borracha_conj_ond.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Repetição de conjuntos de onda – cada três grupos de conjuntos de onda é repetido dezesseis vezes – de um ruído rosa intercalado com os seguintes sons: excerto do Áudio 3.17 com repetições de conjuntos de onda em sons vocálicos; excerto do Áudio 3.12 com inversão de conjunto de onda de uma nota C3 tocada em um trompete e em um piano; colagem e superposição de vários sons de borracha retirados de <https://freesound.org/s/88502/>, <https://freesound.org/s/429388/> e <https://freesound.org/s/88502/>.

Como último ponto, ressalta-se que sinais que alternem entre seções mais claramente tônicas e mais claramente ruidosas ao longo do tempo, irão apresentar diferentes características perceptivas em cada uma destas seções para uma mesma transformação de *conjunto-de-onda*. Em sons vocais, especialmente os de fala, tal fenômeno ocorre frequentemente, pois uma das características centrais dos sinais de fala humana é a alternância entre vogais (mais claramente tônicas) e consoantes (mais claramente ruidosas). Este ponto será explorado em detalhes no Capítulo 3.

2.3 Transformações espectrais

2.3.1 A transformada de Fourier de tempo curto (STFT) e o vocoder de fase

2.3.1.1 STFT: definição

A transformada de Fourier[10] é um procedimento matemático reversível que converte um sinal temporal em sua representação no domínio da frequência; de outra forma, um sinal temporal qualquer – periódico ou não, determinístico ou não – é convertido em um conjunto de senoides com frequências e fases específicas. Computacionalmente[11], o principal uso da transformada de Fourier em sinais musicais concentra-se em finalidades analíticas, revelando detalhes e estruturas que a representação temporal frequentemente encobre. Contudo, no campo da criação, a análise de Fourier de um sinal é apenas uma das diversas etapas possíveis.

Após a etapa de análise, os dados da transformada de Fourier podem ser infindavel-

[10] O termo *transformada de Fourier* é usado de duas maneiras, primeiramente para referir-se genericamente aos quatro tipos mais comuns da transformada ou para referir-se especificamente ao caso da transformada de Fourier de tempo contínuo.

[11] Existem três tipos comuns de transformada de Fourier, sendo os dois primeiros dirigidos para estudos conceituais e o terceiro para utilização prática: (1) *Transformada de Fourier* - formulação matemática pura ou estritamente analógica, na qual os sinais têm duração infinita e o tempo é considerado estritamente contínuo; (2) - *Transformada de Fourier de Tempo Discreto* (DTFT) - que é operação da *transformada de Fourier* em sinais discretos e de duração infinita, o que resulta em um espectro contínuo; (3) - *Transformada discreta de Fourier*(DFT) - é o caso mais utilizado computacionalmente, onde temos um sinal de tempo discreto de tamanho N e um espectro discreto de mesmo tamanho.

mente manipulados e então convertidos novamente para o domínio do tempo (ressíntese); a observação do espectro pode ajudar na dedução de altura predominante em um som complexo, para então ser executada uma modificação na velocidade de leitura do som; espectros provenientes de outros tipos de ondas (luz visível, microondas, sísmicas, etc) podem ser convertidos em ondas sonoras (sonificação de dados), etc (ROADS, 1996).

A formulação matemática da DFT, descrita abaixo conforme terminologia presente em Smith (2007, p.1-5, 115, 116), nos mostra que o cerne desta operação consiste em correlacionar[12] um sinal temporal $x(n)$ com uma componente senoidal e outra cossenoidal (encapsuladas como a exponencial complexa $e^{-j2\pi nk/N}$). Ao realizarmos essa operação, obtemos um número complexo $X(k)$ – que fornece as informações magnitude e fase – para cada *bin* k – também chamando de canal, ou seja, a faixa de frequência que estamos analisando. A transformada inversa iDFT realiza o processo análogo para converter os componentes de magnitude e fase de $X(k)$ de volta em informação temporais de $x(n)$.

$$DFT \qquad X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N}, \qquad k = 0, 1, 2, \ldots, N-1 \tag{2.7}$$

$$iDFT \qquad x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi nk/N}, \qquad n = 0, 1, 2, \ldots, N-1 \tag{2.8}$$

Nas quais temos

$$\sum_{n=0}^{N-1} f(n) \qquad f(0) + f(1) + \cdots + f(N-1) \tag{2.9}$$

$$x(n) \qquad \text{sinal de entrada no instante } n \text{ (amostras)} \tag{2.10}$$

$$X(k) \qquad \text{espectro da } k\text{-ésima amostra espectral (ou } bin) \tag{2.11}$$

$$N \qquad \text{quantidade de amostras temporais do sinal} \tag{2.12}$$

$$= \text{número de amostras de frequências (inteiro)} \tag{2.13}$$

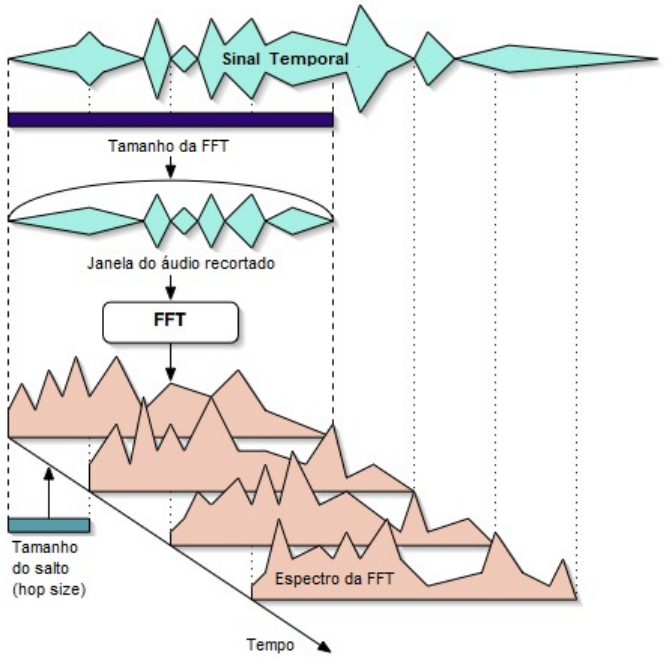
Em geral, os sinais de áudio apresentam forte variabilidade do seu conteúdo espectral ao longo do tempo, que é ocasionada tanto pelas variações rítmicas e melódicas quanto pelas inúmeras variações timbrísticas: como, por exemplo, uma linha melódica tocada por um instrumento monofônico ou uma voz que pronuncia um enunciado. Consequentemente, a aplicação da transformada de Fourier ao longo de toda a extensão destes sinais provê informações espectrais excessivamente abrangentes e vagas. Desta maneira, um procedimento comum consiste dividir o sinal temporal em segmentos iguais (janelas) e aplicar o algoritmo da transformada rápida de Fourier de sinais digitais (FFT)[13] para

[12] Multiplicar ponto a ponto os elementos de dois sinais, o chamado produto escalar, e, então, somar todos os valores resultantes desta operação

[13] Transformada rápida de Fourier (FFT) é a denominação do algoritmo mais eficiente utilizado para o cálculo da *transformada discreta de Fourier*(DFT), também chamado de algoritmo de Cooley-Tukey

cada uma dessas janelas - procedimento denominado transformada de Fourier de tempo curto (STFT) - gerando, assim, um perfil discreto da variação da transformada de Fourier ao longo de todo o sinal[14].

Figura 9 – Transformada de Fourrier de Tempo Curto (STFT): sucessivas transformadas de Fourier aplicadas às janelas do sinal de áudio



Fonte: Dudas e Lippe

2.3.1.2 STFT: artifícios

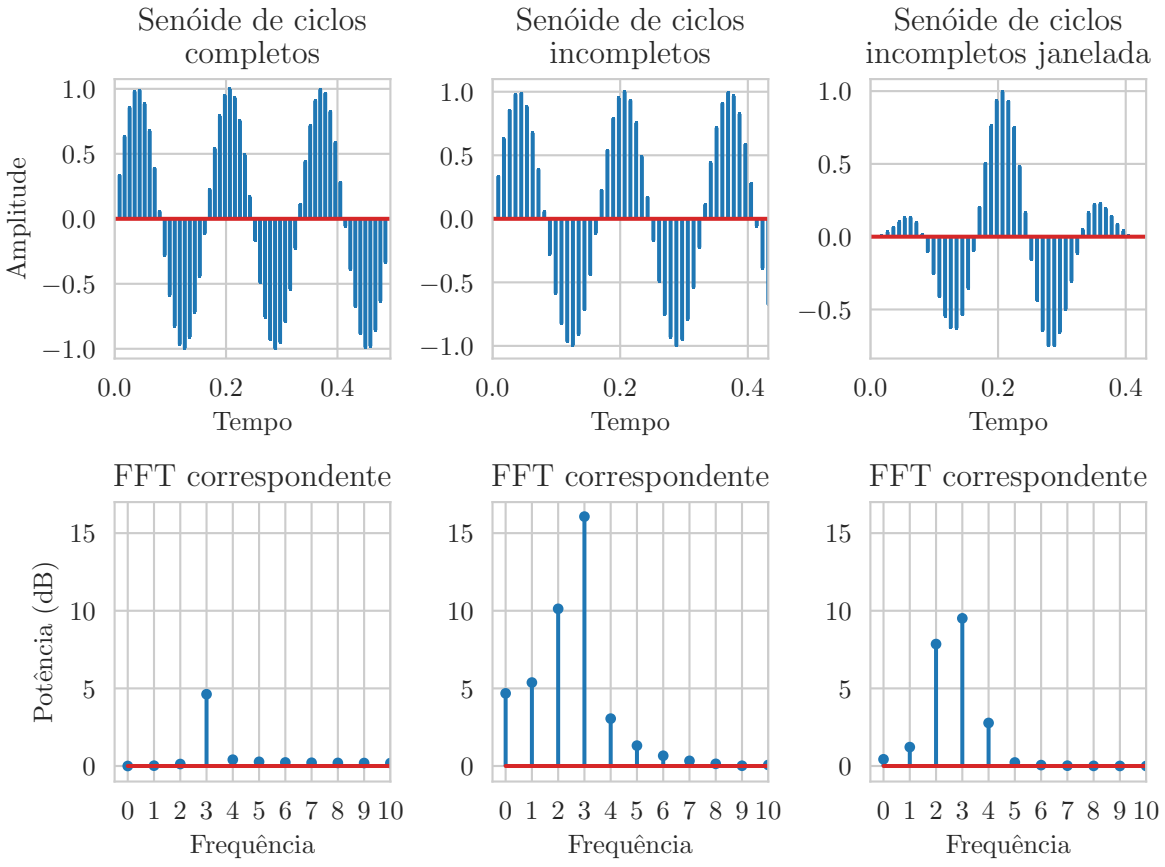
A STFT introduz uma série de artifícios no processamento de um sinal. O primeiro problema é relativo ao chamado princípio da incerteza da STFT: quanto maior for o tamanho da janela temporal escolhida, maior será a resolução de frequências obtida nas FFTs e menor será a precisão temporal das janelas ao longo do sinal, e vice-versa. Por exemplo, se escolhermos uma janela de áudio muito pequena, teremos mais precisão em saber em que instante temporal uma dada frequência começou a aparecer, porém diminuiremos nossa precisão em determinar exatamente qual frequência é essa. Assim, de maneira geral, as transformações espectrais com janelas longas tendem a afetar mais severamente os transientes dos sons e as transformações com janelas longas tendem a alterar mais drasticamente os aspectos harmônicos e de altura dos sons utilizados. Como não há solução geral para esse tipo de problema, na prática ou o janelamento é feito com base em um conhecimento prévio do sinal ou vai se ajustando o janelamento num sistema

[14] A DFT pode ser considera um retrato do espectro e a STFT, por sua vez, um filme do movimento espectral, consistindo em vários retratos sucessivos do espectro

de tentativa e erro com avaliação gráfica, sonora e estatística dos resultados. Além disso, quanto maior o número de pontos da janela, mais tempo será necessário para realizar os cálculos da STFT, portanto, maior será a latência do sistema. Desta maneira, o tamanha da janela tende a ser um dos parâmetros mais cruciais em um processamento espectral, afetando, de maneira acentuada, tanto a sonoridade resultante de uma transformação espectral quanto sua possibilidade de utilização em tempo real.

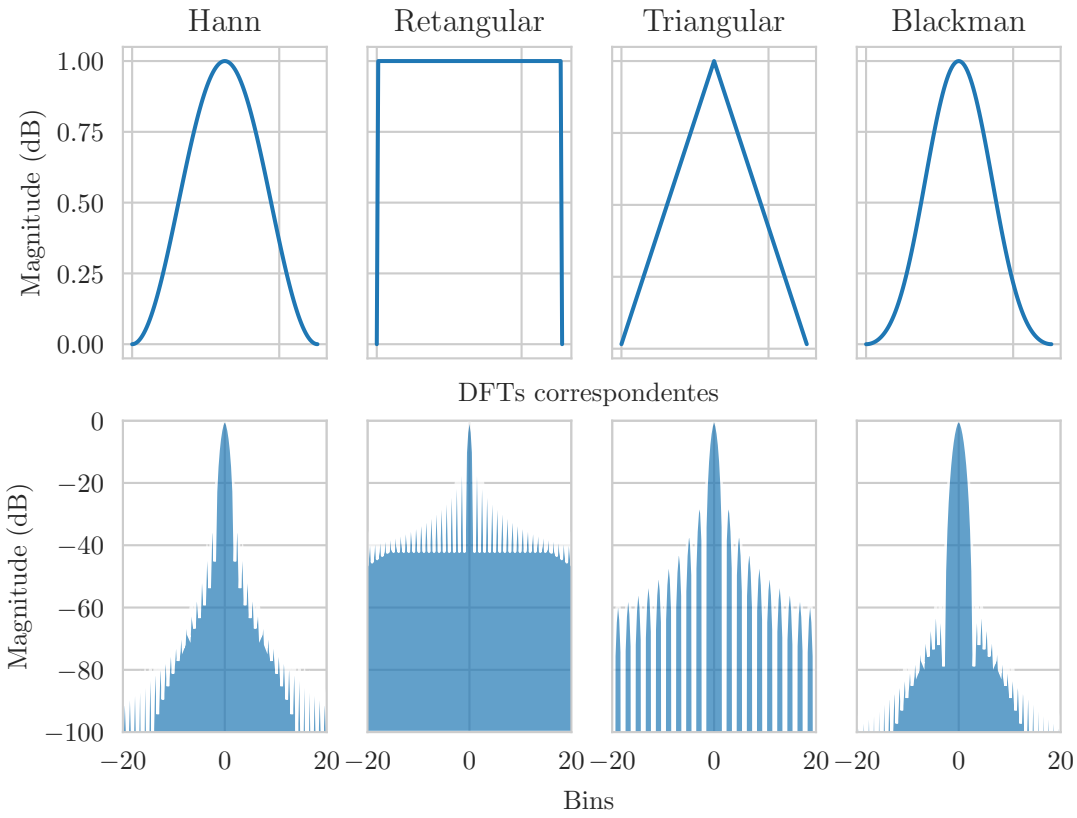
Um segundo artifício introduzido pela STFT é o chamado vazamento espectral. Para representar corretamente o sinal temporal em seu espectro correspondente, a DFT necessita que, idealmente, o sinal temporal contenha ciclos completos da forma de onda. Caso isso não ocorra, os valores de magnitude da frequência correspondente "vazam" para os *bins* laterais, conforme a Figura 10. No caso dos sinais musicais, a situação mais comum é que tenhamos uma variação de frequência ao longo do tempo, o que faz com que a análise de Fourier quase nunca corresponda à situação ideal em que podemos definir com precisão qual é a frequência presente uma dada janela. Uma estratégia para minimizar tais erros consiste na aplicação de funções de janelas que apresentem um lóbulo central proeminente decaindo para valores próximos de zero no seus extremos.

Figura 10 – Vazamento Espectral



Fonte: elaborada pelo autor, baseada em

Figura 11 – Tipos de funções de janela e suas DFTs correspondentes



Fonte: elaborada pelo autor

Existe uma grande variedade de funções de janela, pois há uma relação de compromisso entre faixa dinâmica e resolução de frequência na escolha do formato da janela. Conforme vemos na Figura 11, funções de janela que apresentam descontinuidades e formatos próximos ao retangular tendem a apresentar um espectro com forte presença de lóbulos laterais, porém com um pico central bastante estreito. O contrário pode ser dito para janelas com formato de sino: o pico central é mais largo, porém os lóbulos laterais decaem mais rapidamente.

Se tivermos duas senoides com frequências muito próximas e amplitudes similares, uma janela com lóbulo central muito largo pode impedir a discriminação precisa dessas frequências, portanto dizemos ter baixa resolução de frequência. Por outro lado, se tivermos duas senoides com frequências mais separadas e uma grande diferença de amplitude, uma janela com lóbulos laterais muito fortes poderá mascarar as frequências de amplitudes mais fracas, logo é dita ter pequena faixa dinâmica. Essa relação de compromisso entre largura do lóbulo central *versus* altura (vertical) dos lóbulos laterais é explorada em mais detalhes em Smith (2011), Roads (1996), Reiss e McPherson (2014, p.37, 1102, 193).

De maneira bastante geral, quando se tem um sinal com componentes senoidais estáveis e claramente separadas – por exemplo sons tônicos – é interessante utilizar janelas com lóbulos laterais mais fracos para permitir uma avaliação de magnitude mais precisa. De outra maneira, quando se trabalha com sons que misturam ruído e componentes harmônicas precisas é interessante trabalhar com janelas de lóbulo central mais estreito para permitir uma identificação mais precisa das frequências das componentes harmônicas.

Funções de janelas podem ainda introduzir ruídos e cliques devido à suas transições bruscas, como é o caso do formato retangular. É frequente também observar o uso mais geral de modelos como Hann, que apresentam uma curva intermediaria. Num processo teórico puro de análise-(re)síntese – no qual não se realiza modificações nos dados espectrais – o tipo de janela utilizada tem pouca influência no sinal de áudio reconstruído. Todavia, nos típicos processos de transformação espectral, o formato da janela imprime suas características no som que foi analisado, transformado e (re)sintetizado. O processo de escolha das janelas é portanto um procedimento prático no qual o julgamento auditivo também tem grande importância.

É importante frisar que, em termos práticos, as realizações computacionais da transformada de Fourier são processos de estimação do espectro (JR, 2019), não sendo processos exatos. Além disso, essa é apenas umas das metodologias disponíveis de estimação espectral, que pressupõe que os dados analisados se adequarão bem ao modelo de faixas fixas de frequências (*bins*) – ou seja, pressupõe-se que os dados analisados são compostos de componentes senoidais relacionadas harmonicamente.

A STFT permite a análise de um sinal de áudio, convertendo-o para o domínio da frequência, e a reconstrução idêntica do mesmo por meio da transformada inversa iSTFT, gerando novamente um sinal de áudio. Todavia, quando se deseja realizar uma transformação do sinal espectral visando obter também um novo sinal temporal (procedimento de análise-(re)síntese), é importante acrescentar outros passos e cuidados para evitar a introdução de novos artifícios. Para essa finalidade, foi desenvolvido o *vocoder de fase*, que é uma melhoria do algoritmo da STFT visando especialmente sinais de áudio, que será analisado a seguir.

2.3.1.3 Vocoder de fase: definição e utilização

O termo *VOCODER* – contração de *VOice* (voz) + *CODER* (codificação) – foi cunhado no início do século XX para descrever técnicas de compressão de dados telefônicos, sendo que as primeiras tentativas visavam processar um sinal de voz em um conjunto de filtros passa-faixa, transmitindo-se somente os coeficientes mais importantes de tais conjuntos de filtros, procedimento denominado atualmente *vocoder de canal*. Tal ideia apresentava pontos crucialmente problemáticos : primeiramente, a economia na transmissão era pequena, pois a quantidade de coeficientes necessários para se fazer uma transmissão

inteligível era muito alta; em segundo ponto, somente as informações sobre as variações de magnitude dos filtros eram transmitidas, ou seja, informações sobre a variação do envelope espectral eram transmitidas, porém as informações sobre a variação da frequência fundamental instantânea e seus harmônicos eram descartadas. Uma resposta para este problema foi a introdução do cálculo de diferença de fase instantânea, de onde derivou-se o termo *vocoder de fase*, que introduzia a informação frequência instantânea e permitia uma análise e (re)síntese virtualmente perfeita do sinal.

Historicamente, o vocoder de fase também nasce como uma tentativa de comprimir dados sem introduzir grandes distorções, um aprimoramento do vocoder de canal. Todavia, o primeiro algoritmo desenvolvido por Flagan e Golden (FLANAGAN; GOLDEN, 1966) aumentava enormemente o tamanho dos dados após a análise, o que fez com que a primeiras implementações práticas só surgissem dez anos depois, com um algoritmo otimizado (ROADS, 1996; GORDON; STRAWN, 1987, p.568, p. 2). Existem duas possíveis implementações do vocoder de fase, a primeira utilizando a técnica da heterodinamização (*heterodyning*)[15] e a segunda utilizando a FFT. Ainda hoje o vocoder de fase é computacionalmente intensivo; apesar das implementações em tempo real desse método já terem se tornado populares, algoritmos de modificação dos parâmetros em tempo real em linguagens de alto nível (SuperCollider, Pd e Max) são bastante restritivos, sendo ainda recomendável a utilização de linguagens mais otimizadas como o C++ e C para o desenvolvimento de novas aplicações.

O vocoder de fase é bastante utilizado como ferramenta puramente analítica, como vemos em Beauchamp (2007). Entretanto, seu propósito mais usual é criar uma representação espectral variável no tempo, o que permite modificações independentes na frequência (e também na altura), na escala temporal e no conteúdo espectral – processos de análise-ressíntese. Sem dúvida, o principal uso desta ferramenta é para a realização de modificações independentes da escala temporal ou da escala de frequência, permitindo manipulações como: ajustes finos do tempo musical de gravações de diferentes instrumentos, acréscimo de microtemporalidades, correção de afinação, criação de vozes transpostas, sincronização de gravações, dentre outras.

Cabe ressaltar aqui a crucial diferença de abordagem relativa aos contextos da engenharia e computação musical, de um lado, e da sonologia e música com computadores, de outro. De maneira geral, este primeiro campo se preocupa em garantir que os resultados práticos se aproximem ao máximo das previsões teóricas. No segundo campo, é mais comum a proposição de um dado algoritmo, seguida imediatamente de alguma implementação e sua consequente apreciação auditiva – muitas vezes dispensando-se o enquadramento

[15] Para detalhes ver Dolson (1986), Park (2010), Zolzer (2011) nos quais se relaciona essas duas implementações com as duas perspectivas de interpretação do *vocoder de fase*, a do algoritmo como uma série de filtros passa-faixa ou a interpretação do algoritmo como uma série de DFTs de segmentos de sons sobrepostos temporalmente

de tal processo numa dimensão analítica ou teórica – esse é o caso das transformações propostas pelo CDP.

Os contextos de engenharia e computação musical geralmente se restringem a abordar as aplicações da STFT a assuntos como estimação da F0 a partir de picos espectrais, filtragem espectral, correção de afinação, ajustes rítmicos, dentre outros (REISS; MCPHERSON, 2014; SMITH, 2011; ZOLZER, 2011). Já os contextos de música com computadores e sonologia se concentram em levar os procedimentos do vocoder de fase ao extremo, com propostas que, inicialmente podem não apresentar sentido ou aplicabilidade direta, mas na prática mostram seu potencial criativo. É o caso de transformações como a inversão do espectro em relação ao seu eixo central, o esticamento de só uma parte do espectro, os processos de transformação dos dados do espectro com lógicas fractais, etc. De um lado, preocupa-se em desenvolver uma ferramenta que seja a melhor possível para dada tarefa, de outro, inventam-se diversas novas ferramentas ou utilidades pouco ortodoxas para uma ferramente – desde que a ação final desse uso seja satisfatória.

2.3.1.4 Vocoder de fase: algoritmo clássico

Existem diversas implementações do vocoder de fase, sendo a divisão entre implementações de tempo real e tempo diferido as duas principais categorias. Dentro de cada um desses grupos há variantes como escolha de utilização da DFT ou FFT; tipo de janelamento do sinal; realização ou não de preenchimento com zeros; conversão de coordenadas ou não; travamento da fase ou fase livre, etc. Nos concentraremos, sempre que for necessário, nas consequências da diferença entre implementação de tempo real e tempo diferido, pois esse ponto crucial distancia o CDP do SuperCollider e atualmente impede uma plena realização da totalidade dos processos em ambas as plataformas.

Adaptando definições como as presentes em Dolson (1986), Driedger e Müller (2016), Park (2010), Zolzer (2011), Reiss e McPherson (2014), o algoritmo geral do vocoder de fase pode ser descrito pelos seguintes passos:

- Dado um sinal de tamanho arbitrário, fatiamos este sinal em quadros de N amostras distanciados de R_a pontos(o chamado tamanho de salto de análise, ou *analysis hop size*).
  - Para implementações em tempo real N deve ser uma potência de 2 para a execução do algoritmo mais eficiente da transformada de Fourier (FFT).
- Multiplica-se o sinal por uma função de janela de tamanho N. Denomina-se janelamento estas duas etapas.
- Aplicamos a transformada rápida de Fourier (FFT) nos quadros do sinal.

  - Até esse ponto executamos uma transformada de Fourier de tempo curto (STFT).
  - Cada quadro irá gerar um vetor de tamanho N composto de números complexos. Como o sinal analisado é real, a FFT será simétrica, portanto podemos utilizar somente a primeira metade dos pontos.

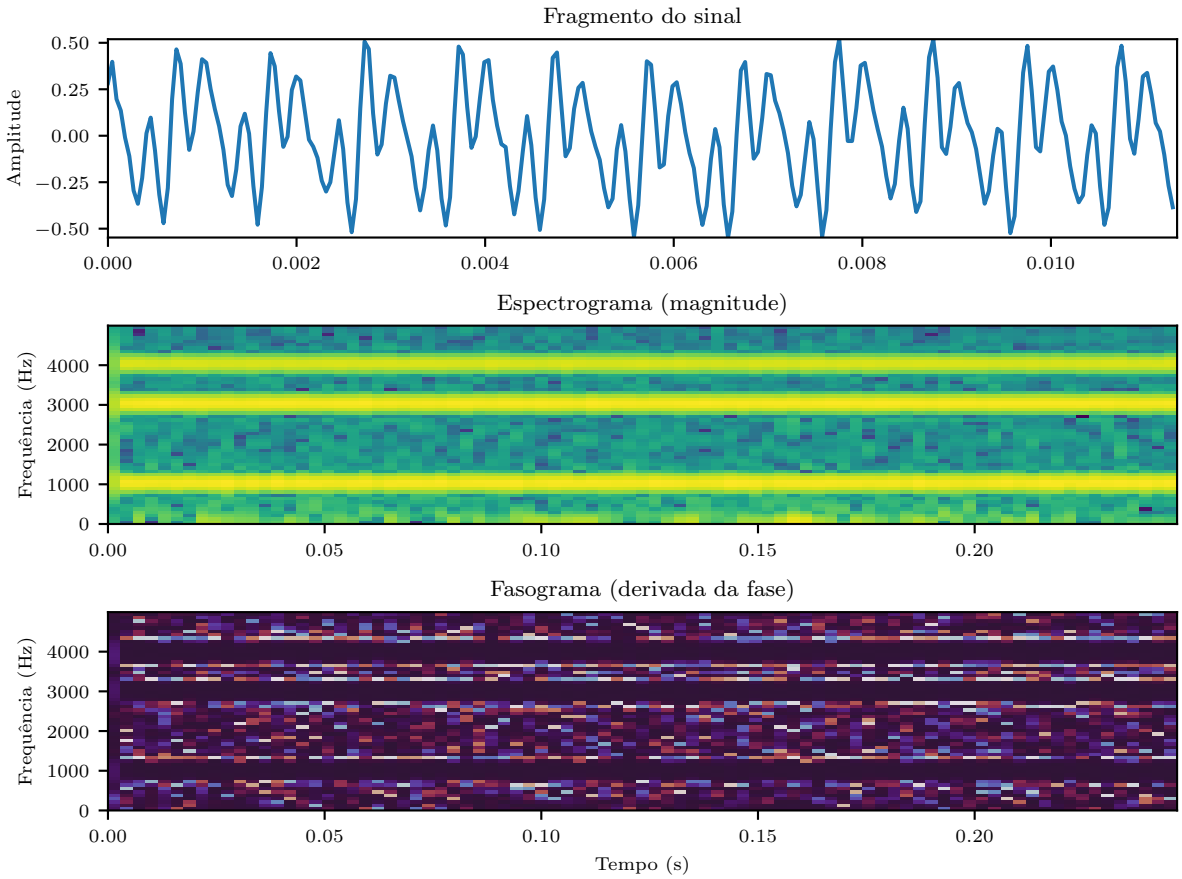
- Como os quadros da FFT são fornecidos em representação cartesiana (XY), converte-se para representação polar (magnitude e fase).
- Os valores de fase são fornecidos de forma enrolada (ou embrulhada) dentro da faixa de 0 a 2π. Para operações em tempo diferido, desenrolamos estes valores para obter valores lineares de fase.
- Para cada canal (*bin*), subtraimos o valor de fase atual do valor de fase do quadro anterior e dividimos pelo R_a (passo entre dois quadros) de modo a obter a taxa de variação instantânea de fase, que na verdade representa uma frequência instantânea.
  - Esse procedimento de cálculo da taxa de variação da fase (frequência instantânea) é específico ao vocoder de fase
- Converte-se os valores polares (magnitude e fase) de cada quadro para a forma cartesiana (XY).
- Executa-se a inversa transformada rápida de Fourier (iFFT) em cada quadro.
- A cada R_s pontos(o chamado tamanho de salto de síntese, ou *synthesis hop size*), sobrepomos e somamos os quadros de sinal temporal obtidos no processo anterior.

A transformada de Fourier, seja uma FFT ou DFT, não é capaz de lidar, por si só, com os pontos problemáticos introduzidos pelo janelamento. Os dois principais são: (1) o fato de um canal (*bin*) apresentar somente uma faixa de frequência, e não a frequência exata da senoide analisada; (2) a perda da *coerência de fase* entre dois quadros da STFT, a chamada *coerência horizontal de fase*. O ponto central que diferencia o vocoder de fase de outros métodos espectrais – especialmente a STFT – é o cálculo de uma frequência instantânea, obtida por meio da diferença de fase[16] (taxa de variação) entre os quadros.

Para uma dada senoide em regime permanente, o janelamento costuma recortar trechos da onda que não apresentam a mesma fase inicial, o que causará saltos no momento em que acontecer a ressíntese. A solução apresentada pelo algoritmo é calcular qual foi a

[16] A fase a qual o nome do algoritmo se refere é o conceito de *fase instantânea*, e não a ideia de *fase inicial* ou *diferença de fase*. A *fase instantânea* é o instante angular no qual uma oscilação se encontra; a *fase inicial* é o ponto de partida de uma oscilação, o instante angular do início da oscilação; já a *diferença de fase* é uma medida entre duas oscilações da diferença entre as suas *fases instantâneas*. Assim, para uma cossenoide, por exemplo, teremos uma *fase inicial* igual a $\pi/2$ ou $90°$ e uma *fase instantânea* que circula linearmente entre os valores de 0 a 2π.

Figura 12 – Espectrograma e fasograma de três senoides (1000, 3000 e 4000 Hz) somadas a um ruído



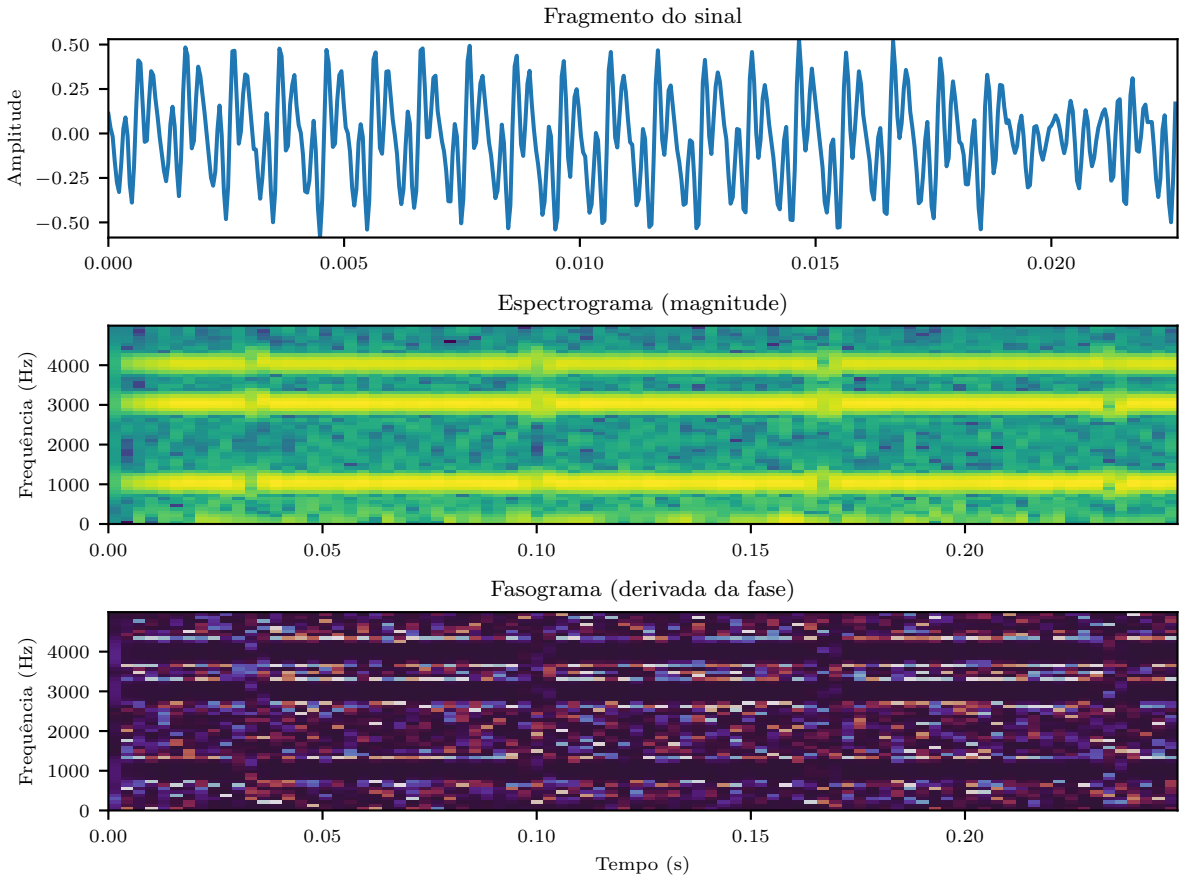
Fonte: elaborada pelo autor a partir de adaptações em Serra et al. (2020)

diferença de fase entre dois quadros consecutivos e dividir o resultado pelo tempo passado entre os dois quadros, o que resulta em uma frequência instantânea, melhor estimada do que a frequência crua da faixa do canal (*bin*). Este refinamento nos dados da STFT é especialmente útil quando realizam-se modificações da escala de temporal, pois a garantia da continuidade da fase possibilitará o cálculo de quadros intermediários com maior precisão.

Graficamente o vocoder de fase fornece duas visualizações complementares: o espectrograma – representação da magnitude, em geral, de cada *bin* ao longo do tempo – e o fasograma – representação da fase de cada canal desenrolada e derivada . Ao contrário do espectrograma, a Figura 12 permite avaliar como se faz difícil a obtenção de informações significativas a partir de uma avaliação crua do fasograma. Neste é possível perceber que há faixas estáveis (representando que há frequências estáticas) e regiões de variações mais bruscas (mais fortemente preenchidas por ruídos), porém é difícil ir além desse tipo de interpretação.

A modificação mais direta no espectrograma é a filtragem espectral, que consiste

Figura 13 – Espectrograma e fasograma de três senoides (1000, 3000 e 4000 Hz) somadas a um ruído, no som total as fases são embaralhadas a cada 66 ms



Fonte: elaborada pelo autor a partir de adaptações em Serra et al. (2020)

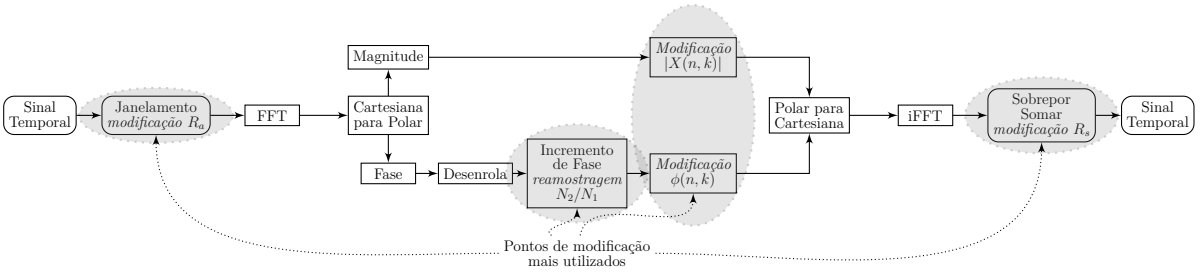
em uma manipulação no valor de um dado canal ao longo dos quadros. No fasograma, todavia, manipulações *per se* do conteúdos dos canais somente causarão descontinuidades na forma de onda – mudanças bruscas de fase. A Figura 13 permite antecipar que uma permutação aleatória dos canais de fase não irá gerar grandes mudanças no timbre, mas sim descontinuidades na onda, que são percebidas tanto com cliques quanto como inicio de novos eventos tônicos idênticos aos anteriores, conforme pode ser escutando em Áudio 2.7.

Áudio 2.7 (2-7_sen_ruido_comp_fase_embaralhada.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.* Três senoides (1000, 3000 e 4000 Hz) somadas a um ruído, seguidas deste mesmo som com as fases embaralhadas a cada 66 ms.

Quando se manipula as informações de magnitude e fase conjuntamente ou em associação com outros pontos do algoritmo, transformações mais diversificadas são possíveis. Os pontos estratégicos em que podemos realizar modificações nos dados ou em certos parâmetros de processamento para obter efeitos sonoros criativos estão apontados na

Figura 14.

Figura 14 – Etapas tradicionais de análise-ressíntese do vocoder de fase e pontos estratégicos para modificações



Fonte: elaborada pelo autor

Algumas utilizações menos ortodoxas do vocoder de fase, bem como apontamentos em direção das consequentes características perceptivas, são estruturadas e detalhadas em Zolzer (2011), conforme vemos na versão adaptada da Figura 15. Esse tipo de estruturação visivelmente privilegia o processamento de dados e tende a não se ocupar da relação entre tipos de fontes e o resultado sonoro final após processamento.

As propostas de Wishart, bem como as presentes no SuperCollider, para manipulações espectrais utilizando o vocoder de fase expandem consideravelmente o horizonte presente nas literaturas tradicionais como Zolzer (2011), Reiss e McPherson (2014), Park (2010), além de aprofundarem muitas das metodologias propostas em tais autores. A princípio, não há modelos teóricos que justifiquem ou embasem os procedimentos propostos no CDP e no SuperCollider. Além disso, a documentação dos algoritmos utilizados é frequentemente vaga, portanto, na seção 3.2 discutiremos, sempre que possível, os detalhes das funções escolhidas.

2.3.1.5 Vocoder de fase: artifícios sonoros

Por trabalhar com uma discretização do espectro de frequências e uma granulação do eixo temporal, o vocoder de fase apresentará características sonoras que são inerentes a tal tipo de processamento. Em relação a esse ponto é importante ressaltar que as funções de janela promovem vazamento de frequências entre os canais – perda de resolução horizontal (ao longo do tempo) – e que a sobreposição de janelas promove o vazamento dos canais entre diferentes quadros – perda de resolução vertical (em um instante de tempo específico, entre os *bins* de uma janela). Assim, a estratégia básica da STFT faz com que a manipulação de dados espectrais seja um processo de trabalhar com dados "borrados" em uma grade de espaçamento considerável.

O cerne da transformada de Fourier é a análise de um sinal por meio de um conjunto de filtros do tipo passa-faixa. Assim, sempre que transformarmos estes dados

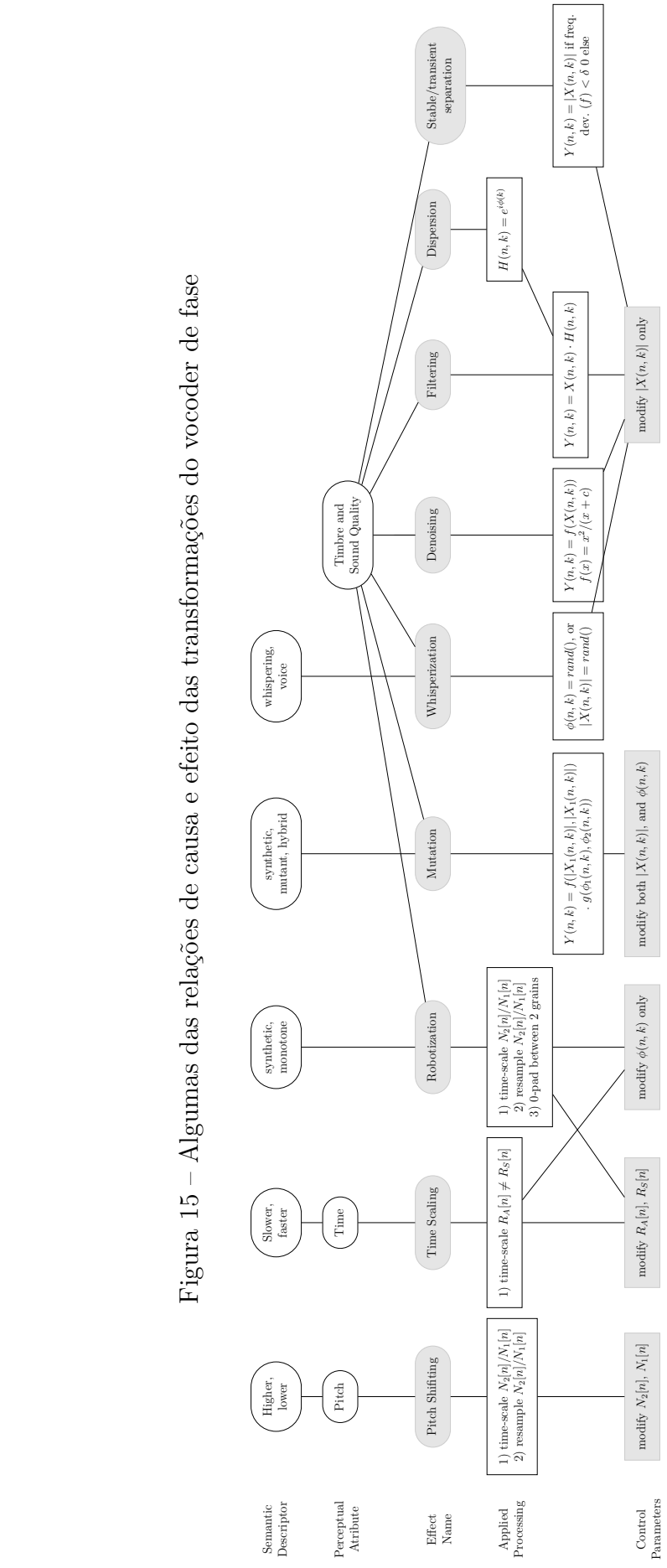


Figura 15 – Algumas das relações de causa e efeito das transformações do vocoder de fase

Fonte: Adaptação nossa de Zolzer (2011, p. 244)

iremos acentuar as características sonoras que são típicas de bancos de filtros, como por exemplo os sons canelados de Schaeffer. Esse tipo de característica pode ser facilmente percebida nos exemplos da seção 3.2.

Outro artifício frequente é o chamado "faseamento", que é uma espécie de característica reverberante que aparece especialmente quando realizamos esticamento temporal. Em resumo, o que acontece é que o algoritmo mostrado aqui é projetado para manter a *coerência de fase horizontal* apenas, não conseguindo lidar com a *coerência de fase vertical*. Esse fenômeno também aparece quando temos sons com ataques rápidos que são esticados, é comum vermos os transitórios serem "esmagados" (comprimidos em amplitude) no caso de esticamento temporal (DRIEDGER; MüLLER, 2016).

Recentemente, diversas alternativas foram propostas para melhorar tal produção de artifícios sonoros, como é o caso do vocoder de fase travada, vocoder de phase com sobreposição-soma sincronizada (PVSOLA), além de variados métodos de cálculo da frequência instantânea (DRIEDGER; MüLLER, 2016).

2.4 Modificações da escala de tempo-frequência

Um ponto especial que interliga as diversas técnicas de transformação utilizadas por Wishart são as modificações da da escala de tempo-frequência[17]: a possibilidade de esticar ou contrair, independentemente ou interdependentemente, a duração, a faixa de frequência e a altura[18] de um dado objeto sonoro. Apesar das modificações interdependentes trazerem consigo um forte potencial criativo, a independência em tais fatores sempre foi almejada por possibilitar expandir possibilidades físicas (*e.g.* afinar notas, sem alteração timbrística ou temporal, de instrumento de museu que não pode ser fisicamente modificado). Tanto os métodos de operação temporal (conjuntos de onda, estratégias de reprodução, transformações de microssom) quanto os métodos de operação espectral (vocoder de fase) apresentam formas de utilização que contemplam a realização independente de modificações da escala de tempo-frequência, cada qual com suas vantagens e artifícios produzidos, e esta é a principal utilização comercial de tais algoritmos.

As modificações de escala de tempo-frequência encontram-se em um nível de desenvolvimento avançado uma vez que sua aplicação prática tem ao menos duas fortes demandas na indústria da tecnologia musical e do áudio: do ponto de vista da escala

[17] Tradicionalmente, na literatura encontram-se nomes separados para cada fenômeno, modificações da escala temporal (*time scale modification* - TSM) e deslocamento de altura (*pitch shifting*). Todavia, em literaturas mais recentes costuma-se encontrar o termo modificações da escala de tempo-frequência (*time-frequency scaling*) devido ao inevitável acoplamento entre estes dois domínios.

[18] Preferimos aqui não utilizar o termo modificação de altura de forma irrestrita pois não é possível modificar computacionalmente um atributo percetivo. Assim diferenciaremos quando o foco da modificação for espectral, ou nas frequências, que pode ou não modificar a altura, das modificações de altura, que geralmente visa manter as relações entre harmônicos de um som, garantindo a percepção do atributo altura

de tempo, são utilizadas para ajustar a sincronia de gravações, *e.g.* estabelecimento de sincronia entre múltiplas faixas de uma gravação de bateria que apresentam pequenos desvios temporais e ajuste de uma faixa vocal à um andamento metronômico; sob a lógica da escala de frequência, exemplos de grande utilização são a correção de afinação de instrumentos e vozes – sem que se promova alterações na escala temporal e no timbre das fontes – e também a simulação de instrumentos similares, como é o caso da criação de corais de vozes paralelas à partir de uma única faixa vocal.

Ainda assim, modificações na escala de tempo-frequência são uma das tarefas mais difíceis de serem realizadas visto que a interdependência entre objetivo sonoro a ser alcançado e a fonte sonora utilizada costuma impedir a adoção de soluções genéricas. Como exemplo inicial, tomemos o caso em que a soma de duas senoides é equivalente, do ponto de vista perceptivo, ao produto da sua portadora e da moduladora, o fenômeno dos batimentos. O esticamento temporal deste sinal deve resultar na soma de duas senoides com frequências igualmente alteradas ou na mesma portadora mas com uma moduladora mais grave? Ambas as soluções parecem estar corretas, sendo que a primeira tenta privilegiar o rigor matemático de tal transformação e a segunda busca encontrar um procedimento matemático que se adéque a uma representação auditiva de um esticamento temporal de um objeto sonoro. Outro exemplo problemático clássico é o esticamento de um som que apresentam um gesto percussivo – ataque rápido e decaimento exponencial. As tentativas de esticamento costumam romper com os limites da percepção humana do que é um ataque rápido e um decaimento exponencial, limitando o esticamento deste tipo de som a uma faixa estreita de valores. O mais comum aqui é que a maioria dos casos seja suficientemente problemática e apenas alguns exemplos específicos possam ser considerados como resolvidos (ZOLZER, 2011, p. 249).

