UNIVERSIDADE FEEVALE

GUSTAVO HENRIQUE STRASSBURGER

TRANSCRIÇÃO AUTOMÁTICA DE MÚSICA DE ÁUDIOS MONOFÔNICOS

Novo Hamburgo
2016

GUSTAVO HENRIQUE STRASSBURGER

TRANSCRIÇÃO AUTOMÁTICA DE MÚSICA DE ÁUDIOS MONOFÔNICOS

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Sistemas de Informação pela
Universidade Feevale

ORIENTADOR: GABRIEL SIMÕES

Novo Hamburgo
2016

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, aos meus pais por me proporcionarem este momento. Definitivamente, sem o apoio deles, não estaria passando por esta etapa. Quero também agradecer à minha irmã, por sempre me apoiar e exigir o máximo do meu potencial. Também desejo agradecer à minha namorada, pela parceria nessa etapa, assim como em todas as demais. Ao meu orientador, por todo o suporte e auxílio que tem me dado durante este período. Enfim, meus mais sinceros agradecimentos a todos que, de alguma forma, estiveram comigo durante esta etapa da vida.

RESUMO

O campo da transcrição automática de música tem crescido de forma contundente nos últimos anos, sendo, cada vez mais, objeto de estudo de artigos científicos, dissertações e teses. Em razão disto, novos métodos têm surgido a fim de solucionar as lacunas ainda existentes. Sendo objetivo da área transcrever sinais de áudio de música em notações de música conhecidas, as pesquisas, em grande parte, objetivam melhorar o processo de detecção de melodias, de tal forma que seja possível produzir softwares mais confiáveis e com menos restrições de uso. Sendo assim, este trabalho tem como objetivo estudar as principais etapas do processo de transcrição de música, sendo elas: detecção de *onsets* e *offsets*, estimativa da frequência fundamental do sinal e a conversão para uma notação conhecida, no caso deste trabalho, para o protocolo MIDI. Desta forma, este trabalho tem como objetivo desenvolver um protótipo de software capaz de transcrever um sinal monofônico, empregando as técnicas que apresentarem os melhores resultados em trabalhos relacionados.

Palavras-chave: Transcrição automática de música. Sinais de áudio monofônicos. Detecção de *onsets*. Detecção de *offsets*. Estimativa da frequência fundamental.

ABSTRACT

Automatic music transcription has been growing in the last few years, being object of study of many papers and theses, resulting in new methods to solve still remaining problems on the area. The main purpose of this field of study it is to transcribe onto a well-known musical notation. These studies presents enhancements on the process of transcribing music automatically, as they are improving existing methods or presenting new ones, so that they became more reliable and with less restrictions of use. This paper aims to present a prototype capable of transcribing monophonic music applying the methods with best results in co-related researches. It will be studied the main stages of the process: onset and offset detection, fundamental frequency estimation and the transcription onto a musical notation to enable this implementation. For the purpose of this study, the MIDI protocol will be used to present the outputs of this software.

Keywords: Automatic music transcription. Monophonic audio signals. Onset detection. Offset detection. Fundamental frequency estimation.

LISTA DE FIGURAS

Figura 1 – Etapas da execução de uma nota musical 14
Figura 2 – Fluxograma de um algoritmo comum de detecção de *onsets* 15
Figura 3 – Sinal no domínio de tempo (a). Função de janelamento (b). Magnitude (c) e fase (d) da STFT de duas janelas do sinal. 16
Figura 4 – Diagrama da fase do sinal ao longo de *frames* adjacentes. 19
Figura 5 – Diagrama fasorial demonstrando a variação entre vetor alvo e vetor medido, além da distância euclidiana entre ambos 21
Figura 6 - Diagrama da detecção de onsets baseada em pitch. 22
Figura 7 – Limiar dinâmico calculado sobre uma função de detecção gerada utilizando o algoritmo de domínio complexo 24
Figura 8 – Envelope do modelo de execução de uma nota de um violino (dedilhada) 25
Figura 9 – Função de autocorrelação 31
Figura 10 – Função da diferença quadrática 31
Figura 11 – Função da média normalizada acumulada da diferença quadrática 32
Figura 12 – Formato de onda de um áudio de um triângulo 34
Figura 13 – Formato de onda de áudio de guitarra com *legatos* 34
Figura 14 – Formato de onda de áudio sintetizado 35
Figura 15 – Código-fonte do método de diferença espectral do protótipo 37
Figura 16 – Formato de onda de áudio sintetizado e *onsets* detectados do trecho 38
Figura 17 – Código-fonte do método de desvio de fase do protótipo 39
Figura 18 – Código-fonte do cálculo do fluxo de uma janela no método de desvio de fase 40
Figura 19 – Código-fonte do método de diferença espectral na detecção de *offsets* 41
Figura 20 – Código-fonte do método de detecção dos picos a partir da média da função de detecção 42
Figura 21 – Código-fonte do método de detecção da nota por meio da autocorrelação 43
Figura 22 – Frequências de um piano de 88 teclas 44
Figura 23 – Código-fonte para conversão de frequências em notas na escala MIDI 45
Figura 24 – Modelo de visualização dos eventos extraídos de arquivo MIDI 47
Figura 25 –Visualização dos *onsets* detectados em comparativo com onda do sinal. 48