Wishart mostra que um dos principais potenciais criativos desta técnica está justamente na operação fora das tradicionais faixas permissíveis, fator que gera alterações radicais nos objetos sonoros a ponto de transformá-los em outros tipos que são pouco ou nada associáveis à suas fontes originárias, mas associáveis a outros sons naturais. Essa capacidade funciona como uma engrenagem mestre na ideia de *metamorfose sonora*: um sílaba pode ser percebida como uma versão encolhida de outra sílaba, um som percussivo pode ser transformado num *glissando* suave, sons contínuos podem ser encolhidos a ponto de se parecerem com sons percussivos, etc.

Em pontos ainda mais extremos, as modificações da escala de tempo-frequência também são utilizadas para extrapolar os domínios da percepção clássica do fenômeno sonoro. Um exemplo amplamente utilizado aquilo que chamamos *dessonificação/ressonificação* – quando se estica um dado objeto sonoro a ponto que as oscilações sejam tão lentas que não são mais perceptíveis como sons ao ouvido humano (dessonificação), para em seguida, utilizar estas ondas como dados de controle que dirigem algoritmos de produção sonora

(ressonificação). Outro exemplo vastamente empregado é um procedimento analogamente inverso a este: encolhimento e iteração. Aqui, encolhe-se um dado objeto sonoro até que seu tamanho esteja na faixa dos microssons e então repete-se iteradamente este som na faixa de frequências audíveis.

Quando o foco da transformação é realizar uma modificação de altura que não altere o timbre e a duração do som fonte adentramos em outros pontos problemáticos. O primeiro deles está no fato da percepção de altura ser um atributo psicoacústico que não apresenta uma definição estrita e que varia em diferentes contextos. Tendo isso em vista, não é possível manipular a altura de um som diretamente, mas sim parâmetros que alteram a sensação de altura, tais como, velocidade de reprodução, posição da fundamental e harmônicos no espectro, dimensão das estruturas repetitivas de uma onda, dentre outros. Em segundo ponto, altura e timbre são parâmetros intimamente relacionados fisicamente e, do ponto de vista da realização sonora prática, mudanças de altura geralmente estão condicionadas a alterações timbrísticas, como no caso evidente do piano e da voz humana – mudanças de altura implicam necessariamente mudanças de registro, timbre. De modo semelhante, as alterações computacionais de altura também promoverão alguma modificação no grande guarda-chuva de parâmetros denominado *timbre*.

Nos casos limítrofes deste problema, há sons que apresentam altura menos definida, como é o caso de sinos, sons com vibrato e ruidosos. Aqui o fato da altura ser menos definida não implica que tenhamos sons pouco conhecidos ou modelados fisicamente, muito pelo contrário, significa que a percepção auditiva humana se confunde com tais sons. Tal como o problema das modificações da escala temporal, aqui temos de escolher entre realizar uma modificação que seja mais coerente física e matematicamente ou perceptivamente.

Por fim, quando desejamos modificar a escala de frequência de um som, como é o caso de diversas transformações espectrais propostas por Wishart, é bastante provável que ocorram modificações na altura, e não é incomum que algumas destas modificações também alterem a duração dos sons. Especialmente no caso de transformações sonoras criativas, modificações de tempo, altura e frequência costumam ser fortemente interdependentes, pois são parâmetros e atributos constituintes da nossa percepção do fenômeno sonoro. Assim, as transformação detalhadas daqui em diante tenderam a gravitar em torno destes pontos.

3 Transformações sonoras no SuperCollider

> *Cientistas, estejam avisados! Nós iremos embarcar em procedimentos de processamento de sinal que parecerão bizarros diante dos procedimentos cientificamente sofisticados, que fornecerão resultados relativamente imprevisíveis ou que serão fortemente dependentes de propriedades únicas dos sinais particulares aos quais serão aplicados. Como músicos, a questão que deveremos nos perguntar não é entretanto se esses procedimentos são cientificamente válidos, ou mesmo previsíveis, mas ao contrário, se eles produzem resultados estéticos úteis em pelo menos alguns tipos de materiais sonoros.*
>
> —Trevor Wishart, *Audible Design*

O SuperCollider, por também ser um ambiente de programação geral e não apenas musical, possibilita diversas maneiras de execução de uma mesma ideia. Neste capítulo, selecionamos algumas destas possíveis abordagens que permitem executar os procedimentos presentes no CDP, algumas vezes adaptando-os. Além disso, fazemos também uma discussão sobre a sonoridade de cada processo e abordamos em exemplos a maneira como a idiossincrasia dos sons fontes reflete-se fortemente na sonoridade de cada transformação.

Na *seção 3.1 - Processos de transformação no domínio do tempo*, avaliam-se as transformações nas quais os cálculos computacionais são feitos no domínio do tempo, utilizando uma subdivisão mais organizacional do que prática em envelopamento e estratégias de ataque-continuação; estratégias de reprodução, granulação e microsom; e transformações de conjuntos de onda. Na *seção 3.2 - Processos de transformação no domínio da frequência* esta avaliação é feita para os cálculos no domínio da frequência, mais especificamente, utilizando os dados do vocoder de fase. Aqui não adotou-se a subdivisão proposta pelo CDP, porém selecionamos os subgrupos mais relevantes do CDP para serem adaptados e implementados.

A avaliação da mecânica interna dos processos de transformação do CDP foi feita por meio do estudo das descrições presentes na documentação e nas explicações contidas em (WISHART, 1994), sendo que o código fonte deste foi analisado apenas de maneira superficial, somente para a verificação de detalhes e pormenores. Todavia, sempre que diferenças críticas ou intransponíveis apareceram, elucidações gerais da implementação foram mencionadas no formato de notas.

Devido à inexistência de traduções das obras de Wishart para o português, optamos aqui por traduzir de maneira mais direta possível os nomes dos processos. Estes nomes, que são bastante poéticos e pouco científicos, refletem mais a maneira como os dados são manipulados do que suas consequências sonoras; essa escolha de Wishart possivelmente foi

devido à intensa depedência entre a sonoridade do processo e o tipo de som fonte utilizado, dificultando uma categorização pela sonoridade. Por fim, como a terminologia adotada no CDP variou bastante ao longo dos anos, optamos aqui em utilizar os termos mais atuais.

Diversas das transformações avaliadas computacionalmente neste capítulo ocorrem em tempo real. Todavia, para possibilitar o fornecimento de exemplos sonoros de maneira adequada ao modelo brasileiro de trabalhos acadêmicos (essencial textual), as transformações são realizadas a partir de sons gravados, carregados em um *buffer*. As utilizações destes processos de transformação sonora envolvendo entradas em tempo real (microfone, GUI, dentre outras) são vastas e demandariam outros recursos de exemplificação, portanto, não foram incluídas neste texto.

* * *

3.1 Processos de transformação no domínio do tempo

3.1.1 Envelopamento e estratégias de ataque-continuação

Existem dois principais seguidores de envoltória no SuperCollider, ambos operando em tempo real: *EnvDetect*, que é baseado em um filtro passa-baixas de um polo e apresenta controles de ataque e decaimento; *EnvFollower*, que é baseado um retificador de onda completa seguido de um filtro. Optamos por utilizar *EnvDetect* neste trabalho por preferir a sonoridade do mesmo.

No CDP as funções categorizadas como manipulação de envelope (*envel* e *envnu*) realizam as operações de corte e edição típicas das DAW (*fade in*, *fade out*, *cross-fade*, modulações de amplitude customizadas, etc) e também as possibilidades de extração e imposição de envelope temporal de um dado *sample*. De maneira geral, tais métodos inclinam-se à uma manipulação artesanal dos envelopes, pois são executadas por meio dos arquivos de *breakpoints* (.bkr), ou seja, geralmente é feita uma edição ponto a ponto dos momentos de transição, assim como nos processos de automação típicos das DAW.

Quando comparada com a geração sintética, a extração de envelopes nos permite alcançar curvas mais próximas dos modelos físicos, porém geralmente às custas de introdução de algum tipo de ruído – devido às transições bruscas nestas curvas – ou oscilações espúrias que vazam do conteúdo harmônico.

Em Áudio 3.1 e Áudio 3.2 é possível perceber como a alteração do ataque de um som modifica a percepção de suas características timbrísticas, da sua evolução espectromorfológica e da possível associação com a fonte.

Áudio 3.1 (3-1_C3_marimba_violino_trompa_fagote_trompete_piano.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota dó-3 (C3)[a] tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano, retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) (<https://vis.versilstudios.com/vsco-community.html>).

[a] Pelo fato da biblioteca utilizada não dispor do *sample* da nota C3, optamos pela utilização da nota C3 simulada, neste caso, a tecla responsável pela nota C2 é percutida com grande intensidade de forma que sua altura percebida soe como seu segundo harmônico (nota C3). Esse detalhe será omitido daqui por diante no texto.

Áudio 3.2 (3-2_env_multifon_mrb_vln_tro_fgt_tru_pno.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Fagote tocando multifônicos, seguido dos envelopes dos sons Áudio 3.1 – marimba, violino, trompa, fagote, trompete e piano – impostos sobre este som do fagote multifônico. Gravação de fagote retirada de <https://freesound.org/s/90026/>.

No Código 3.1 mostramos como é possível realizar vários envelopamentos em uma só fonte utilizando uma renderização em tempo diferido no SuperCollider. Este tipo de técnica no SuperCollider mostra também como a renderização em tempo diferido nativa deste ambiente (*Score*) demanda uma alta verbosidade e dificulta a utilização de estruturas mais abstratas e de alto nível.

```
1  (var server = Server(\nrt,
2          options: ServerOptions.new
3          .numOutputBusChannels_(2)
4          .numInputBusChannels_(2)
5  );
6  ~b1 = Buffer.new(server, 0, 2); ~c1 = Buffer.new(server, 0, 2); ~c2 =
       Buffer.new(server, 0, 2); ~c3 = Buffer.new(server, 0, 2); ~c4 = Buffer.
       new(server, 0, 2); ~c5 = Buffer.new(server, 0, 2); ~c6 = Buffer.new(
       server, 0, 2);
7  a = Score([
8          [0.0, ['/d_recv',
9                  SynthDef(\NRTrec, { |bEnv|
10                         var source = PlayBuf.ar(2, bEnv, doneAction: 2);
11                         var env = EnvDetect.ar(source, 0.0001, 0.0001);
12                         OffsetOut.ar(0, LeakDC.ar(env * PinkNoise.ar(2.25))
                               )
13                 }).asBytes
14         ]],
15         [0.0, ~c1.allocReadMsg(PathName(thisProcess.nowExecutingPath).
              parentLevelPath(3) +/+ "Audios/C3-marimba-stereo.wav")],
16         [0.0, ~c2.allocReadMsg(PathName(thisProcess.nowExecutingPath).
              parentLevelPath(3) +/+ "Audios/C3-violino-stereo.wav")],
17         [0.0, ~c3.allocReadMsg(PathName(thisProcess.nowExecutingPath).
              parentLevelPath(3) +/+ "Audios/C3-trompa-stereo.wav")],
```

```
18          [0.0, ~c4.allocReadMsg(PathName(thisProcess.nowExecutingPath).
               parentLevelPath(3) +/+ "Audios/C3-fagote-stereo.wav")],
19          [0.0, ~c5.allocReadMsg(PathName(thisProcess.nowExecutingPath).
               parentLevelPath(3) +/+ "Audios/C3-trompete-stereo.wav")],
20          [0.0, ~c6.allocReadMsg(PathName(thisProcess.nowExecutingPath).
               parentLevelPath(3) +/+ "Audios/C3-piano-stereo.wav")],
21          [0.01, (x = Synth.basicNew(\NRTrec, server, 1000)).newMsg(args: [
               bEnv: ~c1])],
22          [3.01, (x = Synth.basicNew(\NRTrec, server, 1001)).newMsg(args: [
               bEnv: ~c2])],
23          [6.01, (x = Synth.basicNew(\NRTrec, server, 1002)).newMsg(args: [
               bEnv: ~c3])],
24          [9.01, (x = Synth.basicNew(\NRTrec, server, 1003)).newMsg(args: [
               bEnv: ~c4])],
25          [12.01, (x = Synth.basicNew(\NRTrec, server, 1004)).newMsg(args: [
               bEnv: ~c5])],
26          [15.01, (x = Synth.basicNew(\NRTrec, server, 1005)).newMsg(args: [
               bEnv: ~c6])],
27  ]);
28  a.sort;
29  a.recordNRT(
30          outputFilePath: thisProcess.nowExecutingPath.dirname.dirname.
               dirname +/+ "Audios/env-vazamento.wav",
31          headerFormat: "wav",
32          sampleFormat: "int16",
33          options: server.options,
34          duration: 18.0,
35          action: { "done".postln }
36  );
37  server.remove;)
```

Código 3.1 – Envelopamento em tempo diferido

Quando utilizamos uma fonte mais neutra, como é o caso de um ruído rosa, em associação com parâmetros de ataque e relaxamento de cálculo do envelope que permitam uma extração do envelope com oscilações mais rápidas, é auditivamente claro o vazamento do timbre na curva de envelope, conforme exemplifica o Áudio 3.3. O Código 3.2 implementa esse tipo de envelopamento em tempo real, mostrando também como tal prática no SuperCollider é menos verbosa se comparada à renderização em tempo diferido.

Áudio 3.3 (3-3_env_vazamento.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Envelopes dos sons Áudio 3.1 – marimba, violino, trompa, fagote, trompete e piano – impostos sobre o som de um ruído rosa.

```
(s = Server.local;
~caminho = PathName(thisProcess.nowExecutingPath).parentLevelPath(3) +/+ "
    Audios/";
~inicializa = {
        ~instOrq = [
                Buffer.read(s,~caminho +/+ "C3-marimba-stereo.wav"),
                Buffer.read(s,~caminho +/+ "C3-violino-stereo.wav"),
                Buffer.read(s,~caminho +/+ "C3-trompa-stereo.wav"),
                Buffer.read(s,~caminho +/+ "C3-fagote-stereo.wav"),
                Buffer.read(s,~caminho +/+ "C3-trompete-stereo.wav"),
                Buffer.read(s,~caminho +/+ "C3-piano-stereo.wav"),
        ];
};
ServerBoot.add(~inicializa);
s.waitForBoot({
        s.sync;
        fork {
                ~instOrq.do{|item|
                        3.wait;
                        {EnvDetect.ar(PlayBuf.ar(2, item, doneAction: 2),
                            0.0001, 0.0001) * PinkNoise.ar(2.25)}.play;
                        3.wait;
                };
        }.play;
});)
```

Código 3.2 – Envelopamento em tempo real

Um procedimento semelhante ao envelopamento é a chamada *substituição de ataque*, discutido na subseção 2.2.2. Como esse procedimento está originalmente ligado à uma edição minuciosa da forma de onda, sua realização no SuperCollider requer uma codificação dura[1], bem como análise prévia dos pontos de início e fim dos segmentos do envelope, conforme pode ser observado no Código 3.3. Alternativamente, se implementarmos uma solução mais automatizada para esse procedimento podemos incorrer em uma solução que é muito específica aos arquivos de sons que utilizamos.

```
~c1 = Buffer.readChannel(s, PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3) +/+ "Audios/C3-marimba-stereo.wav",channels: [1]);
~c2 = Buffer.readChannel(s, PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3)+/+ "Audios/C3-violino-stereo.wav",channels:[ 1]);
~c3 = Buffer.readChannel(s,PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3) +/+ "Audios/C3-trompa-stereo.wav",channels:[1]);

```

[1] Tradução nossa para *hard coding*, prática na qual se evita automações, iterações e estruturas abstratas, escrevendo o código de forma mais explícita o possível.

```
(SynthDef(\sampler, {| out = 0, buffer = 0, bStart = 0.0, dur = 0.145,
   curve = -10, bSus=0.3, amp = 0.001|
    Out.ar(out,
        BufRd.ar(1, buffer, Phasor.ar(0, BufRateScale.kr(buffer),  (
           SampleRate.ir * bStart), (SampleRate.ir * dur)), loop: 0.0,
           interpolation:  2).dup * amp
    ) * EnvGen.ar(Env.pairs([[bStart,0],[(dur-bStart)/2,1],[dur-bStart,
       0.0]], -3.1), doneAction: Done.freeSelf)
}).add;)

(~p1= Pbind(
        \instrument, \sampler,
        \buffer, ~c1,
        \bStart, 0.0,
        \dur, Pn(0.1028,1),
        \amp, 1.2
);
~p2= Pbind(
        \instrument, \sampler,
        \buffer, ~c2,
        \bStart, 0.081,
        \dur, Pn(~c2.duration,1),
        \amp, 0.8,
);
~p3= Pbind(
        \instrument, \sampler,
        \buffer, ~c3,
        \bStart, 0.07901,
        \dur, Pn(~c2.duration,1),
        \amp, 0.7,
);
Ptpar([
        0.0, ~p1, 0.1, ~p2,
        3.0, ~p1, 3.1, ~p3,
]).play;)
```

Código 3.3 – Substituição de ataque

O Áudio 3.4 mostra como esse procedimento dificilmente promove uma fusão entre os dois sons, mas permite uma alusão à uma mudança de fonte ou das propriedades físicas do instrumento (quando podem ser detectadas nos sons originais) devido à transição muito rápida e sem cliques – quando recorta-se os sinais nos pontos de amplitude próxima a zero.

Áudio 3.4 (3-4_substituicao-ataque.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Substituição de ataque dos sons de Áudio 3.1 – violino, trompa, fagote, trompete e piano – pelo ataque de marimba da referida gravação.

A realização manual e mais minuciosa deste procedimento não promove grandes melhoria na fusão dos dois sons, apenas melhora o corte entre os dois sons. Isso ocorre essencialmente porque as formas de ondas continuam radicalmente diferentes na transição, não tendo a evolução contínua, gradual e redundante que os sons naturais costumam apresentar. Em conformidade com o Áudio 3.5, mesmo reduzindo a duração do ataque, o efeito mais próximo que conseguimos atingir é o de quase sincronia de dois ataques ou mesmo de uma troca rápida de dois instrumentos.

Áudio 3.5 (3-5_subs-atq-manual-mrb-vln.wav). *Arquivo de áudio disponível em <http://www.musica.ufmg.br/lapis/?p=1141>.*

Substituição de ataque do som de violino (Áudio 3.1) pelo ataque do som de marimba da referida gravação. Neste caso foi executada uma edição manual da forma de onda, selecionando uma porção menor do ataque da marimba.

3.1.2 Estratégias de reprodução, granulação e microssom

Os grupos e funções do CDP para granulação, micromontagem e diferentes estratégias de reprodução de arquivos sonoros são: *sfedit*, programas que possibilitam edições e montagens diversas entre dois ou mais arquivos de áudio; *extend*, conjunto de funções que permitem estender um dado arquivo de áudio utilizando diferentes estratégias de reprodução; *grains*, grupo de funções com estratégias mais radicais de transformação de um som compostos por grãos, ou seja, formado por ondas separadas por momentos de silêncio; *modify brassage*, função que permite diversos tipos de granulação de um dado arquivo de áudio; *modify sausage*, função é semelhante à anterior, porém opera com diversos arquivos simultaneamente. Embora o CDP apresente esta variedade de funções dedicada ao tema, para realizar outras lógicas de processamento ainda se faz necessário o uso de codificações duras ou de combinações altamente complexas das suas funções. Neste texto, optamos por não investigar detalhadamente tais processo dada a dimensão deste assunto e também pelo fato de Roads (2001) e Wishart (1994) apresentarem um estudo minucioso do tema, porém apresentaremos alguns apontamentos gerais de tais técnicas.

Devido ao seu enfoque em processos algorítmicos no domínio sonoro, o SuperCollider disponibiliza um vasto e maduro leque de opções para a exploração de modificações parametrizadas. Desta maneira, apresentaremos aqui alguns procedimentos gerais para granulação, micromontagem e diferentes estratégias de reprodução no SuperCollider, cuja modificação pode levar à processos suficiente similares aos do CDP. Destaca-se ainda que tal utilização permite não somente o controle das transformações do CDP por processos

caóticos, estocásticos, recursivos e realimentados, mas, sobretudo, permite um avançado grau de controle entre continuidade e descontinuidade, possibilitando a exploração de diversas nuances do *contínuo sonoro*.

O Código 3.4, e seu arquivo sonoro correspondente, Áudio 3.6, mostram como diferentes granularidades de parâmetros sonoros semicorrelacionados contribuem para ressaltar as nuances entre o contínuo e descontínuo. Neste caso, lemos um *buffer* de forma senoidal (*SinOsc*) – indo e voltando com acelerações e desacelerações cossenoidais – porém a frequência de leitura salta discretamente com frequência contínua (*LFNoise1*). Este efeito causa uma espécie de vibrato que é acompanhado por uma modificação na largura de banda do áudio, podendo ser compreendido fisicamente como o *scratching* feito por DJs, porém com controle e precisão improváveis de serem alcançados humanamente.

Áudio 3.6 (3-6_3_galos_reprod_senoidal_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Som de três galos lidos de forma padrão, seguido deste mesmo som lido de forma senoidal, cíclica, com velocidade variando em saltos. Os sons originais foram retirados de <https://freesound.org/s/435506/>, <https://freesound.org/s/391313/> e <https://freesound.org/s/316920/>.

```
1 b = Buffer.read(s, PathName(thisProcess.nowExecutingPath).parentLevelPath
      (3) +/+ "Audios/3_galos_original_stereo.wav");
2
3 ({ BufRd.ar(2, b,
4         SinOsc.ar(
5                 LFNoise1.ar(18.0).range(0.038,0.05)) * BufFrames.ir(b)
6 ) }.play;)
7
8 b.free;
```

Código 3.4 – Leitura senoidal de velocidade variável

No limiar da turva divisão entre estratégias de reprodução com salto e granulação, costuma-se utilizar a primeira denominação quando existem padrões cíclicos de leitura ou quando a granulação não leva a criação de texturas – mantendo um fluxo auditivo coeso em um só objeto. No Código 3.5, e Áudio 3.7 seu arquivo sonoro correspondente, primeiramente transformamos o som utilizando grãos de duração mais longa e tempos de relaxamento do envelope também mais longos, levemente superpostos e a velocidade de leitura de cada grão (*rate*) varia browniamente. Um ponto importante aqui está no padrão *Pseg* que é uma estrutura do SuperCollider que faz com que os métodos de criação de envelope sejam suportados pelas estruturas de *Patterns* – em relação ao CDP, é uma maneira automática de se criar arquivos de *breakpoint* algoritmicamente e também incluir tais arquivos na lógica de eventos dos padrões do SuperCollider.

Áudio 3.7 (3-7_3_galos_reprod_granular_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Som de três galos lidos de forma padrão, seguido deste mesmo som lido de forma granular, com velocidade de reprodução variando brownianamente e a posição de leitura variando conforme o envelope descrito no Código 3.5. Os sons originais foram retirados de <https://freesound.org/s/435506/>, <https://freesound.org/s/391313/> e <https://freesound.org/s/316920/>.

```
SynthDef(\tocador, {
        |out=0, pan=0, buf=0, rate=1.0, amp=0.6, pos=0.0, rel=1.0|
        var sig, env;
        env = EnvGen.ar(Env.perc(0.001,rel), doneAction:2);
        sig = PlayBuf.ar(2, buf, BufRateScale.ir(buf) * rate, 1,
                BufFrames.kr(buf) * pos);
        OffsetOut.ar(out, Pan2.ar(sig, pan, amp) * env);
}).add;

~galo = Buffer.read(s, PathName(thisProcess.nowExecutingPath).
   parentLevelPath(3) +/+ "Audios/3_galos_original_stereo.wav");

(Pbindef(\t1,
        \instrument,\tocador,
        \buf, ~galo.bufnum,
        \rate, Pbrown(0.75, 1.25, 0.01),
        \pos, Pseg(Pseq([0.01,0.4,0.9,0.5]),Pseq([8,2,16]),\sine),
        \dur, 0.25,
        \rel, 1.5).play;)

Pbindef(\t1, \dur, 0.05, \rel, 0.1).play;
```

Código 3.5 – Leitura senoidal de velocidade variável

O reprodutor de som anterior apresenta problemas práticos para funcionar como granulador ou motor de transformações microssonoras, especialmente por não suportar uma maior quantidade de eventos por segundos (densidade de textura), seja na criação mais frequente de eventos ou por uma maior quantidade de eventos criados a cada instante. Quando deseja-se atingir tal nível de densidade, é recomendável utilizar um granulador implementado diretamente no servidor e otimizado para atuar de tal forma. O Código 3.6, e seu arquivo sonoro correspondente Áudio 3.8, mostra uma das implementações de síntese granular nativas do SuperCollider.

Dentro da função (código entre colchetes { }) há somente *UGens*, portanto estamos realizando todo o processamento no lado do servidor. Além disso, quando há a necessidade de se implementar alguma sequência ou estrutura mais específica utilizamos os *Demand UGen*, neste caso o *Dxrand* para circular aleatoriamente dentro de uma lista sem repetir

os elementos consecutivamente. Este tipo de UGen garante que possamos enviar valores escolhidos no lado da linguagem para o lado do servidor com precisão e sincronia com o *clock* do servidor.

Áudio 3.8 (3-8_3_galos_textura_granular_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Som de três galos lidos de forma padrão, seguido deste mesmo som lido de forma granular, com sobreposição, velocidade e pan variando estocasticamente. Os sons originais foram retirados de <https://freesound.org/s/435506/>, <https://freesound.org/s/391313/> e <https://freesound.org/s/316920/>.

```
1  ~galo = Buffer.readChannel(s, PathName(thisProcess.nowExecutingPath).
       parentLevelPath(3) +/+ "Audios/3_galos_original_stereo.wav", channels:
       [0]);
2  ({
3      var trate, dur, clk, pos, pan, rate;
4      trate = LinExp.kr(LFTri.kr(15),-1,1,8,120);
5      dur = LFDNoise1.ar(2).range(5,12) / trate;
6      clk = Impulse.ar(trate);
7          rate = Dxrand([1.0, 0.9, 1.1, 0.8, 1.2], inf);
8      pos = LFSaw.ar(BufDur.ir(~galo).reciprocal).range(0, BufFrames.ir(~galo
           ));
9      pan = WhiteNoise.kr(0.6);
10     TGrains.ar(2, clk, ~galo, rate, pos, dur, pan, 0.1);
11 }.play;)
```

Código 3.6 – Criação de textura granular a partir de leitura de som monofônico

3.1.3 Transformações de conjuntos de onda

A descrição e referência mais atualizada das transformações de conjuntos de onda do CDP está em CDP (2015), todavia explicações mais detalhadas bem como ilustrações dos procedimentos podem ser encontradas ao longo de Wishart (1994). Como a nomenclatura das funções do CDP variou bastante ao longo dos anos, optamos aqui em utilizar uma tradução dos termos mais atuais, conforme a Tabela 3.1.3.

Tabela 1 – Transformações de *conjuntos de ondas* (*wavesets*)

Transformação	Descrição
Ponderação (distort average)	escala conjuntos de onda adjacentes e pondera seu comprimento e amplitude
Apagamento (*distort delete*)	contração temporal de um arquivo de áudio apagando seus *conjuntos de onda*, ou mais simplemeste somente deixando de lê-los
Divisão (*distort divide*)	sem alterar a duração, substitui cada bloco de n conjuntos de onda pelo seu primeiro conjunto-de-onda (esticado para caber neste espaço)
Envelopamento (*distort envel*)	Impõe um envelope em cada grupo de N conjuntos de onda

Filtragem (*distort filter*)	Comprime temporalmente um som descartando conjuntos de onda de acordo com uma dada regra
Fractalização (*distort fractal*)	Superpõe cópias em miniatura da fonte sobre ela mesma
Ressíntese aditiva (*distort harmonic*)	acrescenta conjuntos de onda com velocidade aumentadas (dobradas, triplicadas) à cada conjunto de onda, depois pondera e as soma.
Interação (*distort interact*)	Substitui ou impõe o comprimento de conjuntos de onda de um arquivo A em um arquivo B
Esticamento temporal (*distort interpolate*)	repete cada grupo ou conjunto de onda n vezes (criando "miçangas"de alturas)
Multiplicação (*distort multiply*)	sem alterar a duração, substitui o conjuto-de-onda original por n cópias(comprimidas para caber neste espaço)
Omissão (*distort omit*)	Substitui conjuntos de onda por silêncios
Saturação (*distort overload*)	aumenta a amplitude de um conjunto de onda, mas grampeia seu valor em um dado limiar (*cliping*)
Altura Aleatória (*distort pitch*)	Retorce aleatoriamente a altura dos conjuntos de onda
Pulsação (*distort pulsed*)	Impõe pulsações regulares em um som
Substituição (*distort Reform Substitution*)	Em um dado grupo de conjutos-de-onda, substitui os conjuntos de onda mais fracos pelo conjunto de onda mais forte do grupo
Repetição (*distort Repeat*)	estica o arquivo de som repetindo seus conjuntos de onda
Repetição 2 (*distort Repeat 2*)	repete os *conjuntos de onda* sem esticar o arquivo de som
Sustentação *distort replace*	Substitui os conjuntos de onda menos intensos de um grupo pelos mais fortes
Repetição limitada (*distort replim*)	Repete somente os conjuntos de onda de um dado tamanho
Reversão (*distort reverse*)	toca em reversão cada conjunto ou grupo de conjuntos de onda.
Permutação (*distort shuffle*)	altera, dentro de um grupo de conjuntos de onda, a posição relativa entre os elementos
Telescopia (*distort telescope*)	Para um dado grupo de conjuntos de onda, as ondas curtas são esticadas e mixadas (somadas) com a onda mais longa, comprimindo temporalmente o sinal
Inversão (*distort Reform Inversion*)	inverte, em relação ao eixo temporal, cada metade de um conjunto de onda

Tradução e adaptação nossa de CDP (2019b), CDP (2015), Roads (2001, p. 207-208)

Como a descrição do site oficial do CDP não desvela detalhadamente os algoritmos utilizados bem como as explicações em Wishart (1994) muitas vezes são desatualizadas, nas reimplementações em que se teve dúvida sobre o algoritmo executado optou-se por avaliar graficamente as formas de onda das saídas do CDP para verificação experimental dos procedimentos. Como alguns pontos problemáticos foram encontrados em tais algoritmos, modificações básicas são descritas ao longo de cada processo.

Conforme mencionado na seção 2.1 a arquitetura cliente-servidor do SuperCollider permite ao cliente um controle de alto nível dos parâmetros sonoros às custas de um menor

controle dos parâmetros computacionais. Essa diferença se torna mais acentuada à medida que os parâmetros temporais controlados pelo cliente se aproximam das grandezas da taxa de amostragem, ou seja, à medida que as manipulações feitas se aproximam cada vez mais da manipulação direta de *samples* de áudio, à medida que entramos do domínio dos microsons (ROADS, 2001).

A realização de transformações de microssom em tempo real é computacionalmente intensa, mesmo para CPUs modernas. Para lidar com essa questão no SuperCollider, temos duas estratégias básicas: ou implementamos o procedimento microssonoro no servidor de forma específica porém otimizada ou utilizamos o cliente com um alto nível de abstração para enviar mensagens ao servidor em uma escala temporal na qual haja tempo o suficiente o processamento e tempo de envio das mensagens. Em termo práticos, se decidirmos utilizar *patterns* para enviar mensagens de transformação de microssons no servidor, essas mensagens devem ser muito leves ou, se forem pesadas, acontecer numa escala de tempo mais lenta.

A primeira implementação das transformações de conjuntos de onda no SuperCollider[2] foi feita por Campo (2020), Wilson, Cottle e Collins (2011, p. 491-500), no formato de *Quark*, utilizando exclusivamente o lado do cliente via *sclang* e limitando-se às funcionalidades mais centrais de tais transformações. Uma atualização dessa implementação encontra-se em Campo, Bovermann e Rohrhuber (2020), possivelmente ainda em desenvolvimento, e será a principal utilizada ao longo desse texto. Há ainda duas implementações feitas do ponto de vista da linguagem Mayer (2020) e Hochherz (2008) e uma implementação de tempo real[3] Seidl (2016), que naturalmente não contempla todas as possibilidades de tempo diferido e atualmente encontra-se em caráter experimental.

A implementação de conjuntos de onda com a arquitetura de *Patterns*, ou seja controle pelo lado do cliente, nos permite uma grande maleabilidade na definição e execução das transformações, porém, frequentemente perdemos a garantia da baixa latência (menos do que vinte milissegundos) ou mesmo de uma execução síncrona determinística (saber exatamente quanto tempo de processamento cada operação demandará). Por outro lado, uma possível implementação de transformações de conjuntos de onda sob a premissa de tempo real, executada no lado do servidor, implicaria na impossibilidade de se operar com grandes quantidades de conjuntos de onda, ou com uma análise mais longa de som, pois frequentemente incorreríamos na situação de um bloco de processamento não conter a quantidade necessária de *conjuntos de onda* para uma desejada transformação.

[2] Todas as implementações deste capítulo, tanto as de conjuntos de ondas quanto as do vocoder de fase, são baseadas ou adaptadas da literatura mencionada aqui ou da própria documentação do SuperCollider, sempre que houver a utilização de alguma fonte mais específica, mencionaremos os detalhes.

[3] Existe ainda uma proposta de linguagem de programação específica para trabalho com microsons (NISHINO, 2014), originalmente uma linguagem para controle do *scsynth* (NISHINO; OSAKA, 2012) que evoluiu também para um mecanismo de síntese e motor de áudio, que contempla diversas possibilidades de trabalho com conjuntos de onda.

O funcionamento de Campo, Bovermann e Rohrhuber (2020) consiste na leitura de um arquivo de som, armazenamento em um *buffer* e análise completa deste arquivo em tempo diferido[4] (o que é feito pelo método *.read*). Em seguida carregamos as definições dos *SynthDef* por meio do método *.prepareSynthDefs*, adicionamos os *WavesetsEvent* à um dicionário interno (*.add*), e por fim executamos a leitura dos conjuntos-de-onda utilizando a classe que acharmos mais conveniente na linguagem (*Task*, *Routine*, *Pattern*, etc). O *SynthDef* para execução de transformações de conjuntos de onda é essencialmente um leitor de pequenos segmentos, ou seja, irá ler cada conjunto de onda um a um conforme o Código 3.7.

```
1 SynthDef(\wvst1gl, { | out = 0, buf = 0, startFrame = 0, numFrames = 441,
      rate = 1, rate2 = 1, sustain = 1,
2         amp = 0.1, pan, interpolation = 2 |
3         var rateEnv = Line.ar(rate, rate2, sustain) * sign(numFrames);
4         var phasor = Phasor.ar(0, BufRateScale.ir(buf) * rateEnv, 0, abs(
              numFrames)) + startFrame;
5         var env = EnvGen.ar(Env([amp, amp, 0], [sustain, 0]), doneAction:
              2);
6         var snd = BufRd.ar(1, buf, phasor, 1, interpolation) * env;
7
8         OffsetOut.ar(out, Pan2.ar(snd, pan));
9 }, \ir.dup(10)).add;
```

Código 3.7 – *SynthDef* leitor de conjuntos de onda

Na análise de arquivo de áudio, um padrão de evento é criado, daí o nome *WavesetEvent*, que mapeia as seguintes chaves para os argumentos do *SynthDef*:

- *start* – conjunto-de-onda inicial
- *num* – quantidade de conjuntos de onda por evento
- *end* – determina o último conjunto-de-onda a ser tocado (se for fornecido, sobrepõe *num*)
- *repeats* – quantas vezes o conjunto de onda deste evento será repetido
- *startTime* – instante de início de leitura dentro do *buffer*, se fornecido sobrescreve *start*
- *endTime* – instante de final de leitura dentro do *buffer*, se fornecido sobrescreve *end*
- *amp* – altera a amplitude do arquivo de som como um todo
- *wsamp* – normaliza a amplitude do conjunto-de-onda
- *rate* – velocidade de reprodução do arquivo de áudio
- *rate2* – velocidade final de reprodução do arquivo de áudio (se utilizado, permite variação na taxa de leitura) de cada conjunto

[4] A análise dos conjuntos de onda consiste resumidamente na medição de pontos de cruzamento de sinal – mudança de sinal da onda – conjuntamente com um limiar que definição qual é o tamanho mínimo dos conjuntos de onda.

- *pan* – panorama estéreo
- *useFrac*[5] – valor booleano que habilita a utilização de valores fracionários para os índices dos conjuntos

A execução dos eventos por meio de padrões pode ser feita via a classe *Pwavesets*, que permite o encapsulamento de num formato próximo ao estilo de *Pbind*. Em Código 3.8 temos a definição mais genérica possível para os casos em questão:

```
c = WavesetsEvent.read(Platform.resourceDir +/+ "sounds/a11wlk01.wav");
WavesetsEvent.prepareSynthDefs;
c.add(\somPadrao);
(Pwavesets(
        Pbind(
                \name, \somPadrao, // buffer a ser utilizado
                \start, Pn(Pseries(0, 1, 3435),1), // toca o buffer sem
                   saltos e até o final
                \num, 1, // quantidade de waveset que será lida do buffer
                   em um evento
                \repeats, 1, //quantas vezes irá tocar cada waveset
                \rate, 1.0, // velocidade de leitura
                \amp, 0.1, //amplitude
                \pan, 0, // pan estéreo
        )
).play;)
```

Código 3.8 – *Pattern* leitor de conjuntos de onda

É nitidamente perceptível que os algoritmos de conjuntos de onda do CDP, em comparação aos do SuperCollider, apresentam uma maior quantidade de ruído espúrios – e.g. picos bruscos em senoides em pontos dos conjuntos de ondas nos quais não se realizou nenhuma modificação – o que ocorre possivelmente devido à não utilização da metodologia de cálculo de cruzamentos de zero pelo CDP. O SuperCollider também apresentou um problema similar, cuja causa exata não foi possível detectar: ciclicamente em alguns pontos dos cruzamentos de zeros há repetição de duas amostras. Isso faz com que a execução da leitura regular, sem transformação, dos conjuntos de onda via *Pwavesets* soe com um ruído maior do que a leitura típica de um *buffer* carregado com o mesmo arquivo.

3.1.3.1 Apagamento de conjuntos de onda (*Distort Delete*)

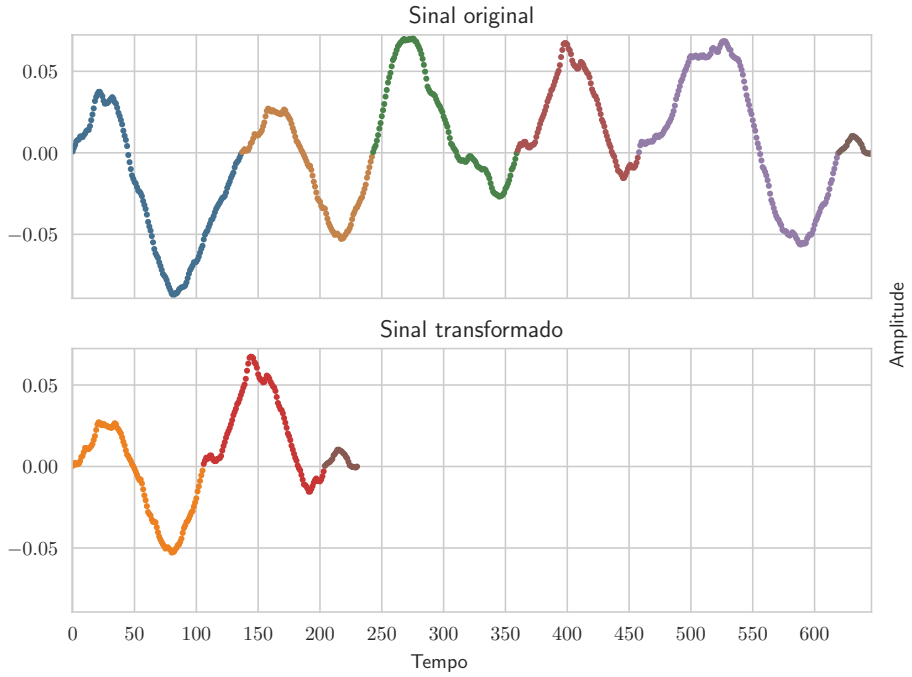
3.1.3.1. a Definição

O apagamento de conjuntos de onda consiste na contração temporal de um arquivo de áudio apagando seus *conjuntos de onda*, ou mais simplesmente somente deixando de

[5] A utilização desta chave costuma introduzir ruídos, mesmo quando é executada apenas a leitura regular, sem transformação, dos conjuntos de onda. Como tal parâmetro parece ainda está em implementação, tenderemos a fixá-lo como falso

lê-los, realizando uma leitura em saltos, conforme ilustra a Figura 16[6]. No caso da escolha de parâmetros de apagamento igual a dois, teremos dois conjuntos lidos, seguidos de dois apagados, depois mais dois lidos, seguidos de dois apagados e assim por diante.

Figura 16 – Apagamento de conjuntos de onda



Fonte: Elaborada pelo autor

3.1.3.1. b Sonoridade

Como os sinais sonoros tendem a ser contínuos e de alta redundância, quanto maior for a quantidade de conjuntos de onda apagados, maior será a tendência a termos transições abruptas entre os conjuntos, o que causa aumento de frequências de alta ordem e também traz sons ruidosos e impulsivos.

Por estarmos reduzindo a informação do sinal, descartando conjuntos de onda, naturalmente teremos um tipo de distorção um pouco similar ao esmagamento de bits[7].

3.1.3.1. c Exemplos

No Código 3.9 mostramos como é possível realizar o apagamento sucessivo de um mesmo som com diferentes parâmetros de apagamento bem como é possível realizar

[6] Os gráficos das transformações de conjuntos de onda apresentados neste capítulo foram obtidos por meio de uma portabilidade que fizemos dos códigos de (CAMPO, 2020) para a linguagem Python.

[7] Tradução nossa para o termo *bitcrushing*, um tipo de efeito sonoro digital de distorção no qual se reduz a taxa de amostragem descartando amostras e/ou se reduz a quantização por meio da diminuição do número de bits que guardam os valores de amplitude.

um apagamento contínuo – circulando dentro do *buffer* – com uma taxa de apagamento variável. Neste exemplo, o passo de apagamento muda a cada evento, variando entre um e oito, e o quantidade total de conjuntos de onda que iremos ler varia a cada três eventos, podendo ir de 64 a 214 conjuntos de onda.

Áudio 3.9 (3-9_eletron_volt_apag_conj_ond.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Próprio autor pronunciando "elétron-volt", seguido deste som com apagamento de conjuntos de onda com o parâmetro de apagamento (grupos de conjuntos de onda apagados) sendo 1, 2, 3, 4, 6 e 8.

O Áudio 3.10 – resultante do Código 3.9 – mostra como é possível migrar de um som vocálico para um som com características sonoras típicas de motor – presença de vibrações ruidosas, repetições paralelas fora de fase, filtragem com frequência variável, etc – como é o caso do som de motor a diesel do Áudio 3.11.

```
~w = WavesetsEvent.read(PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3) +/+ "Audios/eletron_volt.wav");
WavesetsEvent.prepareSynthDefs;
~w.add(\eletronvolt);
(Pwavesets(
        Pbind(
                \name, \eletronvolt, // identificação do buffer
                \start, Pseq([
                        Pn(Pseries(0, 1, (1603/1).asInt), 1),
                        Pn(Pseries(0, 2, (1603/2).asInt), 2), // apaga 2
                        Pn(Pseries(0, 3, (1603/3).asInt), 3), // apaga 3
                        Pn(Pseries(0, 4, (1603/4).asInt), 6), // ...
                        Pn(Pseries(0, 5, (1603/5).asInt), 7),
                        // taxa de apagamento variável
                        Pn(Pn(Pseries(0, Pwhite(1,8),  Pstutter(3,Pbrown
                            (64,214,20,inf)).asStream),5),40)
                ])
        )
).play;)
```

Código 3.9 – Apagamento de Conjunto-de-Onda

Áudio 3.10 (3-10_ev_conj_apag_var_motor.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Próprio autor pronunciando "elétron-volt"; este som com apagamento de conjuntos de onda por parâmetros (1, 2, 3, 4 e 5) e tocados em *loop* 0, 1, 2,3, 6 e 7 vezes; apagamento contínuo – circulando dentro do *buffer* – com uma taxa de apagamento variável.

3.1.3.2 Divisão e multiplicação conjuntos de onda (*Distort Divide* e *Distort Multiply*)

3.1.3.2. a Definição

A *divisão de conjunto-de-onda* consiste em, a cada grupo de conjuntos de onda, retirar alguns conjuntos e interpolar os outros para que caibam no mesmo segmento temporal. Assim, em uma divisão de ordem 2, a cada 2 conjuntos de onda, descartamos o segundo e interpolamos o primeiro para caber no espaço ocupados anteriormente pelos outros dois conjuntos. A *multiplicação de conjunto-de-onda* por um fator n consiste em repetir o conjunto-de-onda n vezes interpolando para que sua duração seja n vezes menor, ou seja, lendo o arquivo n vezes mais rápido.

3.1.3.2. b Sonoridade

Os processos de divisão e multiplicação de conjuntos de onda são tentativas de se alterar a altura de um dado som sem alterar a duração do mesmo, ou seja, uma forma modificação da escala de altura. Esse procedimentos possivelmente são os mesmos dos chamados *transposição de conjunto-de-onda* em Wishart (1994, p. 51, apênd. 2) e Wilson, Cottle e Collins (2011, p. 498).

3.1.3.2. c Exemplos

Como esse referido processo de interpolação é equivalente a uma mudança na velocidade de leitura do *buffer*, aproximaremos o processo de divisão de conjuntos de onda no SuperCollider por uma divisão de n vezes a velocidade de leitura dada a ordem n de descarte de conjuntos de onda, conforme Código 3.10 .

```
1  ~w1 = WavesetsEvent.read(PathName(thisProcess.nowExecutingPath).
       parentLevelPath(3) +/+ "Audios/f4000-stereo.wav");
2  WavesetsEvent.prepareSynthDefs;
3  ~w1.add(\f4000);
4
5  (Pwavesets(
6          Pbind(
7                  \name, \f4000,
8                  \start, Pn(Pseries(0, 2, (57092/2).asInt),1), // salta de
                       dois em dois
9                  \rate, 0.5, // metade da velocidade de leitura original
10         )
11 ).play;)
```

Código 3.10 – Divisão de *Conjunto-de-Onda*

A realização deste processo é feita como no Código 3.11 [8].

```
~w1 = WavesetsEvent.read(PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3) +/+ "Audios/f4000-stereo.wav");
WavesetsEvent.prepareSynthDefs;
~w1.add(\f4000);

(Pwavesets(
        Pbind(
                \name, \f4000,
                \start, Pn(Pseries(0, 1, 57092),1), // toca o buffer sem
                    saltos e até o final
                \repeats, 2, //quantas vezes irá tocar cada waveset
                \rate, 2.0, // velocidade de leitura
        )
).play;)
```

Código 3.11 – Multiplicação de *Conjunto-de-Onda*

Este processo costuma não apresentar fortes artifícios quando utiliza sons ruidosos. Escolhemos um som de caminhonete diesel dando partida para exemplificar as alterações, conforme Áudio 3.11: no caso da divisão de conjuntos de onda, o som se torna mais grave e os transitórios mais lentos, a modificação no som faz parecer que migramos de um som de caminhonete para um som de caminhão, ou uma máquina industrial mais lenta; no caso da multiplicação, como temos alturas dobradas, a modificação alude à migração de um som de caminhonete para um som de automóvel pequeno ou de motocicleta.

Áudio 3.11 (3-11_f4000_cnjonda_mul_div.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Caminhonete F4000 diesel tentando dar a partida, seguida da divisão de conjuntos de onda por fator 2 do som do som fonte, por último multiplicação de conjuntos de onda por fator 2 do som do som fonte. Som fonte retirado de <https://freesound.org/s/72027/>.

3.1.3.3 Inversão de conjuntos de onda (*Distort Reform 5*)

3.1.3.3. a Definição

Na *inversão de conjuntos de onda* - inverte-se, em relação ao eixo temporal, cada metade de um conjunto de onda (WISHART, 1994, p. 51, apênd. 2) e (WILSON; COTTLE; COLLINS, 2011, p. 498).

[8] Devido à alta redundância, daqui em diante iremos omitir a etapa de carregamento e análise dos arquivos de som, bem como o carregamento dos *SynthDefs*. Como o acréscimo de operações do lado da linguagem costuma sobrecarregar os *Patterns*, também evitaremos o cálculo do tamanho do *buffer* na chave *start*, utilizando o valor explícito do tamanho desejado.

3.1.3.3. b Sonoridade

Este processo costuma gerar harmônicos de alta ordem pois, caso ocorra a inversão num ponto em que a onda vinha de uma excursão de valores decrescentemente negativos e depois seguia para valores crescentemente positivos, teremos a formação de uma espécie de "bico" ou ponta no cruzamento de zero. Outro fenômeno escutado é que a inversão de conjuntos de onda costuma introduzir ou trazer à tona glissandos internos à estrutura sonora.

3.1.3.3. c Exemplos

Ambos fenômenos explicados na seção anterior podem ser escutados noÁudio 3.12.

Áudio 3.12 (3-12_inv_conj_onda_C3_mrb_vln_tro_fgt_tru_pno.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.* Nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano; em seguida é aplicada uma inversão de conjuntos de onda a esta sequência de som, utilizando um conjunto invertido em cada grupo de dois conjuntos lidos; em seguida a mesma transformação porém com um conjunto invertido em cada grupo de oito conjuntos lidos; idem porém com um conjunto invertido em cada grupo de doze conjuntos lidos. Sons de intrumentos orquestrais retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

Para a geração do Áudio 3.12 utilizamos o Código 3.12 numa das maneiras típicas de utilização de linguagens interpretadas: para cada nova transformação, retirou-se os comentários da chave *\amp* desejada e executou-se o código. Daqui em diante neste texto iremos omitir as chaves não modificadas – neste exemplo *\num*, *\repeats*, *\rate*, etc – pois os valores padrão de tais chaves serão utilizados.

```
(Pwavesets(
        Pbind(
                \name, \somPadrao, // buffer a ser utilizado
                \start, Pn(Pseries(0, 1, 24185),1), // toca o buffer sem
                   saltos e até o final
                \amp, Pseq([0.4, -0.4], inf), //amplitude
/*              \amp, Pseq(
        [0.4, 0.4, 0.4, 0.4,
        0.4, 0.4, 0.4, -0.4], inf), // um em cada 8 */
/*              \amp, Pseq(
        [0.4, 0.4, 0.4, 0.4,
         0.4, 0.4, 0.4, 0.4,
         0.4, 0.4, 0.4, -0.4], inf), // um em cada 12 */
                \useFrac, false,
        )
).play;)
```

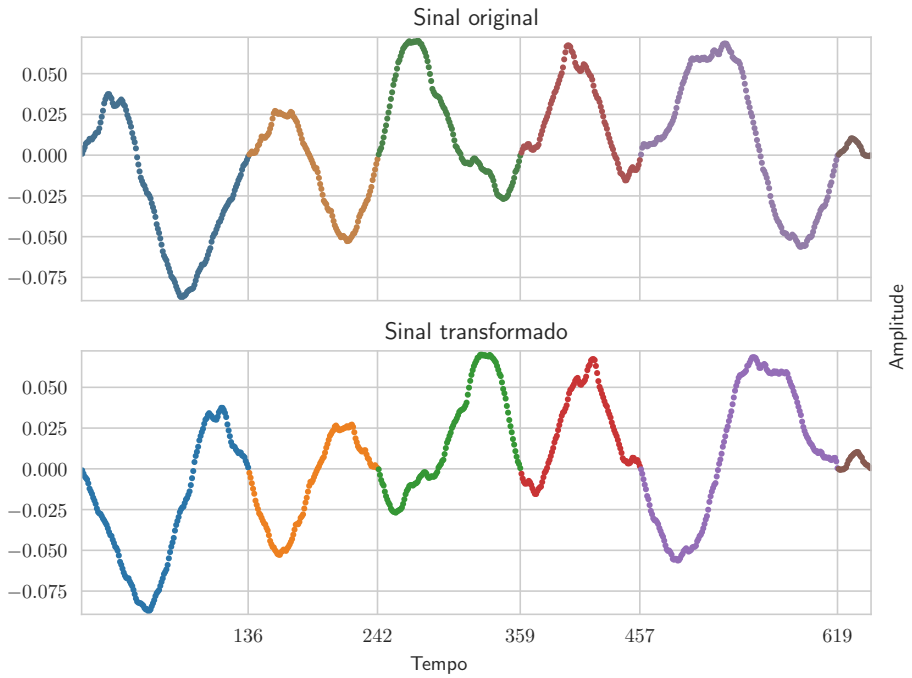
Código 3.12 – Inversão de conjunto de onda

3.1.3.4 Reversão de conjuntos de onda (*Distort Reverse*)

3.1.3.4. a Definição

Na *reversão de conjuntos de onda* toca-se em reversão cada conjunto ou grupo de conjuntos de onda, conforme ilustra a Figura 17.

Figura 17 – Reversão de conjuntos de onda



Fonte: Elaborada pelo autor

3.1.3.4. b Sonoridade

Esse procedimento pode causar transições abruptas, como é o caso da transição entre os dois últimos conjuntos transformados da Figura 17, especialmente quando há algum tipo de componente contínua em pontos específicos do sinal. Não há aqui a sonoridade de áudio sendo tocado de trás para a frente, fisicamente o que parece ocorrer é uma alternância de inversão de fase.

A principal característica sonora observada, de maneira geral, foi uma espécie de dissolução dos sons ruidosos e ligeira acentuação dos sons tônicos.

3.1.3.4. c Exemplos

No Áudio 3.13, realizado por meio do Código 3.13, é possível perceber como as transformações de conjuntos de onda variam sua sonoridade em relação às características tipomorfológicas de cada som. Em especial, vemos que sempre que o ataque e as porções finais do som (decaimento, sustentação e relaxamento) apresentarem sonoridades suficientemente diferentes, o efeito das transformações também apresentará uma proporcional diferença de sonoridade. Isso pode ser percebido na grande diferença de sonoridade entre o ataque e a porção final dos sons de marimba e do piano, todavia é bem menos discrepante nos sons do violino, trompa, fagote e trompete.

Áudio 3.13 (3-13_revers_conj_onda_C3_mrb_vln_tro_fgt_tru_pno.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano; em seguida é aplicada uma reversão de conjuntos de onda à esta sequência de som, utilizando um conjunto revertido em cada grupo de dois conjuntos lidos; em seguida a mesma transformação porém com um conjunto revertido em cada grupo de oito conjuntos lidos; idem porém com um conjunto revertido em cada grupo de doze conjuntos lidos. Sons de instrumentos orquestrais retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

```
(Pwavesets(
        Pbind(
                \name, \somPadrao, // buffer a ser utilizado
                \start, Pn(Pseries(0, 1, 24185),1), // toca o buffer sem
                    saltos e até o final
                \amp, 0.8, //amplitude
                \rate, Pseq([1.0, -1.0], inf), //velocidade de leitura
/*              \rate, Pseq(
        [1.0, 1.0, 1.0, 1.0,
        1.0, 1.0, 1.0, -1.0], inf), // um em cada 8 */
/*              \rate, Pseq(
        [1.0, 1.0, 1.0, 1.0,
         1.0, 1.0, 1.0, 1.0,
         1.0, 1.0, 1.0, -1.0], inf), // um em cada 12   */
                \useFrac, false,
        )
).play;)
```

Código 3.13 – Reversão de conjunto de onda

É importante ressaltar que as transformações aqui mostradas estão em estado original, sendo que na prática composicional dos usuários do CDP vários outros processos costumam ser aplicados. Um dos mais básicos, e também mais utilizados, destes é a ideia

de balanço *dry/wet*: dois arquivos são tocados simultaneamente, um com a transformação e o outro sendo o arquivo original e a diferença de volume entre tais sons constituindo este balanço. No Áudio 3.14 podemos ouvir a diferença entre a transformação pura e seu som mixado com balanço em 30%.

Áudio 3.14 (3-14_revers_grupo_conj_onda_C3_mrb_vln_tro_fgt_tru_pno.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Sons de instrumentos de orquestra transformados por reversão de seis conjuntos de onda a cada 12 grupos de conjuntos, seguidos de sua respectiva transformação com balanço *dry/wet* de 30% (ver Código 3.14). Na ordem de aparição, os sons utilizados foram uma nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano; retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

O Áudio 3.14 mostra que para transformações mais radicais nem sempre a ideia de balanço *dry/wet* é trazer à tona o som original, mas sim uma versão intermediária com novas características sonoras. Neste caso é possível perceber que geralmente há o acréscimo de uma lenta modulação de amplitude após a realização de tal balanço.

O Código 3.14 mostra uma das formas mais compactadas dos SuperCollider de realizar este tipo mixagem: a expansão multicanal. Aqui, o par de números dentro das chaves, por exemplo $[1.0, -1.0]$ irá tocar dois eventos paralelos, um com cada valor.

```
1  (Pwavesets(
2          Pbind(
3                  \name, \somPadrao, // buffer a ser utilizado
4                  \start, Pn(Pseries(0, 1, 24185),1), // toca o buffer sem
                       saltos e até o final
5                  \amp, 0.6, //amplitude
6                  \rate, Pseq(
7          [ 1.0,  1.0,  1.0,  1.0,
8            1.0,  1.0, -1.0, -1.0,
9           -1.0, -1.0, -1.0, -1.0], inf), // seis em cada 12
10                 \useFrac, false,
11         )
12 ).play;)
13
14 (Pwavesets(
15         Pbind(
16                 \name, \somPadrao, // buffer a ser utilizado
17                 \start, Pn(Pseries(0, 1, 24185),1), // toca o buffer sem
                       saltos e até o final
18                 \amp, [0.7, 0.3]*0.6, //amplitude [dry,wet]
19                 \rate, Pseq(
20         [[1.0,1.0],  [1.0,1.0],  [1.0,1.0],  [1.0,1.0],
21          [1.0,1.0],  [1.0,1.0],  [1.0,-1.0], [1.0,-1.0],
```

```
22          [1.0,-1.0], [1.0,-1.0], [1.0,-1.0], [1.0,-1.0]], inf), // seis em
                cada 12
23                  \useFrac, false,
24          )
25  ).play;)
```

Código 3.14 – Reversão de grupos de conjunto de onda e comparação com balanço *dry/wet*

No caso de sons mais ruidosos, esta transformação é menos agressiva e supreendemente apresenta o efeito de não atenuar os sons tônicos que se encontram misturados aos ruidosos, ressaltando-os levemente como é o caso do Áudio 3.15. Neste som, os glissandos cíclicos da partida do motor são atenuados – ou talvez encobertos ciclicamente pelos novos ruídos inseridos.

Áudio 3.15 (3-15_revers_grupo_conj_onda_f4000.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Som de caminhonete F4000 diesel tentando dar a partida, seguida da reversão de conjuntos de onda deste som – um conjunto revertido a cada dois. Som fonte retirado de <https://freesound.org/s/72027/>.

3.1.3.5 Omissão de conjuntos de onda (*Distort omit*)

3.1.3.5. a Definição

Na *omissão de conjuntos de onda*, insere-se silêncios dentro de um grupo de conjuntos de onda, ou de outro modo, toca-se os conjuntos-de-onda espaçadamente, conforme a Figura 18.

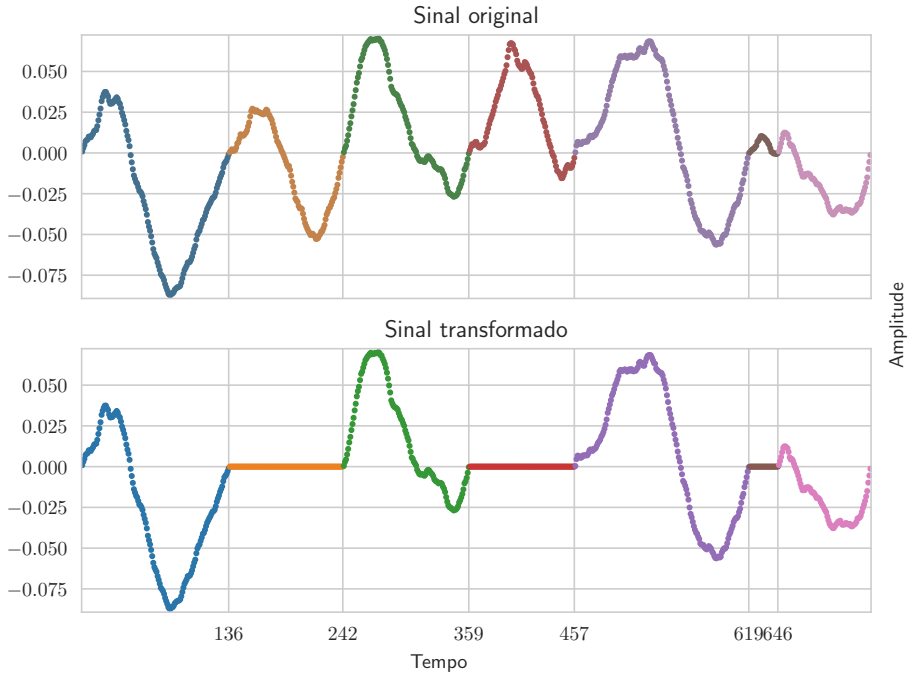
3.1.3.5. b Sonoridade

Este procedimento pode ser compreendido como uma espécie de granulação na qual há uma redução da informação total presente no sinal, um descarte de amostras em blocos que introduz pulsações no som original.

3.1.3.5. c Exemplos

O Código 3.15 mostra uma maneira idiossincrática dos *Patterns* de realizar a inserção de silêncios: em primeiro lugar toca-se um evento de silêncio (um evento no qual nada acontece) por meio da classe *Rest()* ou dos seus encurtadores \ e *rest*; em segundo lugar utilizar-se um par chave-valor no qual a chave não representa nenhum argumento no *Synth*, o que faz com que controlemos somente os eventos internos ao padrão – nesse caso a chave *nada* fará o macroevento alternar entre eventos de silêncio e eventos de execução do *Synth*.

Figura 18 – Omissão de conjuntos de onda



Fonte: Elaborada pelo autor

```
(Pwavesets(
        Pbind(
                \name, \somPadrao,
                \start, Pn(Pseries(0, 1, 37194),1),
                \nada, Pxrand([1, Rest()], inf),                    //  Rest()
                    como silêncio
                //\nada, Pxrand([1, Rest(), Rest()], inf),
                //\nada, Pxrand([1, \, \, \, \], inf),  //  \ como silêncio
                //\nada, Pxrand([1, 1, \rest], inf),                    //
                     \rest como silêncio
                //\nada, Pxrand([1, 1, 1, \], inf),
                \useFrac, false,
        )
).play;)
```

Código 3.15 – Omissão de Conjunto-de-Onda

O Áudio 3.16 apresenta a sucessão de transformações produzidas no Código 3.15. Aqui, optamos por alterar a definição do *Quark* para permitir uma maior resolução nas transformações de conjuntos de onda[9]. Esse tipo de transformação tende a ser menos agressiva em sons ruidosos, muitas vezes gerando consequências sonoras bastante sutis –

[9] Isso pode ser feito alterando o arquivo *Wavesets2.sc* na linha "*minLength = minLength ? 10;* ", na qual o valor 10 representa o tamanho do menor conjunto de onda em amostras.

como é o caso dos dois últimos exemplos do referido áudio. Além disso, quando queremos alterar as características do ruído de um som sem utilizar processos que suavizem a percepção do mesmo – como é o caso da filtragem – este procedimento pode auxiliar na fornecimento sons ruidosos intermediários.

Áudio 3.16 (3-16_omissao_conj_onda_gongo_.wav). *Disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

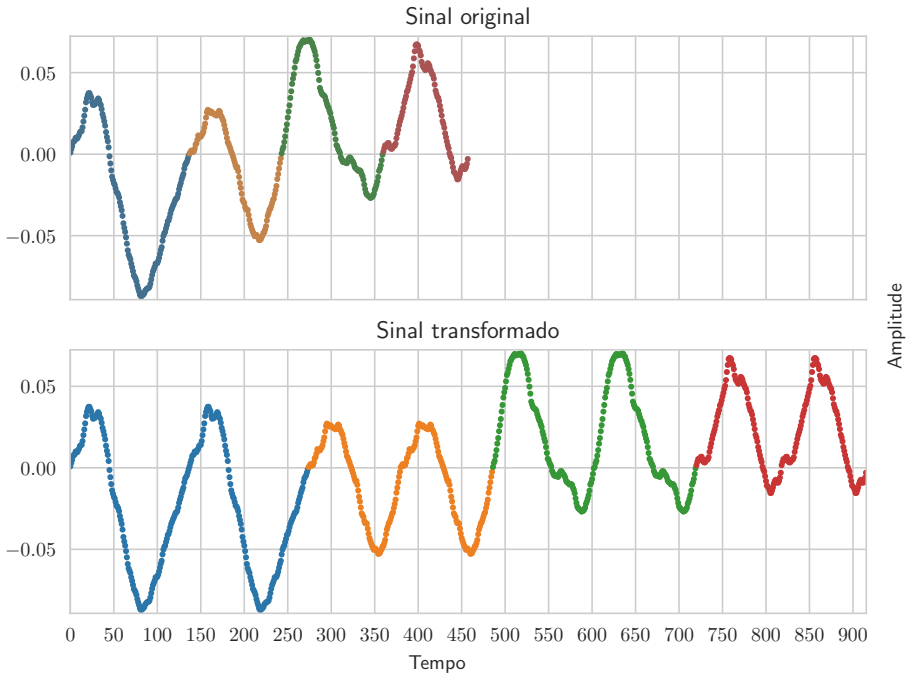
Conforme ordem apresentada em Código 3.15, som original de um gongo de orquestra; omissão, em ordem aleatória, de um conjunto de onda a cada grupo de dois; idem, porém omissão de dois conjuntos de onda a cada grupo de três; idem, porém omissão de quatro conjuntos de onda a cada grupo de cinco; omissão, em ordem aleatória, de um conjunto de onda a cada grupo de três; idem, porém omissão de um conjunto de onda a cada grupo de quatro . Som fonte retirado de <https://freesound.org/s/266566/>.

3.1.3.6 Repetição de conjuntos de onda (*Distort Repeat*)

3.1.3.6. a Definição

Na *repetição de conjuntos de onda* estica-se o arquivo de som repetindo seus conjuntos de onda, tanto por uma repetição de cada conjunto individualmente quanto por uma repetição de grupos de conjuntos de onda. Este processo está ilustrado em Figura 19.

Figura 19 – Repetição de conjuntos de onda



Fonte: Elaborada pelo autor

3.1.3.6. b Sonoridade

De forma geral, essa transformação é aquela que mais introduzirá o artifício referido com "som de borracha atritando". Além disso, os pontos extremos dessa transformação costumam gerar uma espécie de som senoidal, ou um som com poucos harmônicos, com frequências aleatórias se sucedendo. Sons que apresentam evolução espectromorfológica pouco variável tendem a apresentar poucos artifícios nessa transformação, além da esperada consequência do esticamento temporal.

Por funcionar como um dos métodos de realização de esticamento temporal, em tese com poucas alterações na altura dos sons, essa transformação tenderá a apresentar também as sonoridades gerais oriundas do esticamento temporal : dissolver transientes rápidos, revelar rápidas estruturas internas, facilitar a percepção de detalhes e introduzir iterações lentas.

Como esta modificação transforma radicalmente o som em questão, a possibilidade de realização de um balanço *dry/wet* geralmente não é pensada como tal, mas sim como estratégia criativa de criação de textura, disposição ou sincronização dos materiais.

3.1.3.6. c Exemplos

O Código 3.16 mostra como podemos executar cinco leituras sucessivas de um arquivo de som com repetição individual de todos conjuntos de ondas em uma, duas, três, quatro e oito vezes.

```
1 (Pwavesets(
2         Pbind(
3                 \name, \somPadrao,
4                 \start, Pn(Pseries(0, 1, 17555),5),
5                 \repeats,Pstutter(17555, Pseq([1,2,3,4,8],1) ),
6                 \useFrac, false,
7         )
8 ).play;)
```

Código 3.16 – Repetição de conjuntos de onda

Em Wishart (2012a) o problema do esticamento temporal é exemplificado para duas diferentes sílabas "ma" e "ra". No primeiro caso, por se tratar basicamente de um som tônico e com ataque mais suave, o esticamento temporal é mais facilmente alcançado. No segundo caso, por se tratar de um som iterativo, a realização deste procedimento não é trivial.

O Áudio 3.17 utiliza diversas pronúncias de r para avaliar tal aspecto, em especial o r vibrante simples alveolar que tem características um pouco mais próximas da sílaba "ma" que os demais. Para sons iterativos, como é o caso aqui, começamos a selecionar e repetir

os impulsos que acionam cada iteração, possibilitando percebê-los mais detalhadamente. No Áudio 3.17, para a vibrante múltipla uvular podemos ouvir os impulsos típicos de sons de língua batendo no palato vocal e nos sons da vibrante múltipla uvular podemos ouvir os típicos sons de batidas em líquidos causados pela vibração da úvula.

Áudio 3.17 (3-17_repet_conj_onda_rato_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte consiste no autor pronunciando a palavra "rato" com o r vibrante simples alveolar (*Voiced alveolar tap*); seguido deste mesmo som mas com duração mais longa; em seguida r vibrante múltipla alveolar (*voiced alveolar trill*); seguido de r vibrante múltipla uvular (*voiced uvular trill*). Após o som fonte, segue-se a transformação de repetição de conjunto de onda de uma, duas, três, quatro e oito repetições de cada conjunto, cada uma apresentada em sequência. Para demarcar a transformações, foi inserido um curto ruído rosa entre cada bloco.

Para os valores mais extremos de repetição, no Áudio 3.17 temos uma espécie de (re)sonificação – utilizamos os dados sonoros de forma tão lenta que passam a funcionar como uma espécie de dados de controle. Estes últimos promovem lentas mudanças de alturas, pois estamos repetindo formatos de onda quase senoidais, criando, assim, uma espécie de melodia com o que antes era a própria onda sonora.

Um modo de operação desta função consiste em repetir os grupos de conjuntos de onda, ao invés de executar a repetição dos conjuntos individuais como feito anteriormente. No Código 3.17 isso é feito ajustando-se a chave *num* de forma que cada grupo de um, dois, três, quatro e oito conjuntos de onda sejam repetidos duas vezes.

```
(Pwavesets(
        Pbind(
                \name, \somPadrao,
                \start, Pn(Pseries(0,2, (17556/2).asInt),1),
                \num, 2,
                \repeats, 2,
                \useFrac, false,
        )
).play;)

(Pwavesets(
        Pbind(
                \name, \somPadrao,
                \start, Pn(Pseries(0,8, (17556/8).asInt),1),
                \num, 8,
                \repeats, 2,
                \useFrac, false,
        )
).play;)
```

Código 3.17 – Repetição de grupos de conjuntos de onda

No Áudio 3.18 à medida que o tamanho dos grupos vai crescendo, começamos a selecionar fragmentos que, no caso da voz humana, misturam porções de ondas relativas às vogais com porções relativas às consoantes, o que gera pequenos loops com novas características. Um efeito natural aqui é a mudança de percepção do fonema dito: à medida que executamos o esticamento a palavra "rato" começa ser confundida com "trato", "frato', "prato", dentre outras.

Áudio 3.18 (3-18_repet_grup_conj_onda_rato_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte consiste no autor pronunciando a palavra "rato" com o r vibrante simples alveolar (*Voiced alveolar tap*); seguido deste mesmo som mas com duração mais longa; em seguida vibrante múltipla alveolar (*voiced alveolar trill*); seguido de vibrante múltipla uvular (*voiced uvular trill*). Após o som fonte, segue-se a transformação de repetição de duas vezes cada dois, três, quatro e oito grupos de conjuntos de onda, cada uma apresentada em sequência. Para demarcar a transformações, foi inserido um curto ruído rosa entre cada bloco.

Por fim, quando levamos o procedimento anterior à valores mais extremos – repetição de quatro vezes cada grupo de oito conjuntos de onda no Áudio 3.19 – podemos perceber o efeito da ressonificação (mencionado mais anteriormente) de forma mais branda, todavia aqui há uma espécie de introdução de reverberação. Além disso, a confusão entre o tipo de fonema dito é um fator destacado aqui.

Áudio 3.19 (3-19_4repet_8grup_conj_onda_rato_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte consiste no autor pronunciando a palavra "rato" com o r vibrante simples alveolar (*Voiced alveolar tap*); seguido deste mesmo som mas com duração mais longa; em seguida vibrante múltipla alveolar (*voiced alveolar trill*); seguido de vibrante múltipla uvular (*voiced uvular trill*). Após o som fonte, segue-se a transformação de repetição de quatro vezes cada grupo de oito conjuntos de onda.

3.1.3.7 Repetição 2 de conjuntos de onda (*Distort Repeat 2*)

3.1.3.7. a Definição

Na *Repetição 2* repete-se os conjuntos de onda sem esticar o arquivo de som, mais especificamente repetimos uma dada quantidade de conjuntos de onda e saltamos a mesma quantidade de conjuntos. Por exemplo, se repetirmos o conjunto de onda de índice 1 quatro vezes, a próxima leitura será do conjunto de onda de índice 5, que também será repetido

cinco vezes. Dado que o tamanho dos conjuntos de onda de um arquivo é geralmente diferente, sempre haverá alguma pequena diferença de tamanho após tal transformação.

3.1.3.7. b Sonoridade

Como nessa transformação existe um descarte de informação, é natural que se escute efeitos próximos aos de algoritmos de compressão de dados. Nos casos mais extremos, quando a quantidade de repetições e o tamanho dos saltos são grandes, obtém-se sequências de alturas mais definidas, porém suas transições são mais bruscas, com ataque e decaimento rápidos.

3.1.3.7. c Exemplos

O Código 3.18 gera uma sequência de transformações *repetição 2* que podem ser escutadas no Áudio 3.20. À medida que se aumenta a quantidade de repetições podemos perceber que ocorre uma espécie de quantização das alturas do som fonte assim como ocorre nos procedimentos analógicos chamados *sample and hold*, as estruturas internas do som, que ocorrem em uma velocidade mais rápida, vão sendo congeladas logo trazidas à uma escala escala temporal mais lenta, onde podem ser apreendidas de outra maneira.

```
(Pwavesets(
        Pbind(
                \name, \somPadrao,
                \start, Pseq([ Pn(Pseries(0,1, 17555),1),
                        Pn(Pseries(0,2, 8776),1), Pn(Pseries(0,3, 5850),1),
                        Pn(Pseries(0,4, 4387),1), Pn(Pseries(0,8, 2194),1)
                        ]),
                \repeats, Pseq([ Pn(1,17555),
                        Pn(2,8776), Pn(3,5850),
                        Pn(4,4387), Pn(8,2194)
                        ]),
                \useFrac, false,
        )
).play;)
```

Código 3.18 – Repetição 2 de conjuntos de onda

Áudio 3.20 (3-20_repet2_conj_onda_rato_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte consiste no autor pronunciando a palavra "rato" com o r vibrante simples alveolar (*Voiced alveolar tap*); seguido deste mesmo som mas com duração mais longa; em seguida r vibrante múltipla alveolar (*voiced alveolar trill*); seguido de r vibrante múltipla uvular (*voiced uvular trill*). Após o som fonte, segue-se a transformação de repetição 2 de conjunto de onda de uma, duas,

três, quatro e oito repetições de cada conjunto, cada uma apresentada em sequência.

Além da semelhança com o congelamento, esta transformação também é análoga aos processos no quais extrai a curva de alturas de um dado som (*pitch follower*) e então se reproduz tais alturas utilizando um oscilador senoidal. Procedimento este que denominamos *ressonificação*, sendo a extração de atributos de uma dado arquivo de áudio e utilização destes como dados de controle de unidades produtoras de som (osciladores, moduladores, etc).

3.1.3.8 Permutação de conjuntos de onda (*Distort Shuffle*)

3.1.3.8. a Definição

Na permutação de conjuntos de onda troca-se a ordem dos conjuntos de onda dentro de um dado grupo de conjuntos. Por exemplo, se há dez conjuntos de onda [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] e troca-se a ordem dos dois primeiros conjuntos a cada grupo de cinco conjuntos, teremos a seguinte organização [2, 1, 3, 4, 5, 7, 6, 8, 9, 10].

3.1.3.8. b Sonoridade

Quando se faz trocas entre conjuntos que são próximos entre si os efeitos dessa transformação são mais brandos, devido à típica redundância dos sinais de áudio; à medida que os conjuntos permutados estão mais distantes entre si a transformação tende a gerar um novo objeto sonoro como sendo uma versão fragmentada e repetida do objeto original. No caso de sons iterativos e não sintetizados, essa transformação tende a alterar as naturais variações sutis presentes entre cada iteração deste som.

Destaca-se uma consequência deste tipo de transformação: a inserção de silêncios ou de conjuntos de onda de amplitudes muito pequenas em meio à oscilações mais regulares e intensas. Isso aproxima esta transformação à *omissão de conjuntos de onda*, nas quais é possível perceber um introdução de ruído (algumas vezes análogo à distorção) ou de granulação mais grossa.

3.1.3.8. c Exemplos

No Código 3.19 realizamos permutações nas quais a cada execução completa do arquivo de áudio diminui-se a quantidade global de conjuntos embaralhados, permutados. Neste caso, iniciamos com dois conjuntos permutados a cada grupo de dessezeis conjuntos; em seguida dois permutados a cada grupo de oito conjuntos, e assim por diante, exemplo este disponível no Áudio 3.21.

```
(Pwavesets(
        Pbind(
                \name, \somPadrao, // buffer a ser utilizado
                \start, Pseq([Pseries(0,1, 17555),
                        Pstutter(16,Pseries(0,16,1097))+Pseq
                            ([0,1,2,3,4,5,6,7,9,8,10,11,12,13,14,15], inf),
                        Pstutter(8, Pseries(0,8,2194))+ Pseq
                            ([0,1,2,4,3,5,6,7], inf),
                        Pstutter(4, Pseries(0,4,4388))+ Pseq([0,2,1,3], inf
                            ),
                        Pstutter(2, Pseries(0,2,8776))+ Pseq([1,0], inf),
                ]),
                \useFrac, false,
        )
).play;)
```

Código 3.19 – Permutação browniana de conjuntos de onda

Áudio 3.21 (3-21_permut_gradual_conj_onda_rato_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte consiste no autor pronunciando a palavra "rato" com o r vibrante simples alveolar (*Voiced alveolar tap*); seguido deste mesmo som mas com duração mais longa; em seguida r vibrante múltipla alveolar (*voiced alveolar trill*); seguido de r vibrante múltipla uvular (*voiced uvular trill*). Após o som fonte, segue-se a transformação de permutação de conjuntos de onda, sendo que os conjuntos do meio do grupo permutados a cada dezesseis, oito, quatro e dois grupos.

Além da execução de algoritmos de permutação tradicionais, uma expansão natural de tal técnica é a realização de uma leitura linear modulada por variações brownianas, o que garante tanto a permutação de conjuntos de onda quanto a inserção de variabilidade nas mesmas. Tal resultado, apresentado no Código 3.20 e Áudio 3.22, pode ser compreendido como uma transformação que mescla a ideia de *permutação* e *repetição* de conjuntos de onda.

```
(Pwavesets(
        Pbind(
                \name, \somPadrao, //buffer a ser utilizado
                \start, Pseq([Pseries(0,1, 17555),
                        (Pseries(0, 1, 17555) + Pbrown(-5, 5, 1)),
                        (Pseries(0, 1, 17555) + Pbrown(-10, 10, 2)),
                        (Pseries(0, 1, 17555) + Pbrown(-50, 50, 4)),
                        (Pseries(0, 1, 17555) + Pbrown(-100, 100, 8)),
                        ]).fold(0, 17555), //quando exceder os limites,
                            rebate ao centro
                \amp, 0.8, //amplitude
```

```
                \useFrac, false,
        )
).play;)
```

Código 3.20 – Permutação browniana de conjuntos de onda

Áudio 3.22 (3-22_permut_brown_conj_onda_rato_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte consiste no autor pronunciando a palavra "rato" com o r vibrante simples alveolar (*Voiced alveolar tap*); seguido deste mesmo som mas com duração mais longa; em seguida r vibrante múltipla alveolar (*voiced alveolar trill*); seguido de r vibrante múltipla uvular (*voiced uvular trill*). Após o som fonte, segue-se a transformação de leitura linear modulada por variações brownianas, no primeiro caso desvio mínimo de -5, desvio máximo de 5 e passo máximo de 1 conjunto; no segundo caso desvio mínimo de -10, desvio máximo de 10 e passo máximo de 2 conjuntos; no terceiro caso desvio mínimo de -50, desvio máximo de 50 e passo máximo de 4 conjuntos; no último caso desvio mínimo de -100, desvio máximo de 100 e passo máximo de 8 conjuntos .

Diferentemente da distorção não linear – tal como nos casos extremos do Código 3.21 e Áudio 3.23, nos quais toda a ordem de uma grande quantidade de conjuntos de onda (32, 64 e 128) é embaralhada – o som transformado ainda mantém as características gestuais do som original (curva de altura, intensidade, iteração, etc) porém introduz um conteúdo espectral que é tipicamente ruidoso. Além disso, os tipos de ruídos introduzidos aqui são altamente idiossincráticos, diferindo qualitativamente sobremaneira dos ruídos estocásticos clássicos (ruído branco, rosa, marrom, etc) e também de osciladores caóticos quando operantes na região paramétrica produtora de ruído (Lorenz, função logística, oscilador neural, etc).

```
(Pwavesets(
        Pbind(
                \name, \somPadrao, //buffer a ser utilizado
                \start, Pseq([Pseries(0,1,17555),
                        Pstutter(32,Pseries(0,32,548))+Pseq((0..31).
                            scramble,inf),
                        Pstutter(64,Pseries(0,64,274))+Pseq((0..64).
                            scramble,inf),
                        Pstutter(128,Pseries(0,128,137))+Pseq((0..128).
                            scramble,inf),
                        ]).fold(0, 17555), //quando exceder os limites,
                            rebate ao centro
                \amp, 0.8, //amplitude
                \useFrac, false,
        )
).play;)
```

Código 3.21 – Permutação aleatória de conjuntos de onda

Áudio 3.23 (3-23_permut_extrem_conj_onda_rato_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte consiste no autor pronunciando a palavra "rato" com o r vibrante simples alveolar (*Voiced alveolar tap*); seguido deste mesmo som mas com duração mais longa; em seguida r vibrante múltipla alveolar (*voiced alveolar trill*); seguido de r vibrante múltipla uvular (*voiced uvular trill*). Após o som fonte, segue-se a transformação de permutação de conjuntos de onda, embaralhando todos os conjuntos em grupos de 32, 64 e 128 unidades, nesta ordem.

3.1.3.9 Ressíntese aditiva de conjuntos de onda (*Distort Harmonic*)

3.1.3.9. a Definição

Na ressíntese aditiva cada conjunto de onda é substituído pela soma de múltiplas cópias que totalizem o mesmo comprimento do conjunto original, de forma análoga aos harmônicos de uma onda fundamental [10]. Para a realização de tal procedimento, multiplica-se a velocidade de reprodução e a quantidade de repetições de um dado conjunto de onda pelo mesmo fator.

Este procedimento é nomeado no CDP por distorção harmônica de conjuntos de onda (*Distort Harmonic*), todavia como tal nome já encontra uma consolidada utilização para outro fenômeno no áudio e processamento de sinais, preferimos aqui fazer alusão à síntese aditiva – processo no qual diversos múltiplos de uma senoide são somados. Tanto a sonoridade quanto o processamento de dados de tal algoritmos estão mais fortemente ligados à ideia de síntese aditiva.

3.1.3.9. b Sonoridade

De maneira geral, a ressíntese aditiva de conjuntos de onda tende a apresentar características de filtro ressonante, como a valorização de determinadas frequências, em associação à introdução de diversos artifícios pouco controláveis, como é o caso de oscilações e iterações de baixa frequência. Quando as frequências adicionadas são de alta ordem, a tendência é que tal acréscimo soe como introdução de ruídos.

Como consequência dessas alterações, é comum que se altere a atribuição de propriedades físicas das fontes sonoras, *e.g.* instrumentos de madeira percutida tendem a soar como vidro percutido, arcos de crina em atrito (violino, violoncelo, etc) podem soar como metais atritados, etc.

[10] Utilizaremos a expressão *harmônico de conjunto de onda* em referência aos múltiplos inteiros de um conjunto de onda, em analogia aos harmônicos de uma onda

3.1.3.9. c Exemplos

O Código 3.22 [11] mostra uma maneira de se acrescentar um *segundo harmônico de conjunto de conjunto de onda* ao som fonte.

```
(var server = Server(\nrt,
    options: ServerOptions.new
    .numOutputBusChannels_(2)
    .numInputBusChannels_(2)
),
def = SynthDef(\wvst0, { arg out = 0, buf = 0, start = 0, length = 441,
   playRate = 1, sustain = 1, amp=0.2, pan;
                        var phasor = Phasor.ar(0, BufRateScale.ir(buf) *
                            playRate, 0, length) + start;
                        var env = EnvGen.ar(Env([amp, amp, 0], [sustain,
                            0]), doneAction: 2);
                        var snd = BufRd.ar(1, buf, phasor) * env;

                        OffsetOut.ar(out, Pan2.ar(snd, pan));
                }, \ir.dup(8));
b = Buffer.new(server, 0, 1);
w = Wavesets.from("Audios/C3-marimba-violino-trompa-fagote-trompete-piano-
   mono.wav", server: server);
def.add;
x = Ppar([
        Pbindef(\ws1,
        \instrument, \wvst0,
        \startWs, Pn(Pseries(0, 1, 54156), 1),
        \numWs, 1, \playRate, 1,
        \bufnum, b, \repeats, 1,
        [\start, \length, \sustain], Pfunc({ |ev|
                var start, length, wsDur;
                #start, length, wsDur = w.frameFor(ev[\startWs], ev[\numWs
                    ]);
                [start, length, wsDur * ev[\repeats] / ev[\playRate].abs]
        }),
        \dur, Pkey(\sustain)
),
        Pbindef(\ws2,
        \instrument, \wvst0,
        \startWs, Pn(Pseries(0, 1, 54156), 1),
        \numWs, 1, \playRate, 2,
        \bufnum, b, \repeats, 2,
        [\start, \length, \sustain], Pfunc({ |ev|
                var start, length, wsDur;
```

[11] Esse tipo de transformação é dispendiosa para ser realizada em tempo real no SuperCollider, portanto optou-se aqui pela realização em tempo diferido. Para tal implementação foi necessária a utilização da classe *Wavesets*, versão anterior da *WavesetEvents* que utilizamos.

```
                #start, length, wsDur = w.frameFor(ev[\startWs], ev[\numWs
                    ]);
                [start, length, wsDur * ev[\repeats] / ev[\playRate].abs]
        }),
        \dur, Pkey(\sustain)
)
]).asScore(duration: 15, timeOffset: 0.1);
x.add([0.0, [\d_recv, def.asBytes]]);
x.add([0.0, b.allocReadMsg("Audios/C3-marimba-violino-trompa-fagote-
    trompete-piano-mono.wav")]);
x.sort;
x.recordNRT(
    outputFilePath: "~/wstHarm1246.wav".standardizePath,
    sampleRate: 44100,
    headerFormat: "WAV",
    sampleFormat: "int16",
    options: server.options,
    duration: 15
);
server.remove;)
```

Código 3.22 – Ressíntese aditiva de conjuntos de onda

No Áudio 3.24, além da já mencionada valorização de certas faixas de frequência por esta transformação, é possível notar como o timbre da marimba parece migrar de um marimba de madeira para uma marimba de vidro. Além disso, cada tipo de harmônico introduzido tende a reforçar alguma certa característica do timbre instrumental bem como inserir artifícios de oscilações e iterações espúrias diferentes.

Além disso, é importante notar que tal transformação afeta de maneira dissimilar o ataque e a sustentação dos objetos sonoros. De maneira bastante geral, a característica mais comum que se pode perceber é que o ataque dos sons tende a permanecer menos alterado e que a seção sustentada, quando harmônica, será reforçada daquele harmônico que foi adicionado.

Áudio 3.24 (3-24_ressint_aditiv_conj_onda_C3_orq.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Sons de instrumentos de orquestra transformados por ressíntese aditiva de conjunto de onda. Primeiramente toca-se os sons fonte, em seguida som fonte acrescido do "segundo harmônico de conjunto de onda", som fonte acrescido do "terceiro harmônico de conjunto de onda", som fonte acrescido do "quarto harmônico de conjunto de onda", som fonte acrescido dos "segundo, quarto e sexto harmônico de conjunto de onda" e por fim o som fonte acrescido do "terceiro, quinto e sétimo harmônico de conjunto de onda". Na ordem de aparição, os sons utilizados foram uma nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano; retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-

community.html>.

As transformações de conjuntos de onda tendem a apresentar importantes diferenças para sons sustentados e *staccatto*. Em sons sustentados instrumentais, sempre há alguma variação ligeira no seu regime permanente, seja na forma de vibrato, tremolo, variações timbrística, *allure*, etc. Assim, como a sonoridade das transformações de conjuntos de onda é muito dependente do som fonte, em uma transformação de conjuntos de onda constante (*e.g.* a cada conjunto de onda somamos seu terceiro harmônico de conjunto do onda) haverá uma variação da sonoridade ao longo do regime permanente devido a estas variações timbrísticas do som fonte. No caso dos sons *staccatto* esse fenômeno é menos perceptível pois ocorre muito rapidamente, mas ainda sim é possível de ser percebido, em especial, em sons cujo ataque é muito rápido e cuja sustentação é harmonicamente mais estável.

O Código 3.23 mostra uma maneira de modular qual "harmônico de conjunto de onda" será adicionado, modulação esta que acentua a diferenciação entre transformações de conjuntos de onda para sons sustentados e *staccatto*. Diferentemente dos sons *stacatto*, no Áudio 3.25 é possível perceber como em sons sustentados esses novos "harmônicos" tendem a se descolar do objeto sonoro original, sendo mais percebidos como sons paralelos.

```
(Pwavesets(
        Pbind(
                \name, \somPadrao, // buffer a ser utilizado
                \start, Pn(Pseries(0, 1, 54156),1),
                // três padrões em paralelo: 1 como valor padrão e dois
                   Pseq
                \repeats, Ptuple([1, Pseq([2,2,2,2,3,3,4,4], inf), Pseq
                   ([5,5,6,6,7,7,7,7,7],inf)]),
                // a variação interna do Pseq é a modulação
                \rate, Ptuple([1, Pseq([2,2,2,2,3,3,4,4],inf), Pseq
                   ([5,5,6,6,7,7,7,7,7],inf)]), // velocidade de leitura
                \useFrac, false,
        )
).play;)
```

Código 3.23 – Ressíntese aditiva de conjuntos de onda com modulação

Áudio 3.25 (3-25_ressint_aditiv_conj_onda_F1_fgt_C3_orq.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota fá-1 (F1) tocada em sustentação por um fagote, seguida da ressintese aditiva de conjuntos de onda conforme Código 3.23. Na sequência, nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano, seguidas da mesma transformação de conjuntos de onda. Amostras retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível

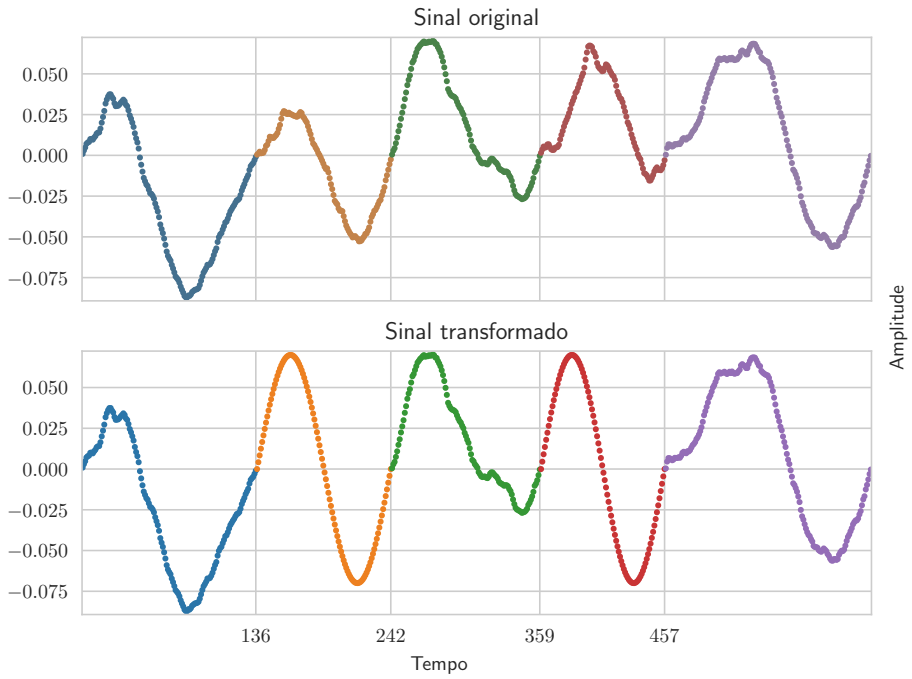
em<https://vis.versilstudios.com/vsco-community.html>.

3.1.3.10 Substituição e intercâmbio de conjuntos de onda (*Distort Reform* e *Distort Interact*)

3.1.3.10. a Definição

O processo de *intercâmbio* de conjuntos de onda consiste alternar entre conjuntos de onda de arquivos diferentes, já no processo de *substituição* conjuntos de ondas do arquivo de destino são substituidos pelos conjuntos de onda do arquivo fonte[12]. Assim como nos demais processos, essa transformação pode ocorrer para grupos ou unidades de conjuntos de onda. Na Figura 20 ilustramos a substiução de conjunto-de-Onda de um sinal gravado por uma onda senoidal.

Figura 20 – Substiução de conjuntos de onda de um sinal gravado por uma onda senoidal



Fonte: Elaborada pelo autor

3.1.3.10. b Sonoridade

Um primeiro ponto a se destacar é a taxa à qual tais conjuntos de onda serão comutados: se realizarmos trocas a cada conjunto, ou a uma pequena quantidade de grupos de conjuntos, por mais que haja uma grande discrepância na forma de onda destes elementos (amplitude, frequência, número de harmônicos, etc) a frequência desta

[12] No CDP, denomina-se *Distort Reform* quando temos um arquivo de áudio gravado e os outros conjuntos de onda são sintéticos (senoide, dente de serra, etc), e *Distort Interact* quando se utiliza dois sinais gravados

alternância tenderá a ser rápida o suficiente para superar o limiar da diferenciabilidade de eventos (cerca de 20 Hz). Na situação oposta, se a comutação se der uma grande quantidade de grupos de conjuntos de onda (cerca de 30) iremos perceber o som como sendo composto de dois sinais rapidamente alternados.

A utilização de sons que apresentam algum tipo de semelhança na sua forma de onda auxilia a manutenção da fusão auditiva dos eventos comutados. Alguns exemplos são utilização de um som fonte e de uma versão levemente modificada do mesmo (reverberação, demais transformações de conjuntos de onda, *chorus*); diferentes execuções de uma mesma nota; diferentes porções de um mesmo som iterativo; sons sintetizados com diferenças pequenas de modulações; etc.

3.1.3.10. c Exemplos

No Código 3.24 fazemos uma avaliação das diferenças básicas entre *substituição* e *intercâmbio* de conjuntos de onda. O resultado sonoro destes algoritmos é bastante similar quando se utiliza sons parecidos, como é o caso do Áudio 3.26. Neste último, só nos casos de pequenas quantidade de grupos de conjuntos de onda transformados foi possível perceber diferenças sonoras mais significativas.

```
// substituição de conjuntos de onda
(Pwavesets(
        Pbind(
                \name, Pseq([Pn(\c1, 13), Pn(\f1,13)],inf), // buffer a ser
                    utilizado
                \start, Pn(Pseries(0, 1, 3456),1),
                \useFrac, false,
        )
).play;)
// intercâmbio de conjuntos de onda
(Pwavesets(
        Pbind(
                \name, Pseq([Pn(\c1, 13), Pn(\f1,13)],inf), // buffer a ser
                    utilizado
                \start, Pswitch1(
                        [Pseries(0, 1, 3456), Pseries(0, 1, 3456)],
                        Pseq([Pn(0,13),          Pn(1,13)], inf)),
                \useFrac, false,
        )
).play;)
```

Código 3.24 – Substituição e intercâmbio de conjuntos de onda

É importante avaliar no Áudio 3.26 que a técnica de substituição e intercâmbio de conjuntos de onda permite a produção de sons fortemente similares aos sons de multifônicos no fagote.