LISTA DE TABELAS

Tabela 1 – Exemplo de cálculo de diferença espectral para dois espectros 36
Tabela 2 – Exemplo de cálculo de desvio de fase para dois espectros..... 39

LISTA DE QUADROS

Quadro 1 – Conversão de frequências para escala MIDI 45

LISTA DE ABREVIATURAS E SIGLAS

ACF	Função de Autocorrelação
AMT	Transcrição Automática de Música
API	Interface de Programação de Aplicação
FN	Falso Negativo
MIDI	Interface Digital de Instrumentos Musicais
MIR	Recuperação de Informação de Música
STFT	Transformada de Curto Tempo de Fourier
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
WAV	Waveform Audio File Format

SUMÁRIO

INTRODUÇÃO **9**

1 TRANSCRIÇÃO AUTOMÁTICA DE MÚSICA **11**

2 DETECÇÃO DE EVENTOS **14**
2.1 Detecção de *onsets* **14**
2.1.1 Função de detecção 16
2.1.1.1 Diferença espectral *17*
2.1.1.2 Desvio de fase *18*
2.1.1.3 Domínio complexo *20*
2.1.1.4 Detecção baseada em pitch *22*
2.1.2 Detecção de picos 23
2.2 Detecção de *offsets* **24**

3 ESTIMATIVA DA FREQUÊNCIA FUNDAMENTAL **28**
3.1 Métodos **28**
3.1.1 Função de autocorrelação 28
3.1.2 YIN 30

4 PROTÓTIPO **33**
4.1 Manipulação de arquivos de áudio **33**
4.2 Detecção de eventos **35**
4.2.1 Detecção de *onsets* 35
4.2.1.1 Diferença espectral *36*
4.2.1.2 Desvio de fase *38*
4.2.2 Detecção de *offsets* 40
4.2.3 Detecção de picos 41
4.3 Estimativa da frequência fundamental **42**

5 RESULTADOS **47**

CONSIDERAÇÕES FINAIS **50**

REFERÊNCIAS BIBLIOGRÁFICAS **53**

INTRODUÇÃO

O crescimento da pesquisa na área da recuperação da informação da música (MIR, do inglês *Music Information Retrieval*) deve-se, sobretudo, em razão do aumento de disponibilidade de obras musicais em formato digital (SANTINI; SOUZA, 2007). Apesar de sua existência desde a década de 1960, apenas recentemente, com maior poder computacional e desenvolvimento de novos algoritmos, a área consegue produzir resultados satisfatórios (ELLIS, 2006).

O processo de conversão de sinais de áudio em notações musicais estabelecidas, estudado pela área da transcrição automática de música (AMT, do inglês *Automatic Music Transcription*) – subárea da MIR – ainda não está inteiramente solucionado. Atualmente, a transcrição de sinais monofônicos já pode ser considerada compreendida, no entanto, transcrever sinais polifônicos ou sem restrição dos instrumentos ainda requer maior aprofundamento (BENETOS et al., 2012).

Na transcrição de áudios monofônicos, somente uma nota pode ser tocada por vez, tendo em vista que podem haver múltiplos instrumentos na faixa, contanto que os sons não se sobreponham. Trabalhar apenas com sinais monofônicos limita o número de aplicações das ferramentas, no entanto, aumentam as chances de ser obtido um retorno confiável e rápido (BELLO; MONTI; SANDLER, 2000).

Ao contrário da transcrição de sinais monofônicos, transcrever áudios polifônicos exige detectar e identificar múltiplos *pitches* concorrentes que podem se sobrepor. Esta etapa permanece sendo o principal desafio a ser resolvido nesta área. Detecção de *onsets* e *offsets*, reconhecimento de instrumentos e extração de ritmo e tempo também são problemas a serem solucionados na área (BENETOS et al, 2013).

Apesar de demonstrarem um avanço significativo nos últimos anos, as áreas da transcrição de sinais monofônicos e, sobretudo, de sinais polifônicos, ainda dependem de resultados mais significativos. Assim, este estudo tem como foco de pesquisa a transcrição de sinais monofônicos, abordando todas as etapas do processo.

Portanto, o objetivo deste trabalho é desenvolver um protótipo de software capaz de interpretar arquivos de áudio e extrair os eventos de *onsets* e *offsets*, assim como as notas de tais eventos, convertidas para a escala MIDI. Os resultados desse protótipo serão avaliados a partir da análise da taxa de acertos e percentual de falsos positivos, com base em áudios

sintetizados, ou seja, arquivos WAV convertidos a partir de arquivos MIDI. Por fim, o desempenho será comparado com o obtido por trabalhos relacionados, a fim de validar a abordagem utilizada.

Este trabalho está dividido em cinco capítulos. O Capítulo 1 apresenta a área da transcrição automática de música, no qual são expostas as etapas do processo e apontados os principais estudos da área. O Capítulo 2 abrange todo o processo da detecção de *onsets*, expondo as etapas e avaliando os principais métodos para a geração da função de detecção. Além disso, apresenta alguns métodos de detecção de *offsets*. O Capítulo 3 expõe dois métodos para estimar a frequência fundamental de um sinal no domínio de tempo, que apresentam resultados satisfatórios quando aplicados em sinais monofônicos. O Capítulo 4 apresenta as implementações do protótipo deste trabalho, a partir do estudo transcrito nos capítulos anteriores. O Capítulo 5 expõe os resultados obtidos pelo protótipo e compara tais resultados com os atingidos por trabalhos relacionados, seguido das considerações finais.