Áudio 3.26 (3-26_C1s_F1_conj_onda_subst_interc_fgt_multif.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Temos os seguintes sons em sequência: nota dó-1 sustenido (C♯1) tocada em sustentação por um fagote, seguida da nota fá-1 (F1); dois tipos de sons multifônicos da nota dó-2 (C2) no fagote; substituição seguida de intercâmbio de dois conjuntos de onda entre os sons do dó-1 sustenido (C♯1) e fá-1 (F1); idem ao anterior entretanto com quatro conjuntos de onda; idem entretanto com treze conjuntos de onda; idem entretanto com vinte e seis conjuntos de onda. Sons de fagote multifônicos disponíveis em <http://www.idrs.org/resources/BSNFING/PUB/BsnMulti/C2_B2MU.html>, demais sons fonte de fagote retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

Na execução desta transformação com comutação entre sons gravados e sintetizados, é comum obter resultados que apresentem menor fusão auditiva. No Código 3.25 escolhe-se duas formas de onda com características sonoras drasticamente distintas, uma senoide e uma onda dente de serra. Além disso, escolheu-se as ondas com a mesma frequência fundamental do som gravado (dó sustenido 1 no fagote) e foram utilizadas duas ondas dente de serra em oitavas para aumentar a presença de harmônicos de alta ordem.

```
1  b = Buffer.alloc(s, 44100 * 16.04, 1); // a four second 1 channel Buffer
2  (SynthDef(\help_RecordBuf, { arg out = 0, bufnum = 0;
3      RecordBuf.ar(LFSaw.ar(49.midicps,mul: 0.25)+LFSaw.ar(25.midicps,mul:
           0.25), bufnum, doneAction: Done.freeSelf, loop: 0);
4  }).play(s,[\out, 0, \bufnum, b]);)
5  b.play;
6  b.write("C:/Users/f11mi/Google Drive/dirce drive/Dirce_1_3-10/Audios/
       C1s_saw_sust_mono.wav", headerFormat: "wav", sampleFormat: "int16")
7
8  ~w = WavesetsEvent.read("C:/Users/f11mi/Google Drive/dirce drive/Dirce_1_3
       -10/Audios/C1_fgt_sust_mono.wav");
9  ~w.add(\c1);
10 ~w = WavesetsEvent.read("C:/Users/f11mi/Google Drive/dirce drive/Dirce_1_3
       -10/Audios/C1_saw_sust_mono.wav"); // 1314xings
11 ~w.add(\c1saw);
12 ~w.add(\somPadrao);
13
14 (Pwavesets(
15         Pbind(
16                 \name, Pseq([Pn(\c1, 16), Pn(\c1saw,16)],inf), //
                       identificação do buffer a ser utilizado
17                 \start, Pn(Pseries(0, 1, 637),1), // toca o buffer sem
                       saltos e até o final
18                 \num, 1, // quantidade de waveset que será lida do buffer
                       em um evento
19                 \repeats, 1, //quantas vezes irá tocar cada waveset
20                 \rate, 1.0, // velocidade de leitura
```

```
                \amp, 0.6, //amplitude
                \useFrac, false,
        )
).play;)
```

Código 3.25 – Substituição de conjuntos de onda com sons sintetizados

Primeiramente, o Áudio 3.27 mostra como a diferença da percepção de nível entre as duas formas de onda tende a gerar um efeito de modulação de amplitude mesmo com as mudanças de parâmetros das transformações. Essa diferença dificilmente é eliminada devido à influência do conteúdo harmônico na percepção de intensidade sonora. Um segundo ponto está no fato de sons senoidais, neste tipo de transformação, muitas vezes tenderem a ser percebidos como silêncios ou serem mascarados tanto pela intensidade quanto pela faixa de frequência dos sons gravados. Por fim, também é importante notar que diversas outras frequências vão sendo ressaltadas ou introduzidas, modificando inclusive a percepção da altura.

Áudio 3.27 (3-27_C1s_fgt_sin_saw_conj_ond_subst.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Notá dó-1 sustenido tocada por um oscilador senoidal; seguida de nota dó-1 sustenido (C ♯ 1) tocada em sustentação por um fagote; substituição de conjuntos de onda da senoide no som de fagote a cada um grupo de conjuntos de onda; idem, todavia a cada dois grupos de conjuntos de onda; idem, todavia a cada quatro grupos de conjuntos de onda; idem, todavia a cada oito grupos de conjuntos de onda; idem, todavia a cada dezesseis grupos de conjuntos de onda. Em seguida este mesmo processo é repetido, porém com a onda dente de serra descrita no Código 3.25. Som de fagote retirado da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

3.2 Processos de transformação no domínio da frequência

3.2.1 Utilização do vocoder de fase

A Tabela 3.2.1 apresenta a referência mais atual dos procedimentos de manipulação espectral disponíveis no CDP. Tais procedimentos se dividem, de maneira geral, em modificações nos *bins*, em estimativas de altura, no envelope espectral e nos picos de tais envelopes (formantes). Naturalmente, o procedimento padrão consiste em uma análise pela função *Pvoc anal*, segue com uma das variadas transformações espectrais e finaliza com a função *Pvoc synth* para gerar um arquivo de áudio. É possível, ainda, realizar uma segunda etapa de análise e ressíntese na qual, a partir de um arquivo contendo o espectro, realiza-se uma extração da curva de altura, em seguida promove-se alguma modificação nesta, para, enfim, converter tal curva de volta à um arquivo de espectro.

Tabela 2 – Transformações espectrais disponíveis no Composers Desktop Project

Transformação	**Descrição**
	1. Borra os dados de análise
Blur avrg	Pondera a energia espectral ao longo de N canais adjacentes
Blur blur	Borra os dados espectrais ao longo do tempo
Blur chorus	Adiciona variações aleatórias à amplitude ou frequência nos canais de análise
Blur drunk	Modifica um som por meio de uma caminhada bêbada (movimento browniano) ao longo das janelas de análise
Blur noise	Adiciona ruído ao espectro
Blur scatter	Emagrece o espectro de forma aleatória
Blur shuffle	Embaralha as janelas de análise de acordo com um dado esquema
Blur spread	Espalha os picos espectrais
Blur suppress	Suprime os dados dos canais mais proeminentes
Blur weave	Oscila entre as janelas de análise de acordo com um dado padrão
Selfsim	Susbtitui janelas espectrais pelas janelas mais similares e mais intensas
	2. Combina os dados de análise de dois ou mais arquivos
Combine cross	Substitui somente as magnitudes espectrais do primeiro arquivo com os do segundo
Combine diff	Procura e retém a diferença entre dois espectros
Combine interleave	Entrelaça janelas ou grupos de janelas de alguns espectros
Combine make	Gera um arquivo de análise a partir somente dos arquivos de dados de formante e altura
Combine make2	Gera um arquivo de análise a partir somente dos arquivos de dados de formante, altura e envelope
Combine max	Retém, para cada janela, os componentes mais intensos dos canais dentre um conjunto de espectros
Combine mean	Gera a média entre dois espectros
Combine sum	Soma um espectro a outro
Specross	Interpola as parciais de dois sons com alturas definidas
Specsphinx	Impõe os canais de amplitude de um som nos canais de frequência de outro som
Spectwin	Combina as formantes e/ou o envelope espectral total de dois espectros
	3. Foca em características dos dados de análise
Focus accu	Sustenta cada banda espectral até que dados mais intensos apareçam naquela banda (descontinuada)
Focus superaccu	Extensão de *Focus accu*, com modos adicionais para afinar os canais espectrais sustentados à um template de ressonâncias.
Focus exag	Exagera o contorno espectral
Focus focus	Foca a energia espectral nos picos do espectro
Focus fold	Transposição de oitava dos componentes espectrais para uma dada faixa de frequência
Focus freeze	Congela o espectro durante um certo tempo, descongelando depois do tempo transcorrido
Focus hold	Congela o espectro durante um certo tempo, descongelando sempre do ponto onde parou
Focus step	Caminha, com passos de duração regular, entre cada quadro espectral congelado
	4. Operações de formantes
Formants get	Extrai a evolução temporal do envelope de formantes de um arquivo de análise.
Formants getsee	Obtém os dados de formante de um arquivo de análise e escreve o arquivo para visualização
Formants put	Impões formantes oriundas de um arquivo de formantes em um arquivo de análise espectral
Formants see	Converte os dados de formante em um arquivo binário para a visualização
Formants vocode	Impõe o envelope espectral de um som em outro som
	5. Realça características dos dados de análise
Hilite arpeg	Arpeja o espectro
Hilite band	Divide o espectro em bandas e as processa individualmente
Hilite bltr	Gera uma média ao longo do tempo e delineia o espectro
Hilite filter	Filtra os dados espectrais utilizando filtros passa-altas, passa-baixas, passa-faixa ou rejeita-faixa.
Hilite greq	Filtro equalizador gráfico atuando no espectro
Hilite pluck	Realça mudanças espectrais (e.g. utilizando com Hilite arpeg)
Hilite trace	Realça as N parciais mais fortes a cada momento (janela) do tempo
Hilite vowels	Impõe vogais em um som
Glisten	Divide o espectro aleatoriamente em bins e reproduz em ordem
	6. Morfa[13] *entre dois sons com transições suaves*
Morph bridge	Cria uma ponte de interpolação entre dois espectros sonoros por meio da interpolação entre janelas temporais

[13] Tradução nossa para o termo *morphing*, ver seção 1.3.

Morph glide	Interpola linearmente entre duas janelas de análise extraídas de Spec grab
Morph morph	Morfa um espectro em outro, aceitando espectros variáveis no tempo
Newmorph	Morfa entre espectros dissimilares
Newmorph2	Morfa as frequências dos picos espectrais

7. Operações em um único quadro de formante

Oneform get	Extrai o envelope de formantes de um só quadro a partir de um arquivo de formante preexistente
Oneform put	Impõe o envelope de formantes em um único quadro no arquivo de destino
Oneform combine	Gera um novo som a partir da informação de altura e de um único quadro de formante

8. Operações de altura

Pitch altharms	Deleta harmônicos alternadamente
Pitch chord	Superpõe versões transpostas de um som com o original
Pitch chordf	Superpõe versões transpostas do espectro com o espectro original
Pitch octmove	Transposição de oitava com deslocamento de formante (torna-se inarmônico)
Pitch pick	Retém somente os canais que devem sustentar parciais específicas
Pitch transp	Desloca a fundamental do espectro (ou parte dela), mantendo as relações harmônicas
Pitch tune	Substitui frequências do espectro por harmônicos de alturas específicas
Tunevary	Idem ao anterior, acrescentada a possibilidade de variação temporal

9. Acessa informações de parciais e traçados de altura

Pitchinfo info	Mostra informações sobre altura de um arquivo de altura
Pitchinfo hear	Converte um arquivo de altura binário em um arquivo de som para teste
Pitchinfo see	Converte um arquivo de altura ou transposição binário em pseudo arquivo de som para visualização
Pitchinfo convert	Converte um arquivo de altura binário em um arquivo texto de *breakpoints* de tempo e frequência
Pitchinfo zeros	Mostra se um arquivo de altura contém zeros não interpolados (janelas sem altura)

10. Análise e ressíntese pelo vocoder de fase

Pvoc anal	Converte um arquivo de som em um arquivo de espectro
Pvoc extract	Analisa e ressintetiza um som com várias opções
Pvoc synth	Converte um arquivo de espectro em arquivo de som

11. Modifica arquivos de altura

Repitch analenv	Extrai o envelope de intensidade de um arquivo de análise
Repitch approx	Faz uma cópia aproximada do arquivo de altura
Repitch combine	Combina arquivos de altura e/ou tranposição para gerar outro arquivo de altura e/ou transposição
Repitch combineb	Combina arquivos de altura e/ou tranposição para gerar um arquivo de *breakpoints*
Repitch cut	Recorta e mantém um segmento de um arquivo binário de altura
Repitch exag	Exagera o contorno de altura
Repitch fix	"Massageia" os dados de um arquivo de altura, deixando os mais suaves
Repitch generate	Cria um arquivo binário de altura a partir de um arquivo texto com pares de valores MIDI
Repitch getpitch	Tenta extrair altura a partir dos dados espectrais
Repitch insertsil	Marca áreas como silêncio em um arquivo de altura
Repitch insertzeros	Marca áreas de altura não definida em um arquivo de altura
Repitch interp	Substitui ruídos ou silêncios por alturas de um arquivo existente
Repitch noisetosil	Substitui janelas de altura pouco definidas por silêncio
Repitch invert	Inverte o contorno de altura em um arquivo de altura
Repitch pchshift	Transpõe alturas a um número constante de semitons (torna-se inarmônico)
Repitch pchtotext	Converte um arquivo de altura binário em arquivo texto
Repitch pitchtosil	Substitui janelas que contenham altura por silêncio
Repitch quantize	Quantifica alturas em um arquivo de alturas
Repitch randomize	Randomiza uma linha de alturas
Repitch smooth	Suaviza o contorno de altura em um arquivo de alturas
Repitch synth	Cria um espectro seguindo o contorno de altura de um arquivo de altura
Repitch transpose	Transpõe o espectro em conjunto com uma transposição do envelope espectral
Repitch transposef	Transpõe o espectro mas mantém o envelope espectral estático
Repitch vibrato	Adiciona vibrato a um arquivo de altura
Repitch vowels	Cria um espectro de vogal, seguindo o contorno de altura de um arquivo de altura

12. Utilitários básicos de espectro

Spec bare	Zera os dados em um canal que não contém harmônicos
Spec clean	Remove ruído de arquivos de análise do vocoder de fase
Spec cut	Corta e descarta uma seção do arquivo de análise
Spec gain	Amplifica ou atenua o espectro
Spec gate	Elimina os dados de um canal que estão abaixo de um limiar de amplitude
Spec grab	Captura uma única janela de análise de um certo instante especificado
Spec magnify	Expande em duração uma única janela de análise a partir de um certo instante até um determinado período

13. Limpa e organiza os dados de análise espectral

Specnu clean	Elimina qualquer sinal que esteja abaixo de um certo limiar (definido no arquivo de ruído)
Specnu rand	Randomiza a ordem das janelas espectrais
Specnu remove	Remove componentes de altura definida de um espectro
Specnu slice	Divide um arquivo de análise em bandas individuais de frequências, salvando cada uma como arquivos de análise separados
Specgrids	Parte o espectro em pedaços, dentro de uma grade
Specnu squeeze	Espreme o espectro dentro de uma faixa de frequência
Specinfo Subtract	simular a *specnu clean*, com o acréscimo de também subtrair as amplitudes do ruído do arquivo original

14. Fornece informações do espectro

Specinfo channel	Retorna o número do canal do vocoder de fase correspondente à frequência pesquisada
Specinfo frequency	Retorna a frequência central do canal pesquisado
Specinfo level	Converte o nível variável da análise em um pseudo arquivo de áudio para possibilitar a visualização
Specinfo octvu	Imprime textualmente a variação temporal da amplitude espectral dentro de bandas de oitava
Specinfo peak	Localiza o centro de energia do espectro, com suas variações temporais, e imprime textualmente
Specinfo print	Imprime os dados de análise como arquivo de texto
Specinfo report	Relatório de texto sobre a localização de picos de frequência ao longo da evolução espectral
Specinfo windowcnt	Retorna o número de janelas de análise
Get partials harmonics	Extrai as amplitudes relativas dos harmônicos em uma fonte de altura definida
Peak Extract	Extrai os picos espectrais de um arquivo de análise e escreve em um arquivo de texto
Peak Find	Encontra os tempos em que ocorrem os picos de um som

15. Operações estranhas

Strange glis	Cria glissandos dentro da evolução do envelope espectral
Strange invert	Inverte o espectro
Strange shift	Deslocamento de frequência linear de parte do espectro
Strange waver	Oscila entre estados harmônicos e inarmônicos

16. Estica os dados de tempo ou frequência

Stretch spectrum	Estica ou comprime as frequências no espectro
Stretch time	Estica ou comprime um som no tempo sem alterar a altura

Traduzido e Adaptado de CDP (2018b)

As transformações mostrada na Tabela 3.2.1 são excessivamente específicas. As estruturas de dados do SuperCollider possibilitam o acesso a tais processos de uma maneira um pouco mais generalizável, portanto selecionamos alguns procedimentos que possam representar globalmente cada grupo de transformações a ser implementado.

Um ponto crucial de distinção entre o CDP e o SuperCollider está na implementação do vocoder de fase: no primeiro, toda a implementação é feita em tempo diferido e no segundo a implementação privilegia fortemente o tempo real. Em termos práticos essa distinção implica que as transformações espectrais do CDP irão permitir a análise-ressíntese de um arquivo como um todo e permitirão a independência entre pontos de aplicação das transformações, *e.g.* é possível retirar um bloco do final do arquivo e utilizá-lo em uma transformação de um bloco do começo do arquivo.

No SuperCollider, de outra maneira, é possível transformar o espectro de capturas de sons em tempo real (microfones e transdutores) ou processos de síntese em tempo real (síntese estocástica, caótica, modelos físicos, etc), realizando e modulando as transformações ao vivo com uma latência baixíssima ou devidamente controlada. Neste caso, para garantir

a baixíssima latência do tempo real nos restringimos a um processamento em pequenos blocos de cálculo, assegurando a entrega rápida dos cálculos computacionais de análise-transformação-ressíntese em troca da impossibilidade de utilização de eventos futuros pelo fato de ainda não terem acontecido fisicamente (garantia de causalidade) ou por termos de esperar um tempo excessivo para que ocorram (alta latência).

Por ser orientado ao funcionamento em tempo real, a grande maioria dos *UGens* do vocoder de fase do Supercollider trabalha com alterações dentro de uma única janela de áudio, ou seja, modificando os *bins* dentro de uma única janela. É possível gravar continuamente os dados fornecidos pelo *UGen FFT* em um *buffer*, modificá-los sem restrições temporais e depois ressintetizá-los, porém a manipulação de tais dados se mostrou complexa e morosa. Ainda devido às restrições impostas pela implementação em tempo real só é possível realizar modificações no tamanho da janela, no tipo de janela e no tamanho de salto no início da execução do *UGen FFT* no servidor, não sendo possível realizar modificações contínuas desses parâmetros ao longo da execução.

Apesar de ser possível a realização de transformações espectrais implementando o vocoder de fase exclusivamente do ponto de vista da linguagem (utilizando a classe *Signal* e os métodos *.ftt* e *.ifft*), a execução desses procedimentos costuma se tornar excessivamente lenta. Assim, iremos nos restringir a processos que operem em uma única janela, ou os processos de múltiplas janelas que já estão implementados no SuperCollider. Quando for necessário avaliar um processo que lide com uma grande quantidade de janelas, geralmente de maneira não-causal, iremos recorrer ao *scripting* do CDP.

O Código 3.26 apresenta um bloco básico de análise-ressíntese em tempo real no SuperCollider, tendo como fonte um oscilador dinâmico estocástico (*Gendy1*) sob o qual se aplica uma filtragem espectral com vocoder de fase para selecionar os canais instáveis (*PV_NoiseSynthP*) – canais cuja fase varia muito rapidamente, ou seja, tendem a não apresentar frequência estável. O nível de seleção de instabilidade é controlado pela posição Y do mouse e o Áudio 3.28 mostra uma gravação na qual aumentamos este nível de seleção gradualmente.

```
1 ({  var in, chain;
2     in = Gendy1.ar(mul: MouseY.kr(0.125,0.75)); // oscilador estocástico
3     chain = FFT(LocalBuf(2048), in); // análise
4     chain = PV_NoiseSynthP(chain,MouseY.kr(0.1,4.9),10); // transformação
5     LeakDC.ar(IFFT(chain)).dup; // ressíntese
6 }.play)
```

Código 3.26 – *SynthDef* básico de manipulação espectral

O Áudio 3.28 nos mostra uma característica inerente às transformações com vocoder de fase: uma sonoridade típica de banco de filtro ou de síntese granular senoidal, essencialmente caracterizada por frequências igualmente espaçadas e que costumam ressoar

ou apresentar a textura típica de granulações aleatórias. Essa característica sonora será ouvida ao longo de todas as transformações daqui em diante, naturalmente de acordo com as especificidades de cada processo.

Áudio 3.28 (3-28_pv_gendy_bins_instaveis.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Oscilador estocástico dinâmico proposto por Iannis Xenakis – *Gendy* – gradualmente transformado por um procedimento que seleciona somente os canais instáveis do vocoder de fase. À medida em que o tempo avança, selecionamos canais cada vez mais instáveis, conforme Código 3.26.

3.2.2 Modificação da escala temporal

3.2.2.0. a Definição

A modificação da escala temporal pelo vocoder de fase é feita utilizando diferentes tamanhos de saltos para a etapa de análise e síntese, ou seja modificando o fator de escala R_s/R_a. Quando o tamanho do salto de síntese é maior do que o tamanho de salto de análise, com fator de escala $R_s/R_a > 1$, o áudio será esticado; no caso contrário, com fator de escala $1 > R_s/R_a > 0$, teremos um encolhimento do áudio.

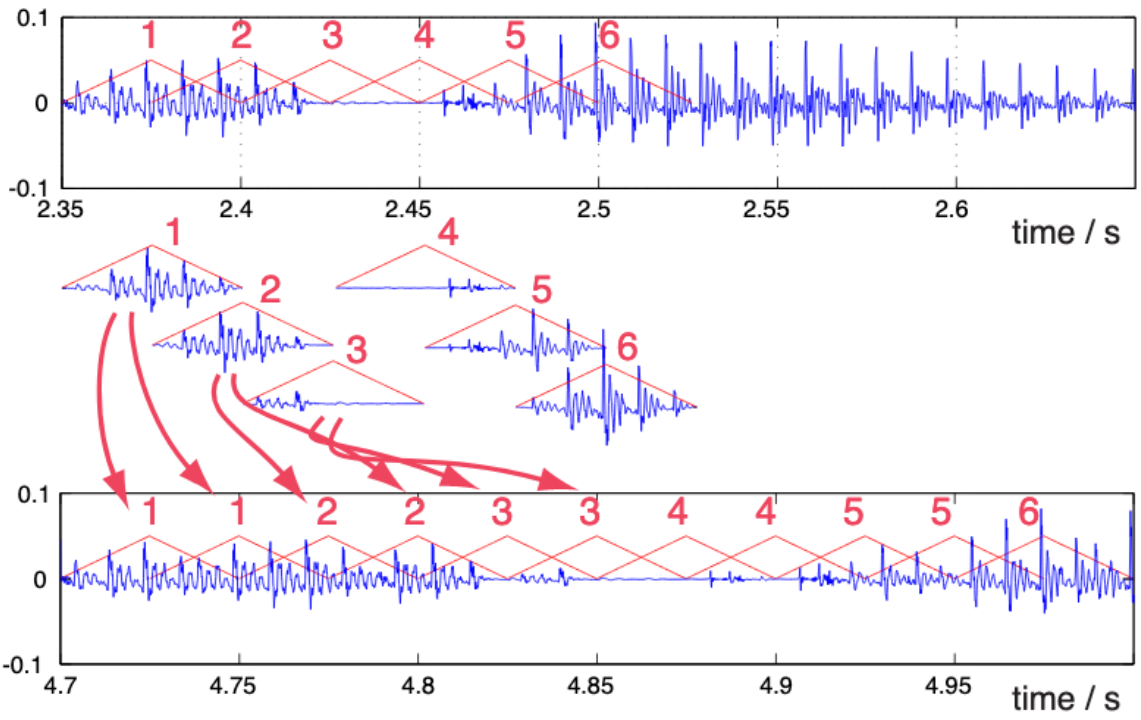
A ideia básica deste tipo de transformação é preservar as estruturas locais do áudio, ou seja fazer com que os pequenos blocos compõem as formas de ondas complexas sejam repetidos ou omitidos ao invés de serem esticados ou encolhidos. Isso garante que a altura e o espectro permaneçam praticamente inalterados, porém permite que a evolução temporal do objeto sonoro seja maleável, à semelhança de uma possível passagem mais lenta ou rápida do tempo (ELLIS, 2014).

A Figura 21 ilustra a ideia de preservação de estruturas de locais, sem significativas alterações na altura, que é utilizada nos vários métodos de esticamento temporal. A principal diferença entre o vocoder de fase e o método puramente temporal ilustrado aqui, é que o vocoder de fase tenta garantir o alinhamento de fase entre a nova sobreposição de quadros, ou seja, garantir que não aconteçam saltos de fase entre os quadros.

3.2.2.0. b Sonoridade

O esticamento temporal tende a revelar detalhes internos das estruturas sonoras cuja velocidade padrão de reprodução costuma mascarar. No casos extremos em que se cruza as fronteiras dos tipos de percepção auditiva, tais estruturas internas passam a ser percebidas como ritmos e mesmo como novas frequências. No caso do encolhimento temporal, estruturas de maior duração são encurtadas e as de menor duração são ainda

Figura 21 – Esticamento temporal por granulação



Fonte: Ellis (2014)

mais encolhidas, portanto obtém-se uma espécie de resumo perceptivo do objeto sonoro, no qual somente as características mais salientes poderão ser percebidas.

Conforme já detalhado na subseção 2.3.1.5, os principais artefatos gerados nesta transformação são o faseamento e uma espécie de reverberação. Com menor intensidade, aqui também é possível escutar o referido som de banco de filtros.

3.2.2.0. c Exemplos

A versão mais elaborada encontrada para a realização de esticamento temporal no SuperCollider utilizando o vocoder de fase foi proposta em Pluta, Alex e Altieri (2020)[14].

A principal dificuldade de ser fazer uma modificação da escala temporal em tempo real é conceitual: não parece ser possível alterar a velocidade do passar do tempo enquanto o tempo passa. Dadas as dificuldades computacionais de se realizar esta transformação em tempo real – explicitadas na citação a seguir – iremos quase sempre utilizar o modo NRT ou carregar *buffers* já previamente gravados.

[14] Essa implementação é um aprimoramento do clássico algoritmo desenvolvido Paul Nasca para esticamento temporal, o *PaulStretch*, que por sua vez também é um aprimoramento do vocoder de fase. Em resumo, descartam-se as fases originais e utilizam-se fases aleatórias bem como utiliza-se um tamanho de quadros decrescente (redução decrescente da resolução espectral) de forma a simular a percepção humana, reduzindo-se os artifícios causados pelo vocoder de fase.

> (...) por definição, as modificações na escala temporal não podem ser um efeito de áudio em tempo real, exceto por limitados períodos de tempo. Se a saída é encolhida em relação à entrada (*i.e.* toca-se mais rapidamente), então em algum ponto o efeito irá se tornar não-causal pois a saída irá depender de amostras futuras da entrada. Por outro lado, se a saída é esticada no tempo (toca-se mais lentamente), a diferença entre o tempo de entrada e saída irá se acumular constantemente e uma quantidade de memória sempre crescente será necessária para carregar o *buffer* do áudio a ser tocado. Entretanto, o caso oposto de modificação da escala de altura e frequência independentemente da escala temporal (...) é um tipo comum de efeito de áudio em tempo real. (REISS; MCPHERSON, 2014, p.207, tradução nossa).

O Código 3.27 realiza três versões de esticamentos sonoros acentuados, utilizando fatores 2, 4 e 8, respectivamente, executados em sons com características fortemente distintas e que podem ser escutadas no Áudio 3.29. Devido à típica dissolução de transientes que ocorre no esticamento temporal do vocoder de fase, o objeto sonoro que se torna mais descaracterizado é o da caminhonete dando partida, pois apresenta cíclicos ataques rápidos e sons bastante ruidosos, com conteúdo de frequência muito amplo, que também serão dissolvidos.

```
1 s.options.memSize = 16384; s.reboot;
2
3 TimeStretch.stretchNRT(PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3) +/+ "Audios/4_sinos_eletron_volt_f4000.wav", Platform
    .recordingsDir +/+ "4_sinos_eletron_volt_f4000_2_split4.wav", 2,
    numSplits: 4);
4
5 TimeStretch.stretchNRT(PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3) +/+ "Audios/4_sinos_eletron_volt_f4000.wav", Platform
    .recordingsDir +/+ "4_sinos_eletron_volt_f4000_4_split4.wav", 4,
    numSplits: 4);
6
7 TimeStretch.stretchNRT(PathName(thisProcess.nowExecutingPath).
    parentLevelPath(3) +/+ "Audios/4_sinos_eletron_volt_f4000.wav", Platform
    .recordingsDir +/+ "4_sinos_eletron_volt_f4000_8_split4.wav", 8,
    numSplits: 4);
```

Código 3.27 – Esticamento temporal utilizando uma variante do vocoder de fase

Áudio 3.29 (3-29_esticamento-2-4-8_SC_4_sinos_eletron_volt_f4000_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte é composto de três partes, na primeira, um som de sino percutido com baqueta rígida, com baqueta de feltro; riscado com metal e com metal serrilhado; na segunda parte temos o autor pronunciando a palavra “elétron-volt”; por fim temos um excerto do som da caminhonete diesel f4000 dando partida. Após o som fonte são apresentadas as versões esticadas em duas, quatro e oito vezes, nesta ordem, utilizando a variante do vocoder de fase do SuperCollider. Som fonte da

caminhonete retirado de <https://freesound.org/s/72027/>.

É possível notar também que o esticamento introduz mudanças no timbre, especialmente uma espécie de reverberação, ou no termo utilizado na literatura, um "faseamento". Esse fator é menos intenso para pequenos valores do fator de esticamento e também tende a ser bastante diferente de acordo com o tipo de vocoder de fase utilizado.

No Áudio 3.30 realizamos o mesmo padrão de esticamento, mas utilizando o algorítmo clássico do vocoder de fase disponível no CDP. Aqui, quando os ataques são esticados, já é possível ouvir uma espécie de variação aleatória de conjunto de banco de filtros ou um som similar à uma textura granular aleatória de senoides.

Áudio 3.30 (3-30_esticamento-2-4-8_CDP_4_sinos_eletron_volt_f4000_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte é composto de três partes, na primeira, um som de sino percutido com baqueta rígida, com baqueta de feltro; riscado com metal e com metal serrilhado; na segunda parte temos o autor pronunciando a palavra "elétron-volt"; por fim temos um excerto do som da caminhonete diesel f4000 dando partida. Após o som fonte são apresentadas as versões esticadas em duas, quatro e oito vezes, nesta ordem, utilizando a versão clássica do vocoder de fase do CDP. Som fonte da caminhonete retirado de <https://freesound.org/s/72027/>.

A variante do vocoder de fase do SuperCollider só permite a realização de encurtamentos temporais por meio da execução de um *script* na linguagem Python, conforme o Código 3.28. No Áudio 3.31 proveniente desta transformação, é possível perceber a introdução de glissandos nos ataques e uma forte dissolução dos transientes (como é o caso da dissolução do *allure* do sino sendo serrilhado).

```
1 "cd /TimeStretch_master/python/ && python /TimeStretch_master/python/
     nessstretch.py /TimeStretch_master/python/sinos.wav TimeStretch_master/
     python/sinos_encurt_r2.wav -r 2".runInTerminal;
2
3 "cd /TimeStretch_master/python/ && python /TimeStretch_master/python/
     nessstretch.py /TimeStretch_master/python/sinos.wav TimeStretch_master/
     python/sinos_encurt_r4.wav -r 4".runInTerminal;
4
5 "cd /TimeStretch_master/python/ && python /TimeStretch_master/python/
     nessstretch.py /TimeStretch_master/python/sinos.wav TimeStretch_master/
     python/sinos_encurt_r8.wav -r 8".runInTerminal;
```

Código 3.28 – Encurtamento temporal utilizando uma variante do vocoder de fase

Para o caso de sons ruidosos, como a gravação da caminhonete, o encurtamento temporal, neste caso, introduz um marcante artifício de sonoridade de banco de filtros ressoantes. Além disso, o encurtamento temporal tende a esmagar os transitórios rápidos

e alterar menos drasticamente os componentes estacionários, como pode ser percebido no sino tocado com baqueta de feltro.

Áudio 3.31 (3-31_encurt_SC_4_sinos_eletron_volt_f4000_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte é composto de três partes, na primeira, um som de sino percutido com baqueta rígida, com baqueta de feltro; riscado com metal e com metal serrilhado; na segunda parte temos o autor pronunciando a palavra "elétron-volt"; por fim temos um excerto do som da caminhonete diesel f4000 dando partida. Após o som fonte são apresentadas as versões encurtadas em duas, quatro e oito vezes (fatores 0.5, 0.25 e 0.125), nesta ordem, utilizando a versão clássica do vocoder de fase do CDP. Som fonte da caminhonete retirado de <https://freesound.org/s/72027/>.

No encurtamento temporal utilizando o algorítmo clássico do vocoder de fase, Áudio 3.32, as alterações no timbre e no ataque dos sons são menos destrutivas, tendendo a reter um pouco mais das características originais. Há, porém, no Áudio 3.32, um problema pouco contornável da STFT: existe um efeito de borda na ideia de janelamento com sobreposição, pois os primeiros e os últimos quadros de áudio não podem ser sobrepostos assim como ocorre nos quadros do meio do vetor de áudio (não existem quandros anteriores ao primeiro, somente após este, tampouco existem quadros após o último, somente antes deste.). Se o tamanho do quadro de janelamento for muito grande ou se arquivo de som for muito, este efeito se torna mais presente e perceptível. Isso, em conjunção com a forma típica das funções de janelas, faz com que tanto o início quanto o final de um áudio a ser janelado perca suas amostras, devido à atenuação da função de janela. No Áudio 3.32, como o primeiro som está muito próximo do início do arquivo de áudio, sem a inserção de silêncio inicial, uma porção do seu ataque será suavizada pelo janelamento.

Áudio 3.32 (3-32_encurt_CDP_4_sinos_eletron_volt_f4000_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

O som fonte é composto de três partes, na primeira, um som de sino percutido com baqueta rígida, com baqueta de feltro; riscado com metal e com metal serrilhado; na segunda parte temos o autor pronunciando a palavra "elétron-volt"; por fim temos um excerto do som da caminhonete diesel f4000 dando partida. Após o som fonte são apresentadas as versões encurtadas em duas, quatro e oito vezes (fatores 0.5, 0.25 e 0.125), nesta ordem, utilizando a versão clássica do vocoder de fase do CDP. Som fonte da caminhonete retirado de <https://freesound.org/s/72027/>.

3.2.3 Modificação da altura

3.2.3.0. a Definição

O conceito básico de modificação de altura de um sinal é o de multiplicação de todas as suas frequências por um dado fator de transposição – geralmente chamado de

transposição ou esticamento espectral – o que mantém as relações harmônicas entre as frequências que constituem os sons de altura definida. No domínio espectral, a forma mais simples de realizar esse procedimento é multiplicando a posição dos canais por um fator de escala, que é a forma disponível no SuperCollider.

Como já mencionado na seção 2.4, conceitualmente não é trivial elaborar um procedimento de modificação de altura, pois este atributo não é, perceptivamente, definido de maneira absoluta e precisa, logo não é possível elaborar uma definição psicoacústica fechada tampouco uma implementação computacional precisa de um conceito que é disforme ou pouco consensual. O CDP apresenta outras possibilidades de modificação de altura no domínio espectral, como a tentativa de preservação do contorno espectral e formantes por meio da extração e re-imposição do contorno espectral. Dada a amplidão deste assunto na literatura pesquisada, iremos nos concentrar somente nas técnicas simples e já implementadas.

3.2.3.0. b Sonoridade

O método de alteração de alturas via transposição espectral tende a introduzir inarmonicidade nos sons pois a divisão do espectro em canais (*bins*) discretos impede a multiplicação das frequências por valores precisos, logo, iremos perder as relações harmônicas necessárias para a percepção de altura definida.

Outro fator de modificação timbrística é a alteração do envelope espectral, pois um simples esticamento ou encurtamento da posição dos canais irá também causar o mesmo efeito no contorno espectral. Naturalmente, alterações em tal contorno provocam sensíveis alterações no timbre, em especial, na percepção de formantes, conforme ilustra a Figura 22.

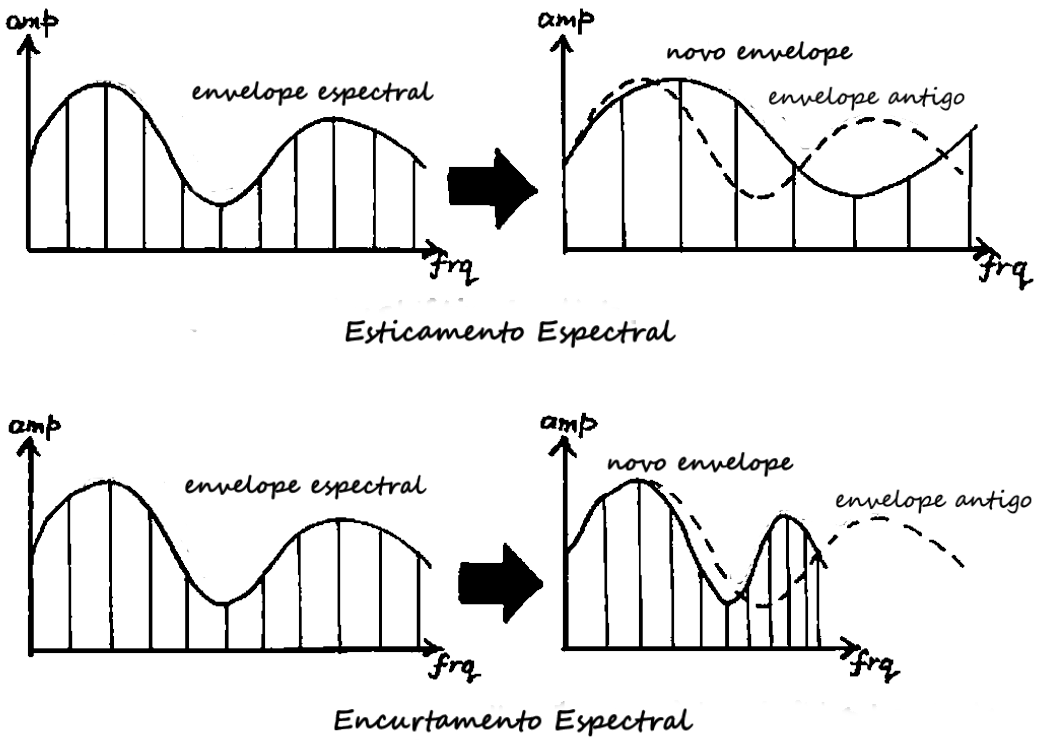
Essa transformação não altera o tamanho a duração do som, pois só são feitas modificações nos canais e não há modificação no passo da sobreposição-soma dos quadros da STFT. Todavia, este procedimento costuma deixar lacunas entre os canais, portanto uma interpolação espectral pode auxiliar a minimizar este fenômeno.

3.2.3.0. c Exemplos

No Áudio 3.33, gerado conforme o Código 3.29, é possível perceber diversos artifícios produzidos por essa manipulação espectral. Aqui é possível observar que há mudanças na evolução timbrística, nos sons de sinos , em especial, há a introdução de batimentos e mudanças nas frequências de ressonância.

```
1 b = Buffer.read(s,PathName(thisProcess.nowExecutingPath).parentLevelPath(3)
      +/+ "Audios/4_sinos_eletron_volt_f4000.wav");
2
3 (SynthDef(\modAltura, { arg out=0, soundBufnum = b.bufnum;
4     var in, chain;
```

Figura 22 – Modificação de altura por esticamento e encurtamento espectral



Fonte: Traduzido e adaptado de Wishart (1994, apêndice 2,p. 16)

```
    in = PlayBuf.ar(1, soundBufnum, BufRateScale.kr(soundBufnum), loop: 1);
    chain = FFT(LocalBuf(2048), in);
        chain = PV_BinShift(chain, 0.5);
    Out.ar(out, IFFT(chain).dup);
}).play(s,[\out, 0, \soundBufnum, b.bufnum]);)
```

Código 3.29 – Modificação de altura por esticamento e encurtamento espectral

Um dos artifícios mais salientes é o fato de modificações no contorno espectral induzirem a uma condicionada mudança de percepção das causas físicas do som transformado. Na parte vocal do Áudio 3.33, quando há um esticamento, existe a sensação de ouvirmos a voz de uma criança. Isso se dá pois o contorno espectral da voz humana, em especial seus picos (formantes), é fisicamente definido pelo tamanho do trato vocal. Assim, a audição humana instantaneamente estabelece uma ligação entre formantes deslocadas e tamanho físico do objeto que as produziu. No caso oposto, do encurtamento espectral, aparentamos ouvir uma voz produzida por um ser humano gigante, e não apenas uma voz com altura mais grave.

Áudio 3.33 (3-33_mod_altura_estic_spec_4_sinos_eletron_volt_f4000_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Para cada tipo de som fonte teremos as seguintes seções: som fonte, seguido de um esticamento espectral com fator 2 e um encurtamento espectral de fator 0.5. Os sons fontes, apresentados nesta ordem, foram um som de sino percutido com baqueta rígida; idem porém com baqueta de feltro; idem, porém riscado com metal ; idem, porém riscado com metal serrilhado; próprio autor pronunciando a palavra "elétron-volt"; excerto do som da caminhonete diesel f4000 dando partida. Som fonte da caminhonete retirado de <https://freesound.org/s/72027/>.

3.2.4 Deslocamento espectral

3.2.4.0. a Definição

No *deslocamento espectral* adiciona-se ou subtrai-se um fator para todos os canais do espectro, gerando um deslocamento linear deste canais. Diferentemente do *esticamento espectral* apresentado anteriormente, que inseria pequenas lacunas entre os canais espectrais, aqui teremos uma maior lacuna no início ou no final do espectro devido a tal deslocamento.

Conforme aponta Smith (2011) a realização de um deslocamento dos *bins* implica, no domínio do tempo, em uma multiplicação do sinal por uma senóide complexa, ou seja uma modulação de amplitude. Essa modulação precisa estar em fase ao longo dos quadros da STFT, logo, para a realização de um deslocamento espectral sem artifícios é necessário realizar uma etapa de correção de fase (SMITH, 2013).

3.2.4.0. b Sonoridade

A percepção de altura fixa está diretamente associada ao fenômeno de componentes de frequências separadas segundo uma relação harmônica, ou seja, múltiplos inteiros de uma frequência fundamental. Quando um fator de deslocamento é somado ou subtraído à som com esta característica, as relações harmônicas vão sendo desfeitas (inarmonicidade) e mais de uma altura tende a ser percebida.

Quando realizamos o deslocamento espectral, é possível obter um algoritmo rudimentar de modificação de altura, porém modificações timbrísticas também serão introduzidas. Além da introdução da inarmonicidade, uma das principais é um tipo de filtragem espectral, pois, quando deslocamos todos os *bins* para alguma direção, esvaziaremos os bins da direção oposta. Assim, um deslocamento espectral para o agudo, tende a funcionar auditivamente como um filtro passa altas, assim como um um deslocamento espectral para o grave tende a funcionar como um filtro passa baixas.

3.2.4.0. c Exemplos

O Código 3.30 implementa sucessivos deslocamentos espectrais para um mesmo *buffer*. Podemos usar essa ideia para aumentar gradualmente a inarmonicidade de um

som de fagote, que naturalmente apresenta pouca ou nenhuma inarmonicidade, conforme realizado no Áudio 3.34 .

```
(SynthDef(\deslocSpec, { arg out=0, buf = b;
    var in, chain;
    in = PlayBuf.ar(1, buf, BufRateScale.kr(buf), loop: 1);
    chain = FFT(LocalBuf(2048), in);
        chain = PV_BinShift(chain, 1.0,
        Demand.kr(
                        Impulse.kr(BufDur.kr(buf).reciprocal),
                        0,
                        Dseq([0,-6,0,-10,0,3,0,4,0,5,0,13], 1)
                )
        );
    Out.ar(out, IFFT(chain).dup);
}).play(s,[\out, 0, \buf, b]);)
```

Código 3.30 – Deslocamento espectral

Áudio 3.34 (3-34_desloc_spec_F1_fagote.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota fá-1 (F1) tocada em sustentação por um fagote, apresentada alternadamente com suas transformações de deslocamento espectral pelos seguintes fatores: -6, -10, 3, 4, 5 e 13. Amostras retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

O sons vocais dificilmente conseguem apresentar inarmonicidade à maneira que sinos e sistemas de modulação de frequência produzem (sons nos quais múltiplas alturas podem ser percebidas), tal característica só pode ser obtida por meio de manipulações eletrônicas ou computacionais dos sons vocais (WISHART, 1988). No Áudio 3.35 além do efeito alteração de formantes apresentada na *modificação de altura*, aqui também percebemos que cada fator de deslocamento irá promover intrínsecas alterações timbrísticas. Segundo terminologia schaefferiana, em alguns casos teremos mais batimentos, em outros perceberemos mais *massa sonora*, em outros maior sensação de *sons canelados*, etc.

Áudio 3.35 (3-35_desloc_spec_rato_vib_mult_alv.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Próprio autor pronunciando a palavra “rato” com a consonante r vibrante múltipla alveolar, apresentada alternadamente com suas transformações de deslocamento espectral pelos seguintes fatores: -6, -10, 3, 4, 5 e 6.

A implementação do deslocamento espectral com correção de fase apresentada por

Smith (2013) – presente no Código 3.31 com as devidas modificações para possibilitar uma leitura de *buffer* – depende da escolha de uma frequência fundamental para ser obter uma referência ao deslocamento. Assim, é natural que sons não tônicos, ausentes de altura claramente definida, seja difícil realizar um deslocamento espectral que gere poucos artifícios.

```
~sino1 = Buffer.read(s,"/sinos-baqueta.wav");
(
x = {
        var in, out, amp, f0=400, fftSize=8192, winLen=2048, hopFrac=0.25,
        chain, mexp, fScaled, df, binShift, phaseShift,
        inWinType=0, outWinType=0;

        amp = MouseX.kr(-60,10).dbamp;
            in = PlayBuf.ar(2,~sino1,loop: 0.0);

        chain = FFT(LocalBuf(fftSize), in, hopFrac, inWinType, 1, winLen);
        mexp = MouseY.kr(-2.0,2.0); // duas oitavas acima ou abaixo
        fScaled = f0 * (2.0 ** mexp);
        df = fScaled - f0;
        binShift = fftSize * (df / s.sampleRate);
        chain = PV_BinShift(chain, stretch:1, shift:binShift, interp:1);
        phaseShift = 2 * pi * binShift * hopFrac * (winLen/fftSize);
        chain = PV_PhaseShift(chain, phaseShift, integrate:1);

        out = IFFT(chain,outWinType,winLen);
        Out.ar(0, out.dup);
}.play
)
```

Código 3.31 – Deslocamento espectral com correção de fase

O exemplos sonoros deste tipo de deslocamento espectral mostrados no Áudio 3.36 revelam que ocorre uma modificação da altura percebida, porém acompanhada de diversos outros efeitos, tais como filtragem e introdução de inharmonicidade. No caso de sons cuja mistura de tonicidade e ruideza é complexa (voz e sons de motores) estes efeitos tendem a ser mais perceptíveis do que uma modificação de altura em si.

Áudio 3.36 (3-36_desloc_spec_correcao_fase.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Para cada tipo de som fonte são apresentados, em sequência, um deslocamento espectral de duas oitavas para baixo, uma oitava para baixo, uma oitava para cima e duas oitavas para cima. Os sons fontes, apresentados nesta ordem, foram um som de sino percutido com baqueta rígida; próprio autor pronunciando a palavra "elétron-volt"; excerto do som da caminhonete diesel f4000 dando partida. Som fonte da caminhonete retirado de <https://freesound.org/s/72027/>.