1 TRANSCRIÇÃO AUTOMÁTICA DE MÚSICA

Música – em seu formato escrito – normalmente encontra-se presente no formato de partitura, contendo os tempos, notas, durações e demais aspectos dos sons ali presentes (KLAPURI, 2003). O processo de obtenção de uma partitura, ou outra forma de notação musical conhecida, a partir de sua execução recebe o nome de transcrição musical (DINIZ, 2009). Klapuri (2003) defende que este pode ser considerado como uma forma de descoberta da "receita" da música, pois, a partir desta, a música poderá ser reproduzida, ou mesmo modificada, posteriormente.

O processo de transcrição musical pode ser considerado exitoso caso seja possível, a partir de suas notações, recriar a obra original de forma idêntica. Sendo um procedimento complexo e que exige uma percepção musical apurada, principalmente se considerados arranjos complexos, este processo acaba ficando restrito a músicos experientes (DINIZ, 2009).

As possibilidades abertas pela automatização deste processo impulsionaram o desenvolvimento da área de transcrição automática de música (DINIZ, 2009), que tem como objetivo extrair, a partir da representação digital do sinal da música, as informações relativas às estruturas presentes nas partituras (BELLO; MONTI; SANDLER, 2000). No entanto, apesar do avanço significativo da área nos últimos anos, ainda não existem aplicações para o usuário final com um nível de transcrição confiável e preciso, capaz de processar diferentes arranjos, com diferentes instrumentos e combinações. Os sistemas atuais ainda revelam grandes dificuldades ao trabalhar com músicas de diferentes gêneros, sendo pouco flexíveis, à medida em que grande parte dos gêneros não possuem muitos aspectos em comum (BENETOS et al., 2012).

Atualmente, a transcrição automática de áudios monofônicos – áudios sem sobreposição de notas – já pode ser considerada compreendida. No entanto, transcrever áudios polifônicos e sem restrição de instrumentos presentes na faixa é uma tarefa a ser solucionada (BENETOS et al., 2012).

A fim de conseguir transcrever uma determinada melodia, o sistema deve ser capaz de extrair as alturas das notas (*pitch*), a duração das mesmas e o momento de início destas (*onset*) dos sinais de áudio. Apesar de instrumentos possuírem timbres diferentes, uma mesma nota sendo tocada por dois instrumentos diferentes irá possuir um mesmo *pitch* (BELLO; MONTI; SANDLER, 2000).

Aspectos físicos do sinal, tais como taxa de repetição, amplitude de onda e formato de onda, correlacionam-se com aspectos menos abstratos para nossa realidade, sendo eles, respectivamente, *pitch*, volume e timbre. Portanto, são utilizados por softwares de transcrição automática no processo (BELLO; MONTI; SANDLER, 2000).

As principais pesquisas na área tendem a não abranger todo o processo da transcrição automática de música. Enquanto uma linha de pesquisa (DIXON, 2006A; DIXON, 2006B; BELLO et al., 2005; COLLINS, 2005A; DUXBURY et al., 2003A; DUXBURY et al., 2003B; DZIUBIŃSKI; KOSTEK, 2004) dedica-se ao desenvolvimento de técnicas de detecção de *onsets*, o restante (BENETOS, 2012; KLAPURI, 2004; BENETOS et al., 2013; MCLEOD, 2008; ZORNDORF; CARREON, 2013) investe seu tempo na detecção de *pitch* na música. Por outro lado, alguns trabalhos (BELLO; MONTI; SANDLER, 2000; DINIZ, 2009; COLLINS, 2005B; ZHOU; REISS, 2007) propõem soluções completas para o problema, envolvendo todas as etapas do processo.

Para Benetos et al. (2012), essa separação entre as pesquisas também é um problema a ser solucionado na área. Hoje, faz-se necessário focar exclusivamente nas tarefas de forma exclusiva devido à complexidade envolvida. No entanto, a fim de desenvolver sistemas mais robustos, as pesquisas podem ser combinadas, buscando atingir resultados mais significativos. Apesar da área já possuir resultados interessantes publicados, muitas pesquisas buscam soluções sem considerar todo o progresso da área até o momento, tornando os resultados menos relevantes.

Avaliar sistemas de transcrição automática de música é uma tarefa complexa, à medida em que não existem formas padronizadas para execução dos testes, o que fica a cargo do pesquisador. Desta forma, cada autor utiliza um conjunto de músicas diferentes para aplicação, podendo, por vezes, mascarar e distorcer resultados, já que os métodos permitem a obtenção de resultados diferentes de acordo com as características das músicas analisadas. Por este motivo, comparar métodos diretamente entre si pode ser um processo difícil e, possivelmente, injusto (DINIZ, 2009).

Apesar de existirem inúmeras partituras e áudios das mais variadas músicas, um esforço muito grande é necessário para alinhar no tempo corretamente ambos, dificultando ainda mais o processo de avaliação dos sistemas existentes ou mesmo a utilização como base de treinamento para novos métodos de transcrição automática (BENETOS et al., 2012).

A seguir, serão apresentadas abordagens para a detecção de eventos mencionados – *onsets* e *offsets* – nos sinais de áudio, assim como a extração das notas relacionadas com estes eventos. Além disso, serão apresentadas as implementações de um protótipo desenvolvido neste trabalho e os resultados obtidos por este.

2 DETECÇÃO DE EVENTOS

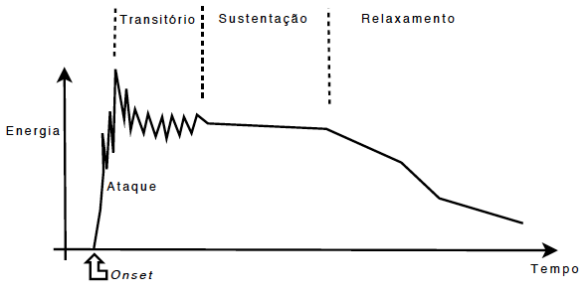
A execução de uma nota sempre irá abranger o início e a finalização desta execução. Assim, tais eventos devem ser detectados no processo de transcrição de uma música. Portanto, a transcrição automática também deve ser capaz de extrair tais informações do sinal. Neste capítulo, serão apresentadas algumas abordagens utilizadas no processo de detecção de *onsets* e *offsets*, eventos de início e finalização da execução de uma nota, respectivamente.

2.1 DETECÇÃO DE *ONSETS*

O instante exato do início de uma nota recebe o nome de *onset* (BELLO et al., 2005), sendo responsabilidade dos sistemas de transcrição automática realizar a detecção deste evento (DINIZ, 2009). Em áudios monofônicos, nas quais as notas apresentam-se isoladas, a tarefa de definição dos *onsets* é relativamente simples. No entanto, para áudios polifônicos, o processo se torna mais complexo, na medida em que notas simultâneas podem estar a milissegundos de distância entre elas (DIXON, 2006A).

O modelo de execução de uma nota, de acordo com Diniz (2009), engloba quatro etapas: ataque, transitório, sustentação e relaxamento (Figura 1). Entende-se por ataque: "o intervalo de tempo em que o envelope da amplitude de um sinal aumenta" (BELLO et al., 2005, p. 1035, tradução nossa). Por sua vez, transitório pode ser definido como o período em que o sinal se comporta de forma relativamente imprevisível, enquanto sustentação é o intervalo de tempo em que a amplitude do sinal tende a manter-se constante. Já a etapa de relaxamento pode ser compreendida como o período em que o sinal sofre desvanecimento, sendo a etapa final de execução de uma determinada nota musical (DINIZ, 2009).

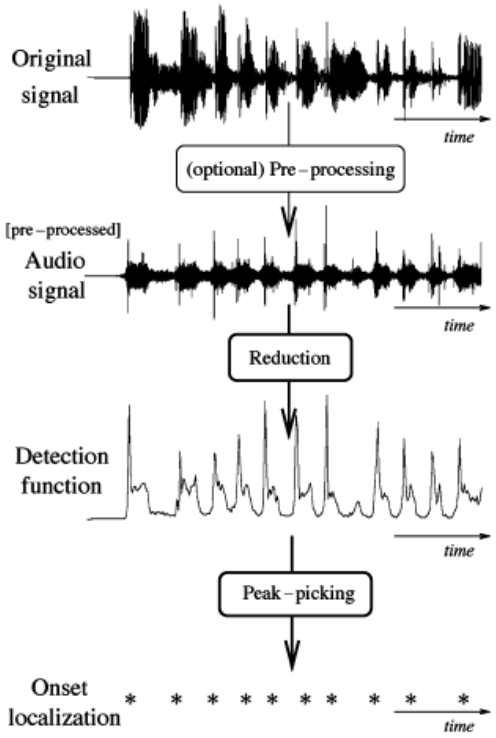
Figura 1 – Etapas da execução de uma nota musical



Fonte: DINIZ, Felipe Castello da Costa Betrão. **Transcrição musical automática usando representação frequencial eficiente por banco de filtros de alta**. 2009. 170 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-graduação em Engenharia Elétrica, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, 2009.

A detecção de *onsets* geralmente é constituída por três etapas, sendo elas: pré-processamento, redução e detecção de picos (Figura 2). A primeira etapa é opcional no processo, já que consiste somente no processo de enaltecer e atenuar aspectos relevantes e irrelevantes, respectivamente, do sinal (BELLO et al., 2005). Desta forma, em alguns trabalhos, como em Duxbury et al. (2003B), esta etapa é desconsiderada.

Figura 2 – Fluxograma de um algoritmo comum de detecção de *onsets*



Fonte: BELLO, Juan Pablo; DAUDET, Laurent; ABDALLAH, Samer; DUXBURY, Chris; DAVIES, Mike; SANDLER, Mark B. A Tutorial on Onset Detection in Music Signals. **IEEE** Transactions on speech and audio processing, v. 14, n. 5, p. 1035-1047, set. 2005.

Principal etapa de uma grande variedade de métodos de detecção de *onsets*, a etapa de redução, também denominada de função de detecção, é responsável por transformar o sinal em uma amostra menor, porém com maior expressividade (BELLO et al., 2005), mantendo somente as informações necessárias para a detecção de *onsets* na função (DIXON, 2006A).