3.2.5 Borramento espectral (*Blur blur* e *Blur average*)

3.2.5.0. a Definição

No borramento espectral faz-se uma média entre os dados espectrais, diminuindo a resolução destes dados. Este processo pode se dar de duas maneiras: horizontalmente, ao longo do tempo, quando faz-se uma média entre os sucessivos canais espectrais (*bins*) de um dado número de quadros; verticalmente, em um certo instante temporal específico, quando faz-se uma média entre os canais espectrais adjacentes, contidos em um único quadro.

No CDP, o borramento entre os sucessivos quadros se dá pela função *Blur blur*, já o borramento entre os canais adjacentes de um mesmo quadro se dá pela função *Blur average*. Como diferença crítica, no SuperCollider, o *UGen* de borramento espectral entre quadros apresenta a forma específica de ser uma média recursiva na qual $magnitude_{atual} = (magnitude_{anterior} * fator) + (magnitude_{atual} * (1 - fator))$, diferentemente do CDP no qual é feita uma interpolação entre uma quantidade ajustável de quadros.

3.2.5.0. b Sonoridade

De maneira geral, quando temos um borramento espectral entre os canais de um mesmo quadro a tendência é de dissolução de alturas fixas, aumento ou introdução de inarmonicidade, batimentos e ruído; no caso do borramento entre diferentes quadros, a tendência geral é que aconteça uma maior sustentação das alturas e parciais definidas, uma dissolução de transientes rápidos e brandas atenuações nos ruídos. Como o borramento espectral ao longo dos quadros tende a dissolver o sinal ao longo do tempo, esta transformação apresentará também características similares à reverberação.

Para os casos de sons sintetizados como ruídos (branco, rosa, acastanhado, etc) e oriundos de modulação em frequência (chilros ou *chirp*) estas transformações espectrais tendem a ter pouco efeito ou demandarem parâmetros agressivos para que suas alterações possam ser percebidas.

3.2.5.0. c Exemplos

No Áudio 3.37, e sua respectiva implementação, Código 3.32, versões cada vez mais borradas verticalmente (entre os canais espectrais adjacentes de um mesmo quadro) são apresentadas, mostrado como esta transformação tende a ressaltar os aspectos ruídosos e reduzir os aspectos tônicos.

```
1 (SynthDef(\borramentoCanais, { arg out=0, buf = b;
2         var in, chain;
3         in = PlayBuf.ar(1, buf, BufRateScale.kr(buf), loop: 1);
```

```
        chain = FFT(LocalBuf(2048), in);
        chain = PV_MagSmear(chain,
                Demand.kr(
                        Impulse.kr(BufDur.kr(buf).reciprocal),
                        0,
                        Dseq([0, 3, 7, 50, 100], 1)
                )
        );
        Out.ar(out, IFFT(chain).dup);
}).play(s,[\out, 0, \buf, b]);)
```

Código 3.32 – Borramento espectral entre canais (*bins*)

Áudio 3.37 (3-37_borra_spec_canais_4096_C3_orq.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano; em seguida é feito o borramento espectral entre os canais (*bins*) adjacentes utilizando uma janela de 4096 pontos, sendo que a quantidade de canais espectrais borrados é, na seguinte ordem, 0, 3, 7, 50 e 100. Sons de intrumentos orquestrais retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

Neste tipo de transformação, o tamanho da janela temporal (e consequentemente a resolução espectral de cada quadro) adotada será fundamental para o tipo de resultado sonoro. O Áudio 3.38 mostra o mesmo processo anterior, porém com uma janela dezesseis vezes menor. Se optarmos por janelas menores teremos uma maior e mais irregular introdução de inarmonicidade (devido à menor resolução espectral); na opção por janelas maiores, teremos mais controle sobre a inarmonicidade porém os transitórios rápidos do sinal tenderão a ser mais dissolvidos (menor resolução temporal).

Áudio 3.38 (3-38_borra_spec_canais_256_C3_orq.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano; em seguida é feito o borramento espectral entre os canais utilizando uma janela de 256 pontos, sendo que a quantidade de canais borrados é, na seguinte ordem, 0, 3, 7, 50 e 100. Sons de intrumentos orquestrais retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

No caso do borramento espectral ao longo do tempo, ou seja, entre quadros, cuja implementação é apresentada em Código 3.34, naturalmente podemos perceber uma dissolução dos ataques mais rápidos e uma manutenção dos sons contínuos. No Áudio 3.39, é possível perceber que ocorre aqui uma espécie de reverberação, ressaltando-se as partes

mais contínuas do espectro. No caso do trava língua russo, por exemplo, a gravação original apresenta uma frequência espúria, que vai se destacando à medida aumentamos o fato de suavização.

```
(SynthDef(\borramentoQuadros, { arg out=0, buf = b;
        var in, chain;
        in = PlayBuf.ar(1, buf, BufRateScale.kr(buf), loop: 1);
        chain = FFT(LocalBuf(2048), in);
        chain = PV_MagSmooth(chain,
                Demand.kr(
                        Impulse.kr(BufDur.kr(buf).reciprocal),
                        0,
                        Dseq([0, 0.8, 0.85, 0.9, 0.95], 1)
                )
        );
        Out.ar(out, IFFT(chain).dup);
}).play(s,[\out, 0, \buf, b]);)
```

Código 3.33 – Borramento espectral entre quadros

Áudio 3.39 (3-39_borra_spec_quadros_2048_gato_russo_cantor.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Três sons fontes foram utilizados: trava língua russo com o dizer "Brejnev raspou as sobrancelhas com uma navalha"; um gato esturrando; um cantor lírico sustentando uma nota lá-4 (A4) com a vogal "a". Apresenta-se o cada som fonte seguido do borramento espectral entre os quadros utilizando uma janela de 2048 pontos, sendo que o fator de suavização é, na seguinte ordem, 0, 0.8, 0.85, 0.9 e 0.95. Os sons originais foram retirados, respectivamente, de <https://freesound.org/s/120160/>, <https://freesound.org/s/475464/>, <https://freesound.org/s/395421/>.

Esta ideia de dissolução dos ataques mais rápidos e uma manutenção dos sons contínuos pode ser usada, contrariamente, em um algoritmo redutor de ruído, conforme indicado na própria documentação deste *UGen*. Se tiver um ruído constante no sinal, o borramento espectral irá ressaltar essa continuidade, que quando subtraída do sinal original atenua os sons contínuos, dentre eles o ruído e a reverberação.

O Código 3.34, adaptado da documentação de *PV_MagSmooth*, utiliza a posição X do cursor do mouse para ajustar o fator de atenuação e o clique do mesmo para alternar, em tempo real, entre a versão original e transformada. NoÁudio 3.40 apresentamos estes dois sons separadamente.

```
c = Buffer.read(s, "~\Audios\f4000—mono.wav")

({
var son, chain, chainorig, chainsmooth, out;

```

```
6  son = PlayBuf.ar(1, c, loop: 1);
7
8  chain = FFT(LocalBuf(2048), son);
9  chainorig = PV_Copy(chain, LocalBuf(2048));
10 chainsmooth = PV_MagSmooth(chain, (1 - MouseX.kr(1, 0.00001, 1)).poll);
11 chain = PV_MagSubtract(chainorig, chainsmooth, 1);
12
13 out = (XFade2.ar(IFFT(chain), son, MouseButton.kr(-1,1))).dup * 0.9;
14 }.play;)
```

Código 3.34 – Redução de ruídos utilizando Borramento espectral entre quadros

Áudio 3.40 (3-40_red_ruido_borra_spec_quadros_f4000.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.* Som de caminhonete f4000 dando partida seguida da sua versão com redução de ruídos por meio do borramento espectral. Som original retirado de <https://freesound.org/s/72027/>.

3.2.6 *Chorus* espectral (*blur chorus*)

3.2.6.0. a Definição

O *chorus espectral* consiste em aplicar diferentes tempos de atraso (*delay*) e diferentes taxas de realimentação (*feedback*) para cada canal do espectro. O nome *chorus* remete à sonoridade de um coral e, historicamente, este efeito é simulado pela a soma de múltiplas cópias levemente defasadas de um mesmo sinal, ou seja, múltiplos *delays* com pequenos e diferentes tempos de atraso, geralmente também acrescidos de lentas modulações.

No CDP, o termo *blur chorus* refere-se às ligeiras randomizações dos canais ao longo do tempo, modificando suas posições para que ocorra esse efeito múltiplas e ligeiras desafinações de um coral. Isso pode ser obtido utilizando combinando ou automatizando os processos de borramento, deslocamento e esticamento espectrais apresentados anteriormente. Todavia, notamos que, sonoramente, os resultados do CDP não são parecidos com os tradicionais *chorus*, portanto, propomos aqui a avaliar um atrasador (*delay*) espectral do SuperCollider como alternativa mais próxima à sonoridade desejada.

3.2.6.0. b Sonoridade

Tradicionalmente o efeito *chorus* se encontra no campo das modifições temporais que tendem a estar associadas à espacialização ou à simulação de grandes grupos de fontes sonoras simultâneas. Visto que técnicas muito semelhantes de manipulação temporal são utilizadas para a produção de reverberação, atraso, eco, efeito doppler e chorus, é

comum alcançar todos estes tipos de sonoridade exclusivamente por meio da modificação dos parâmetros de controle dos atrasadores. O mesmo se aplica na versão espectral, frequentemente migramos da região perceptiva de *chorus* para os demais fenômenos.

No *chorus* espectral aqui proposto, diferentes sonoridades podem ser alcançadas pois ajustes mais finos podem ser feitos: no domínio espectral temos agora todos os parâmetros de controle temporal porém para cada canal de frequência. Esta separação em canais, entretato, tende a gerar o efeito de segregação auditiva das múltiplas cópias para parâmetros mais agressivos.

3.2.6.0. c Exemplos

No Código 3.35, adaptado da sua própria documentação, temos uma implementação na qual os parâmetros são modificáveis por meio de uma GUI. Por ser um cálculo computacional bastante dispendioso, a realização deste processo em tempo real costuma apresentar maior latência e o aumento no tamanho da janela da FFT poderá até mesmo travar o servidor de áudio.

```
1  s.boot;
2  z = Buffer.read(s, "C:/Users/f11mi/Google Drive/dirce drive/Dirce_1_3-11/
       Audios/imitacao-sapo-composto_mono.wav");
3
4  (//Na GUI o controles superiores representam o tempo de atraso e os
       inferiores a quantidade de realimentação
5  s.doWhenBooted({
6          var size, fftSize, awin, delaySilder, fbSlider, maxdel, synth, cond
               , playbuf;
7          var setup, onReadyFunc, fftBuffer, delTimeBuffer, fbAmtBuffer,
               createGUI, liveInputMix;
8
9          size = 512;
10         fftSize = size * 2;
11         maxdel = 0.5;
12         cond = Condition.new;
13         liveInputMix = 0.0; // modifique para um para ouvir a entrada de
               mic ao invés do PlayBuf
14
15         SynthDef(\helpBinDelay, { arg inbus=0, inMix = 0.0, out=0, fftBuf
               =0, delayBuf=0, fbBuf=0;
16               var in, chain;
17               in = (PlayBuf.ar(1, z, BufRateScale.ir(z), loop: 1) * (1.0
                     - inMix)) + (In.ar(inbus, 1) * inMix);
18               chain = FFT(fftBuf, in, 0.25);
19               chain = PV_BinDelay(chain, maxdel, delayBuf, fbBuf, 0.25);
20               OffsetOut.ar(out,
21                       (in + IFFT(chain) * -12.dbamp).dup// inverse FFT
```

```
22                );
23        }).add;
24
25        setup = {
26                Routine.run({
27                        "Allocating FFT buffer".postln;
28                        fftBuffer = Buffer.alloc(s, fftSize, 1);
29                        "Allocating DelTime buffer".postln;
30                        delTimeBuffer = Buffer.alloc(s, size, 1);
31                        "Allocating FB buffer".postln;
32                        fbAmtBuffer = Buffer.alloc(s, size, 1);
33                        s.sync(cond);
34                        onReadyFunc.value()
35                });
36        };
37
38        createGUI = {
39
40                awin = Window("test", Rect(200 , 450, 10 + (size * 1), 10 +
                      (size * 2)));
41                awin.view.decorator = FlowLayout(awin.view.bounds);
42
43                delaySilder = MultiSliderView(awin, Rect(0, 0, size * 1,
                     size * 1));
44                delaySilder.action = {arg xb;
45                        ("Deltime index: " ++ xb.index ++" value: " ++ (xb.
                           currentvalue * maxdel)).postln;
46                        delTimeBuffer.set(xb.index, xb.currentvalue *
                           maxdel)
47                };
48
49                fbSlider = MultiSliderView(awin, Rect(0, 0, size * 1, size
                     * 1));
50                fbSlider.action = {arg xb;
51                        ("FB index: " ++ xb.index ++" value: " ++ xb.
                           currentvalue).postln;
52                        fbAmtBuffer.set(xb.index, xb.currentvalue)
53                };
54
55                [delaySilder, fbSlider].do({arg thisSliderView;
56                        var initDataArray;
57                        initDataArray = Array.fill(size, {0.0});
58                        thisSliderView.value_(initDataArray);
59                        thisSliderView.xOffset_(5);
60                        thisSliderView.thumbSize_(12.0);
61                        thisSliderView.valueThumbSize_(15.0);
62                        thisSliderView.indexThumbSize_( thisSliderView.
```

```
                                bounds.width / initDataArray.size );
63                          thisSliderView.gap = 0;
64                  });
65
66                  awin.front;
67
68                  awin.onClose_({
69                          synth.free;
70                          fftBuffer.free;
71                          fbAmtBuffer.free;
72                          delTimeBuffer.free;
73                  })
74          };
75
76          onReadyFunc = {
77                  createGUI.defer();
78
79                  synth = Synth(\helpBinDelay, [
80                          \inbus, s.options.numOutputBusChannels,
81                          \inMix, liveInputMix,
82                          \out, 0,
83                          \fftBuf, fftBuffer.bufnum,
84                          \delayBuf, delTimeBuffer.bufnum,
85                          \fbBuf, fbAmtBuffer.bufnum,
86                  ]);
87          };
88
89          setup.value();
90  }))
```

Código 3.35 – *Chorus* espectral

O Áudio 3.41, registra uma gravação na qual manipulamos a GUI em tempo real. No início temos valores de chorus mais brandos e seguimos aumentando gradativamente os parâmetros, sempre mantendo alguma aleatoriedade entre os parâmetros de cada canal. Ao final, valores extremos nos parâmetros são testados, o que naturalmente irá gerar sons contínuos e de espectro bastante cheio.

Áudio 3.41 (3-40_chorus_spec_imitacao_sapo.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Um som de imitação de sapo é submetido a contínuas variações no parâmetros de *chorus* espectral. No princípio o tempo de atraso e a quantidade de realimentação são mais brandos e aos poucos selecionamos valores mais intensos. A modificação dos parâmetros se deu de forma a manter sempre alguma diferença entre cada canal, sendo que tal diferença foi crescendo ao longo da gravação. Para o som fonte, trechos de duas gravações foram misturados. Sons originais foram retirados de

<https://freesound.org/s/447917/> e <https://freesound.org/s/66584/>.

3.2.7 Focalização espectral (*Focus Accu*)

3.2.7.0. a Definição

A *focalização espectral* consiste em ressaltar alguns aspectos específicos do conteúdo espectral. Na abordagem utilizada aqui, mantêm-se os canais mais estáveis (geralmente o conteúdo harmônico) ou instáveis (geralmente o conteúdo ruidoso) dentro de uma dada quantidade de janelas e um certo limiar no valor de magnitude e fase.

3.2.7.0. b Sonoridade

Na sustentação dos *bins* mais estáveis, parece haver uma valorização do envelope espectral, especialmente pelo fato dos movimentos entre os picos das formantes ficarem mais evidentes auditivamente. Isso corresponde a uma sonoridade semelhante a uma mudança de vogal na fala ou variações contínuas de filtros de banda passante. Além disso, esse procedimento também tende a suavizar os ataques rápidos, o que pode fornecer a sensação de mudança física do tipo de instrumento ou do mecanismo de excitação do corpo vibrante.

Este procedimento tende a apresentar menos artifícios para sons modais (sons cujos modos de vibração tendem a permanecer estáticos e bem definidos, por exemplo, cordas tensionadas pinçadas, sinos percutidos, dentro outros) como foi o caso em maior grau da marimba e em menor grau do piano. No caso do violino, que é tocado com arcada e não com *pizzicatto*, sons de filtro pente variáveis podem ser ouvidos à medida de mudamos o tamanho da janela de sustentação.

Quando sustentou-se os *bins* mais instáveis, naturalmente diminui-se a tonicidade e a sensação de altura definida dos sons. Este processo tendeu a funcionar bem para os casos de ataques rápidos, como os instrumentos percutidos, porém a sonoridade de sons canelados – bancos de filtros de frequências fixas – pode ser facilmente percebida nestes.

3.2.7.0. c Exemplos

O Código 3.36 mostra um procedimento básico de realização da sustentação dos *bins* estáveis[15], adaptado da documentação do SuperCollider. No Áudio 3.42 podemos ouvir os efeitos mencionados na seção anterior. Naturalmente, os sucessivos processos tiveram de ser normalizados em amplitude devido à redução de informação causar uma grande perda de intensidade.

[15] A focalização dos canais (*bins*) instáveis é feita utilizam o *UGen* complementar *PV_NoiseSynthP* no referido código.

```
b = Buffer.read(s,"~\Audios\C3-marimba-violino-trompa-fagote-trompete-piano
    -mono.wav");

(SynthDef(\focaSpec, { arg out=0, buf = b;
    var in, chain;
    in = PlayBuf.ar(1, buf, BufRateScale.kr(buf), loop: 0);
    chain = FFT(LocalBuf(4096), in);
        chain = PV_PartialSynthP(chain, 0.3, 16);
    Out.ar(out, IFFT(chain).dup);
}).play(s,[\out, 0, \buf, b]);)
```

Código 3.36 – Focalização espectral

Áudio 3.42 (3-42_focal_spec_estaveis_ruido_inst_orq.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota dó-3 (C3) tocada em *staccatto* em uma marimba, violino, trompa, fagote, trompete e piano; em seguida é feita a focalização espectral sustentando-se os *bins* mais estáveis, utilizando um limiar de 0.3 e uma quantidade de quadros em sustentação iguais a 2, 4, 8 e 16, nessa ordem. Por fim, sustentam-se somente os bins não estáveis dentro de dois quadros, com limiares iguais a 0.3 e 0.6, nessa ordem. Sons de intrumentos orquestrais retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

3.2.8 Congelamento espectral (*focus freeze* e *focus hold*)

3.2.8.0. a Definição

No *congelamento espectral* uma dada quantidade de janelas espectrais é tocada em *loop*.

Duas variantes são comuns neste processo, na primeira temos a ideia de se congelar somente a magnitude das janelas e deixar a fase seguir[16] ou então congelar ambas magnitude e fase. Na segunda variante, optamos por um congelamento em que há retenção ou não do material que está sendo tocado(no CDP, *focus hold*), ou seja, se após o congelamento devemos voltar ao ponto de onde paramos ou se devemos seguir para o ponto do transcorrimento normal do tempo.

3.2.8.0. b Sonoridade

O tamanho da janela de análise irá definir se, perceptivamente, estaremos realizando uma espécie de síntese de tabela de onda (*wavetable synthesis*)– repetição rápida e cíclica dos valores tabela que contém uma forma de onda de forma a gerar alturas e frequências

[16] Via o *UGen PV_MagFreeze*

audíveis – ou se estaremos realizando um tipo de granulação – repetição mais lenta de blocos sonoros compostos de algumas dezenas de comprimentos de onda.

Quando realizamos o congelamento somente da magnitude, deixando a fase "correr", há um efeito parecido com o da reverberação e também próximo da robotização (quando ajustamos todas as fases para um valor fixo, tendo assim uma sonoridade do vocoder de canal ao invés do vocoder de fase).

3.2.8.0. c Exemplos

O Código 3.37 expõe o congelamento espectral com e sem retenção, no qual um oscilador aleatório é utilizado para acionar os momentos em que o espectro será congelado. A diferença entre tamanho de janelas pode ser escutada no Áudio 3.43 e também podemos perceber a presença ou não de saltos na leitura do *buffer*.

```
b = Buffer.read(s,"~\Audios\f4000-mono.wav);

(SynthDef(\congelaSpec, { arg out=0, buf=b;
    var in, chain;
    in = PlayBuf.ar(1, buf, BufRateScale.kr(buf), loop: 0);
    chain = FFT(LocalBuf(4096), in);
    chain = PV_Freeze(chain, LFDNoise3.kr(20) > 0.1 );
    Out.ar(out, IFFT(chain).dup);
}).play(s,[\buf, b]);)

//Congelamente espectral com retenção
(SynthDef(\congelaSpecRetem, { arg out=0, buf=b;
 var in, chain, t_trig;
        t_trig = LFDNoise3.kr(1) > 0.1;
    in = BufRd.ar(1, b, Phasor.ar(0, BufRateScale.kr(buf) * abs(t_trig-1),
       0, BufFrames.kr(buf)), );
    chain = FFT(LocalBuf(8192), in);
    chain = PV_Freeze(chain, t_trig );
    Out.ar(0, IFFT(chain).dup);
}).play(s,[\buf, b]);)
```

Código 3.37 – Congelamento espectral

Áudio 3.43 (3-43_congela_spec_simples_e_retencao_f4000.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Som de caminhonete F4000 diesel tentando dar a partida, seguida de congelamento espectral sem retenção (com uma janela de 4096 pontos) e congelamento espectral com retenção (com uma janela de 8192 pontos). Som fonte retirado de <https://freesound.org/s/72027/>.

3.2.9 Síntese cruzada

3.2.9.0. a Definição

O objetivo geral da síntese cruzada é combinar a altura de um sinal de áudio com o timbre de outro sinal. Este modelo baseia-se na ideia de que os sons, em geral, são constituídos por um sinal de excitação e um sinal de filtragem. A nomenclatura padrão é chamar o primeiro de sinal fonte (ou excitação, modulador) e o segundo de sinal filtro (ou portadora). No domínio da frequência, esse processo equivale a impor o envelope espectral de um dado som no conteúdo espectral de outro som.

O algoritmo genérico da síntese cruzada consiste em extrair o envelope espectral de ambos os sons, em seguida divide-se o espectro do sinal filtro (portadora) pelo seu próprio envelope espectral e, por fim, impõe-se (multiplicação) o envelope espectral do sinal fonte (moduladora) no espectro aplainado do sinal filtro obtido anteriormente (SMITH, 2020). Abordaremos aqui quatro maneiras[17] de realizar esse processo:

- *Intercâmbio magnitude-fase* – combina-se a magnitude de um sinal com a fase de outro sinal;
- *Estampagem de timbre* – normaliza-se a magnitude do sinal filtro ($1/magnitude$) e multiplica-se a magnitude sinal fonte por estes valores normalizados. Por fim, multiplica-se este resultado pelas magnitudes do sinal filtro original, mantendo as fases do sinal filtro (PUCKETTE, 2007, p. 278).
- *Filtragem de Cepstrum* – realiza-se uma segunda transformada de Fourier no logaritmo da magnitude do sinal original, e, então, mantêm-se somente os *bins* de baixa ordem, o que permite obter o envelope espectral deste sinal. Realiza-se este procedimento para ambos os sinais (fonte e filtro), de forma obter o envelope espectral de ambos, e então realiza-se a etapa tradicional de normalização e cruzamento dos espectros (PADOVANI, 2009) ;
- *Codificação Linear Preditiva (LPC)* – Aqui o envelope espectral é estimado temporalmente[18] a partir de um conjunto de filtros de resposta ao impulso infinita (IIR), gerando um conjunto de coeficientes temporais que irão ponderar os pontos do sinal fonte (PARK, 2010, p. 283-290) ;

[17] Estes métodos aparecem ao longo da documentação do CDP e de Wishart (1994) como possibilidades, porém de forma pouco detalhada e alguns não estão implementados. Optamos por incluí-los aqui como um acréscimo, assim não fornecemos os diversos detalhes teóricos destes métodos na seção 2.1, o que demandaria um grande espaço neste texto.

[18] Apesar dos cálculos do LPC serem feitos exclusivamente no domínio temporal, este algoritmo está apresentado nessa seção pois todo seu embasamento é feito visando promover alterações no domínio espectral.

3.2.9.0. b Sonoridade

O vocoder de canal é um exemplo rudimentar de síntese cruzada, assim, os resultados sonoros obtidos com os algoritmos propostos nesta seção tendem a ser similares, porém mais flexíveis e complexos. De maneira análoga ao funcionamento deste, espera-se que o sinal filtro (portadora) seja um sinal com espectro complexo e denso – ruídos, ondas com descontinuidades abruptas, sons naturais como vento, mar, explosões ou instrumentos de espectro denso, tais como cravo, percussões e sons orquestrais – o que garantirá que nenhuma faixa do espectro do sinal fonte (moduladora) seja anulada, resultando em perda de informação. Espera-se que o sinal de fonte (moduladora) seja não estacionário, e, preferencialmente, apresente um contorno espectral variável – tipicamente sons de voz ou de animais, sons polifônicos, sons sintéticos com modulações de amplitude facilmente perceptíveis, etc – pois é essa dinamicidade que irá garantir a modulação do filtro, logo, a percepção do efeito final.

De maneira geral, a sonoridade da síntese cruzada tenderá a combinar as características da excitação com as da filtragem, ou seja, se utilizarmos um som de voz como excitação e um som de mar como filtro, termos uma "voz que fala com um timbre de mar", de outra maneira, se utilizarmos um mugido de vaca como modulador de um dó-3 (C3) de uma onda quadrada, teremos uma espécie de mugido de vaca afinado nesta nota e também com características timbrísticas de uma onda quadrada.

Como timbre e altura não são parâmetros facilmente classificáveis e separáveis, tanto teoricamente quanto auditivamente, nos processos de síntese cruzada haverá uma espécie de "percentual de contribuição" deste fatores. Este ponto torna a combinação entre os sinais bastante complexa, o que promove resultados sonoros bastante variáveis e dinâmicos.

3.2.9.0. c Exemplos

A combinação entre vetor de magnitude de um som e vetor de fase de outro som é uma maneira elementar de se realizar a síntese cruzada. Este é um procedimento simples, conforme é apresentado no Código 3.38, é necessário apenas um *UGen* para realizar esta operação.

```
1 ~orq = Buffer.read(s,"~\Audios\orquestra.wav");
2 ~fgtF1 = Buffer.read(s,"~\Audios\fagote.wav");
3
4 SynthDef(\trocaFaseMag, {arg tamanhoQuadro=2048;
5         var inA, chainA, inB, chainB, chain;
6         inA = PlayBuf.ar(1, ~orq, BufRateScale.kr(~orq), loop: 0) * 1.0;
7         chainA = FFT(LocalBuf(tamanhoQuadro), inA);
8         inB = PlayBuf.ar(1, ~fgtF1, BufRateScale.kr(~fgtF1), loop: 0,
            doneAction: 2);
```

```
        chainB = FFT(LocalBuf(tamanhoQuadro), inB);
        chain = PV_CopyPhase(chainA, chainB);
        Out.ar(0, 0.9 * IFFT(chain).dup);
}).play;)
```

Código 3.38 – Síntese cruzada por Intercâmbio de magnitude e fase entre dois sons

Percebe-se no Áudio 3.44, que há, de fato, uma combinação entre altura de um som e timbre do outro som, porém, não há aqui uma separação nítida entre entre estes parâmetros. Assim, é possível perceber a altura dois sons, porém, o sinal do qual foram utilizadas as fases tende a contribuir mais para a percepção da altura.

Áudio 3.44 (3-44_intercambio_magnitude_fase_orquestra_fagote_trovao_cantor.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Nota fá-1 (F1) tocada em sustentação por um fagote, seguida de excerto orquestral, seguidos de intercâmbio de magnitude e fase entre estes dois sons com janela de 2048 pontos, orquestra fornecendo as magnitudes e fagote fornecendo as fases; cantor lírico sustentando uma nota lá-4 (A4) com a vogal "a", seguido por um som de um trovão, seguidos de intercâmbio de magnitude e fase entre estes dois sons com janela de 1024 pontos, trovão fornecendo as magnitudes e cantor fornecendo as fases. Amostra de fagote retirada de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>. Som de trovão e cantor retirados de <https://freesound.org/s/243614/> e <https://freesound.org/s/395421/>, respectivamente. Excerto orquestral retirado de Johannes Brahms - Op.45 *Ein Deutsches Requiem - Denn wir haben hie keine bleibende Statt*, The Holden Consort Orchestra and Choir, CC BY-SA 2.0 <https://commons.wikimedia.org/wiki/File:Johannes_Brahms_-_Op.45_Ein_Deutsches_Requiem_-_(06)_Denn_wir_haben_hie_keine_bleibende_Statt.oga>

A implementação do processo de *estampagem de timbre*, conforme o Código 3.40, inclui uma etapa de compressão espectral proposta por (PUCKETTE, 2007, p. 276). As alterações neste fator de compressão irão definir o quão agressiva ou suave é a extração da altura do sinal de excitação bem como a extração do timbre do sinal de filtragem.

```
~orq = Buffer.read(s,"~\Audios\orquestra.wav");
~fgtF1 = Buffer.read(s,"~\Audios\fagote.wav");

({
    var inControl,inFilter, chainFilter, chainControl, chainMain,
        tamanhoQuadro=2048, fatorComp=40;

    inControl = PlayBuf.ar(1, ~orq, BufRateScale.kr(~orq), loop: 1) *1;
        chainControl = FFT(LocalBuf(tamanhoQuadro), inControl);

        inFilter = PlayBuf.ar(1, ~fgtF1, BufRateScale.kr(~fgtF1), loop: 0,
            doneAction: 2) *1;
        chainFilter = FFT(LocalBuf(tamanhoQuadro), inFilter);
```

```
12
13
14      chainMain = chainFilter.pvcalc2(chainControl, tamanhoQuadro, {|mags,
            phases, mags2, phases2|
15                  [
16                          ((mags + 1e-20).reciprocal.clip(0.0,fatorComp
                                ** 2*0.01)) * mags2,
17                          phases
18                  ]
19      });
20          chainMain = PV_MagMul(chainFilter, chainMain);
21
22      0.8*IFFT(chainMain).dup
23  }.play;)
```

Código 3.39 – Síntese cruzada por estampagem de timbre

Conforme o Áudio 3.45, no processo de *estampagem de timbre* se torna auditivamente mais clara a separação entre sinal de excitação e sinal de filtragem, especialmente em relação à ideia rudimentar do intercâmbio de magnitude e fase. Apesar da síntese cruzada geralmente permitir uma identificação de cada som em separado (de maneira similar ao que ocorre no *efeito festa de coquetel*), é possível perceber que a fusão entre os dois sons ocorre de forma bastante convincente, a ponto de tornar difícil a separação auditiva analítica entre os elementos de cada som. Especialmente no exemplo do fagote e do excerto orquestral, altura e timbre se misturam de maneira complexa.

Áudio 3.45 (3-45_estampagem_timbre_orquestra_fagote_trovao_cantor.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Excerto orquestral, seguido de nota fá-1 (F1) tocada em sustentação por um fagote, seguidos da estampagem timbrística deste dois sons com janela de 2048 pontos, orquestra como som fonte e fagote como som filtro; Som de um trovão, seguido por um cantor lírico sustentando uma nota lá-4 (A4) com a vogal "a", seguidos da estampagem timbrística deste dois sons com janela de 1024 pontos, trovão como som fonte e cantor como som filtro. Amostra de fagote retirada de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>. Som de trovão e cantor retirados de <https://freesound.org/s/243614/> e <https://freesound.org/s/395421/>, respectivamente. Excerto orquestral retirado de Johannes Brahms - Op.45 *Ein Deutsches Requiem - Denn wir haben hie keine bleibende Statt*, The Holden Consort Orchestra and Choir, CC BY-SA 2.0 <https://commons.wikimedia.org/wiki/File:Johannes_Brahms_-_Op.45_Ein_Deutsches_Requiem_-_(06)_Denn_wir_haben_hie_keine_bleibende_Statt.oga>

Na filtragem de cepstrum Hsu (2012), conforme adaptação apresentada em Código 3.40, propõe a utilização de um filtro espectral *brickwall* (muro de tijolos) para suavizar o envelope espectral obtido no cepstrum, um filtro que zera os *bins* a partir de um dado ponto do espectro. A frequência de corte deste filtro irá determinar a referida

proporção da extração altura/timbre de cada sinal.

```
(~envc = Buffer.alloc(s, 1024);
~envm = Buffer.alloc(s, 1024);)

(SynthDef(\cepstralCrossSynth, {|out = 0, modBuf = 0, carBuf = 1, envSmooth
    = -0.99|
var in, in2, chain, chain2, chain3, cepsch, cepsch2, fftsize = 1024;

//STFT e Cepstrum do sinal modulador e da portadora
in = PlayBuf.ar(1, modBuf, BufRateScale.kr(modBuf), 1, 0, 0, 2);
chain = FFT(LocalBuf(fftsize), in);
cepsch = Cepstrum(LocalBuf((fftsize/2).asInt), chain);

in2 = PlayBuf.ar(1, carBuf, BufRateScale.kr(carBuf), 1, 0, 0, 2);
chain2 = FFT(LocalBuf(fftsize), in2);
cepsch2 = Cepstrum(LocalBuf((fftsize/2).asInt), chain2);

//filtragem espectral para suavizar o envelope espectral
cepsch = PV_BrickWall(cepsch, envSmooth);
ICepstrum(cepsch, ~envm);
//envelope espectral suave da portadora
cepsch2 = PV_BrickWall(cepsch2, envSmooth);
ICepstrum(cepsch2, ~envc);

//divisão do espectro da portadora pelo seu envelope espectral suavizado
//gerando um espectro aplainado
chain2 = PV_MagDiv(chain2,~envc);

//multiplicação do espectro "aplainado" da portadora pelo envelope
   espectral suavizado da moduladora
chain2 = PV_MagMul(chain2, ~envm);

Out.ar( out, Pan2.ar(IFFT(chain2) *0.8) );

}).add;)

Synth.new(\cepstralCrossSynth, [\modBuf, ~trovao, \carBuf,  ~cantor,
   \envSmooth, -0.85]);
```

Código 3.40 – Síntese cruzada por filtragem de cepstrum, código adaptado de Hsu (2012)

Dentre todos os algoritmos avaliados nesta seção, a filtragem de cepstrum foi aquele na qual a manipulação do parâmetro relativo à proporção da extração altura/timbre de cada sinal resultou em modificações mais controláveis e variações sonoras mais graduais.

Áudio 3.46 (3-46_Cepstrum_orquestra_fagote_trovao_cantor.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Excerto orquestral, seguido de nota fá-1 (F1) tocada em sustentação por um fagote, seguidos da filtragem de cepstrum deste dois sons com janela de 2048 pontos, orquestra como som fonte e fagote como som filtro; Som de um trovão, seguido por um cantor lírico sustentando uma nota lá-4 (A4) com a vogal "a", seguidos da filtragem de cepstrum deste dois sons com janela de 1024 pontos, trovão como som fonte e cantor como som filtro. Amostra de fagote retirada de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>. Som de trovão e cantor retirados de <https://freesound.org/s/243614/> e <https://freesound.org/s/395421/>, respectivamente. Excerto orquestral retirado de Johannes Brahms - Op.45 *Ein Deutsches Requiem - Denn wir haben hie keine bleibende Statt*, The Holden Consort Orchestra and Choir, CC BY-SA 2.0 <https://commons.wikimedia.org/wiki/File:Johannes_Brahms_-_Op.45_Ein_Deutsches_Requiem_-_(06)_Denn_wir_haben_hie_keine_bleibende_Statt.oga>

O Código 3.41 mostra uma utilização elementar da LPC. É possível também tratar o sinal antes de alimentar o algoritmo principal, à maneira dos algoritmos anteriores, utilizando métodos como filtragem de agudos (de forma a aproximar um sinal da sua altura fundamental, descartando informações sobre o timbre) ou comprimindo os sinais para obter entradas mais normalizadas.

```
1  ~orq = Buffer.read(s,"~\Audios\orquestra.wav");
2  ~fgtF1 = Buffer.read(s,"~\Audios\fagote.wav");
3  ~cantor = Buffer.read(s,"~\Audios\cantor.wav");
4  ~trovao = Buffer.read(s,"~\Audios\trovao.wav");
5
6  ({LPCAnalyzer.ar(
7          PlayBuf.ar(1, ~orq, BufRateScale.kr(~orq), loop: 0, doneAction: 2)
               *1,
8          PlayBuf.ar(1, ~fgtF1, BufRateScale.kr(~fgtF1), loop: 1) *1.0,
9          2048, 99, 1, 0.9999999,windowtype: 1, mul: 0.1).dup}.play)
10
11 ({LPCAnalyzer.ar(
12         PlayBuf.ar(1, ~cantor, BufRateScale.kr(~cantor), loop: 0,
               doneAction: 2) *1,
13         PlayBuf.ar(1, ~trovao, BufRateScale.kr(~trovao), loop: 1) *0.5,
14         2048, 99,1,0.999997, windowtype: 1, mul: 0.1).dup}.play)
```

Código 3.41 – Síntese cruzada por codificação preditiva linear (LPC)

Devido ao seu modelo basear-se na ideia de fonte e filtro do trato vocal, a LPC tende a funcionar de forma mais homogênea quando se utiliza filtros similares aos vocais e excitações próximas às das pregas vocais. Todavia, o que observou-se aqui é que o parâmetro de minimização do erro (*delta*) e o número de pólos do filtro (p) apresentam alta sensibilidade e podem fornecer resultados sonoros bastante complexos. Em comparação

com outros métodos utilizados, no Áudio 3.47 não foi possível ouvir uma mistura tão homogênea entre os sinais. O sinal modulador, de forma geral, tendeu a apresentar mais artifícios e distorções nos seus transitórios rápidos.

Áudio 3.47 (3-47_LPC_orquestra_fagote_trovao_cantor.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Excerto orquestral, seguido de nota fá-1 (F1) tocada em sustentação por um fagote, seguidos da síntese cruzada por codificação preditiva linear (LPC) deste dois sons com janela de 2048 pontos, orquestra como som fonte e fagote como som filtro; Som de um trovão, seguido por um cantor lírico sustentando uma nota lá-4 (A4) com a vogal "a", seguidos da síntese cruzada por codificação preditiva linear (LPC) deste dois sons com janela de 2048 pontos, trovão como som fonte e cantor como som filtro. Amostra de fagote retirada de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>. Som de trovão e cantor retirados de <https://freesound.org/s/243614/> e <https://freesound.org/s/395421/>, respectivamente. Excerto orquestral retirado de Johannes Brahms - Op.45 *Ein Deutsches Requiem - Denn wir haben hie keine bleibende Statt*, The Holden Consort Orchestra and Choir, CC BY-SA 2.0 <https://commons.wikimedia.org/wiki/File:Johannes_Brahms_-_Op.45_Ein_Deutsches_Requiem_-_(06)_Denn_wir_haben_hie_keine_bleibende_Statt.oga>

4 Expansão e recriação de processos de transformação sonora

> *O que nós procuramos é uma forma de transformar materiais sonoros que entregue sons resultantes que são claramente parentes próximos da fonte, mas também claramente diferentes.*
>
> —Trevor Wishart, *Audible Design*

Neste capítulo mostro como a pesquisa realizada ao longo deste trabalho se integra em minhas práticas criativas. Quatro pequenas peças, no formato de estudos sonoros, são propostas aqui de maneira a avaliar as especificidades dos processos de transformação sonora e vasta gama de possibilidades decorrente das individualidades de cada objeto sonoro proposto como material.

Na primeira, avaliamos a utilização combinada de diferentes transformações de conjuntos de onda como processo único. A partir do gesto sonoro de diversas pronúncias da palavra "rato", busco realizar procedimentos de *metamorfose sonora* deste vocábulo em sons de ondas do mar. No segundo estudo, reduzimos a escolha do material a uma única nota de altura fixa tocada por instrumentos orquestrais, explorando de maneira mais intensa o leque de outros parâmetros acústicos e musicais disponíveis. No terceiro, utilizamos as próprias informações acústicas contidas num material polifônico (via descritores) para conduzir e controlar as transformações de uma peça generativa. Por fim, o último estudo explora as modificações da escala de tempo-frequência numa elaboração e montagem dos materiais mais artesanal e ligada ao tempo diferido.

Em virtude deste capítulo concentrar-se na integração entre procedimentos computacionais, utilização dos materiais e ideias/poéticas de organização sonora, optamos aqui por não apresentar textualmente os códigos na sua totalidade, o que iria requerer um espaço excessivamente grande, prejudicando o equilíbrio textual. Os códigos completos podem ser encontrados nos seus respectivos anexos ao final deste texto.

* * *

4.1 Transformações algorítmicas contínuas e combinadas

Todas as transformações de conjuntos de onda apresentadas na subseção 3.1.3 foram realizadas dentro de uma só macro estrutura, *Pbind* e *Pwavesets*, por meio de combinações das suas subestruturas internas, os pares de chave-valor. Assim, é possível realizar diversos processos intermediários por meio de alterações nestes pares, *e.g.* combinar

num só processo inversão e omissão de conjuntos de onda. O SuperCollider suporta várias estruturas para composição algorítmica, sendo que a utilização mais comum dos *Patterns* é para tal finalidade. Objetivando explorar essa funcionalidade, propomos aqui a realização de um pequeno procedimento generativo que produza sucessivas transformações sonoras combinadas.

Por meio das diferentes pronuncias da palavra "rato" que foram apresentadas no Capítulo 3, utilizamos o gesto espectro-morfológico da mesma para simular o gesto de ondas do mar. Buscamos gerar diversas *metamorfoses sonoras* que oscilem entre uma maior percepção de sons vocalicos e um maior percepção de sons marítmos visando, assim, manter a sensação dúbia entre qual objeto sonoro de fato o ouvinte está apreendendo.

É possível, com relativa facilidade, utilizar sons marítimos como base para a imposição de gestos espectromorfológicos vocálicos, gerando assim um espécie de som de mar falante (WISHART, 1994, p. 47). Todavia, o procedimento inverso, utilizar sons vocálicos para gerar sons semelhantes aos de ondas do mar não pode ser feito tão diretamente. Aqui a escolha do palavra "rato" é crucial por apresentar ataque e decaimento suaves além de sua terminação "to" apresentar um som semelhante ao de ruído filtrado que ocorre após a quebra das ondas do mar com formação de espuma. Os sons iterativos das pronúncias com vibrante múltipla alveolar e vibrante múltipla uvular também são cruciais por dois motivos: primeiramente, seu esticamento ressalta características típicas de sons envolvendo líquidos (iterações lentas do aparelho vocal) e, em segundo lugar, estas interações são ligeiramente diferentes umas das outras, o que facilita a simulação de sons de ondas no mar, igualmente redundantes.

Para a realização deste estudo sonoro, uma primeira etapa analisa os conjuntos de onda e carrega os *SynthDefs*. Em seguida, define-se diversos blocos de combinação de procedimentos de transformações de conjuntos de onda, os *Pdef* apresentados no Código 4.1. Para a execução propriamente dita de tais processos, utiliza-se padrões de sequenciamento e paralelização – final do Código 4.1 – no qual utiliza-se codificação dura, porém todos os processos internos são generativos e probabilísticos.

```
1  (Pdef(\saTotalAlea,
2          Pwavesets(
3                  Pbind(
4                  \name, \simpAlveo,
5                  \start, Pn(Pseries(0, 1, 1552),1),
6                  \num, Pxrand([2,2,2,3,5,1,4,7,1,1,1,1,1],inf),
7                  \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
8                  \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
9                  \rate2, 1.0,
10                 \amp, 0.4,
11                 \pan, Pbrown(-0.5, 0.5, 0.001),
12                 \useFrac, false)        )       );
```

```
Pdef(\saSaltRateBrown,
        Pwavesets(
                Pbind(
                \name, \simpAlveo,
                \start, Pn(Pseries(0, 4, 388),1),
                \num,1,
                \repeats,4,
                \rate, Pbrown(0.5, 1.5, 0.001),
                \amp, 0.4,
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false)        )       );
)
// Organização semi-generativa da forma musical
(Pseq([
        Pseq([Pdef(\vuTotalAlea), Pdef(\vaTotalAlea), Pdef(\saTotalAlea)],
            2), // apresentação
        Ptpar([0.0, Pdef(\vuTotalAlea), 2.9, Pdef(\vaTotalAlea), 2.1, Pdef(
            \saTotalAlea)], 2),
        Ptpar([0.0, Pdef(\vaNum7a12Rate8), 4.9, Pdef(\saNum7a12Rate8), 8.1,
             Pdef(\vuNum7a12Rate8)], 2),
        Ptpar([0.0, Pdef(\vaRecipRep1), 0.9, Pdef(\saRecipRep1), 1.1, Pdef(
            \vuRecipRep1)], 4),
        Ppar([Ptpar([0.0, Pdef(\vaRecipRep1), 0.9, Pdef(\saRecipRep1), 1.1,
             Pdef(\vuRecipRep1)], 4),    Ptpar([0.0, Pdef(\vaRecipRep1),
            1.9, Pdef(\saRecipRep1), 2.1, Pdef(\vuRecipRep1)], 5),
]),
        Ppar([Pdef(\vuTotalAlea), Pdef(\vaTotalAlea), Pdef(\saTotalAlea),
            Pdef(\vaNum7a12Rate8),  Pdef(\saNum7a12Rate8), Pdef(
            \vuNum7a12Rate8)]),
        Ptpar([1.1, Pdef(\vuTotalAlea2), 1.9, Pdef(\vaTotalAlea2), 2.3,
            Pdef(\saTotalAlea2), 0.5, Pdef(\vaNum7a12Rate8), 4.9, Pdef(
            \saNum7a12Rate8), 8.1, Pdef(\vuNum7a12Rate8)]),
]).play;)
```

Código 4.1 – Combinação de transformações e organização da macro estrutura semi-generativa

De acordo com código do Apêndice B, na primeira transformação valores aleatórios, bastante redundantes e pouco agressivos são utilizados nos diversos parâmetros de modificação dos conjuntos de onda: quantidade de grupos de conjuntos, números de repetições, taxa de leitura, etc. Blocos paralelos ou semi-defasados são apresentados aqui. Num segundo momento, utilizam-se transformações em que a taxa de leitura dentro de um conjunto de onda é variável, uma espécie de *glissando* interno ao conjunto, sendo estes *glissandos* primeiramente ascendentes e depois, majoritariamente, descendentes. Por fim, todos os processos são executados juntos, com pequenos defasamentos entres eles de forma a simular o gesto de uma grande onda.

Estas transformações estão contidas numa primeira seção que vai até o 2'05", a partir daí utilizei as mesmas transformações, organizadas em ordem aleatórias, porém executando duas instâncias do mesmo código, uma em cada canal, de forma a aumentar a densidade da textura marítima pretendida. Nesta seção final acrescentamos também uma transformação na qual realizamos processo mais extremos de esticamento e com taxas de leituras mais lentas (por volta da metade da velocidade original), o que gera os sons graves e contínuos ao fundo da textura.

No Áudio 4.1 temos o resultado final dos processos gerados. Aplicou-se um algoritmo de redução de ruído com parâmetros bastantes suaves, um filtro passa-baixas e reverberação disponíveis no software Audacity. A ideia aqui é deixar os processos mais difusos e aproximar-se ainda mais da sonoridade espacial de um mar aberto.

A ideia em se utilizar apenas um material ou poucos materiais de sonoridade bastante similar proporciona, consequentemente, transformações bastante similares entre si. Esta alta redundância facilita a realização de transições mais suaves entre materiais fortemente distintos e também faz com que um maior leque de texturas homogêneas possam ser criadas. Além disso, como temos um material suficientemente coeso, a percepção das transformações sonoras é mais facilmente guiada por sempre estarmos rememorando tal objeto.

Áudio 4.1 (4-1_estudo_1_mar_ratos_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Estudo sonoro 1 – *Transformações algorítmicas contínuas e combinadas.* Diferentes pronuncias da palavra "rato" são submetidas a processos combinados de transformação de conjuntos de onda de forma a simular o gesto espectromorfológico do som de ondas do mar.