Uma função de detecção efetiva deve gerar picos pontiagudos nos transientes do sinal e gerar o mínimo possível de outros picos para outros eventos do sinal (DUXBURY et al., 2003B), sendo um processo complexo, na medida em que os sinais de áudios sempre estão em estado de mudança contínua. Os algoritmos devem ser capazes de distinguir entre os variados tipos de mudanças presentes nos sinais de áudio: *onset*, *offset*, vibrato, ruído e outros (DIXON, 2006A).

Por fim, o sistema deve ser capaz de analisar a função de detecção, a fim de identificar os *onsets* presentes na mesma. O processo é feito através de um limiar, com o qual são comparados os pontos presentes na função de redução. Com isso, é possível definir como *onsets* os períodos em que a função ultrapassar o valor definido (BELLO et al., 2005).

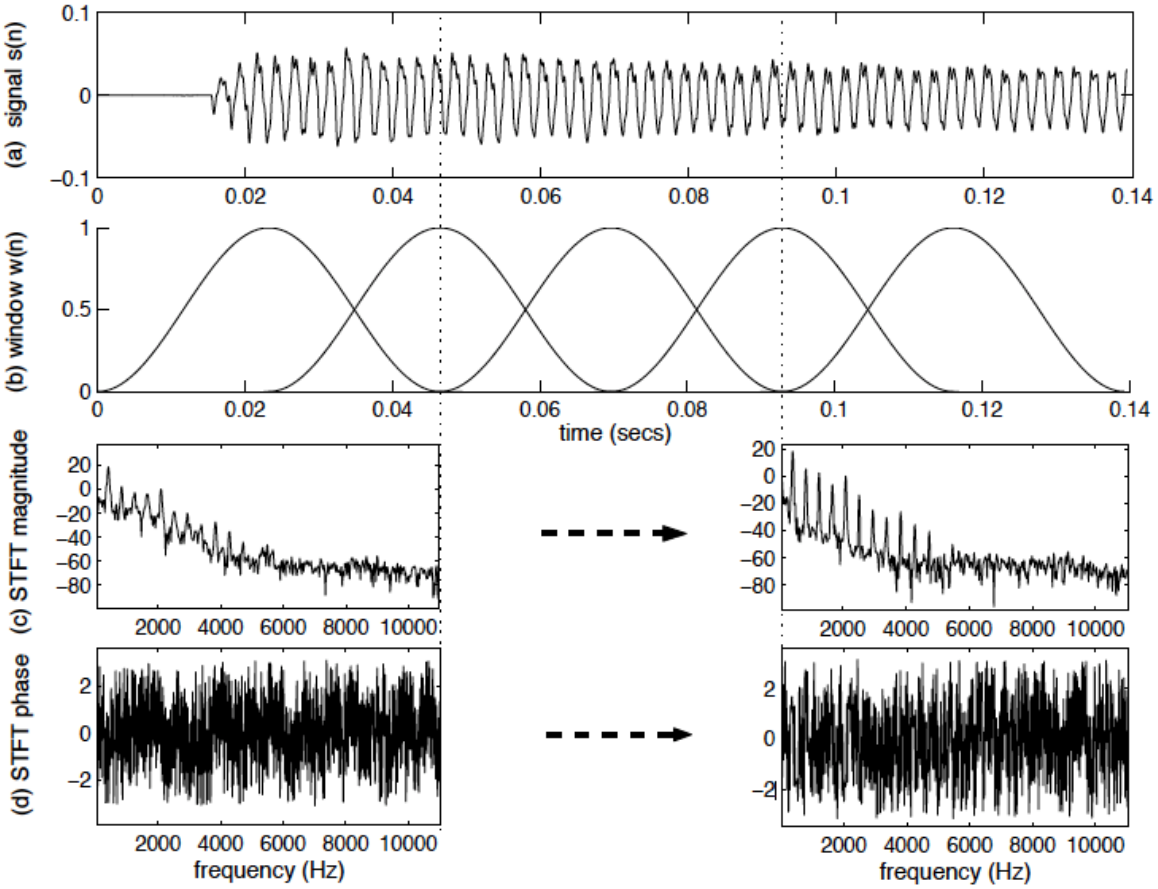
2.1.1 Função de detecção

Os métodos baseados na energia ou amplitude para geração da função de detecção presentes neste trabalho são baseados na representação do sinal no domínio da frequência (gerada a partir da Transformada de Curto Tempo de Fourier - STFT). A STFT permite que as informações referentes às frequências do sinal sejam visualizadas ao longo do tempo ao quebrar o sinal em pequenas janelas. Esta transformação pode ser calculada conforme:

$$S(n,k) = \sum_{m=-\infty}^{\infty} s(m)w(n-m)e^{-\frac{j2\pi mk}{N}} \quad (1)$$

para qual, $s(n)$ é o sinal no domínio de tempo, k indica a frequência utilizada para computação da transformada, enquanto $w(n)$ representa uma função de janelamento do sinal. O tamanho da janela, definido por N, é um parâmetro crucial na etapa do cálculo da STFT, na medida em que influencia diretamente no resultado da função. Conforme o tamanho da janela aumenta, mais precisa torna-se a representação do sinal no domínio da frequência. No entanto, a representação do mesmo no domínio de tempo é enfraquecida, porque quantidade menor de intervalos é considerada. O processo da STFT pode ser observado na Figura 3 (BELLO, 2003).