4.2 Variações em uma nota fixa de instrumentos orquestrais

Neste estudo, utilizei a nota dó 3 (C3) executada em diferentes instrumentos orquestrais – marimba, violino, trompa, fagote, trompete e piano – em transformações sonoras que ressaltem os diversos aspectos sonoros além da altura musical. Busquei explorar aqui a tênue relação entre altura e timbre, mais especificamente em como mudanças no espectro podem sugerir novas alturas; a desafinação como possibilidade de criação textural ou de variação timbrística, disrupções no ataque e continuação das notas como estratégia para sugerir saltos de altura; mudanças de escala temporal como procedimento de geração de alturas – ritmos acelerados se convertendo em alturas definidas; por fim, exploração das fronteiras entre inarmonicidade, timbre e afinação.

Os instrumentos de orquestra, em especial, são fruto de uma longa trajetória de pesquisa que visa facilitar a execução de notas de alturas fixas e temperadas dentro de

variações timbrísticas relativamente delimitadas (GIESELER et al., 1985; PADOVANI; FERRAZ, 2011)[1]. Isso tende a facilitar a diferenciabilidade entre instrumentos que executam melodias paralelas, acrescentando novas formas de percepção do timbre. A ideia de se fixar uma altura e trabalhar com os demais parâmetros sonoros permite que, por meio da escuta, possamos refletir e questionar qual é de fato a importância das relações de altura para a percepção humana e como esse parâmetro se integra ou dissocia daquilo que chamamos de timbre.

Esse estudo foi motivado por quatro peças que abordam essa temática, cada uma à sua maneira: *De Natura Sonorum* (1975) de Bernard Parmegiani, *Partiels* (1975) de Gérard Grisey, *Quattro Pezzi su una nota sola* (1959) de Giacinto Scelsi e o primeiro movimento de *Musica Ricercata* (1953) de György Ligeti. Em cada uma destas peças a redução drástica do material primordial da música ocidental – o conjunto de alturas definidas dentro de uma oitava – demanda diferentes estratégias para evitar a repetição demasiada. Nos casos orquestrais e no eletroacústico, este último em especial, há uma grande variedade de timbres, técnicas estendidas, sons de altura não definida e recursos de processamento que abrem o leque das possibilidades no trabalho com uma nota só. No caso de *Musica Ricercata* a redução é ainda mais drástica pois a peça é escrita para a execução ao piano, instrumento que representa, por excelência, a supervalorização ocidental do atributo altura e sua divisão em doze partes. Além disso, instrumento no qual o timbre foi historicamente condicionado a tornar-se parâmetro secundário, funcionando apenas como "colorido" devidamente controlado e homogêneo (ao longo das oitavas) (WISHART, 1996; WISHART, 1994; SCHAEFFER, 2017).

O que resta além da altura? No caso de *Musica Ricercata* com a extrema redução de possibilidades, as possíveis explorações parecem ser no campo rítmico, gestual, fraseológico, dos registros e oitavas. Em *Partiels* e *Quattro Pezzi su una nota sola*, a enorme gama de possibilidades oferecida pelos instrumentos da orquestra permite longas explorações e, ainda assim, cada uma destas peças irá se concentrar em certos aspectos exploratórios: na primeira, a ideia de utilização dos intrumentos orquestrais como geradores de parciais de um som harmônico e, na segunda, uma maior concentração na orquestração em si. No caso de *De Natura Sonorum*[2], que foi realizada exclusivamente com manipulação de fita magnética e eletrônica analógica, abre-se um campo maior de possibilidades, pois é possível fazer uma investigação da inarmonicidade, batimentos, ressonâncias, continuidade de *streams* auditivos, possibilidades texturais, percepção da série harmônica, influência dos ataques para a identificação dos instrumentos, dentre outros pontos. Quando utilizamos recursos computacionais é possível ainda expandir um pouco mais este leque.

Ao buscar compreender como nossa escuta compara diferentes timbres que possuem

[1] Gieseler et al. (1985) em tradução para o português de José Henrique Padovani, ainda não publicada.
[2] Em especial nos movimentos como *Accidents/Harmoniques*, *Conjugaison du timbre* e *Ondes croisées*

uma mesma altura, Schaeffer et al. (1967) busca na linguística regras que possam apresentar paralelos com o domínio musical.

> JAKOBSON define uma das leis fundamentais da linguagem como uma "relação de alternância", quer dizer, "a possibilidade de substituir um termo por outro, equivalente sob um aspecto mas diferente sob outro." (SCHAEFFER et al., 1967, p. 17).

Esta ideia nos auxilia a repensar as referidas peças pelo prisma da forma musical, buscando compreender se há algum discurso no encadeamento de objetos sonoros, ou, ao menos, se alguma linha da percepção auditiva se estabelece ao longo dessa sucessão de eventos no tempo. Se fixarmos uma só altura, podemos fazer algo além da alternância entre timbres/gestos/durações? Esta alternância entre parâmetros diferentes da altura, fica retida como um *stream* auditivo? Associando essa perspectiva com as ideias de Wishart, avaliamos neste estudo sonoro se, para realizar exploraraçõnes nas fronteiras propostas pelas referidas peças, seria a alternância gradual de parâmetros o elemento essencial para a metamorfose sonora?

Além disso, no estudo proposto, avaliamos maneiras de dissolver o timbre do ataque em meio às ressonâncias de cada nota, maneiras de mudar o domínio perceptivo auditivo do objeto sonoro (ritmo, batimento, altura definida e timbre), formas de incorporar e integrar *allures* dos objetos sonoros às texturas e forma, e ainda possibilidades de produzir objetos intermediários entre os dois objetos sonoros distintos. Abaixo temos uma divisão básica das seções deste estudo:

1. Substituição de ataque e notas simultâneas em diferentes acelerações
    a) Substituição de ataque marimba-instrumento
        i. C3 marimba-violino, marimba-trompa, marimba-fagote, marimba-trompete e marimba-piano
    b) notas simultâneas de ataque sincronizado
        i. C3 violino, trompa, fagote, trompete e piano
2. Ressíntese aditiva de conjunto de onda e estratégia de repodução
    a) Ressíntese aditiva de conjunto de onda
        i. C3 Piano + "harmônicos" de conjunto de onda 2 e 4
        ii. C3 Piano + "harmônicos" de conjunto de onda 5 e 2
        iii. C3 Violino + "harmônicos" de conjunto de onda 1 e 3
        iv. C3 Violino + "harmônicos" de conjunto de onda 1 e 3 + esticamento de conjunto de onda
    b) estratégia de repodução - C3 violino, trompa, fagote, trompete e piano

- oitavas diferentes
- oitavas diferentes
- mesma oitava desafinada

3. Estratégias de reprodução e focalização espectral associadas (todos C3 violino, trompa, fagote, trompete e piano)
   a) *chorus* em oitavas
   b) somente *chorus*
   c) *chorus* em oitavas
4. Deslocamento espectral e ataques dessincronizados
   a) C1 de fagote
      i. original
      ii. deslocamento espectral
   b) C3 marimba, violino, trompa, fagote, trompete e piano
      i. ataques dessincronizados sob diferentes taxas
5. Intercâmbio de conjuntos de onda
   a) 64 conjuntos - C3 trompa e fagote
   b) 64 conjuntos - C3 fagote e C1 fagote
   c) 32 conjuntos - C3 fagote e C1 fagote
   d) 64 conjuntos - C3 piano e marimba
   e) 3 conjuntos - C3 violino e trompete
6. *Delay* conjuntos de onda intercambiados
   a) 64 conjuntos - C3 piano e marimba
   b) 7 conjuntos - C3 piano e C1 fagote
   c) 3, 6, 32 e 64 conjuntos - C3 violino e trompete
7. Estratégias de reprodução
   a) C3 em todos os instrumentos sob diferentes acelerações e desacelerações rítmicas

No primeiro trecho os sons de *substituição de ataque* das notas em instrumentos de orquestra obtidos na subseção 3.1.1 são intercalados com duas notas com ataque sincronizado, todavia com uma variação aleatória entre os instrumentos executados. O ritmo de intercalação entre estes dois objetos ora está sincronizado e ora dessincronizado, pois cada apresenta uma aceleração geométrica de velocidade distinta. Ao final deste trecho, com a alta frequência de execução das notas, forma-se um contínuo no qual o ataque da marimba e arcada do violino se destacam.

No caso de sons instrumentais, o ataque e seu conteúdo espectral tendem a ter uma maior importância para a identificação do timbre (ROADS, 1996; SCHAEFFER et al., 1967, p. 544). Nestas primeira seção, como ocorre uma mistura entre ataques e continuações de forma rápida, tendemos a ter mais dificuldade em identificar os instrumentos, o que pode liberar a percepção para se concentrar em outros fatores, como as mudanças nas formantes espectrais e nas relações entre alturas oriundas das mudanças bruscas no espectro.

O contínuo da primeira seção é interrompido por uma nota de piano transformada por ressíntese aditiva de conjuntos de onda, inciando-se aqui uma nova seção formada por dois elementos alternados: a referida transformação de conjuntos de onda para som de piano ou violino e um par de sons de instrumentos orquestrais executados com velocidade original, dobrada ou pela metade, acionados em tempos irregulares. Esta seção termina quando os dois blocos anteriores são executados em sincronia, porém a segunda transformação é feita de modo a causar pequenas desafinações – desvios na relação inteira entre velocidades de reprodução.

A partir deste momento três transformações de focalização espectral – mantêm-se somente os canais mais estáveis ao longo do tempo – são executadas: na primeira temos velocidades de reprodução original, dobrada ou pela metade com pequenos desvios (*chorus em oitavas*) e uma focalização espectral com parâmetros suaves (pouca alteração); na segunda o mesmo tipo de focalização espectral porém com o referido *chorus* em uma única oitava; por fim voltamos ao (*chorus em oitavas* mas agora com uma focalização espectral agressiva.

Aqui podemos perceber como os ataques são desmanchados na focalização espectral agressiva e também como as alterações mais agressivas na velocidade de reprodução promovem variações timbrísticas agressivas. Parece haver a introdução de instrumento de cordas metálicas pinçadas, resultante da modificação timbrística na velocidade de reprodução.

A quarta seção inicia com uma nota dó-1 (C1) tocada por um fagote com deslocamento espectral. Aqui teremos esta nota de fagote sendo tocada na sua versão original alternada com transformações de deslocamento espectral sob diferentes fatores. Entre cada uma destas notas fagote executam-se todas as notas dó-3 (C3) de marimba, violino, trompa, trompete e piano semi-sincronizadas, ou seja, com o ataque levemente fora de sincronia. A quantidade de perda de sincroniza é variada a cada execução.

Na quinta seção explorei as possibilidades do intercâmbio de conjuntos de onda. De forma geral, esse procedimento tende a gerar uma sonoridade parecida a modulação de amplitude síncrona entre sons diferentes, o que produz como resultante um som iterativo misto entre as duas fontes. Em alguns casos ocorre produção de um objeto que tem suas características próprias, geralmente imprevisíveis, e bastantes diferentes do efeito mencionado anteriormente. Este foi o caso do intercâmbio de conjuntos de onda entre as

notas do violino e trompete, que gerou uma espécie de som de balão de borracha sendo raspado. Ao final desta seção, utilizei diversas cópias do mesmo executadas em tempos e ritmos diversos, gerando assim uma forma de *delay* aleatório.

Na última seção, busquei uma forma de utilizar as sonoridades oriundas de modulações algorítmicas, como aceleração e desaceleração de execuções de notas de forma a sempre cruzar as fronteiras perceptivas entre ritmo, batimento (ou também chamado de rugosidade), altura definida e timbre. Além disso, diferença de sincronia entre os processos algorítmicos faz com mudanças de formantes no todo ouvido possam ser percebidas. Esta diferença auxilia também na segregação perceptiva de cada linha de aceleração/desaceleração.

O arquivo de som completo está presente no Áudio 4.2. Neste tipo de estudo a metodologia adota foi uma utilização teste, avaliação e refatoração dos códigos. Assim, começamos com códigos exclusivamente generativos e probabilísticas e vamos aos poucos realizando ajustes finos em cada um destes. Devido a tais ajuste o código final se torna muito grande, neste caso cerca de 300 linhas de códigos que exigiram demasiado espaço neste documento. O código completo está apresentado no Apêndice B.

Áudio 4.2 (4-2_estudo_2_nota_orquestral_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Estudo sonoro 2 – *Variações em uma nota fixa de instrumentos orquestrais*. Neste estudo a nota dó 3 (C3) executada em diferentes instrumentos orquestrais – marimba, violino, trompa, fagote, trompete e piano – passando especialmente pelas transformações de substituição de ataque, modificação da velocidade de reprodução, ressíntese aditiva de conjuntos de onda, deslocamento e focalização espectral. Sons fonte retirados da biblioteca de *samples* VSCO 2 Community Edition (CE) disponível em<https://vis.versilstudios.com/vsco-community.html>.

4.3 Processos adaptativos e ressonificação

Em virtude das transformações espectrais e temporais propostas por Wishart serem altamente não lineares, seu efeito depende fortemente da tipomorfologia do som utilizado, logo, uma melhoria de controle do resultado sonoro destas transformações está associada ao balanço entre conhecimento prático envolvendo tipos de objetos e as consequências de cada transformação para estes tipos. Esse fator também dificulta a transformação de sons polifônicos, texturais ou espacializados, pois tais características tendem a ser fundamentalmente perdidas nestas transformações mais radicais, daí a marca tecnográfica de Wishart: transformar sons que sejam objetos sonoros mais claramente delimitados, à partir de gravações bem tratadas.

Assim, Wishart adota uma metodologia de trabalho que costuma partir da análise,

avaliação e experimentação com um dado som gravado para então repensar e elaborar os demais propósitos de uma peça. Deste ponto em diante, o compositor segue um processo quase artesanal de avaliar como cada trecho de um som, cada inflexão de sua tipomorfologia pode influenciar suas transformações. Somado a isso, existe uma preocupação na inter-relação entre os materiais, logo, Wishart também faz uma avaliação de como os materiais e suas derivações transformadas poderão ser articuladas entre si, geralmente de forma a produzir algum tipo de metamorfose sonora. Este tratamento fino auxilia a elaboração de transformações sonoras suaves e graduais à percepção humana, porém demanda uma grande quantidade de tempo.

Dois pontos importantes escapam à metodologia anterior: a não utilização de processos algorítmicos cuja sonoridade pode ser incorporada às características tipomorfológicas dos objetos sonoros; a não avaliação de características sonoras cuja complexidade exige recursos computacionais para sua extração e interpretação[3]. Propomos aqui dois tipos de procedimentos que possam avançar neste sentido, expandindo e complementando a metodologia utilizada por Wishart. Uma primeira ideia é utilizar processos adaptativos – procedimentos cuja resposta se adequa ao tipo de sinal de entrada – visando aproveitar, assim, a própria tipomorfologia do objeto como orientadora e fornecedora de transformações.

> Quando extrai-se descritores sonoros, executamos algum tipo de aquisição indireta dos gestos físicos utilizados para produzir o som. Portanto, o gesto musical transmitido pelo som utilizado para a extração de descritores é implicitamente utilizado para modelar, com significado, a transformação sonora, pois 'a inteligência (já) está no sinal sonoro'. (ZOLZER, 2011, p. 337, tradução nossa).

Uma vez feita a extração destes dados, iremos utilizá-los, mais tradicionalmente, como dados de controle para os processos de modificação no sinal. Contudo, utilizaremos estes dados também como estratégia de ressonificação, transformando dados de controle em novos sinais sonoros.

A segunda ideia parte do detalhamento da tipomorfologia das *allures*, que divide-as de acordo com seu agente de produção e sua forma de produção: regularidade estrita revelando um agente mecânico, flutuações periódicas revelando um agente vivo e irregularidade imprevisível revelando um fenômeno natural como agente (SCHAEFFER, 2017; CHION, 2009)[p. 444-445; p. 179]. Busquei investigar em que medida a sincronização de *allures* presentes ou esperados no objeto sonoro com novas modulações temporais e espectrais traziam a sensação de integração ou dissociação ao novo objeto sonoro, isto é, se ao realizamos modulações síncronas – porém flutuantes, naturalísticas, caóticas e/ou desordenas, em oposição às modulações estritamente regulares – seria possível obter novas

[3] Vale destacar que a escuta humana também tem papel essencial na definição de quais destes recursos computacionais serão de fato relevantes para utilização.

allures que soassem como provenientes de agentes vivos ou fenômenos naturais. Assim, a partir de algumas informações apreendidas auditivamente sobre o sinal, realizei cálculos de parâmetros simples mas que puderam guiar as modulações algorítmicas, buscando integrar a sonoridade dessas modulações com a percepção auditiva das estruturas internas do sinal.

De modo a avaliar o referido problema, escolhemos como som fonte uma gravação de campo de escola de samba que é um som polifônico, multicanal (estéreo), com presença de ruídos e em alguma medida impregnado por características espaciais (variações no distanciamento da captação podem ser ouvidas). Como a gravação apresentava um compasso 2/4 e ciclos de 16 compassos (quadratura), recortamos um trecho desta gravação contendo 12 compassos de forma a permitir desvios na quadratura, mas manter a fórmula de compasso. A partir dessa primeira informação analisada, dividimos o tamanho do *buffer* de áudio em 16, 32 e 128 partes de modo a obter *triggers* dentro de uma grade teoricamente síncrona com este áudio.

Utilizamos unicamente transformações espectrais, sendo elas (na ordem em que aparecem no Apêndice C): congelamento, somente da magnitude e de ambas magnitude e fase; permutação de canais espectrais (*bins*), borramento espectral, entre quadros e dentro de um mesmo quadro; focalização espectral. Optamos pelo formato de uma peça generativa pela semelhança com as longas e contínuas variações desenvolvidas tradicionalmente pelas escolas de samba. Assim, cada tipo de transformação é executada num ciclo de dois compassos, alternando-se para outra transformação ou para a gravação original, sem que haja, no entanto, repetição de transformações consecutivamente.

Uma das ideias de ressonificação utilizadas foi a extração das batidas (detecção de *onsets*) e utilização de sinal de controle para acionar o congelamento de magnitudes. Neste caso, como utilizamos uma janela grande, o congelamento trazia novos ostinatos, que são quase sincronizados com a pulsação do áudio original, gerado uma forte sensação de polirritmia. Acrescida à esta polirritmia, cada canal do estéreo apresenta um limiar de detecção de *onsets*, portanto, um lado estará, por exemplo, mais síncrono com o surdo e outro com o tamborim.

Um outro exemplo de ressonificação está na transformação em que se podem ouvir sons tônicos. Aqui utilizamos janelas muito curtas em conjunto com congelamento de magnitude e fase, circulando nestes dados de forma tão rápida que geramos alturas mais definidas. Outro fator que auxilia este procedimento é o *trigger* utilizado, neste caso, para um canal o *trigger* fixo mais rápido que dispunhamos (divisão por 128) e, no outro canal, a detecção de *onsets*.

Seguindo variantes e combinações destas lógicas estão as transformações de focalização, permutação e borramento. No Áudio 4.3, estas transformações tendem a ser ouvidas como filtragem, e a permutação, mais especificamente, como uma espécie de síntese granular senoidal aleatória.

Áudio 4.3 (4-3_estudo_3_ressonificacao_adaptativa_.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Estudo sonoro 3 – *Processos adaptativos e ressonificação.* Neste estudo generativo, um *loop* de bateria de escola de samba é transformado em diversos processos espectrais adaptativos e de ressonificação. Som fonte de bateria de escola de samba no estilo da escola Mocidade Independente de Padre Miguel, disponível em<https://freesound.org/s/123634/>.

4.4 Potenciais das modificações de altura e escala de tempo-frequência

Figura 23 – Cândido Portinari. Cabeça de Galo (O olho), 1941. Óleo sobre tela. 55 x 46 cm. Coleção particular. Renato Whitaker.

Fonte: <https://criticadeartebh.com/2016/11/10/cabeca-de-galo-o-olho-1941/>

As modificações de altura, tempo e frequência ocupam um ponto central na obra de Wishart e cada uma de suas peças busca uma heurística para os problemas relativos a estas modificações. Na seção 2.4, mencionamos alguns dos problemas técnicos da realização desta modificações, aqui pretendemos discorrer um pouco mais sobre as questões estéticas e poéticas desta seara, que utilizamos como motivação para um novo estudo sonoro.

Como ferramenta de pesquisa, no início da elaboração desde trabalho nos propusemos a retrilhar os caminhos composicionais de Wishart de modo a compreender as motivações, as ferramentas, as escolhas, os atalhos e obstáculos intransponíveis do seu fazer musical. Assim, na peça *G4LO M1L GR4U* nos dispomos a realizar todas as

transformações sonoras por meio do CDP, realizar a montagem dos materiais em uma estação de trabalho de áudio digital (DAW) gráfica, trabalhando exclusivamente com processos em tempo diferido. O principal fator técnico que motivou a elaboração desta peça foi a grande quantidade e variedade de materiais oriundos da modificação das escala temporal, tanto esticamento quanto encurtamento, um recurso aparentemente inesgotável.

Em *Tongues of Fire* (1994), Wishart utiliza a ideia de aglomerar um conjunto de materiais distintos em um só bloco, criando uma espécie de enunciando vocálico polifônico. Isso facilita a futura divisão dos blocos constitutivos do material para sua consequente transformação, pois o material já está previamente e naturalmente segmentado, e também facilita a criação de pontes entre as seções por meio da escolha da quantidade de blocos constitutivos a serem utilizados. Aqui, buscamos realizamos o mesmo processo: utilizamos um som de galo cacarejando somado a uma variedade de outros sons emitidos por frangos e pintos. Este tema inicial é o primeiro som executado em Áudio 4.4, seguido de duas transposições de altura.

A partir deste momento diversos esticamentos temporais são realizados, transformação central desta peça. Sonoramente, o esticamento temporal apresenta três funções ao longo do Áudio 4.4: 1 - tornar os sons mais lentos de modo a permitir melhor apreensão das estruturas internas destes sons; 2 - gerar materiais lentos para utilização como planos de fundo que retenham alguma semelhança timbrística em relação aos sons de primeiro plano; 3 - gerar materiais que possam ser compreendidos como outros tipos de objetos sonoros.

O fenômeno sonoro ocorre essencialmente em uma dimensão temporal. Assim, a riqueza das modificações na escala de tempo e frequência está no fato de se tentar modificar as dimensões mais intrínsecas deste acontecimento físico: alterar o tempo em si, a forma como o tempo passa, nas modificações da escala temporal; e a regularidade de acontecimento de eventos nesta dimensão, as modificações na escala de frequência. Aqui nos parece que esta ideia é realização de uma metatransformação, uma alteração naquilo que a princípio serviria de base para o fenômeno. Longe desse tipo de abordagem, as demais ideias de modificação sonora – distorção não linear, *chorus*, reverberação, etc –tendem a se concentrar na ideia de série temporal, ou seja, em dados variando no tempo, portanto, nestas, é mais importante transformar os dados e não tanto sua variação no tempo ou as próprias variações do tempo.

O esticamento temporal dos sons destas aves, perceptivamente parece induzir a sensação de sons de outros tipos de animais: uivos de cachorros ou gritos de pessoas (5'50"); latidos de cachorros (4'30"-5'00" e 3'55") e sons de focas (2'30"). Além destes, há diversos sons que não apresentam um correspondente físico muito direto, como é o caso dos sons agudos em 2'10" que se assemelham ao som de mangueiras girantes utilizadas como instrumentos de percussão. Os artifícios gerados pelo vocoder de fase parecem gerar ainda

Figura 24 – Dirk Roy. Shape Study, 2019. Animação.

Fonte: <https://vimeo.com/370684822>

sons de fundos que se parecem ruídos de fala, como é possível perceber nos momentos de intensidade mais baixa da peça 4'12"- 4'20" ou 3'10 - 3'15".

Wishart ressalta que as *metamorfoses sonoras* apresentam uma qualidade onírica, especialmente pelo fato de estarmos trabalhando com música acusmática. O que poderiam representar essas transformações de sons de um animal em outro? Quando fazemos uma transformação na qual cacarejar de galo se converte num uivo de cachorro haveria algum corresponde no mundo físico? Ao menos um paralelo visual? Estariamos fazendo uma espécie de manipulação genética ou uma deformação cirúrgica em tais animais?

Um exemplo imagético que nos instigou a escolha de sons de galo para a realização de metamorfoses sonora está na Figura 23. Seria um galo, um ganso, estaria de cabeça torcida ou normalmente, teria no bico um sorriso ou apenas um furioso olho vermelho de galo de briga? O paralelo visual do esticamento temporal seria uma imagem também esticada nos eixos XY ou, mantendo a lógica de alterações perceptivas, teríamos de tratar uma imagem de ave como textura para uma forma de outro animal, conforme Figura 24, esticando somente certas áreas desta imagem ?

Áudio 4.4 (4-4_G4LO_M1L_GR4U.wav). *Arquivo de áudio disponível em <https://musica.ufmg.br/lapis/?p=1141>.*

Peça G4LO M1L GR4U que utiliza sons de galos, galinhas, frangos e pintos para realizar *metamorfoses sonoras* em sons de outros animais, valendo especialmente do esticamento e encurtamento

temporal. Som de galo cantando disponível em <https://freesound.org/s/439472/>, demais sons fontes gravados pelo autor.

Considerações finais

No decorrer da pesquisa que resultou neste trabalho pudemos avaliar com profundidade o estilo, os processos, os materiais, as ferramentas e as motivações poéticas de Trevor Wishart. A partir desse caldeirão de ideias geralmente dispersas e estruturadas de maneira relaxada, buscamos traçar um recorte voltado ao uso de ferramentas de transformação sonora no CDP nos *Capítulo 1 - Ideias e ferramentas de Trevor Wishart* e *Apêndice A - Trevor Wishart: biografia, peças sonoras e textos*. Uma reestruturação e reorganização das ferramentas desenvolvidas por Wishart é realizada no *Capítulo 2 - Processos de transformação sonora*, e sua realização prática, uma adaptação destas técnicas no ambiente SuperCollider, é feita no *Capítulo 3 - Transformações sonoras no SuperCollider* já sob a nossa perspectiva de utilização destas ferramentas para um fazer musical próprio. Por fim, no *Capítulo 4 - Expansão e recriação de processos de transformação sonora*, demos forma mais desenvolvida a algumas das ideias e questionamentos que foram se cristalizando ao longo da elaboração dos três primeiros capítulos.

Ao longo do texto, tanto como observação geral sobre trabalhos voltados às artes sonoras, quanto como paradigma geral da pesquisa aqui apresentada, perpassa a ideia de que há, entre ferramenta, objetivos estéticos e metodologia de trabalho, uma forte relação de simbiose. Seja na avaliação, implementação e adaptação de procedimentos no *Capítulo 3 - Transformações sonoras no SuperCollider* ou nas escolhas pessoais e desenvolvimento dos estudos no *Capítulo 4 - Expansão e recriação de processos de transformação sonora*, é possível perceber, mesmo nos detalhes mais particulares de uma criação musical ou das ferramentas tecnológicas utilizadas nos processos composicionais, uma ligação intrínseca entre: processos técnicos/artesanais relacionados à criação musical; elaborações estéticas/poéticas; e comportamento humano. A todo instante nos deparamos em problemas da fronteira tempo real e tempo diferido, muitas vezes inviabilizando certas proposições estéticas e frequentemente facilitando ou dificultado a realização de uma processo técnico. Seja por uma interface gráfica cujo design modesto e básico desmotiva psicologicamente o usuário na sua utilização inicial, ou por um mecanismo computacional cuja lógica se afasta demais do raciocínio usual dos sistemas musicais, nos parece pouco frutífero abordar estas questões somente por um aspecto estrito.

Dentre os desafios enfrentados ao longo do trabalho, podemos destacar aquele de traduzir processos — e, mais amplamente, toda uma lógica de operação — que tem como pressuposto a realização de determinados procedimentos em tempo diferido, para a operação em tempo real. Longe de se limitar a um problema meramente "técnico", como foi dito anteriormente, essa adaptação, reimplementação ou reinvenção de processos de transformação sonora de maneira a funcionarem em um ambiente fortemente voltado à

operação em tempo real (*SuperCollider*) implicou, evidentemente, em uma reelaboração estética e poética de diversas influências composicionais; em especial, aquelas relacionadas a Trevor Wishart. Algo dessa reelaboração pode ser percebida nas criações e estudos apresentados no *Capítulo 4* que, ainda que não reflitam a totalidade de propostas artísticas desenvolvidas durante a pesquisa de mestrado, permitem entrever esse aspecto.

É importante destacar que a exploração das fronteiras entre tempo real e tempo diferido constitui uma frutífera área de possibilidade ainda a ser pela arte sonora. Este fator tem ganhado cada vez mais destaque, em especial, devido à alta demanda por eventos de tempo real provocada pela pandemia da Covid-19 bem como seus inúmeros problemas técnicos associados. O som apresenta uma natureza intrinsecamente temporal, assim, alterar a base de sustentação mais fundamental desse fenômeno abre não somente um leque de possibilidades artísticas, mas também de importantes indagações físicas e filosóficas.

Os processos abordados no *Capítulo 2 - Processos de transformação sonora* apresentam uma infinidade de detalhes técnicos acerca da sua concepção, modelamento formal e implementação que fogem do propósito deste texto. Este detalhes, no entanto, além de constituírem parte integral das soluções técnicas encontradas neste trabalho, naturalmente ocupando significativa porção do tempo de pesquisa, guardam em potencial diversas outras possibilidades de transformações, aprimoramentos e respostas para muitos dos problemas aqui encontrados.

Ainda no tocante às possibilidades das técnicas aqui avaliadas, a pesquisa que resultou no conteúdo prático do *Capítulo 3 - Transformações sonoras no SuperCollider* nos mostrou que as transformações e manipulações dos conjuntos de onda ainda são um objeto de pesquisa restrito à música com computadores, pouco ou nada explorados pela computação musical e engenharia de áudio. Acreditamos que o material apresentado aqui ainda encontra-se em estado bruto, com latentes aplicações práticas e mais generalizadas. Das inúmeras consequências naturais da investigação dos conjuntos de onda, não puderam ser avaliadas, por exemplo, metodologias de janelamento que considerassem conjuntos de ondas, transformações de conjuntos de onda casadas com transformadas espectrais, procedimentos adaptativos envolvendo conjuntos de onda, aplicação de algoritmos de processamento de imagens e dados gráficos, dentre outros.

A ideia de conjuntos de onda apresenta uma natureza aberta: pode ser utilizada como descritor de áudio, como teorização matemática dos componentes dos fenômenos oscilatórios, como ferramenta analítica para segmentação, dentre outras possibilidades. Essa importante ideia trazida por Wishart e pelo CDP, ainda requer uma avaliação mais consistente a ser feita por diferentes áreas como matemática, física e computação, pois não foi encontrada nenhuma elaboração analítica sobre essa ideia vinda destes campos do conhecimento.

Avaliando o vocoder de fase e os conjuntos de ondas sob uma perspectiva tanto

sonora quanto mecanológica, nos pareceu que o vocoder de fase, por ser uma ferramenta cujas bases buscam ser universais e generalistas (transformada de Fourier), tende a apresentar tanto uma sonoridade mais homogênea (efeitos das transformações são mais parecidos entre si) quanto responder de forma menos personalista a cada tipo de fonte sonora; já nos conjuntos de onda, por se tratarem menos de uma transformação e mais uma forma de seleção de segmentos para aplicação de possíveis transformações, há uma tendência às transformações serem altamente dependentes da fonte, em especial fortemente dependentes da tipomorfologia do objeto fonte, de como sua onda sonora se comporta graficamente de acordo com cada porção do som.

Estas duas principias transformações avaliadas apresentam sua assinatura sonora particular e seus artifícios idiossincráticos. Não há nomes de consenso na literatura para atribuir à tais sonoridades – amplamente abordadas aqui como *som de banco de filtros* e *som de borracha atritando* – e a tipomorfologia schaefferiania nos parece fornecer termos mais próprios aos objetos do que às sonoridades resultantes dos processos. O mesmo se aplica aos procedimentos de transformação algorítmicos, que permitem um controle extremamente fino das modulações e da organização do material. Estes frequentemente impregnam no objetos sonoros fonte uma assinatura sonora que lhes é própria, Nos termos schaefferianos poderiamos falar em modificação ou introdução da *allure*, todavia ainda nos parece necessário um maior detalhamento e expansão desta denominação.

Após as diversas tentativas, erros e acertos envolvendo os procedimentos de modificação espectral, nos sobressalta que a premissa de reversibilidade exata[4] da transformada de Fourier implica também, no seu âmago, a imposição de não modificação nestes dados. Assim, por ter sido concebida inicialmente como modelo de representação dos dados temporais no domínio outro das frequências, a modificação dos dados espectrais irá implicar em diversos artifícios no momento da sua transformação inversa (ressíntese), pois tais estruturas não foram concebidas para tal finalidade. O vocoder de fase apresenta uma das maneiras de redução de alguns destes artifícios, porém este não é uma solução geral para tal problema. No decorrer da pesquisa, algumas metodologias alternativas foram investigadas brevemente, como a síntese de modelamento espectral (SMS) e transformada de Q constante, e foi possível perceber que tal problema ainda aguarda por novas soluções criativas.

O imenso campo de estudos sobre som e espaço não pode ser abordado em mais detalhes aqui pelas limitações de tamanho físico deste trabalho. A intersecção entre espacialização e transformações sonoras (ou mesmo "efeitos sonoros" tradicionais) ainda encontra-se na aurora do seu desenvolvimento e será tema de trabalhos futuros. O mesmo aplica-se às transformações sonoras adaptativas, informadas por descritores, com a utilização de retroalimentação, dentre outras.

[4] Ou quase exata nos casos computacionais.

Este texto também foi pensado como material didático para o campo da música eletroacústica e arte sonora. Conforme já mencionado, a quantidade de trabalhos neste campo que apresenta exemplos sonoros e práticos (seja eles códigos ou técnicas de operação de dispositivos físicos) é muito reduzida. Isso dificulta não só um maior conhecimento da técnicas utilizadas por artistas mas também impede importantes dimensões de aproximação das obras. Buscamos aqui, oferecer uma contribuição que permita e instigue pontes entre a escuta e a realização técnica.

Referências

AMELIDES, Panos. Acousmatic Storytelling. **Organised Sound**, v. 21, n. 3, p. 213–221, dez. 2016. ISSN 1355-7718, 1469-8153. Tex.ids: amelides_acousmatic_2016. Disponível em: <https://www.cambridge.org/core/product/identifier/S1355771816000182/type/journal_article>. Citado 2 vezes nas páginas 40 e 194.

BEAUCHAMP, James (Ed.). **Analysis, Synthesis, and Perception of Musical Sounds: The Sound of Music**. New York: Springer-Verlag, 2007. (Modern Acoustics and Signal Processing). ISBN 978-0-387-32496-8. Disponível em: <https://www.springer.com/gp/book/9780387324968>. Citado na página 77.

BENESTY, Jacob; SONDHI, M. M.; HUANG, Yiteng (Ed.). **Springer Handbook of Speech Processing**. Berlin Heidelberg: Springer-Verlag, 2008. (Springer Handbooks). ISBN 978-3-662-53300-0. Disponível em: <https://www.springer.com/gp/book/9783662533000>. Citado na página 201.

BLACKWELL, Alan F.; COLLINS, Nick. The Programming Language as a Musical Instrument. In: **Proceedings of PPIG05**. Susex: [s.n.], 2005. p. 11. Disponível em: <https://www.ppig.org/papers/2005-ppig-17th-blackwell/>. Citado na página 52.

BLECHMANN, Tim. supernova, a multiprocessor-aware synthesis server for SuperCollider. In: **Proceedings of the International Computer Music Conference**. Huddersfield: [s.n.], 2011. v. 2011, p. 5. Tex.ids: blechmann_supernova_2011. Disponível em: <http://hdl.handle.net/2027/spo.bbp2372.2011.128>. Citado na página 57.

CAETANO, Marcelo. **Morphing isolated quasi-harmonic acoustic musical instrument sounds guided by perceptually motivated features**. Tese (PhD thesis) — Université ParisVI - Pierre et Marie Curie (UPMC), Paris, 2011. Disponível em: <http://recherche.ircam.fr/anasyn/caetano/caetano_morphing_musical_instrument_sounds.pdf>. Citado na página 41.

CAMPO, Alberto de. **Wavesets**. SuperCollider Quarks, 2020. Original-date: 2014-11-30T15:34:50Z. Disponível em: <https://github.com/supercollider-quarks/Wavesets>. Citado 2 vezes nas páginas 98 e 101.

CAMPO, Alberto de; BOVERMANN, Till; ROHRHUBER, Julian. **WavesetsEvent**. musikinformatik, 2020. Original-date: 2017-06-07T12:54:17Z. Disponível em: <https://github.com/musikinformatik/WavesetsEvent>. Citado 2 vezes nas páginas 98 e 99.

CDP, Composers Desktop Project Ltd. **CDP DISTORT Functions**. 2015. Disponível em: <http://www.ensemble-software.net/CDPDocs/html/cdistort.htm>. Citado 2 vezes nas páginas 96 e 97.

CDP, Composers Desktop Project Ltd. **CDP DOCUMENTATION MAIN INDEX**. 2016. Disponível em: <http://www.ensemble-software.net/CDPDocs/html/ccdpndex.htm>. Citado 2 vezes nas páginas 45 e 46.

CDP, Composers Desktop Project Ltd. **CDP Files & Codes: Input and Output Formats**. 2016. Disponível em: <http://www.ensemble-software.net/CDPDocs/html/filestxt.htm>. Citado na página 45.

CDP, Composers Desktop Project Ltd. **About CDP**. 2018. Disponível em: <https://www.composersdesktop.com/about.html>. Citado 2 vezes nas páginas 48 e 198.

CDP, Composers Desktop Project Ltd. **CDP Spectral Processing Function Groups**. 2018. Disponível em: <http://www.ensemble-software.net/CDPDocs/html/cspecndx.htm>. Citado na página 129.

CDP, Composers Desktop Project Ltd. **CDP PSOW Functions**. 2019. Disponível em: <http://www.ensemble-software.net/CDPDocs/html/cgropsow.htm#APPENDIX1>. Citado na página 65.

CDP, Composers Desktop Project Ltd. **CDP Time-Domain Processing Function Groups**. 2019. Disponível em: <http://www.ensemble-software.net/CDPDocs/html/cgroundx.htm>. Citado na página 97.

CEMM, Ro. **Trevor Wishart**. 2009. Disponível em: <https://www.flickr.com/photos/roendoftheroad/4075004227/>. Citado na página 34.

CHION, Michel. **Guide to sound objects: Pierre Schaeffer and musical research**. London: [s.n.], 2009. ISBN 2-7020-1439-9. Citado 2 vezes nas páginas 60 e 168.

DOLSON, Mark. The Phase Vocoder: A Tutorial. **Computer Music Journal**, v. 10, n. 4, p. 14, 1986. ISSN 01489267. Disponível em: <https://www.jstor.org/stable/3680093?origin=crossref>. Citado 3 vezes nas páginas 77, 78 e 195.

DRIEDGER, Jonathan; MüLLER, Meinard. A Review of Time-Scale Modification of Music Signals. **Applied Sciences**, v. 6, n. 2, p. 57, fev. 2016. Tex.copyright: http://creativecommons.org/licenses/by/3.0/ number: 2 publisher: Multidisciplinary Digital Publishing Institute. Disponível em: <https://www.mdpi.com/2076-3417/6/2/57>. Citado 2 vezes nas páginas 78 e 84.

DUDAS, Richard; LIPPE, Cort. Comercial, **Tutorial: The Phase Vocoder - Part II | Cycling '74**. 2006. Disponível em: <https://cycling74.com/tutorials/the-phase-vocoder-part-ii/replies/1>. Citado na página 73.

EDGERTON, Michael. **The 21st Century Voice, 2nd edition**. 2011. Disponível em: <https://michaeledwardedgerton.wordpress.com/the-21st-century-voice/>. Citado na página 200.

ELLIS, Dan. Lecture, **Lecture 9: Time & Pitch Scaling**. Columbia University: [s.n.], 2014. Disponível em: <https://www.ee.columbia.edu/~dpwe/e4896/lectures/E4896-L09.pdf>. Citado 2 vezes nas páginas 131 e 132.

ENDRICH, Archer. Composers' Desktop Project: a musical imperative. **Organised Sound**, v. 2, n. 1, p. 29–33, abr. 1997. ISSN 1469-8153, 1355-7718. Publisher: Cambridge University Press. Disponível em: <https://www.cambridge.org/core/journals/organised-sound/article/composers-desktop-project-a-musical-imperative/6139819E116A7C1F964AD4D4473441A0>. Citado na página 45.

ENDRICH, Archer. Tutorial, **CDP Tutorial Workshop 1**. 2016. Disponível em: <https://www.composersdesktop.com/workshops.html>. Citado na página 31.

ENDRICH, Archer. Tutorial, **CDP Tutorial Workshop 2**. 2016. Disponível em: <https://www.composersdesktop.com/workshops.html>. Citado na página 31.

ENDRICH, Archer. **CDP's Formative Decisions**. 2016. Disponível em: <http://www.hithergatemusic.com/uplymcdp/CDPdecisionsprintable.html>. Citado 2 vezes nas páginas 45 e 47.

ENDRICH, Archer; FRASER, Robert; DOBSON, Richard. **CDP DISTORT Functions (with Command Line Usage)**. 2015. Disponível em: <http://www.ensemble-software.net/CDPDocs/html/cdistort.htm#RWDDIST>. Citado na página 68.

FLANAGAN, J. L.; GOLDEN, R. M. Phase Vocoder. **Bell System Technical Journal**, v. 45, n. 9, p. 1493–1509, 1966. ISSN 1538-7305. Tex.copyright: © 1966 The Bell System Technical Journal. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1966.tb01706.x>. Citado na página 77.

GARRARD, Chris; WILLIAMS, Duncan. Tools for Fashioning Voices: An Interview with Trevor Wishart. **Contemporary Music Review**, v. 32, n. 5, p. 511–525, out. 2013. ISSN 0749-4467, 1477-2256. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/07494467.2013.849878>. Citado 2 vezes nas páginas 189 e 190.

GIESELER, Walter et al. **Instrumentation in der Musik des 20. Jahrhunderts: Akustik - Instrumente - Zusammenwirken**. Celle: Moeck, 1985. ISBN 978-3-87549-018-3. Citado na página 163.

GORDON, John W.; STRAWN, J. An introduction to the phase vocoder. In: STRAWN, J. (Ed.). **Digital audio signal processing: an anthology**. Los Altos, CA: William Kaufmann, 1987. p. 283. ISBN 978-0-86576-082-0. Disponível em: <https://ccrma.stanford.edu/files/papers/stanm55.pdf>. Citado na página 77.

GORNE, Annette Vande. **Traité d'écriture sur support**. Musiques & Recherches, 2017. (Revue LIEN VIII). ISBN 978-2-9600201-1-3. Disponível em: <https://sites.inagrm.com/avdg/index.xhtml>. Citado na página 62.

HOCHHERZ, Olaf. SPList, a Waveset synthesis library and its usage in the composition "draussen". In: . Cologne: [s.n.], 2008. p. 6. Disponível em: <http://lac.linuxaudio.org/2008/download/papers/19.pdf>. Citado na página 98.

HSU, Jenifer. **It's morphin' time**. Stanford, 2012. 12 p. Disponível em: <https://ccrma.stanford.edu/~jhsu/421b/>. Citado 3 vezes nas páginas 18, 154 e 155.

JR, S. Lawrence Marple. **Digital Spectral Analysis: Second Edition**. [S.l.]: Courier Dover Publications, 2019. Tex.googlebooks: uEOjngEACAAJ googlebooksid: uEOjngEACAAJ. ISBN 978-0-486-78052-8. Citado na página 76.

LANDY, Leigh. **Sound Transformations in Electroacoustic Music**. 1991. Disponível em: <https://www.composersdesktop.com/landyeam.html>. Citado 3 vezes nas páginas 40, 41 e 194.

MARTY, Nicolas. "Creavolution" with Trevor Wishart. **Research Report**, v. 10, p. 27, 2011. Citado na página 190.

MAYER, Daniel. **miSCellaneous_lib**. 2020. Original-date: 2017-08-21T20:51:03Z. Disponível em: <https://github.com/dkmayer/miSCellaneous_lib>. Citado na página 98.

MCCARTNEY, James. Rethinking the Computer Music Language: SuperCollider. **Computer Music Journal**, v. 26, n. 4, p. 61–68, dez. 2002. ISSN 0148-9267. Publisher: MIT Press. Disponível em: <https://doi.org/10.1162/014892602320991383>. Citado 2 vezes nas páginas 56 e 59.

MILANI, Matteo. **Trevor Wishart: Chemistry Of Sound • Digicult | Digital Art, Design and Culture**. 2009. Disponível em: <http://digicult.it/digimag/issue-041/trevor-wishart-chemistry-of-sound/>. Citado 3 vezes nas páginas 50, 190 e 198.

MüLLER, Meinard; ZALKOW, Frank. Python Notebooks, **Timbre**. 2019. Disponível em: <https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Timbre.html>. Citado 2 vezes nas páginas 61 e 62.

NISHINO, Hiroki. **LC: A Mostly-strongly-timed Prototype-based Computer Music Programming Language that Integrates Objects and Manipulations for Microsound Synthesis**. Tese (Thesis) — National University of Singapore, Singapore, jan. 2014. Accepted: 2014-07-31T18:01:04Z. Disponível em: <https://scholarbank.nus.edu.sg/handle/10635/78945>. Citado na página 98.

NISHINO, Hiroki; OSAKA, Naotoshi. **LCSynth: A Strongly-timed Synthesis Language That Integrates Objects and Manipulations for Microsounds**. Copenhagen: [s.n.], 2012. Library Catalog: Zenodo Publisher: Zenodo. Disponível em: <https://zenodo.org/record/850112>. Citado na página 98.

PADOVANI, José. Spectral envelope extraction by means of cepstrum analysis and filtering in Pure Data. In: **Proceedings of the 3rd International Convention for Puredata**. São Paulo: [s.n.], 2009. p. 4. Disponível em: <http://puredata.info/community/conventions/convention09/padovani.pdf>. Citado na página 151.

PADOVANI, José Henrique; FERRAZ, Silvio. Proto-história , Evolução e Situação Atual das Técnicas Estendidas na Criação Musical e na Performance. **Revista Música Hodie**, v. 11, n. 2, 2011. ISSN 1676-3939. Number: 2. Disponível em: <https://www.revistas.ufg.br/musica/article/view/21752>. Citado na página 163.

PARK, Tae Hong. **Introduction to Digital Signal Processing: Computer Musically Speaking**. World Scientific, 2010. Google-Books-ID: 10hpDQAAQBAJ. ISBN 978-981-279-027-9. Disponível em: <https://www.worldscientific.com/worldscibooks/10.1142/6705>. Citado 6 vezes nas páginas 43, 66, 77, 78, 82 e 151.

PLUTA, Sam; ALEX, Ness; ALTIERI, Jem. **TimeStretch**. 2020. Original-date: 2020-06-23T00:29:29Z. Disponível em: <https://github.com/spluta/TimeStretch>. Citado na página 132.

PUCKETTE, Miller. **The Theory and Techniques of Electronic Music**. Hackensack, NJ: World Scientific Publishing Company, 2007. ISBN 978-981-270-077-3. Disponível em: <http://msp.ucsd.edu/techniques.htm>. Citado 2 vezes nas páginas 151 e 153.

REISS, D. Joshua; MCPHERSON, P. Andrew. **Audio Effects: Theory, Implementation and Application**. 1 edition. ed. Boca Raton: CRC Press, 2014. ISBN 978-1-4665-6028-4. Disponível em: <https://www.oreilly.com/library/view/audio-effects/9781466560284/>. Citado 6 vezes nas páginas 43, 66, 75, 78, 82 e 133.

ROADS, Curtis. **The Computer Music Tutorial**. Edition unstated edition. Cambridge, Massachussets: The MIT Press, 1996. ISBN 978-0-262-68082-0. Disponível em: <https://mitpress.mit.edu/books/computer-music-tutorial>. Citado 4 vezes nas páginas 72, 75, 77 e 166.

ROADS, Curtis. **Microsound**. 1. ed. Cambridge, Massachussets: The MIT Press, 2001. ISBN 978-0-262-18215-7. Disponível em: <https://mitpress.mit.edu/books/microsound>. Citado 6 vezes nas páginas 64, 65, 66, 93, 97 e 98.

SCHAEFFER, Pierre. **Treatise on Musical Objects: An Essay Across Disciplines**. [S.l.]: Univ of California Press, 2017. Google-Books-ID: faswDwAAQBAJ. ISBN 978-0-520-29430-1. Citado 6 vezes nas páginas 30, 60, 61, 64, 163 e 168.

SCHAEFFER, Pierre et al. **Solfège de l'objet sonore**. [S.l.]: INA GRM, 1967. ISBN B0024N6KKA. Citado 4 vezes nas páginas 31, 61, 164 e 166.

SEIDL, Fabian. **Granularsynthese mit Wavesets für Live-Anwendungen**. Tese (Mestrado) — Technische Universität, Berlin, abr. 2016. Disponível em: <https://www2.ak.tu-berlin.de/%7Eakgroup/ak_pub/abschlussarbeiten/2016/Seidl_MasA.pdf>. Citado na página 98.

SERRA, Xavier et al. **sms-tools**. Pompeu Fabra, 2020. Original-date: 2014-12-29T20:56:13Z. Disponível em: <https://github.com/bzamecnik/sms-tools>. Citado 2 vezes nas páginas 80 e 81.

SMITH, Julius O. **Mathematics of the Discrete Fourier Transform (DFT): with Audio Applications**. 2 edition. ed. North Charleston: W3K Publishing, 2007. ISBN 978-0-9745607-4-8. Disponível em: <https://ccrma.stanford.edu/~jos/mdft/>. Citado na página 72.

SMITH, Julius O. **Spectral Audio Signal Processing**. Stanford, Calif.: W3K Publishing, 2011. ISBN 978-0-9745607-3-1. Disponível em: <https://ccrma.stanford.edu/~jos/sasp/>. Citado 3 vezes nas páginas 75, 78 e 138.

SMITH, Julius O. Personal Website, **Viewing FFT Windows and STFT Overlap-Add, Frequency Shifting and Scaling**. 2013. Disponível em: <https://ccrma.stanford.edu/~jos/intro421b/Viewing_FFT_Windows_STFT.html>. Citado 2 vezes nas páginas 138 e 140.

SMITH, Julius O. Personal Website, **MUS421 Lecture 8B: Cross Synthesis Using Cepstral Smoothing or Linear Prediction for Spectral Envelopes**. 2020. Disponível em: <https://ccrma.stanford.edu/~jos/SpecEnv/>. Citado na página 151.

STONE-DAVIS, Férdia J. Vocalising Home: An Interview with Trevor Wishart. **Contemporary Music Review**, v. 34, n. 1, p. 5–21, jan. 2015. ISSN 0749-4467. Disponível em: <https://doi.org/10.1080/07494467.2015.1077562>. Citado 2 vezes nas páginas 189 e 190.

SUPERCOLLIDER. **SuperCollider**. 2019. Disponível em: <https://supercollider.github.io/>. Citado 2 vezes nas páginas 56 e 57.

VALLE, Andrea. **Introduction to Supercollider**. Edição: Translation. Berlin: Logos Verlag Berlin, 2016. ISBN 978-3-8325-4017-3. Disponível em: <https://www.logos-verlag.de/cgi-bin/engbuchmid?isbn=4017&lng=eng&id=>. Citado na página 57.

VASSILANDONAKIS, Yiorgos. An Interview with Trevor Wishart. **Computer Music Journal**, v. 33, n. 2, p. 8–23, jun. 2009. ISSN 0148-9267, 1531-5169. Disponível em: <http://www.mitpressjournals.org/doi/10.1162/comj.2009.33.2.8>. Citado 4 vezes nas páginas 51, 189, 190 e 197.

WILSON, Scott; COTTLE, David; COLLINS, Nick (Ed.). **The SuperCollider Book**. 1. ed. Massachusetts: [s.n.], 2011. (The MIT Press). ISBN 0-262-23269-3. Disponível em: <https://mitpress.mit.edu/books/supercollider-book>. Citado 6 vezes nas páginas 53, 59, 60, 98, 103 e 104.

WISHART, Trevor. The Composer's View: Extended Vocal Technique. **The Musical Times**, v. 121, n. 1647, p. 313, maio 1980. ISSN 00274666. Disponível em: <https://www.jstor.org/stable/963728?origin=crossref>. Citado na página 200.

WISHART, Trevor. The Composition of "Vox-5". **Computer Music Journal**, v. 12, n. 4, p. 21–27, 1988. ISSN 0148-9267. Disponível em: <https://www.jstor.org/stable/3680150>. Citado 2 vezes nas páginas 139 e 195.

WISHART, Trevor. The function of text in the VOX Cycle. **Contemporary Music Review**, v. 5, n. 1, p. 189–197, jan. 1989. ISSN 0749-4467, 1477-2256. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/07494468900640631>. Citado na página 195.

WISHART, Trevor. Music and technology: Problems and possibilities. In: **Companion to contemporary musical thought**. [S.l.: s.n.], 1992. v. 1, p. 565–582. Citado 2 vezes nas páginas 36 e 48.

WISHART, Trevor. From Architecture to Chemistry. **Interface**, v. 22, n. 4, p. 301–315, nov. 1993. ISSN 0303-3902. Tex.ids: wishart_architecture_1993-3. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/09298219308570639>. Citado na página 37.

WISHART, Trevor. **Audible Design: A Plain and Easy Introduction to Sound Composition**. York: Orpheus The Pantomime Ltd., 1994. ISBN 978-0-9510313-1-5. Disponível em: <http://www.trevorwishart.co.uk/AuD.html>. Citado 26 vezes nas páginas 31, 34, 37, 41, 44, 48, 49, 51, 60, 62, 63, 66, 87, 93, 96, 97, 103, 104, 137, 151, 160, 163, 192, 196, 198 e 202.

WISHART, Trevor. **On Sonic Art**. York: Routledge, 1996. v. 12. (Contemporary Music Studies, v. 12). ISBN 978-1-134-37326-0. Disponível em: <https://www.taylorfrancis.com/books/9781134373260>. Citado 11 vezes nas páginas 35, 36, 37, 163, 190, 191, 192, 194, 196, 199 e 200.

WISHART, Trevor. Personal Website, **Computer Sound Transformation : A personal perspective from the U.K.** 2000. Disponível em: <http://www.trevorwishart.co.uk/transformation.html>. Citado 2 vezes nas páginas 45 e 198.

WISHART, Trevor. **Red Bird and Voiceprints - CD Sleeve Notes**. 2006. Disponível em: <http://www.digital-music-archives.com/webdb2/application/Application.php?> Citado na página 194.

WISHART, Trevor. Globally Speaking. **Organised Sound**, v. 13, n. 2, p. 137–140, ago. 2008. ISSN 1469-8153, 1355-7718. Disponível em: <https://www.cambridge.org/core/journals/organised-sound/article/globally-speaking/8EAC372B10F14CCB631C9590F04CC14F>. Citado na página 189.

WISHART, Trevor. Computer Music: Some Reflections. In: **The Oxford Handbook of Computer Music**. [s.n.], 2011. Disponível em: <http://www.oxfordhandbooks.com/abstract/10.1093/oxfordhb/9780199792030.001.0001/oxfordhb-9780199792030-e-007>. Citado 2 vezes nas páginas 36 e 38.

WISHART, Trevor. Encounters in the Republic of Heaven by Trevor Wishart. In: **eContact!** Toronto: Canadian Electroacoustic Community (CEC), 2012. v. 15.2. Disponível em: <https://econtact.ca/15_2/wishart_encounters.html>. Citado 3 vezes nas páginas 112, 196 e 197.

WISHART, Trevor. **Sound Composition**. York, UK: Orpheus The Pantomime Ltd., 2012. ISBN 978-0-9510313-3-9. Citado 13 vezes nas páginas 34, 39, 40, 43, 49, 189, 193, 194, 195, 196, 197, 199 e 201.

WISHART, Trevor. **The Secret Resonance of Things**. [S.l.: s.n.], 2016. Encarte de CD. Citado na página 197.

WISHART, Trevor. Personal Website, **The Sound Loom**. 2018. Disponível em: <http://www.trevorwishart.co.uk/slfull.html>. Citado na página 31.

YOUNG, Dr Miriama. **Singing the Body Electric: The Human Voice and Sound Technology**. [S.l.]: Ashgate Publishing, Ltd., 2015. Google-Books-ID: KEchCgAAQBAJ. ISBN 978-0-7546-6986-9. Citado 2 vezes nas páginas 190 e 199.

ZOLZER, Udo (Ed.). **DAFX: Digital Audio Effects**. Edição: 2nd. Chichester, West Sussex, England: John Wiley & Sons, 2011. ISBN 978-0-470-66599-2. Disponível em: <http://dafx.de/DAFX_Book_Page_2nd_edition/index.html>. Citado 10 vezes nas páginas 42, 43, 65, 66, 77, 78, 82, 83, 85 e 168.

Apêndices

APÊNDICE A – Trevor Wishart: biografia, peças sonoras e textos

A.1 Trevor Wishart

A.1.1 Biografia condensada

Trevor Wishart é um compositor e performer inglês nascido em 11 de outubro de 1946 e radicado em York. Oriundo de uma família de classe operária de baixa renda, sua trajetória na musica é fortemente influenciada por seu local de origem e pela estrutura cultural, social e econômica da sua família. Wishart iniciou seus estudos universitários em química após receber uma bolsa da Universidade de Oxford, mas rapidamente transferiu-os para o curso de Música. Segundo o mesmo, a escolha inicial do curso se deu pela típica pressão familiar das classes emergentes inglesas que desejam aos filhos profissões com melhores salários e condições trabalhistas (VASSILANDONAKIS, 2009; STONE-DAVIS, 2015; WISHART, 2008; GARRARD; WILLIAMS, 2013; WISHART, 2012b).

Durante seu mestrado no final dos anos 1960, Wishart pesquisava a música de compositores contemporâneos, em especial Iannis Xenakis, quando a ocasião da morte do pai o fez mudar drasticamente a direção dos seus estudos e composições: o autor decidiu comprar um gravador de fita para registrar sons de máquinas e da fábrica de tratores na qual o pai trabalhara. Desse momento em diante, Wishart começou uma intensa exploração das possibilidades de trabalho com sons gravados. Ele relata que, em meados dos anos 1970, havia pouquíssimos locais que ofereciam equipamentos e cursos voltados à composição eletroacústica na Inglaterra – muito menos oferecendo os novos recursos computacionais – e que a situação no interior do país era ainda mais desfavorável, diferentemente do que acontecia em países como França, Holanda, Alemanha e Estados Unidos.

A Universidade de York era um dos poucos lugares que oferecia cursos nesta área, sendo naturalmente o local que Wishart escolheu para cursar seu doutorado. Neste período, segunda metade dos anos 1970, o compositor produziu uma das suas mais populares peças *Red Bird: A Political Prisoner's Dream* (Pássaro Vermelho: um Sonho de um Prisioneiro Político) . Nesta, o artista implementa – arduamente em meios puramente analógicos – várias de suas proposições técnicas e estéticas tais como a metamorfose sonora e o contínuo sonoro.

Paralelamente à composição de tais obras, Wishart iniciou sua pesquisa em técnicas estendidas de voz de forma mais sistemática após a visita do compositor Warren Burt a

York (YOUNG, 2015, p. 33). Iniciou então um projeto de coleta exaustiva de todos os tipos de sons possíveis de serem produzidos pela voz humana, bem como a emissão múltipla e interação destes. Resultaram disso duas peças, Anticredos e VOX Cycle, além de um livreto *The book of the lost voices*, que depois foi incluído em *On Sonic Art* (WISHART, 1996). A partir de tais estudos, Wishart também desenvolveu uma carreira de improvisação solística com técnicas estendidas de voz, geralmente sem a utilização de recursos eletrônicos, exceto amplificação.

Ao longo desse período Wishart teve contato com diversos materiais teóricos sobre computação e vislumbrou que várias das suas proposições técnicas para transformações vocais só poderiam ser implementadas utilizando tal ferramenta. Receoso de não conseguir acessos à computadores para a utilização musical na Inglaterra, Wishart inicia a escrita do livro *On Sonic Art* (WISHART, 1996) para registrar suas ideias estetico-computacionais e também se aplica para uma residência artística no *Institut de Recherche et Coordination Acoustique/Musique* (IRCAM) no ano de 1986. A partir daí desenvolve os seus primeiros primeiros programas de análise-resíntese com o vocoder de fase para a composição da peça VOX-5 .

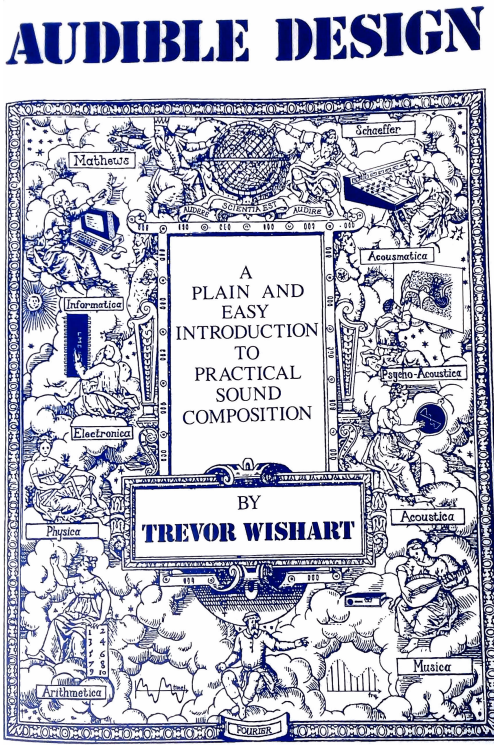
Após a residência no IRCAM, Wishart se reúne com um grupo de artistas e programadores ingleses para iniciar o desenvolvimento do Composers Desktop Project (CDP), uma suíte de programas dedicada à transformação de amostras de sons gravados, que foi lançada em 1987. Um dos intuitos centrais do CDP era a viabilização de música com computadores *desktop*, que antes era realizada somente em computadores *mainframe*. Wishart conclui a primeira obra feita inteiramente no CDP em 1994, *Tongues of Fire*, considerada um importante marco na música eletroacústica.

Wishart sempre optou por não ter cargos e posições fixas, tanto em universidades quanto em instituições privadas. A maior parte suas publicações de discos e livros também foi feita de forma totalmente independente, valendo-se apenas de alguns financiamentos governamentais esporádicos. Entretanto, apenas recemente algumas de suas obras foram disponibilizadas gratuitamente, como o livro *Audible Design* e o código fonte do CDP (GARRARD; WILLIAMS, 2013; MARTY, 2011; MILANI, 2009; STONE-DAVIS, 2015; VASSILANDONAKIS, 2009; YOUNG, 2015). Além disso, o compositor sempre frisa que deliberamente nunca residiu em Londres, pelo contrário, em cidades pequenas do interior da Inglaterra, especialmente York, onde o acesso a recursos tecnológicos sempre foi relativamente mais limitado.

A.1.2 Produção bibliográfica

Da produção bibliográfica de Wishart, destacam-se três livros nos quais o presente trabalho se baseou: *On Sonic Art* (1985), *Audible Design* (1994) e *Sound Composition* (2012). Além destes, existem vários livretos que acompanham seus discos e CDs bem

Figura 25 – Capa de *Audible Design* (1994) - Trevor Wishart



Fonte: (WISHART, 1996)

Figura 26 – Capa de *Plaine and Easie Introduction to Practicall Musicke* (1597) - Thomas Morley

Fonte: Peter Short (estampador). Artista desconhecido, domínio público, via Wikimedia Commons

como vários artigos em revistas científicas. Wishart tem um estilo de escrita fortemente personalista e sempre faz questão de frisar que o conteúdo dos seus textos não é científico, mas sim técnico e altamente especulativo.

On Sonic Art (1985) é um grande tratado-manifesto sobre *arte sonora*[1] mostrando os direcionamentos teóricos, composicionais, estéticos, filosóficos, científicos, sociológicos e políticos de sua visão pessoal sobre o campo do sonoro. Wishart faz fortes críticas à música instrumental e eletroacústica do final do século XX, em especial às correntes ligadas à tradição serialista, mostrando diversos pontos restritivos de tais estéticas e clamando por uma libertação de tais regras por meio de uma mudança de paradigma: ao invés de restringir a *arte sonora* a objetos discretos situados em *treliça* (*lattice*)[2], liberará-la

[1] Como o termo música é frequentemente utilizado em sentido restrito, abarcando somente manifestações que geralmente envolvam instrumentos acústicos guiados por parâmetros sonoros geralmente discretos(em especial altura, duração e intensidade), *arte sonora* é um termo usado por Wishart para abarcar tanto a música tradicional quanto as variadas manifestações artísticas, especialmente surgidas século XX, que valem-se de sons como aspecto fundamental: paisagem sonora, música sobre mídia fixa, *noise-music*, música cênica, etc (WISHART, 1996, p. 3-8).

[2] Wishart usa constantemente tal termo para referir-se ao fato de na notação da música ocidental

para utilizações e explorações mais amplas, utilizando objetos quaisquer situados em um *contínuo sonoro* (WISHART, 1996).

O livro também apresenta uma sólida quantidade de reflexões e apontamentos sobre morfologia de sons, espacialização, paisagem sonora, acusmática, dentre outros, sempre alinhavados com discussões sobre a percepção humana. Além disso, as seções finais trazem importantes e pioneiras contribuições de Wishart para vários temas relacionados ao uso da voz dentro da *arte sonora*, assunto este que é transversal em todo o seu trabalho. O autor inclui um manual exaustivo das emissões vocálicas , fruto de sua pesquisa e experimentação de técnicas estendidas de voz, em conjunto com um sistema de notação. Nas seções mais especulativas, também são apresentadas comparações entre os mecanismos intra-corpóreos humanos e animais de produção de sons, discussões sobre o conceito de enunciado vocal, sobre as variedades de situações do uso da voz, reflexões sobre a voz e os diversos sons vocálicos na comunicação humana, sempre atravessadas por reflexões sobre o uso da voz na *arte sonora* (WISHART, 1996).

Audible Design (1994) foi escrito depois do CDP ter sido lançando comercialmente e imediatamente depois da conclusão de *Tongues of Fire*, a primeira peça de Wishart feita inteiramente neste software utilizando um computador pessoal. Como consequência natural, é um manual transformação de sons gravados utilizando o CDP, especialmente da transformação dos *samples* de voz e dos exemplos da referida peça. Uma das principais razões para a escrita desse tipo de literatura está na concepção estética do autor que bons processos de transformação só podem ser alcançados por meio de uma cuidadosa seleção dos materiais e dos processos, visando sempre resultados específicos, concepção que será aprofundada nas próximas seções. Além de desvelar grande parte dos processos temporais e espectrais do CDP, *Audible Design* faz importantes contribuições à síntese granular, à análise e composição de texturas sonoras, propõe novos tratamentos à forma musical na *arte sonora* e, especialmente, desvela as abordagens de Wishart para a interpolação de sons, *modificação do tempo* e *modificação da altura*[3] (WISHART, 1994).

Em *Sound Composition* (2012), o autor faz descrições detalhadas de cada uma das suas peças, dos processos de composição utilizados nas mesmas, das ideias poéticas e técnicas motivadoras, relacionando esse todo com sua biografia e com o desenvolvimento do CDP. Todas as principais peças do referido compositor – eletroacústicas, para coro, mídia

tradicional alturas, durações e dinâmicas serem tratados como pontos discretos localizados em um reticulado uniforme. A tradução mais coerente é treliça, referindo-se não ao arranjo triangular da engenharia de estruturas, mas sim à estrutura de decoração e jardinagem compostas de ripas formando um reticulado.

[3] Os termos genérico utilizados por Wishart são *time stretching* (na tradução direta, esticamento temporal) para referir-se tanto ao esticamento quanto à contração temporal; e *pitch-shifting*, referindo-se a alterações na altura. Aqui preferimos utilizar o termo geral *modificação da escala temporal (TSM)*, que é mais frequente na literatura recente, bem como os termos *esticamento da escala temporal* *contração da escala temporal* quando for necessário realizar uma especificação; o termo *pitch-shifting* será traduzido como *modificação da altura*

fixa, etc – são contempladas e cada obra é abordada individualmente em um pequeno capítulo. Enquanto *Audible Design* se ocupa da criação e transformação de materiais para a composição eletroacústica, *Sound Composition*, por sua vez, se propõe a dissertar sobre a organização dos materiais ao longo das peças do compositor, ou seja, da forma e estrutura na sua arte sonora; mais especificamente sobre sua proposição de *contínuo sonoro* para organização dos materiais, detalhada na seção 1.2. No texto, Wishart desvela seus dois principais pontos de partida para a criação: propostas/desafios estéticos e técnicos de cada obra. Ao final, o compositor também detalha seus processos de difusão de obras estéreo para configurações multicanal, apresentando várias partituras de difusão de suas peças estéreo (WISHART, 2012b) .

A.1.3 Produção artística

A obra de Trevor Wishart é constituída majoritariamente de composições para mídia fixa, apresentando também diversas peças para teatro, improvisações ao vivo utilizando técnicas estendidas de voz, peças para conjuntos vocais e eletrônica, instalações sonoras e jogos musicais educativos. O compositor raramente utiliza recursos de síntese de som, preferindo trabalhar com transformações de sons gravados, e a grande maioria das suas obras utiliza a voz humana de alguma maneira. Como Wishart geralmente se propõe a abordar ao menos dois aspectos em suas composições – um técnico e um estético – é natural observar vinculações entre suas peças, seja na continuação da investigação de questões surgidas na peça anterior ou na maneira como seus recursos técnicos vão se aprimorando. Faremos aqui um breve panorama de algumas obras selecionadas, que ajudará na compreensão dos processos utilizados pelo compositor.

Red Bird: A Political Prisoner's Dream (1977) - Feita nos típicos estúdios analógicos utilizando quase unicamente corte e colagem de fitas, mixagem e filtros, *Red Bird* levou 4 anos para ser finalizada uma vez que Wishart trabalhava no estúdio da Universidade de York diariamente das 4:00 às 10:00 da manhã, período em que os estudantes e funcionários não utilizavam as instalações. Prestigiada e especialmente singular na música eletroacústica, *Red Bird* explora pontos estruturantes da música acusmática: primeiramente, o fato de que um dado som pode remeter à várias fontes sonoras (um som de um livro batendo em uma mesa é, sonoramente, muito próximo ou igual ao som de um murro, de um instrumento de percussão, de passos, etc); em segundo lugar, o fenômeno anterior pode ser utilizado para sugerir e induzir que sons possam ser transformados uns nos outros (e.g. sons de abelhas transformados em sons de consoantes "zzzz" por sua vez transformados em sons repetitivos de máquinas), processo de *metamorfose sonora*; em terceiro, a possibilidade trazida pela escuta acusmática de sugestão de um mundo fantasioso, onírico, mágico e alternativo; em quarto ponto, a possibilidade de se criar peças eletroacústicas que tenham caráter narrativo ou estruturas similares às estórias, mais precisamente *estórias das transformações sonoras*

(LANDY, 1991; AMELIDES, 2016). Sendo profudamente influenciada por obras como *O Cru e o Cozido* de Levi-Strauss e *O Rebelde* de Albert Camus, a peça apresenta variadas dimensões simbólicas, mitológicas e filosóficas, como o conflito entre concepções abertas e fechadas da realidade; as diversas invocações provocadas por um som (afetivas, políticas, religiosas, sociológicas, etc); o antagonismo entre a ideia de linguagem como único meio preciso, articulado, de elucidação objetiva da realidade e a música, em contraste, como atividade periférica, puramente epifenomenal em relação à realidade concreta e racional, funcionando apenas como meio de transmissão impreciso de emoções vagas (WISHART, 2006; WISHART, 2012b).

Anticredos (1979) - Peça para 6 vozes amplificadas utilizando técnicas estendidas de voz, *Anticredos* marca uma passagem do uso de *metamorfoses sonoras* como metáforas externas ao sonoro, em *Red Bird*, para o uso de *transformações sonoras* como um princípio geral de organização em *Tongues of Fire* . Após a finalização de *Red Bird*, Wishart vislumbrou as infindas possibilidades de transformação de sons via meios eletrônicos e seus possíveis sucessores digitais, aplicando-se para uma residência artística no IRCAM na qual poderia utilizar um computador *mainframe*. Devido ao futuro incerto do projeto, Wishart decide continuar suas explorações da *metamorfose sonora* utilizando a voz humana, "instrumento" mais maleável existente, que permitia a geração de uma enorme quantidade de espectros bem como rápidas transições entre os mesmos (WISHART, 2012b). Em *Anticredos*, há uma exploração puramente sonora das emissões vocálicas – naturalmente com o foco nas *transformações sonoras* possíveis entre tais sons – com uma intencional ausência de metáforas musicais deliberadas tal como em *redbird* e ainda experimentações das heurísticas de Wishart em contextos de apresentação ao vivo. Desta maneira, *Anticredos* tem caráter de análise exploratória do catálogo de sons possíveis via técnicas vocais estendidas, preocupando-se mais em abarcar o grande conjunto de sons possíveis do que concentrar-se em um aspecto específico. Naturalmente, engendrou o desenvolvimento de um sistema de notação de técnicas estendidas de voz. A peça também marca o início da investigação mais sistemática das técnicas de espacialização – apresentando um sistema quadrafônico em que os intérpretes controlavam a espacialização via *joysticks* – (WISHART, 2012b; WISHART, 1996).

Vox Cycle (1980-1988) - Como natural prosseguimento de *Anticredos*, o ciclo *VOX* é composto de 6 peças para quatro vozes amplificadas, geralmente acompanhadas por fita, cada uma explorando alguma característica específica da voz humana, do tipo de estrutura musical, da função do texto e da experiência humana. *VOX 1* trata da dissolução da ordem no caos e, vice-versa, da organização do caos na ordem : *streams* de emissões vocais vão se amalgamando ou se dissociando entre si ou no acompanhamento eletrônico dos sons da fita. *VOX 2* é baseada no *Buranku* – teatro de bonecos japonês – concentrando-se em desenvolver como ornamentos vocais(portamentos, glissandos, trinados, vibratos, etc) podem emergir de estados meditativos e longos contínuos sonoros. A peça também explora

as possibilidades de controle de texturas de ornamentos e suas possíveis metamorfoses. *VOX 3* não apresenta acompanhamento de fita e concentra-se em explorar diversas possibilidades rítmicas entre os quatro cantores, sugerindo brigas, diálogos, disputas e jogos. Apesar de não ter nenhuma conotação direta em qualquer língua, a peça sugere diversos elementos da linguagem por meio das intenções redigidas na bula e dos fonemas utilizados. Além disso, Wishart faz vários desafios de compreensão e produção dos sons, ao propor que os cantores emitam sons como "tktktk" e "dgdgdg" em velocidades altas, desafiando tanto a percepção dos ouvintes quanto a produção por parte dos cantores. *VOX 4* propõe a constituição de uma espécie de teatro puramente formal, abstrato, feito exclusivamente de elementos sonoros. Além de explorar vários elementos da narratividade para sons abstratos, a peça investiga os limites de coesão e disrupção dos cenários vocais formados. *VOX 5* é a única peça do ciclo feita inteiramente para mídia-fixa e, segundo o compositor, funciona como um *Arioso*, formando uma ponte para o último movimento, que é uma típica música eletrônica dançante. *VOX 6* é, ironicamente, talvez a única composição de Wishart em que há o uso tradicional de quase todos os parâmetros: alturas e ritmos definidos numa canção de *disco music*, uso de sintetizadores MIDI (que o compositor chama de "caixas-pretas comerciais"), letras típicas de canção com uso de rimas, padrões e fortes repetições, dentre outros. Tais fatores indicavam, para o compositor, um aspecto fundamental da cultura ocidental: o êxtase da dança na cultura *disco* como um tipo fruição hedonística não-reflexiva. (WISHART, 1989; WISHART, 2012b).

VOX 5 (1980-1988) - realizada durante período de residência artística no IRCAM, *VOX 5* é também um marco na música eletroacústica por ser uma das primeiras peças a utilizar algoritmos de análise-ressíntese (vocoder de fase), à época quase nada conhecidos no IRCAM e re-implementados por Wishart decifrando os arquivos binários originais de San Diego (DOLSON, 1986). Quase todas as transformações da peça iniciam-se com um enunciado vocal monoaural do próprio compositor utilizando-se técnicas estendidas de voz, metamorfoseando-se em algum som não humano (sinos, cavalos, enxames de abelhas, etc) estereofónico e depois voltando aos sons vocálicos, para então seguir para o próximo enunciado vocal e consequente *metamorfose sonora*. Como Wishart tem grandes preocupações em manter a *credibilidade de fonte* na perspectiva do público em suas transformações, importantes compromissos entre a escolha do material fonte e destino (incluindo as técnicas estendidas), o tipo de transformação espectral, a espacialização e a duração de todo o processo são feitos de forma a tentar garantir a manutenção de tal critério (WISHART, 1988; WISHART, 2012b).

Tongues of Fire (1994) - Pedra angular e divisor de águas no desenvolvimento da *metamorfose sonora* e do próprio CDP, a peça aprofunda diversas transformações sonoras que começaram a ser desenvolvidas em *VOX 5* e *Red Bird* e também inaugura várias outras como *distorção de pacotes de onda* , novas técnicas de síntese granular e de análise-resíntese. O material tema é um complexo enunciado vocal, produzido e editado

por Wishart, utilizando várias técnicas estendidas de voz que vai sendo, em partes ou inteiramente, transformado e metamorfoseado continuamente ao longo da peça. Wishart ressalta que, por ter sido a primeira peça feita inteiramente em um computador pessoal, rapidamente pode-se perceber que as possibilidades de transformação e metamorfose de sons eram infinitas, portanto a partir daí era imprescindível diferenciar entre processos técnicos e processos auditivamente perceptíveis - preocupação que marcaria fortemente toda sua carreira e obra(WISHART, 1996; WISHART, 1994; WISHART, 2012b).

Imago (2002) - é uma peça acusmática baseada inteiramente no curto som de um clique entre duas taças de uísque, de cerca de 100ms, que se metamorfoseam em uma grande variedade de sons, aludindo a pássaros, vozes, gamelão e oceano. Também realizada inteiramento no CDP, *Imago* foi motivada como uma resposta a duas críticas que colegas do compositor faziam à música eletroacústica: o primeiro que seria impossível de se seguir alguma lógica de montagem de materiais na música acusmática; o segundo de que toda a música eletroacústica seria monofônica, sendo impossível a distinção entre objetos, linhas e texturas (WISHART, 2012b).

Globalalia (2004) - Se em *Imago* Wishart trabalha com um único material, em *Globalalia* o compositor utiliza uma grande base de dados: gravações de rádio e TV de 26 diferentes línguas e 134 diferentes falantes que foram segmentadas em 8300 sílabas. A ideia central da peça é buscar qual é o cerne da similaridade retida pelos diversos falantes em algumas sílabas, criando, assim, uma fala híbrida entre um possível falante médio mundial. Apesar de termos milhares de palavras nas centenas de línguas existentes no planeta, além de uma língua geralmente ser totalmente incompreensível aos falantes de outra língua, todas essas são constituídas de um pequeno, reduzido e apreensível número de fonemas e sílabas – este sim geralmente comum à toda comunidade humana. Concentrando-se mais em aspectos fonéticos do que fonológicos, Wishart tanto busca criar tanto enunciados encadeando sílabas de diferentes falantes quanto elabora pequenos movimentos que exploram as variedades, similaridades e transformações de cada tipo de sílaba escolhida para a formação dos enunciados. O compositor afirma que a peça é uma celebração da fala humana por meio do seu vocabulário compartilhado de objetos sonoros (WISHART, 2012a; WISHART, 2012b)

Encounters in the Republic of Heaven (2011) - peça acusmática de oito canais e mídia-fixa na qual o compositor explora, diferentemente de *Globalalia*, algumas das características da fala natural em mais longas escalas temporais, concentrando-se nos aspectos fonológicos, especialmente no nível de duração das frases e orações linguísticas: melodia, entoação e campo harmônico implícito; andamento, compasso e ritmo; e timbre específico de cada falante. O peculiar sotaque dos habitantes de Durham, nordeste da Inglaterra, foi escolhido para mostrar as nuances sonoras de cada indivíduo dentro de uma comunidade de semelhanças sonoras. Variados indivíduos foram selecionados de forma a

permitir maior dissimilaridade dentro do grupo: homens, mulheres, crianças, idosos, poetas com fala naturalmente empostadas e pessoas com uso regular da fala. Por ser a primeira peça de oito canais feita por Wishart, grande tempo da feitura foi dedicado à adaptação das funções do CDP para arquivos multicanal e ao desenvolvimento de texturas específicas para o contexto. (WISHART, 2012a; WISHART, 2012b)

The Secret Ressonance of Things (2016) - conjunto de peças acusmáticas de oito canais para mídia-fixa com diferentes propostas de sonificação de dados científicos, uma das raras peças eletrônicas de Wishart em que não há o uso de sons gravados e também não utiliza o CDP extensivamente. Em *Supernova* os dados do espectro eletromagnético de uma supernova são trasnformados e mapeados para um espectro sonoro audível. Já em *Signatures of Chaos*, há uma exploração de processos que tendem a resultados caóticos, como a função logística (que modela o crescimento de populações) e o escoamento de Taylor-Couette (surgimento de turbulência em fluidos rotacionais) . Por fim, em *Dithyramb - Kepler 63c* Wishart faz referência aos primeiros planetas que foram descobertos fora da nossa galáxia e que podia prover condições para a vida humana. Assim, valendo-se do sistema de modelamento físico NESS (Próxima Geração de Síntese Sonora), o compositor simula condições terrenas físicas impossíveis para os instrumentos existentes, mas que talvez sejam possíveis em outros planetas (WISHART, 2016; WISHART, 2012b).

A.1.4 Desenvolvimentos computacionais

> (...) eu me envolvi com desenvolvimento de software porque eu simplesmente não podia arcar com a infinita cadeia de atualizações de caixas pretas que universidades e estúdios comerciais podiam comprar seus orçamentos (escrever software toma tempo, mas, uma vez escrito, é gratuito e você pode atualizar sem pedir autorização). (VASSILANDONAKIS, 2009, *apud* WISHART, tradução nossa).

Wishart foi o principal fundador e idealizador do software colaborativo Composers Desktop Project (CDP), descrito em maiores detalhes na seção 1.5, no qual atuou em diversas áreas do desenvolvimento, desde a idealização e desenvolvimento das funções básicas até a elaboração de interfaces de usuário. O autor sempre se coloca todavia como um *compositor* que desenvolve ferramentas computacionais para possibilitar sua criação artística – que depois de uma estabilização são partilhadas com o público – em oposição a um desenvolvedor de software que algumas vezes utiliza suas ferramentas criativamente. Além disso, afirma que seu contato com a computação se deu especialmente porque sua condição socioeconômica não permitia o acesso à infinidade de softwares (geralmente de alto custo), hardwares e equipamentos oferecida por universidades e empresas, em conjunto com as suas constantes atualizações (VASSILANDONAKIS, 2009).

Os primeiros contatos de Wishart com a programação foram ainda durante seu período de estudos em York, no qual o compositor desenvolveu um sistema de controle

da espacialização de *VOX-1* em Basic utilizando um computador Sinclair QL (MILANI, 2009). Seu aprendizado de programação e computação aconteceu de maneira informal ou auto-didata, entretanto sempre auxiliada por grandes nomes da computação musical e musicologia empírica como Miller Puckette, Stephen McAdams, Philippe Depalle, dentre outros nomes contemporâneos de Wishart no IRCAM, bem como seus colegas de fundação de CDP Martin Atkins e Archer Endrich (WISHART, 2000; WISHART, 1994; MILANI, 2009).

Apesar de propor inúmeros novos paradigmas na computação musical, o estilo de programação e desenvolvimento de Wishart é despreocupado com preciosismos formais e organizacionais; assim como sua literatura, que a despeito de inaugurar novos paradigmas científicos, dispensa o rigor e o método da ciência contemporânea. Além disso, o referido autor frequentemente prefere adotar um tom fortemente pessoal na sua escrita textual e de software; característica que se sobressai na maneira como o CDP é organizado para atender a um método específico de composição bem como evitar integração e comunicação com outros ambientes. Mesmo com grandes e criativas contribuições para a computação musical, Wishart se considera um "*amador prendado*" no campo da programação (MILANI, 2009).

Visto que as mais numerosas e fundamentais contribuições ao CDP são de Trevor Wishart, naturalmente várias das concepções projetivas do ambiente se moldaram aos sistemas e metodologias específicas de composição do referido autor (CDP, 2018a). Um ponto essencial dessa interdependência está na posição do compositor perante ao debate entre a computação de áudio em tempo real e tempo diferido, este se posicionando fortemente à favor dos processos de tempo diferido. Vale destacar que Wishart não somente desenvolveu variados programas para a manipulação espectral e temporal de amostras de áudio mas elaborou fecunda literatura e realizações musicais das relações entre o resultado de tais transformações sonoras e a natureza da cada específica de cada amostra, especialmente as amostras de conteúdo vocal.

A.2 O papel da voz: técnicas estendidas, função do texto e computação da voz

> (...) A laringe e o trato vocal humano bem como a siringe dos pássaros são, dentro da totalidade, sistemas mais maleáveis, permitindo o controle ao longo do *contínuo de altura* (como em cordas sem trastes), o ajuste contínuo do espectro formante e das transformações em direção ao ruído, grão, rangido [4] ou de texturas harmônicas ou multiplexadas. O sutil controle do aparato físico que governa a emissão de sons garante que gestos intencionais de entrada possam ser excessivamente sutis e multi-dimensionais. Assim, a voz se encontra no mais distante extremo em

[4] *grit*, tradução nossa

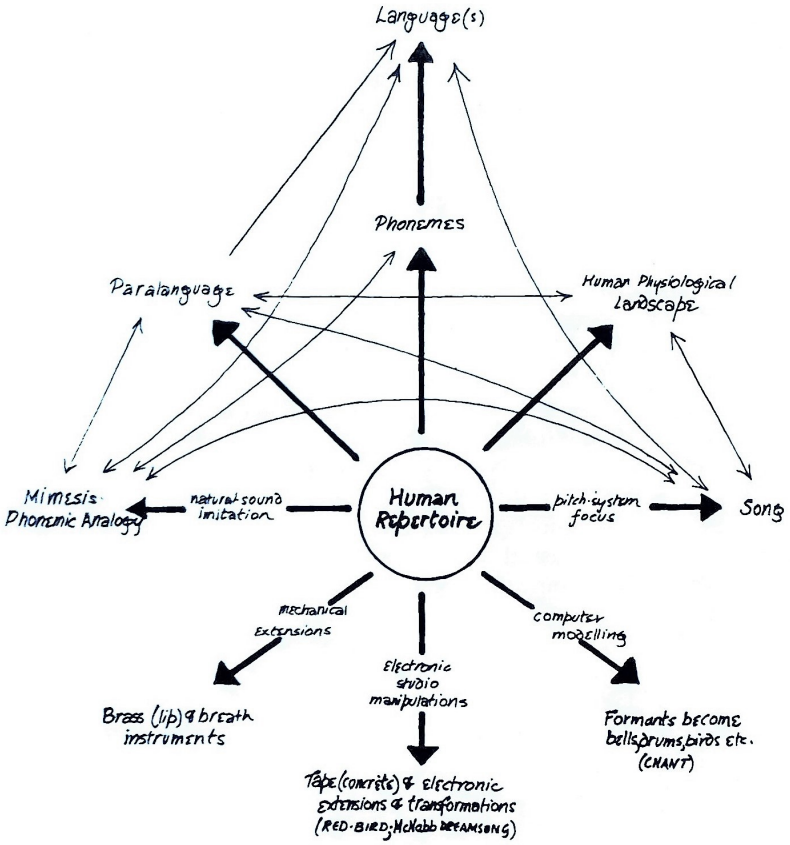
> relação aos instrumentos musicais desenvolvidos no ocidente. (WISHART, 1996, p. 103).

On Sonic Art, especialmente, apresenta uma longa e profunda, mas também despojada, reflexão sobre as potencialidades e conhecimentos envolvendo a voz humana: técnicas estendidas; transmissão de sentido, gênero e emoção; comparações com outros seres vivos; voz como estruturante da música e da razão, etc. Como a estrutura vocal é o mais robusto, amplo, independente e sensível aparato fisiológico de controle humano, é natural que tantas associações possam ser feitas em relação àquele que é um dos nossos principais mecanismos de contato com outros humanos. Ainda mais natural é o fato de um compositor como Wishart valer-se de tal aparato a fim de ampliar as possibilidades musicais da chamada estrutura de *treliça*.

De fato, nenhum instrumento acústico tradicional, especialmente aqueles de parâmetros contínuos, permite tamanhas possibilidades de controle: se avaliarmos um exemplo vocal simples, onde a altura é mantida constante e as formantes são alteradas (por exemplo, diferentes vogais faladas a uma mesma altura) nada semelhante pode ser realizado em um dos instrumentos da orquestra. Por outro lado, se tentamos construir um instrumento acústico que permita tal experimento – um tubo de borracha acoplado a um sistema de palheta – conseguiremos ter mais controle das formantes, mas dificilmente conseguiremos controlar alturas com a exigência que um instrumento de orquestra demanda. Mesmo no caso dos modernos instrumentos digitais, ainda que exista uma maior possibilidade de controle, nada pode ser comparado com a resposta fina do mecanismo vocal.

> (...) a voz é intrinsecamente multimídia. Mais do que um instrumento musical, ela pode transmitir a idade, sexo, uma imagem da saúde do emissor, sua personalidade (ou aquela simulada), a intenção e o estado de espírito. (...) tudo isso pode ser rastreado nas propriedades acústicas do fluxo vocal.
>
> Nós não podemos lidar com todas essas perspectivas em uma peça – a construção de uma forma musical exige foco. Nós articulamos certos aspectos do *stream* sonoro enquanto ignoramos, ou retemos relativamente constante, os outros aspectos. Mover todos eles de uma só vez destrói qualquer clareza do discurso musical. VOX, desta maneira, adota diferentes técnicas e diferentes perspectivas perceptivas em seus diferentes movimentos. (WISHART, 2012b, p.52-53, tradução nossa).
>
> Quando eu comecei a trabalhar no estúdio eu estava interessado na ideia de transformação, e eu estava trabalhando no estúdio analógico, assim era quase impossível fazer qualquer coisa. Mas o que eu descobri rapidamente foi que você pode imitar quase qualquer coisa com sua voz, logo, se você desejasse fazer uma transformação da coisa real no estúdio analógico você precisava transformar sua voz em algo similar a àquilo que você que você queria, como assoviar parecido com um canto de pássaro ou algo assim, e então, você poderia executar algum tipo de mixagem e obter algo algumas metamorfoses relativamente boas dessa maneira. (YOUNG, 2015, p. 33, tradução nossa).

Figura 27 – Relação entre elementos do repertório humano



Fonte: Wishart (1996, p. 286)

Num período marcado por aguda especialização de funções nas ciências, artes e no mundo do trabalho, Wishart destoa não somente por integrar ocupações como compositor, desenvolvedor de software e teórico, mas especialmente por ter sido, por muito tempo, talvez o único intérprete capaz de executar suas próprias obras. Em *Book of lost voices* (1979)[5], Wishart foi um dos pioneiros a fazer um catálogo extensivo da produção vocálica humana, concentrando-se em mapear exaustivamente tanto o repertório de emissões irredutíveis quanto o teste de possibilidade de combinação entre todas elas. Além de elaborar um sistema de notação para as transformações de tais sons vocais, o compositor ainda costumava gravar versões integrais das suas peças para 4 vozes, como *Anticredos*, de forma a facilitar a aprendizagem dos cantores e das cantoras que fariam a interpretação ao vivo (WISHART, 1980).

Além da pesquisa pioneira e exploratória das técnicas estendidas de voz, Wishart também foi um dos pioneiros em testar os limites do uso da voz para imitação e simulação

[5] Libreto que depois foi incorporado como o capítulo *The human repertoire* de *On Sonic Art* sendo ainda um das poucas bibliografias existentes que catalogam e descrevem técnicas estendidas de voz. Além deste referido texto, só encontramos em Edgerton (2011) material que aborda essa temática.

de sons naturais – como em *Red Bird* e *VOX-5* – bem como a metamorfose entre esses diversos sons. O outro limite extremo de extensão das técnicas vocais explorado por Wishart é das transformações aumentadas por computador, na qual o compositor utiliza tanto excertos convencionais quanto excertos de técnicas estendidas de voz para serem ampliados via procedimentos computacionais diversos.

Wishart fala que, de uma forma geral, está mais interessado nos aspectos sonoros da linguagem e da fala do que nos seus aspectos linguísticos e semânticos. Também diz que frequentemente os aspectos semânticos atrapalham seus processos de composição pois introduzem uma infinidade de variáveis na já árdua tarefa de controle dos incomensuráveis parâmentros do *contínuo sonoro*. Ainda assim, em cada uma de suas peças que apresentam texto (ou frases curtas) o mesmo manifesta uma função específica. Temos o uso do texto como jogos rítmicos, construção de situação social (via linguagem inventada), produção de sentido mítico, incorporação de aspectos culturais de uma dada comunidade, etc. (WISHART, 2012b).

O ser humano possui capacidades auditivas extremamente acuradas para diversas habilidades relativas à percepção de sons vocálicos: reconhecimento e distinção de falante, distinção entre fala natural e sintética, distinção entre emissão acústica e eletromecânica, apreensão de sentido, percepção de características paralinguísticas, separação de fontes em meio a densas misturas (*cocktail party effect*), dentre outras (BENESTY; SONDHI; HUANG, 2008). Se ocorre algum processamento mínimo na fala humana, mesmo que involuntário ou indesejável (e.g LPC, VOIP, telefone) a escuta humana detecta imediatamente que há algo diferente da fala natural ali. Outro ponto importante é que mesmo que a voz sofra alguma transformação extremamente radical, as características vocais costumam persistir solidamente e, em geral, os ouvintes conseguem apontar fortes indícios vocais(sentido linguístico, evolução das formantes, articulação, etc). Assim, é natural que existam um grande número de ferramentas desenvolvidas visando as especificidades dos sons vocálicos e que ferramentas computacionais de uso geral tendam a não funcionar em sua totalidade quando aplicadas aos sons da fala e canto humanos.

Wishart, além das suas diversas técnicas estendidas de voz e manipulação de sons vocais em estúdio analógico, desenvolveu algumas metodologias de transformação específicas para tais sons. Em resumo, o compositor explora as características da percepção da voz de duas maneiras opostas: por um lado, desvelando um objeto-sonoro cujo início é um excerto vocal sem modificações para então realizar suas transformações, aumentando, assim, a sensação de vocalidade da transformação; por outro lado, realizando transformações radicais para chegar a sons que são mais dificilmente alcançados pela voz (ou por só um falante) , e.g. quando se faz um processo de síntese cruzada entre o envelope de uma amostra vocal e o conteúdo espectral de ruído natural (som de mar), resultando na permanência da inteligibilidade do discurso mas sob um timbre de oceano.

Por fim, destacamos aqui duas inclinações estéticas de Wishart: a quase total exclusividade no uso de sons gravados – consequentemente ausência de produção de sons via mecanismos de síntese. O compositor justifica esse fato, dizendo que trabalhar com síntese é como pintar somente com cores primárias, é possível alcançar resultados complexos, porém somente via caminhos igualmente intrincados. O segundo ponto é que Wishart quase nunca trabalha com softwares de áudio em tempo real [6], traço fundacional que perpassa o CDP, abordado mais detalhadamente na seção 1.5.

Reconhecendo que o advento tecnológico do final do século XX, em especial a computação em tempo real, trouxe a contribuição de borrar as fronteiras entre as antes definidas funções de compositor, intérprete-*performer* e construtor de instrumentos, especialmente por permitir aos artistas uma exploração dessas fronteiras de forma mais direta e menos amarrada a concepções teóricas, Wishart prefere utilizar o conceito de composição de maneira menos categórica e como um processo geralmente de tempo diferido. Isso se dá especialmente devido ao fato de Wishart explorar meticulosamente amostras de áudio gravadas, num procedimento que, conceitualmente, não permite uma exploração das vantagens do tempo real pois geralmente avalia as amostras do início ao fim para então executar o processamento(WISHART, 1994). Nesse caso específico, tarefas de tempo real poderiam acelerar alguns processos de composição e da interface homem-computador – o que aconteceu na indústria da computação e do áudio desde os anos 1980 – todavia ainda que tenhamos recentemente diversas manifestações de acionamento e transformação de *samples* ao vivo (DJ, VJ, *live-coding*, *Dub*, etc) é impensável e impraticável uma performance ao vivo de transformação de sons gravados no estilo minucioso, analítico e rememorativo de Trevor Wishart.