Figura 3 – Sinal no domínio de tempo (a). Função de janelamento (b). Magnitude (c) e fase (d) da STFT de duas janelas do sinal.



Fonte: BELLO, Juan Pablo. **Towards the Automated Analysis of Simple Polyphonic Music**: A Knowledge-based Approach. 2003. 239 f. Tese (Doutorado em Engenharia Eletrônica), University of London, Queen Mary, 2003.

2.1.1.1 Diferença espectral

Método comum utilizado na área de detecção de *onsets*, o método de diferença espectral – também citado como fluxo espectral – analisa a diferença (distância) entre dois espectros consecutivos do sinal (BELLO et al., 2005). A diferença é calculada a partir da soma das diferenças de magnitude ao longo dos canais, considerando apenas os valores positivos (DIXON, 2006B apud MASRI, 1996). Desta forma, consegue-se dar ênfase somente à detecção de *onsets*, e não de *offsets* (BELLO et al., 2005).

Este método utiliza como base o princípio de que, durante o ataque de uma nota, a distribuição de energia tende a sofrer alterações, enquanto que durante a sustentação de uma nota ou de uma pausa a tendência é de que essa distribuição se mantenha estável (DINIZ, 2009). A função da diferença espectral utiliza as normas 1 e 2 de diferença entre os vetores,

dependendo do autor. Enquanto alguns autores, como Dixon (2006A) e Masri (1996), propõem a utilização da norma 1 da diferença:

$$SD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X_k(n)| - |X_k(n-1)|) \quad (2)$$

outros autores, como Bello et al. (2005) e Duxbury et al. (2003B), sugerem o uso da norma 2 da diferença entre os vetores:

$$SD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \{H(|X_k(n)| - |X_k(n-1)|)\}^2 \quad (3)$$

Em ambas as fórmulas $H(x)$ equivale a $(x + |x|)/2$, ou seja, valores negativos são desconsiderados, zerados (DIXON, 2006A). Para Bello et al. (2004), o método de diferença espectral, assim como os demais métodos baseados em energia do sinal, é de um processo rápido e de fácil implementação. No entanto, por basear-se somente na energia do sinal, possui uma efetividade menor quando estão sendo analisados sinais de instrumentos não-percussivos.

2.1.1.2 Desvio de fase

Ao contrário do método de diferença espectral, este não se baseia nas mudanças de magnitude ao longo espectro do sinal para identificar *onsets* (BELLO et al., 2005). O desvio de fase fundamenta-se no fato de que, no início da execução de uma nota, a frequência tende a sofrer alterações, enquanto que, na sustentação da nota, a tendência é de que se mantenha estável (DUXBURY et al., 2003B). Sendo a frequência a derivada da fase, pode-se calcular a diferença entre as frequências de dois *frames* consecutivos do sinal, a fim de identificar a presença de um *onset* neste período (DINIZ, 2009).

Considerando $\varphi_k(n)$ como a fase de um determinado coeficiente do espectro do sinal $X_k(n)$, para calcular a frequência deste *frame* – para um sinal em estado de sustentação – a fase do *frame* atual ($\varphi_k(n)$) e do *frame* anterior ($\varphi_k(n-1)$) são utilizadas da seguinte maneira (BELLO et al., 2005):

$$f_k(n) = \left(\frac{\varphi_k(n) - \varphi_k(n-1)}{2\pi h} \right) f_s \quad (4)$$

sendo h a quantidade de amostras que constituem o salto entre as janelas e f_s a taxa de amostragem do sinal (DINIZ, 2009).

Assim, pode-se perceber que, de acordo com a Equação 4, a tendência é de que a variação entre *frames* subsequentes seja similar à do *frame* atual e do anterior (BELLO et al., 2005). Desta forma, é possível prever a fase de um determinado *frame* n, na medida em que:

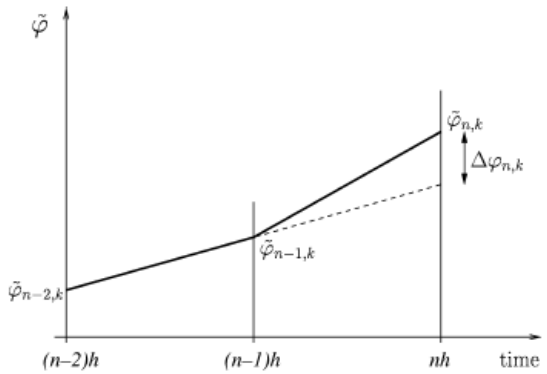
$$\tilde{\varphi}_k(n-1) - \tilde{\varphi}_k(n-2) = \tilde{\varphi}_k(n) - \tilde{\varphi}_k(n-1) \quad (5)$$

em que o operador $\tilde{\varphi}$ denota a fase desempacotada (DUXBURY et al., 2003B). Portanto, a partir da segunda diferença da fase do sinal, é possível calcular o desvio da mesma, conforme:

$$\Delta\varphi_k(n) = \varphi_k(n) - 2\varphi_k(n-1) + \varphi_k(n-2) \cong 0 \quad (6)$$

Durante a etapa de sustentação de uma determinada nota musical, a tendência é de que esta variação se mantenha próxima a zero. Entretanto, durante transientes de ataque, a variação costuma ser grande. Na Figura 4 é possível identificar a variação na fase do sinal em $\Delta\varphi_k(n)$. Todavia, caso este gráfico representasse um sinal durante um período de sustentação de uma nota, o gráfico se manteria constante, conforme a linha pontilhada (BELLO et al., 2005). Logo, uma variação na frequência entre dois *frames* consecutivos torna-se um indicativo de um possível *onset* neste ponto (DIXON, 2006A).