[6] um sistema computacional de tempo real é aquele em que as tarefas de processamento de dados têm um tempo limite severamente estrito para serem cumpridas, no caso dos sistemas de áudio é desejável que esse tempo seja inferior aos limiares perceptivos da audição humana

Anexos

ANEXO A – Código SuperCollider : transformações algorítmicas contínuas e combinadas

```
(
c = WavesetsEvent.read("~/Audios/rato_part2.wav");
c.add(\simpAlveo);
c = WavesetsEvent.read("~/Audios/rato_part3.wav");
c.add(\vibrAlveo);
c = WavesetsEvent.read("~/Audios/rato_part4.wav");
c.add(\vibrUvula);
WavesetsEvent.prepareSynthDefs;
)

(
Pdef(\saTotalAlea,
        Pwavesets(
                Pbind(
                \name, \simpAlveo,
                \start, Pn(Pseries(0, 1, 1552),1),
                \num, Pxrand([2,2,2,3,5,1,4,7,1,1,1,1,1,1],inf),
                \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1,1],inf),
                \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
                \rate2, 1.0,
                \amp, 0.4,
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false,
                )
        )
);
Pdef(\vaTotalAlea,
        Pwavesets(
                Pbind(
                \name, \vibrAlveo,
                \start, Pn(Pseries(0, 1, 1054),1),
                \num, Pxrand([2,2,2,3,5,1,4,7,1,1,1,1,1,1],inf),
                \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1,1],inf),
                \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
                \rate2, 1.0,
                \amp, 0.4,
                \pan, Pbrown(-0.5, 0.5, 0.001),
                )
```

```
        )
);
Pdef(\vuTotalAlea,
        Pwavesets(
                Pbind(
                \name, \vibrUvula,
                \start, Pn(Pseries(0, 1, 1658),1),
                \num, Pxrand([2,2,2,3,5,1,4,7,1,1,1,1,1],inf),
                \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
                \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
                \rate2, 1.0,
                \amp, 0.4,
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false,
                )
        )
);
//////////////////
//////////////////
//////////////////
Pdef(\saNum7a12Rate8,
        Pwavesets(
                Pbind(
                \name, \simpAlveo,
                \start, Pn(Pseries(0, 1, 1552),1),
                \num, Pstutter(1658, Pwhite(7,12)),
                \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
                \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
                \rate2, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1]*8,inf),
                \amp, 0.4,
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false,
                )
        )
);
Pdef(\vaNum7a12Rate8,
        Pwavesets(
                Pbind(
                \name, \vibrAlveo,
                \start, Pn(Pseries(0, 1, 1054),1),
                \num, Pstutter(1658, Pwhite(7,12)),
                \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
                \rate2, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1]*8,inf),
                \amp, 0.4,
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false,
                )
```

```
86          )
87  );
88  Pdef(\vuNum7a12Rate8,
89          Pwavesets(
90                  Pbind(
91                  \name, \vibrUvula,
92                  \start, Pn(Pseries(0, 1, 1658),1),
93                  \num, Pstutter(1658, Pwhite(7,12)),
94                  \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
95                  \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
96                  \rate2, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1]*8,inf),
97                  \amp, 0.4,
98                  \pan, Pbrown(-0.5, 0.5, 0.001),
99                  \useFrac, false,
100                 )
101         )
102 );
103 ////////////////////
104 ////////////////////
105 ////////////////////
106 Pdef(\saRecipRep1,
107         Pwavesets(
108                 Pbind(
109                 \name, \simpAlveo,
110                 \start, Pn(Pseries(0, 7, 155),1),
111                 \num,Pstutter(1658, Pwhite(7,12)),
112                 \repeats, 1,
113                 \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
114                 \rate2, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1].reciprocal,
                        inf),
115                 \amp, 0.4,
116                 \pan, Pbrown(-0.5, 0.5, 0.001),
117                 \useFrac, false,
118                 )
119         )
120 );
121 Pdef(\vaRecipRep1,
122         Pwavesets(
123                 Pbind(
124                 \name, \vibrAlveo,
125                 \start, Pn(Pseries(0, 7, 105),1),
126                 \num,Pstutter(1658, Pwhite(7,12)),
127                 \repeats, 1,
128                 \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
129                 \rate2, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1].reciprocal,
                        inf),
130                 \amp, 0.4,
```

```
131                 \pan, Pbrown(-0.5, 0.5, 0.001),
132                 \useFrac, false,
133                 )
134         )
135 );
136 Pdef(\vuRecipRep1,
137         Pwavesets(
138                 Pbind(
139                 \name, \vibrUvula,
140                 \start, Pn(Pseries(0, 7, 165),1),
141                 \num,Pstutter(1658, Pwhite(7,12)),
142                 \repeats, 1,
143                 \rate, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1],inf),
144                 \rate2, Pxrand([2,2,2,1.5,1.5,1,0.75,0.5,1,1,1].reciprocal,
                        inf),
145                 \amp, 0.4,
146                 \pan, Pbrown(-0.5, 0.5, 0.001),
147                 \useFrac, false,
148                 )
149         )
150 );
151 ////////////////////
152 ////////////////////
153 ////////////////////
154 Pdef(\saSaltRateBrown,
155         Pwavesets(
156                 Pbind(
157                 \name, \simpAlveo,
158                 \start, Pn(Pseries(0, 4, 388),1),
159                 \num,1,
160                 \repeats,4,
161                 \rate, Pbrown(0.5, 1.5, 0.001),
162                 \amp, 0.4,
163                 \pan, Pbrown(-0.5, 0.5, 0.001),
164                 \useFrac, false,
165                 )
166         )
167 );
168 Pdef(\vaSaltRateBrown,
169         Pwavesets(
170                 Pbind(
171                 \name, \vibrAlveo,
172                 \start, Pn(Pseries(0, 4, 263),1),
173                 \num,1,
174                 \repeats,4,
175                 \rate, Pbrown(0.5, 1.5, 0.001),
176                 \amp, 0.4,
```

```
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false,
                )
        )
);
Pdef(\vuSaltRateBrown,
        Pwavesets(
                Pbind(
                \name, \vibrUvula,
                \start, Pn(Pseries(0, 4, 414),1),
                \num,1,
                \repeats,4,
                \rate, Pbrown(0.5, 1.5, 0.001),
                \amp, 0.4,
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false,
                )
        )
);
//////////////////
//////////////////
//////////////////
Pdef(\saTotalAlea2,
        Pwavesets(
                Pbind(
                \name, \simpAlveo,
                \start, Pn(Pseries(0, 1, 1552),1),
                \num, Pxrand([2,2,2,3,5,1,4,7,1,1,1,1,1],inf),
                \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
                \rate, 1,
                \rate2, Pxrand([2,1.3,2,1.5,1.5,1.2,0.75,0.65,0.9,1.1,0.8],
                   inf),
                \amp, Pbrown(0.3, 0.5, 0.001),
                \pan, Pbrown(-0.5, 0.5, 0.001),
                \useFrac, false,
                )
        )
);
Pdef(\vaTotalAlea2,
        Pwavesets(
                Pbind(
                \name, \vibrAlveo,
                \start, Pn(Pseries(0, 1, 1054),1),
                \num, Pxrand([2,2,2,3,5,1,4,7,1,1,1,1,1],inf),
                \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
                \rate, 1,
```

```
222                 \rate2, Pxrand([2,1.3,2,1.5,1.5,1.2,0.75,0.65,0.9,1.1,0.8],
                        inf),
223                 \amp, Pbrown(0.3, 0.5, 0.001),
224                 \pan, Pbrown(-0.5, 0.5, 0.001),
225                 )
226         )
227 );
228 Pdef(\vuTotalAlea2,
229         Pwavesets(
230                 Pbind(
231                 \name, \vibrUvula,
232                 \start, Pn(Pseries(0, 1, 1658),1),
233                 \num, Pxrand([2,2,2,3,5,1,4,7,1,1,1,1,1],inf),
234                 \repeats, Pxrand([2,2,2,3,3,1,4,5,1,1,1,1,1],inf),
235                 \rate, 1,
236                 \rate2, Pxrand([2,1.3,2,1.5,1.5,1.2,0.75,0.65,0.9,1.1,0.8],
                        inf),
237                 \amp, Pbrown(0.3, 0.5, 0.001),
238                 \pan, Pbrown(-0.5, 0.5, 0.001),
239                 \useFrac, false,
240                 )
241         )
242 );
243 )
244 // Organização semi-generativa da forma musical
245 (Pseq([
246         // apresentação
247         Pseq([Pdef(\vuTotalAlea), Pdef(\vaTotalAlea), Pdef(\saTotalAlea)],
                2),
248         //glissandos intra conjuntos de onda
249         Ptpar([0.0, Pdef(\vuTotalAlea), 2.9, Pdef(\vaTotalAlea), 2.1, Pdef(
                \saTotalAlea)], 2),
250         Ptpar([0.0, Pdef(\vaNum7a12Rate8), 4.9, Pdef(\saNum7a12Rate8), 8.1,
                 Pdef(\vuNum7a12Rate8)], 2),
251         //glissandos descendentes
252         Ptpar([0.0, Pdef(\vaRecipRep1), 0.9, Pdef(\saRecipRep1), 1.1, Pdef(
                \vuRecipRep1)], 4),
253         //combinação dos procesos anteriores
254         Ppar([Ptpar([0.0, Pdef(\vaRecipRep1), 0.9, Pdef(\saRecipRep1), 1.1,
                 Pdef(\vuRecipRep1)], 4),    Ptpar([0.0, Pdef(\vaRecipRep1),
                1.9, Pdef(\saRecipRep1), 2.1, Pdef(\vuRecipRep1)], 5),
255         ]),
256         //tutti
257         Ppar([Pdef(\vuTotalAlea), Pdef(\vaTotalAlea), Pdef(\saTotalAlea),
                Pdef(\vaNum7a12Rate8),  Pdef(\saNum7a12Rate8), Pdef(
                \vuNum7a12Rate8)]),
```

```
258         Ptpar([1.1, Pdef(\vuTotalAlea2), 1.9, Pdef(\vaTotalAlea2), 2.3,
                Pdef(\saTotalAlea2), 0.5, Pdef(\vaNum7a12Rate8), 4.9, Pdef(
                \saNum7a12Rate8), 8.1, Pdef(\vuNum7a12Rate8)]),
259 ]).play;)
```

Código A.1 – Código completo do estudo 1 - Transformações algorítmicas contínuas e combinadas

ANEXO B – Código SuperCollider : estratégias de ataque-continuação em uma nota orquestral

```
(SynthDef(\tocarEnv, {
        |out=0, pan=0, buf=0, rate=1.0, amp=0.6, pos=0.0, rel=1.0, atq
           =0.0001|
        var sig, env;
        env = EnvGen.ar(Env.perc(atq,rel), doneAction:2);
        sig = PlayBuf.ar(2, buf, BufRateScale.ir(buf) * rate, 1,
                BufFrames.kr(buf) * pos);
        OffsetOut.ar(out, Pan2.ar(sig, pan, amp) * env);
}).add;
SynthDef(\tocarEnvMono, {
        |out=0, pan=0, buf=0, rate=1.0, amp=0.6, pos=0.0, rel=1.0, atq
           =0.0001|
        var sig, env;
        env = EnvGen.ar(Env.perc(atq,rel), doneAction:2);
        sig = PlayBuf.ar(1, buf, BufRateScale.ir(buf) * rate, 1,
                BufFrames.kr(buf) * pos);
        OffsetOut.ar(out, Pan2.ar(sig, pan, amp) * env);
}).add;
SynthDef(\tocar, {
        |out=0, pan=0, buf=0, rate=1.0, amp=0.6, pos=0.0, rel=1.0, loop=0|
        var sig;
        sig = PlayBuf.ar(1, buf, BufRateScale.ir(buf) * rate, 1,
                BufFrames.kr(buf) * pos, loop, 2);
        OffsetOut.ar(out, Pan2.ar(sig, pan, amp));
}).add;
SynthDef(\tocarST, {
        |out=0, pan=0, buf=0, rate=1.0, amp=0.6, pos=0.0, rel=1.0, loop=0|
        var sig;
        sig = PlayBuf.ar(2, buf, BufRateScale.ir(buf) * rate, 1,
                BufFrames.kr(buf) * pos, loop, 2);
        OffsetOut.ar(out, Pan2.ar(sig, pan, amp));
}).add;
SynthDef(\estaveisST, {
        |out=0, pan=0, buf=0, rate=1.0, amp=0.6, pos=0.0, rel=1.0, loop=0,
           limiar=0.1|
        var sig, chain, chOut;
        sig = PlayBuf.ar(2, buf, BufRateScale.ir(buf) * rate, 1,
                BufFrames.kr(buf) * pos, loop, 2);
```

```
36          chain = FFT(LocalBuf(2048), sig);
37      chain = PV_PartialSynthP(chain, limiar,4);
38      chOut = IFFT(chain).dup;
39          OffsetOut.ar(out, Pan2.ar(chOut, pan, amp));
40  }).add;
41  SynthDef(\deslocSpec, { arg out=0, buf = b, amp=1.0, pan=0, atq=0.0001, sus
        =1.3, rel=0.7, desloc=0;
42      var in, chain, env;
43          env = EnvGen.ar(Env.linen(atq,sus,rel,amp, \lin), doneAction:2);
44      in = PlayBuf.ar(1, buf, BufRateScale.kr(buf));
45      chain = FFT(LocalBuf(4096), in);
46          chain = PV_BinShift(chain, 1.0, desloc);
47      Out.ar(out, Pan2.ar(IFFT(chain)*env,pan));
48  }).add;
49  )
50
51  (
52  c = WavesetsEvent.read("~\Audios\C3-marimba-mono.wav");//696
53  c.add(\marimba);
54  c = WavesetsEvent.read("~\Audios\C3-violino-mono.wav");//11764
55  c.add(\violino);
56  c = WavesetsEvent.read("~\Audios\C3-trompete-mono.wav");//4848
57  c.add(\trompete);
58  c = WavesetsEvent.read("~\Audios\C3-trompa-mono.wav");//11212
59  c.add(\trompa);
60  c = WavesetsEvent.read("~\Audios\C3-piano-mono.wav");//943
61  c.add(\piano);
62  c = WavesetsEvent.read("~\Audios\C3-fagote-mono.wav");//1664
63  c.add(\fagoteC3);
64  c = WavesetsEvent.read("~\Audios\C1_natural_fgt_dec_mono.wav");//4197
65  c.add(\fagoteC1);
66  WavesetsEvent.prepareSynthDefs;
67  )
68
69  (
70  ~sa1 = Buffer.read(s, "~\Audios\subs_atq_mrb_vln.wav");
71  ~sa2 = Buffer.read(s, "~\Audios\subs_atq_mrb_pno.wav");
72  ~sa3 = Buffer.read(s, "~\Audios\subs_atq_mrb_tru.wav");
73  ~sa4 = Buffer.read(s, "~\Audios\subs_atq_mrb_fgt.wav");
74  ~sa5 = Buffer.read(s, "~\Audios\subs_atq_mrb_tro.wav");
75  ~oq1 = Buffer.read(s, "~\Audios\C3-violino-stereo.wav");
76  ~oq2 = Buffer.read(s,"~\Audios\C3-fagote-stereo.wav");
77  ~oq3 = Buffer.read(s, "~\Audios\C3-piano-stereo.wav");
78  ~oq4 = Buffer.read(s, "~\Audios\C3-trompa-stereo.wav");
79  ~oq5 = Buffer.read(s, "~\Audios\C3-trompete-stereo.wav");
80  ~oq6 = Buffer.read(s, "~\Audios\C3-marimba-stereo.wav");
81  ~fgt = Buffer.read(s,"~\Audios\C1_natural_fgt_sust_mono.wav")
```

```
)

(
Pdef(\piano142,
        Ppar([
                Pwavesets(
                        Pbind(
                                \name, \piano, // identificação do buffer a ser utilizado
                                \start, Pn(Pseries(0, 1, 943),1), // toca o buffer sem saltos e até o final
                                \num, 1, // quantidade de waveset que será lida do buffer em um evento
                                \repeats, 1, //quantas vezes irá tocar cada waveset
                                \rate, 0.5, // velocidade de leitura
                                \amp, 0.8, //amplitude
                                \useFrac, false,
                        )
                ),
                Pwavesets(
                        Pbind(
                                \name, \piano, // identificação do buffer a ser utilizado
                                \start, Pn(Pseries(0, 1, 943),1), // toca o buffer sem saltos e até o final
                                \num, 1, // quantidade de waveset que será lida do buffer em um evento
                                \repeats, 4, //quantas vezes irá tocar cada waveset
                                \rate, 2.0, // velocidade de leitura
                                \amp, 0.8, //amplitude
                                \useFrac, false,
                        )
                )
        ])
);
Pdef(\piano152,
        Ppar([
                Pwavesets(
                        Pbind(
                                \name, \piano, // identificação do buffer a ser utilizado
                                \start, Pn(Pseries(0, 1, 943),1), // toca o buffer sem saltos e até o final
                                \num, 1, // quantidade de waveset que será lida do buffer em um evento
```

```
118                                     \repeats, 1, //quantas vezes irá tocar cada
                                            waveset
119                                     \rate, 1.0, // velocidade de leitura
120                                     \amp, 0.8, //amplitude
121                                     \useFrac, false,
122                             )
123                     ),
124                     Pwavesets(
125                             Pbind(
126                                     \name, \piano, // identificação do buffer a
                                            ser utilizado
127                                     \start, Pn(Pseries(0, 1, 943),1), // toca o
                                            buffer sem saltos e até o final
128                                     \num, 1, // quantidade de waveset que será
                                           lida do buffer em um evento
129                                     \repeats, 5, //quantas vezes irá tocar cada
                                            waveset
130                                     \rate, 2.0, // velocidade de leitura
131                                     \amp, 0.8, //amplitude
132                                     \useFrac, false,
133                             )
134                     )
135             ])
136 );
137 //////////////////////
138 //////////////////////
139 //////////////////////
140 Pdef(\violino13lento,
141             Ppar([
142                     Pwavesets(
143                             Pbind(
144                                     \name, \violino, // identificação do buffer
                                            a ser utilizado
145                                     \start, Pn(Pseries(0, 1, 11764),1), // toca
                                            o buffer sem saltos e até o final
146                                     \num, 3, // quantidade de waveset que será
                                           lida do buffer em um evento
147                                     \repeats, 1, //quantas vezes irá tocar cada
                                            waveset
148                                     \rate, 1.0, // velocidade de leitura
149                                     \amp, 0.8, //amplitude
150                                     \useFrac, false,
151                             )
152                     ),
153                     Pwavesets(
154                             Pbind(
```

```
155                                     \name, \violino, // identificação do buffer
                                            a ser utilizado
156                                     \start, Pn(Pseries(0, 1, 11764),1), // toca
                                            o buffer sem saltos e até o final
157                                     \num, 3, // quantidade de waveset que será
                                           lida do buffer em um evento
158                                     \repeats, 3, //quantas vezes irá tocar cada
                                            waveset
159                                     \rate, 3.0, // velocidade de leitura
160                                     \amp, 0.8, //amplitude
161                                     \useFrac, false,
162                             )
163                     )
164             ])
165 );
166 Pdef(\violino13,
167             Ppar([
168                     Pwavesets(
169                             Pbind(
170                                     \name, \violino, // identificação do buffer
                                            a ser utilizado
171                                     \start, Pn(Pseries(0, 1, 11764),1), // toca
                                            o buffer sem saltos e até o final
172                                     \num, 2, // quantidade de waveset que será
                                           lida do buffer em um evento
173                                     \repeats, 1, //quantas vezes irá tocar cada
                                            waveset
174                                     \rate, 1.0, // velocidade de leitura
175                                     \amp, 0.8, //amplitude
176                                     \useFrac, false,
177                             )
178                     ),
179                     Pwavesets(
180                             Pbind(
181                                     \name, \violino, // identificação do buffer
                                            a ser utilizado
182                                     \start, Pn(Pseries(0, 1, 11764),1), // toca
                                            o buffer sem saltos e até o final
183                                     \num, 2, // quantidade de waveset que será
                                           lida do buffer em um evento
184                                     \repeats, 3, //quantas vezes irá tocar cada
                                            waveset
185                                     \rate, 3.0, // velocidade de leitura
186                                     \amp, 0.8, //amplitude
187                                     \useFrac, false,
188                             )
189                     )
```

```
190         ])
191 );
192 Pdef(\violino14,
193         Ppar([
194                 Pwavesets(
195                         Pbind(
196                                 \name, \violino, // identificação do buffer
                                        a ser utilizado
197                                 \start, Pn(Pseries(0, 1, 11764),1), // toca
                                        o buffer sem saltos e até o final
198                                 \num, 1, // quantidade de waveset que será
                                       lida do buffer em um evento
199                                 \repeats, 1, //quantas vezes irá tocar cada
                                        waveset
200                                 \rate, 1.0, // velocidade de leitura
201                                 \amp, 0.8, //amplitude
202                                 \useFrac, false,
203                         )
204                 ),
205                 Pwavesets(
206                         Pbind(
207                                 \name, \violino, // identificação do buffer
                                        a ser utilizado
208                                 \start, Pn(Pseries(0, 1, 11764),1), // toca
                                        o buffer sem saltos e até o final
209                                 \num, 1, // quantidade de waveset que será
                                       lida do buffer em um evento
210                                 \repeats, 4, //quantas vezes irá tocar cada
                                        waveset
211                                 \rate, 4.0, // velocidade de leitura
212                                 \amp, 0.8, //amplitude
213                                 \useFrac, false,
214                         )
215                 )
216         ])
217 );
218 ////////////////////
219 ////////////////////
220 ////////////////////
221 Pdef(\inter7PianoFagoteC1,
222         Pwavesets(
223         Pbind(
224                 \name, Pseq([Pn(\piano, 7), Pn(\fagoteC1,7)],inf),
225                 \start, Pswitch1(
226                         [Pseries(0, 1, 943), Pseries(0, 1, 943)],
227                         Pseq([Pn(0,7),        Pn(1,7)], inf)),
228                 \num, 1,
```

```
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf), //
                \useFrac, false,
        )
)
);
Pdef(\inter7MarimbaFagoteC1,
        Pwavesets(
        Pbind(
                \name, Pseq([Pn(\marimba, 7), Pn(\fagoteC1,7)],inf),
                \start, Pswitch1(
                        [Pseries(0, 1, 696), Pseries(0, 1, 696)],
                        Pseq([Pn(0,7),        Pn(1,7)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf),
                \useFrac, false,
        )
)
);
Pdef(\inter16MarimbaFagoteC1,
        Pwavesets(
        Pbind(
                \name, Pseq([Pn(\marimba, 16), Pn(\fagoteC1,16)],inf),
                \start, Pswitch1(
                        [Pseries(0, 1, 696), Pseries(0, 1, 696)],
                        Pseq([Pn(0,16),        Pn(1,16)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf),
                \useFrac, false,
        )
)
);
Pdef(\inter32MarimbaFagoteC1,
        Pwavesets(
        Pbind(
                \name, Pseq([Pn(\marimba, 32), Pn(\fagoteC1,32)],inf),
                \start, Pswitch1(
                        [Pseries(0, 1, 696), Pseries(0, 1, 696)],
                        Pseq([Pn(0,32),        Pn(1,32)], inf)),
```

```
276                 \num, 1,
277                 \repeats, 1,
278                 \rate, 1.0,
279                 \pan, Pbrown(-1.0, 1.0, 0.01),
280                 \amp, Pseq([0.6, 0.6],inf),
281                 \useFrac, false,
282         )
283 )
284 );
285 Pdef(\inter64MarimbaFagoteC1,
286         Pwavesets(
287         Pbind(
288                 \name, Pseq([Pn(\marimba, 64), Pn(\fagoteC1, 64)],inf),
289                 \start, Pswitch1(
290                         [Pseries(0, 1, 696), Pseries(0, 1, 696)],
291                         Pseq([Pn(0,64),         Pn(1,64)], inf)),
292                 \num, 1,
293                 \repeats, 1,
294                 \rate, 1.0,
295                 \pan, Pbrown(-1.0, 1.0, 0.01),
296                 \amp, Pseq([0.6, 0.6],inf),
297                 \useFrac, false,
298         )
299 )
300 );
301 Pdef(\inter7FagoteC3FagoteC1,
302         Pwavesets(
303         Pbind(
304                 \name, Pseq([Pn(\fagoteC3, 7), Pn(\fagoteC1,7)],inf),
305                 \start, Pswitch1(
306                         [Pseries(0, 1, 1664), Pseries(0, 1, 1664)],
307                         Pseq([Pn(0,7),         Pn(1,7)], inf)),
308                 \num, 1,
309                 \repeats, 1,
310                 \rate, 1.0,
311                 \pan, Pbrown(-1.0, 1.0, 0.01),
312                 \amp, Pseq([0.6, 0.6],inf),
313                 \useFrac, false,
314         )
315 )
316 );
317 Pdef(\inter16FagoteC3FagoteC1,
318         Pwavesets(
319         Pbind(
320                 \name, Pseq([Pn(\fagoteC3, 16), Pn(\fagoteC1,16)],inf),
321                 \start, Pswitch1(
322                         [Pseries(0, 1, 1664), Pseries(0, 1, 1664)],
```

```
323                         Pseq([Pn(0,16),         Pn(1,16)], inf)),
324                 \num, 1,
325                 \repeats, 1,
326                 \rate, 1.0,
327                 \pan, Pbrown(-1.0, 1.0, 0.01),
328                 \amp, Pseq([0.6, 0.6],inf),
329                 \useFrac, false,
330         )
331 )
332 );
333 Pdef(\inter32FagoteC3FagoteC1,
334         Pwavesets(
335         Pbind(
336                 \name, Pseq([Pn(\fagoteC3, 32), Pn(\fagoteC1,32)],inf),
337                 \start, Pswitch1(
338                         [Pseries(0, 1, 1664), Pseries(0, 1, 1664)],
339                         Pseq([Pn(0,32),         Pn(1,32)], inf)),
340                 \num, 1,
341                 \repeats, 1,
342                 \rate, 1.0,
343                 \pan, Pbrown(-1.0, 1.0, 0.01),
344                 \amp, Pseq([0.6, 0.8],inf),
345                 \useFrac, false,
346         )
347 )
348 );
349 Pdef(\inter64FagoteC3FagoteC1,
350         Pwavesets(
351         Pbind(
352                 \name, Pseq([Pn(\fagoteC3, 64), Pn(\fagoteC1, 64)],inf),
353                 \start, Pswitch1(
354                         [Pseries(0, 1, 696), Pseries(0, 1, 696)],
355                         Pseq([Pn(0,64),         Pn(1,64)], inf)),
356                 \num, 1,
357                 \repeats, 1,
358                 \rate, 1.0,
359                 \pan, Pbrown(-1.0, 1.0, 0.01),
360                 \amp, Pseq([0.7, 0.6],inf),
361                 \useFrac, false,
362         )
363 )
364 );
365 Pdef(\inter64ViolinoTrompete,
366         Pwavesets(
367         Pbind(
368                 \name, Pseq([Pn(\violino, 64), Pn(\trompete, 64)],inf),
369                 \start, Pswitch1(
```

```
                        [Pseries(0, 1, 4848), Pseries(0, 1, 4848)],
                        Pseq([Pn(0,64),        Pn(1,64)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.5, 0.6],inf),
                \useFrac, false,
        )
)
);
Pdef(\inter32ViolinoTrompete,
        Pwavesets(
        Pbind(
                \name, Pseq([Pn(\violino, 32), Pn(\trompete, 32)],inf),
                \start, Pswitch1(
                        [Pseries(0, 1, 4848), Pseries(0, 1, 4848)],
                        Pseq([Pn(0,32),        Pn(1,32)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf),
                \useFrac, false,
        )
)
);
Pdef(\inter12ViolinoTrompete,
        Pwavesets(
        Pbind(
                \name, Pseq([Pn(\violino, 12), Pn(\trompete, 12)],inf),
                \start, Pswitch1(
                        [Pseries(0, 1, 4848), Pseries(0, 1, 4848)],
                        Pseq([Pn(0,12),        Pn(1,12)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf),
                \useFrac, false,
        )
)
);
Pdef(\inter6ViolinoTrompete,
        Pwavesets(
        Pbind(
                \name, Pseq([Pn(\violino, 6), Pn(\trompete, 6)],inf),
```

```
                \start, Pswitch1(
                        [Pseries(0, 1, 4848), Pseries(0, 1, 4848)],
                        Pseq([Pn(0,6),        Pn(1,6)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf),
                \useFrac, false,
        )
)
);
Pdef(\inter3ViolinoTrompete,
        Pwavesets(
        Pbind(
                \name, Pseq([Pn(\violino, 3), Pn(\trompete, 3)],inf),
                \start, Pswitch1(
                        [Pseries(0, 1, 4848), Pseries(0, 1, 4848)],
                        Pseq([Pn(0,3),        Pn(1,3)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf),
                \useFrac, false,
        )
)
);
Pdef(\inter64TrompaFagoteC3,
Pwavesets(
        Pbind(
                \name, Pseq([Pn(\trompa, 64), Pn(\fagoteC3,64)],inf),
                \start, Pswitch1(
                        [Pseries(0, 1, 943), Pseries(0, 1, 943)],
                        Pseq([Pn(0,64),        Pn(1,64)], inf)),
                \num, 1,
                \repeats, 1,
                \rate, 1.0,
                \pan, Pbrown(-1.0, 1.0, 0.01),
                \amp, Pseq([0.6, 0.6],inf), //amplitude
                \useFrac, false,
        )
)
);
Pdef(\inter32TrompaFagoteC3,
Pwavesets(
        Pbind(
```

```
464                 \name, Pseq([Pn(\trompa, 32), Pn(\fagoteC3,32)],inf),
465                 \start, Pswitch1(
466                         [Pseries(0, 1, 943), Pseries(0, 1, 943)],
467                         Pseq([Pn(0,32),         Pn(1,32)], inf)),
468                 \num, 1,
469                 \repeats, 1,
470                 \rate, 1.0,
471                 \pan, Pbrown(-1.0, 1.0, 0.01),
472                 \amp, Pseq([0.6, 0.6],inf), //amplitude
473                 \useFrac, false,
474         )
475 )
476 );
477 Pdef(\inter64PianoMarimba,
478         Pwavesets(
479         Pbind(
480                 \name, Pseq([Pn(\piano, 64), Pn(\marimba,64)],inf), //
                        identificação do buffer a ser utilizado
481                 \start, Pswitch1(
482                         [Pseries(0, 1, 696), Pseries(0, 1, 696)],
483                         Pseq([Pn(0,64),         Pn(1,64)], inf)), // toca o
                            buffer sem saltos e até o final
484                 \num, 1, // quantidade de waveset que será lida do buffer
                        em um evento
485                 \repeats, 1, //quantas vezes irá tocar cada waveset
486                 \rate, 1.0, // velocidade de leitura
487                 \pan, Pbrown(-1.0, 1.0, 0.01),
488                 \amp, Pseq([0.6, 0.6],inf), //amplitude
489                 \useFrac, false,
490         )
491 )
492 );
493
494 Pdef(\inter32PianoMarimba,
495         Pwavesets(
496         Pbind(
497                 \name, Pseq([Pn(\piano, 32), Pn(\marimba,32)],inf), //
                        identificação do buffer a ser utilizado
498                 \start, Pswitch1(
499                         [Pseries(0, 1, 943), Pseries(0, 1, 943)],
500                         Pseq([Pn(0,32),         Pn(1,32)], inf)), // toca o
                            buffer sem saltos e até o final
501                 \num, 1, // quantidade de waveset que será lida do buffer
                        em um evento
502                 \repeats, 1, //quantas vezes irá tocar cada waveset
503                 \rate, 1.0, // velocidade de leitura
504                 \amp, Pseq([0.6, 0.7],inf), //amplitude
```

```
505                 \useFrac, false,
506         )
507 )
508 );
509
510 )
511
512
513
514 (
515 Pseq([
516         //primeira seção
517         Ptpar([
518                 0.0, Pbind(\instrument, \tocar, \buf, Pxrand([~sa1, ~sa2,
                        ~sa3, ~sa4, ~sa5], inf), \dur, Pgeom(2, 0.95, 200).clip
                        (1/32,2), \amp, Pgeom(0.6, 0.985, 200).clip(0.25,0.6),
                        \pan, -0.5),
519                 1.0, Pbind(\instrument, \tocarST, \buf, Pxrand([[~oq1, ~oq2
                        ], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3], [~oq1, ~oq4
                        ,], [~oq3, ~oq5], [~oq1, ~oq3,]], inf), \dur, Pgeom(2,
                        0.95, 200).clip(1/32,2), \amp, Pgeom(0.6, 0.985, 200).
                        clip(0.25,0.6), \pan, 0.75)
520         ]),
521
522         //segunda seção ressintese aditiva + estratégia de repodução em
                oitavas ou desafinadas.
523         Pdef(\piano142),
524         Pbind(\instrument, \tocarST, \buf, Pxrand([[~oq1, ~oq2], [~oq3,
                ~oq4], [~oq4, ~oq5], [~oq2, ~oq3], [~oq1, ~oq4,], [~oq3, ~oq5],
                [~oq1, ~oq3,]], inf), \dur, Pseries(1/32, 1/24, 12).clip(1/32,1)
                , \rate, Prand([0.5, 1.0, 2.0], 12), \amp, 0.6),
525         Pdef(\piano152),
526         Pbind(\instrument, \tocarST, \buf, Pxrand([[~oq1, ~oq2], [~oq3,
                ~oq4], [~oq4, ~oq5], [~oq2, ~oq3], [~oq1, ~oq4,], [~oq3, ~oq5],
                [~oq1, ~oq3,]], inf), \dur, Pseries(1/32, 1/24, 12).clip(1/32,1)
                , \rate, Prand([0.5, 1.0, 2.0], 12), \amp, 0.6),
527         Pdef(\violino13),
528         Ppar([Pbind(\instrument, \tocarST, \buf, Pxrand([[~oq1, ~oq2], [
                ~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3], [~oq1, ~oq4,], [~oq3,
                ~oq5], [~oq1, ~oq3,]], inf), \dur, Pseries(1/32, 1/24, 12).clip
                (1/32,1), \rate, Prand([0.99, 1.0, 1.01, 0.98, 1.02], 12), \amp,
                 0.3), Pdef(\violino13lento)]),
529
530         //Terceira seção desafinação e sustentação espectral:
531
532         Pseq([
533                 //chorus+oitavas
```

```
534                 Ppar([Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1,
                        ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3], [~oq1,
                        ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]], inf), \dur,
                        Pseries(1/32, 1/24, 20).clip(1/32,1), \rate, Prand([0.5,
                         1.0, 0.502, 2, 2.02], 20), \amp, Pwhite(0.3,0.4),
                        \limiar, 6, \pan, Pwhite(-1.0, 1.0)),
535                         Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1
                                , ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3
                                ], [~oq1, ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]],
                                inf), \dur, Pseries(1/32, 1/24, 20).clip(1/32,1)
                                , \rate, Prand([0.5, 1.0, 0.502, 2, 2.02], 20),
                                \amp, Pwhite(0.3,0.4), \limiar, 6, \pan, Pwhite
                                (-1.0, 1.0)),
536                         Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1
                                , ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3
                                ], [~oq1, ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]],
                                inf), \dur, Pseries(1/32, 1/24, 20).clip(1/32,1)
                                , \rate, Prand([0.5, 1.0, 0.502, 2, 2.02], 20),
                                \amp, Pwhite(0.3,0.4), \limiar, 6, \pan, Pwhite
                                (-1.0, 1.0)),
537                 ]),
538
539                 //somente chorus
540                 Ppar([
541                         Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1
                                , ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3
                                ], [~oq1, ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]],
                                inf), \dur, Pseries(1/32, 1/24, 20).clip(1/32,1)
                                , \rate, Prand([0.997, 1.0, 1.007, 0.995,
                                1.005], 20), \amp, Pwhite(0.3,0.4), \limiar,
                                0.68, \pan, Pwhite(-1.0, 1.0)),
542                         Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1
                                , ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3
                                ], [~oq1, ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]],
                                inf), \dur, Pseries(1/32, 1/20, 20).clip(1/32,1)
                                , \rate, Prand([0.998, 1.0, 1.002, 0.999,
                                1.004], 20), \amp, Pwhite(0.3,0.4), \limiar,
                                0.68, \pan, Pwhite(-1.0, 1.0)),
543                         Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1
                                , ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3
                                ], [~oq1, ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]],
                                inf), \dur, Pseries(1/32, 1/18, 20).clip(1/32,1)
                                , \rate, Prand([0.999, 1.0, 1.001, 0.996,
                                1.003], 20), \amp, Pwhite(0.3,0.4), \limiar,
                                0.68, \pan, Pwhite(-1.0, 1.0))
544                 ]),
545
```

```
546                 //chorus+oitavas
547                 Ppar([Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1,
                        ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3], [~oq1,
                        ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]], inf), \dur,
                        Pseries(1/32, 1/24, 20).clip(1/32,1), \rate, Prand([0.5,
                         1.0, 0.502, 2, 2.02], 20), \amp, Pwhite(0.3,0.4),
                        \limiar, 0.14, \pan, Pwhite(-1.0, 1.0)),
548                         Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1
                            , ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3
                            ], [~oq1, ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]],
                            inf), \dur, Pseries(1/32, 1/24, 20).clip(1/32,1)
                            , \rate, Prand([0.5, 1.0, 0.502, 2, 2.02], 20),
                            \amp, Pwhite(0.3,0.4), \limiar, 0.14, \pan,
                            Pwhite(-1.0, 1.0)),
549                         Pbind(\instrument, \estaveisST, \buf, Pxrand([[~oq1
                            , ~oq2], [~oq3, ~oq4], [~oq4, ~oq5], [~oq2, ~oq3
                            ], [~oq1, ~oq4,], [~oq3, ~oq5], [~oq1, ~oq3,]],
                            inf), \dur, Pseries(1/32, 1/24, 20).clip(1/32,1)
                            , \rate, Prand([0.5, 1.0, 0.502, 2, 2.02], 20),
                            \amp, Pwhite(0.3,0.4), \limiar, 0.14, \pan,
                            Pwhite(-1.0, 1.0)),
550         ])]
551         ),
552         //Quarta seção som de fagote que vai sendo alternado e
                inharmonicizado em paralelo com ataques de strum (deslocamento
                espectral)
553         Ptpar([
554                 0.0, Pbind(\instrument, \deslocSpec, \buf, ~fgt, \desloc,
                        Ptuple([Pseq([Pwhite(-12,12,1),0],8),0]), \dur, 2, \amp,
                         [0.6, 0.8]),
555                 3.0, Pbind(\instrument, \tocarST, \buf, Pfunc({[~oq1, ~oq2,
                         ~oq3, ~oq4, ~oq5, ~oq6].scramble}), \strum, Pwhite
                        (-0.1, 0.1), \dur, 2, \rate, 1, \amp, Pbrown(0.3, 0.5,
                        0.05, 10))
556         ]),
557         //ponte
558         Pbind(\instrument, \tocarEnvMono, \buf, ~fgt, \dur, 2, \rel, 6.0,
                \amp, Pn(0.9, 4)),
559 //Quinta secao - intercambio de conjuntos de onda
560         Pspawner({ | sp |
561                 sp.wait(0.2);
562                 sp.seq(Pdef(\inter64TrompaFagoteC3) );
563                 sp.wait(0.9);
564                 sp.seq(Pdef(\inter64FagoteC3FagoteC1) );
565                 sp.wait(0.4);
566                 sp.seq(Pdef(\inter32TrompaFagoteC3) );
567                 sp.wait(0.4);
```

```
568                 sp.seq(Pdef(\inter32FagoteC3FagoteC1) );
569                 sp.wait(0.2);
570                 sp.seq(Pdef(\inter64PianoMarimba) );
571                 sp.wait(0.1);
572                 sp.seq(Pdef(\inter3ViolinoTrompete) );
573                 10.do {
574                         sp.par(
575                                 Pdef(\inter64PianoMarimba)
576                         );
577                         sp.wait(rrand(0.2,1.0))
578                 };
579                 10.do {
580                         sp.par(
581                                 Pdef(\inter7PianoFagoteC1)
582                         );
583
584                         sp.wait(0.3)
585                 };
586                 5.do {
587                         sp.par(
588                                 Pdef(\inter3ViolinoTrompete)
589                         );
590                         sp.wait(rrand(0.9,1.9));
591                         sp.par(
592                                 Pdef(\inter6ViolinoTrompete)
593                         );
594                         sp.wait(rrand(0.9,1.9));
595                         sp.par(
596                                 Pdef(\inter32ViolinoTrompete)
597                         );
598                         sp.wait(rrand(0.9,1.9));
599                         sp.par(
600                                 Pdef(\inter64ViolinoTrompete)
601                         );
602                         sp.wait(rrand(0.9,1.9));
603                 };
604                 sp.wait(0.1);
605                 sp.suspendAll;
606         }),
607 Ppar([
608         Pbind(
609                 \instrument, \tocarST,
610                 \buf, ~oq1,
611                 \dur, Pseg( Pseq([1/128, 1/4],inf), Pseq([1.5, 2.9],inf),
                        \welch),
612                 \pan, Pbrown(-1.0, 1.0, 0.01, 400),
613                 \amp, Pbrown(0.02, 0.3, 0.001),
```

```
614         ),
615         Pbind(
616                 \instrument, \tocarST,
617                 \buf, ~oq2,
618                 \dur, Pseg( Pseq([1, 1/64],inf), Pseq([2],inf), \welch),
619                 \pan, Pbrown(-1.0, 1.0, 0.01, 120),
620                 \amp, Pbrown(0.02, 0.3, 0.001),
621         ),
622         Pbind(
623                 \instrument, \tocarST,
624                 \buf, ~oq3,
625                 \dur, Pseg( Pseq([1/2, 1/128],inf), Pseq([2, 1],inf), \sin)
                        ,
626                 \pan, Pbrown(-1.0, 1.0, 0.01, 400),
627                 \amp, Pbrown(0.02, 0.3, 0.001),
628         ),
629         Pbind(
630                 \instrument, \tocarST,
631                 \buf, ~oq4,
632                 \dur, Pseg( Pseq([1/2, 1/128],inf), Pseq([3, 2],inf),
                        \welch),
633                 \pan, Pbrown(-1.0, 1.0, 0.01, 300),
634                 \amp, Pbrown(0.02, 0.3, 0.001),
635         ),
636         Pbind(
637                 \instrument, \tocarST,
638                 \buf, ~oq5,
639                 \dur, Pseg( Pseq([1/128, 1/4],inf), Pseq([2.5, 3.9],inf),
                        \sin),
640                 \pan, Pbrown(-1.0, 1.0, 0.01, 400),
641                 \amp, Pbrown(0.02, 0.3, 0.001),
642         ),
643         Pbind(
644                 \instrument, \tocarST,
645                 \buf, ~oq6,
646                 \dur, Pseg( Pseq([1/128, 1/4],inf), Pseq([4.5, 4.9],inf),
                        \welch),
647                 \pan, Pbrown(-1.0, 1.0, 0.01, 200),
648                 \amp, Pbrown(0.02, 0.3, 0.001),
649         ),
650 ])
651
652 ]).play;
653 )
```

Código B.1 – Estratégias de ataque-continuação em uma nota orquestral

ANEXO C – Código SuperCollider : processos adaptativos e ressonificação

```
l = Buffer.read(s, "~\Audios\escola-samba-l-trim1.wav");
r = Buffer.read(s, "~\Audios\escola-samba-r-trim1.wav");

({
        var sigL, chainL, chain2L, onsetsL, sigR, chainR, chain2R, chain3L,
             chain3R, chain4L, chain4R, chain5L, chain5R, chain6L, chain6R,
            chain7L, chain7R, onsetsR, trig16, trig32, trig128;

        //Triggers baseados na subdivisão proporcional da duração do áudio
        trig16 = Impulse.kr(((BufDur.ir(l)/16).reciprocal));
        trig32 = Impulse.kr(((BufDur.ir(l)/32).reciprocal));
        trig128 = Impulse.kr(((BufDur.ir(l)/128).reciprocal));

        // Sinais esquerdo e direito de uma gravação
        sigL = PlayBuf.ar(1, l, BufRateScale.kr(l), loop: 1);
        sigR = PlayBuf.ar(1, r, BufRateScale.kr(r), loop: 1);

        //Conversão dos sinais de áudio para o domínio da frequência
        chainL = FFT(LocalBuf(4096), sigL);
        chain2L = FFT(LocalBuf(4096), sigL);
        chain7L = FFT(LocalBuf(256), sigL);
        chainR = FFT(LocalBuf(4096), sigR);
        chain2R = FFT(LocalBuf(4096), sigR);
        chain7R = FFT(LocalBuf(512), sigR);

        //Extai os onsets do arquivo de áudio
        onsetsL = Onsets.kr(chain2L, 0.1, \rcomplex, 0.5);
        onsetsR = Onsets.kr(chain2R, 0.4, \rcomplex, 0.25);

        //realiza as transformações espectrais
        chainL = PV_MagFreeze(chainL, onsetsL < 0.9);
        chainR = PV_MagFreeze(chainR, onsetsR < 0.9);

        chain3L = PV_BinScramble(chain2L, Demand.kr(trig32,0,Dwhite(0.0,
            0.5)) , 0.2, trig32);
        chain3R = PV_BinScramble(chain2R, Demand.kr(trig32,0,Dwhite(0.0,
            0.25)) , 0.4, trig32);

        chain4L = PV_MagSmear(chain2L, Demand.kr(trig128,0,Dbrown(100,
            230,10)));
```

```
36          chain4R = PV_MagSmear(chain2R, Demand.kr(trig128,0,Dbrown(25,
                180,10)));
37
38          chain5L = PV_MagSmooth(chain2L, Demand.kr(onsetsL,0,Dbrown(0.1,
                1.0, 0.1)));
39          chain5R = PV_MagSmooth(chain2R, Demand.kr(onsetsR,0,Dbrown(0.1,
                1.0, 0.1)));
40
41          chain6L = PV_PartialSynthP(chain5L, Demand.kr(trig16,0,Dbrown(0.10,
                 0.64, 0.05)));
42          chain6R = PV_PartialSynthP(chain5R, Demand.kr(trig16,0,Dbrown(0.10,
                 0.64, 0.05)));
43
44          chain7L = PV_Freeze(chain7L, trig128<0.5);
45          chain7R = PV_Freeze(chain7R, onsetsR<0.5);
46
47          OffsetOut.ar(0,
48                  Select.ar( // seleciona qual chain, logo qual PV, será
                        tocado
49                          Demand.kr(
50                                  Impulse.kr(((BufDur.ir(1)*2).reciprocal)),
51                                  0,
52                                  Dxrand((0..6),inf)), //Escolhe sem repetir
                                        o último
53
54                          //IFFT ressintetiza os sinais espectrais
55                          [
56                                  // original
57                                  [sigL, sigR],
58                                   // PV_MagFreeze
59                                  [IFFT(chainL), IFFT(chainR)],
60                                  // PV_BinScramble
61                                  [IFFT(chain3L), IFFT(chain3R)],
62                                  // PV_MagSmear
63                                  [IFFT(chain4L) * 1.4, IFFT(chain4R)*1.2],
64                                  // PV_MagSmooth
65                                  [IFFT(chain5L) * 1.4, IFFT(chain5R) * 1.4],
66                                  //PV_MagSmooth -> PV_PartialSynthP
67                                  [IFFT(chain6L) * 1.4, IFFT(chain6R)*1.2],
68                                  // PV_Freeze
69                                  [(IFFT(chain7L) *0.8) + (sigL*0.4) , (IFFT(
                                        chain7R)*0.8) + (sigR*0.4)]
70                          ]
71                  )
72          );
73  }.play;)
```

Código C.1 – Processos adaptativos e ressonificação