Figura 4 – Diagrama da fase do sinal ao longo de *frames* adjacentes.



Fonte: BELLO, Juan Pablo; DAUDET, Laurent; ABDALLAH, Samer; DUXBURY, Chris; DAVIES, Mike; SANDLER, Mark B. A Tutorial on Onset Detection in Music Signals. **IEEE** Transactions on speech and audio processing, v. 14, n. 5, p. 1035-1047, set. 2005.

Então, a partir do cálculo da média do módulo da soma das variações – esta abordagem reduz as possibilidades de falhas na detecção de *onsets* –, é possível elaborar a função de detecção baseada na fase do sinal, podendo ser representada, conforme Dixon (2006B):

$$PD(n) = \frac{1}{N}\sum_{k=1}^{N} |\Delta\varphi_k(n)| \tag{7}$$

A detecção de *onsets* baseada na fase do sinal é uma alternativa que permite identificar *onsets* suaves, em contrapartida aos métodos baseados na energia do sinal, que possuem melhores resultados se aplicados em sinais contendo instrumentos percussivos, na medida em que estes tendem a sofrer oscilações mais significativas na energia durante os transientes de ataque (DIXON, 2006A).

2.1.1.3 Domínio complexo

O método do domínio complexo é uma composição do método que atua sobre a fase do sinal e outro sobre a energia do sinal. Este objetiva – através desta abordagem – fortalecer os pontos fortes e atenuar os pontos fracos de cada uma (DINIZ, 2009). Enquanto os métodos baseados na energia do sinal esperam por oscilações na amplitude do sinal a fim de detectar *onsets*, conseguindo detectar com eficiência instrumentos com fortes transientes de ataque, os métodos baseados na fase do sinal são capazes de detectar *onsets* menos salientes, com transientes de ataques menos intensos (DUXBURY et al., 2003B). Bello et al. (2004) também apontam que, por individualmente apresentarem resultados melhores para grupos de frequências opostas, a união destas abordagens permite a criação de uma função de redução mais confiável.

A integração destes métodos foi proposta primeiramente por Duxbury et al. (2003B), posteriormente sendo abordada e revisada em outros trabalhos (BELLO et al., 2005; DIXON, 2006A; DIXON, 2006B). Conforme citado anteriormente, sinais em estado estável tendem a exibir pouca variação na sua magnitude ou frequência. Desta forma, Duxbury et al. (2003B) apontam que a aplicação de ambos os valores no plano complexo permite medir a variação entre o valor do *frame* atual (valor medido) e o valor calculado do *frame* anterior da STFT (valor alvo) para identificar *onsets* no sinal.

O valor alvo, em sua forma polar, pode ser obtido conforme:

$$\hat{S}_k(m) = \hat{R}_k(m)e^{j\hat{\phi}_k(m)} \tag{8}$$

em que a amplitude alvo – representada por $\hat{R}_k(m)$ – corresponde à magnitude do *frame* anterior que, por sua vez, pode ser representado por $|S_k(m-1)|$. A fase alvo, definida por $\hat{\phi}_k(m)$, pode ser calculada a partir da soma da fase do *frame* anterior com a variação a do frame precedente: $\hat{\phi}_k(m) = \text{princarg}[2\tilde{\varphi}_k(m-1) - \tilde{\varphi}_k(m-2)]$, em que a função princarg é responsável por mapear a fase no domínio $[-\pi, \pi]$ (BELLO et al., 2005).

De modo similar, o cálculo do valor medido dá-se por:

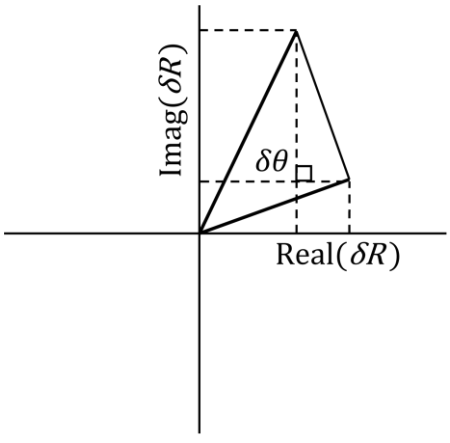
$$S_k = R_k(m)e^{\phi_k(m)} \tag{9}$$

em que R_k e ϕ_k representam a magnitude e a fase, respectivamente, do *frame* atual da STFT. Então, a partir do cálculo da distância euclidiana entre os dois vetores no plano complexo (Figura 5), obtém-se o nível de estacionariedade do *frame* atual (DUXBURY et al., 2003B), conforme:

$$\Gamma_k(m) = \left\{\left[\Re\left(\hat{S}_k(m)\right) - \Re\left(S_k(m)\right)\right]^2 + \left[\Im\left(\hat{S}_k(m)\right) - \Im\left(S_k(m)\right)\right]^2\right\}^{1/2} \tag{10}$$

considerando $\Re$ e $\Im$ como sendo, respectivamente, a parte real e imaginária do número complexo (DINIZ, 2009).

Figura 5 – Diagrama fasorial demonstrando a variação entre vetor alvo e vetor medido, além da distância euclidiana entre ambos



Fonte: DUXBURY, Chris; BELLO, Juan Pablo; DAVIES, Mike; SANDLER, Mark. Complex domain onset detection for musical signals. In: PROCEEDINGS OF THE 6TH INTERNACIONAL CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFx-03), 6., 2003, Londres. **Anais**... Londres: Queen Mary University of London, 2003.

A elaboração da função de detecção a partir do domínio complexo dá-se através do somatório dos níveis de estacionariedade ao longo do sinal, podendo ser representada por:

$$\eta(m) = \sum_{k=1}^{N} \Gamma_k(m) \tag{11}$$

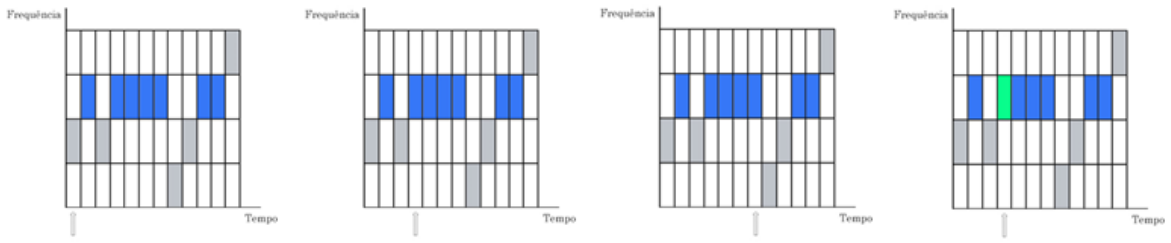
A função de redução gerada a partir deste método apresenta picos agudos em trechos com níveis de estacionariedade baixos, sendo, portanto, um método adequado para detecção de *onsets* (DUXBURY et al., 2003B).

2.1.1.4 Detecção baseada em pitch

A detecção de *onsets* não está exclusivamente vinculada a métodos baseados na análise do espectro do sinal observando parâmetros como energia e amplitude. Collins (2005B) propõe um método baseado na estabilização do *pitch* de uma determinada nota como indicador de detecção de um evento de *onset*, ou seja, no início do período de sustentação de uma nota. Esta abordagem não está vinculada à utilização de um determinado método para detecção de *pitch*, podendo ser adaptada conforme a necessidade da aplicação.

Esta abordagem já havia sido utilizada por Bello, Monti e Sandler (2000), a fim de detectar *onsets* para notas sustentadas por um limite mínimo definido manualmente. Neste trabalho, o método de autocorrelação (abordado no capítulo 3.1.1) foi utilizado para determinar o *pitch* ao longo do sinal. A detecção de *offsets* neste método também se dá através do *pitch* – ao detectar uma nova nota ou através do desvanecimento da nota atual. Na Figura 6 é possível observar o processo de detecção de *onsets* baseada no pitch, onde, após detectar uma mesma frequência durante quatro janelas consecutivas, indica que o momento de *onset* está presente no início desta detecção (destacado em verde).

Figura 6 - Diagrama da detecção de onsets baseada em pitch



Fonte: do autor

A utilização de métodos de detecção de *onsets* baseados no *pitch* tem seu viés voltado à detecção de *onsets* suaves, na medida em que a variação do *pitch* tende a ser o parâmetro que sofre maior influência durante *onsets* nestes casos (ZHOU, 2006). Enquanto Collins (2005B) utilizou esta abordagem somente em áudios monofônicos, Benetos (2012) propôs sua aplicação em conjunto com o método de diferença espectral a fim de detectar *onsets* em sinais polifônicos, abrindo portas para a utilização da abordagem em novos cenários.

2.1.2 Detecção de picos

A última etapa é responsável por detectar os picos da função de detecção, a fim de filtrar somente *onsets* reais, ignorando falsos picos gerados a partir de fenômenos como reverberação e eco (DINIZ, 2009). Desta forma, evidencia-se a influência da etapa de construção da função de redução pois, em uma função bem definida, eventos de *onsets* devem se destacar em meio a outros eventos (BELLO et al., 2005).

No entanto, de acordo com Duxbury et al. (2003B), somente seria possível classificar diretamente os picos da função como *onsets* nos casos em que a função de detecção fosse perfeita. Desta maneira, torna-se necessário aplicar um algoritmo robusto sobre esta função, a fim de detectar os *onsets* presentes no sinal (BELLO et al., 2005).

A identificação dos picos presentes na função é feita por meio do uso de um limiar (BELLO et al., 2005). No entanto, essa abordagem possui uma série de problemas, tais como a presença de ruído na função de detecção e a tendência de que um sinal real sofra variações significativas na magnitude do sinal, dificultando a definição deste limiar (DUXBURY et al., 2003B).

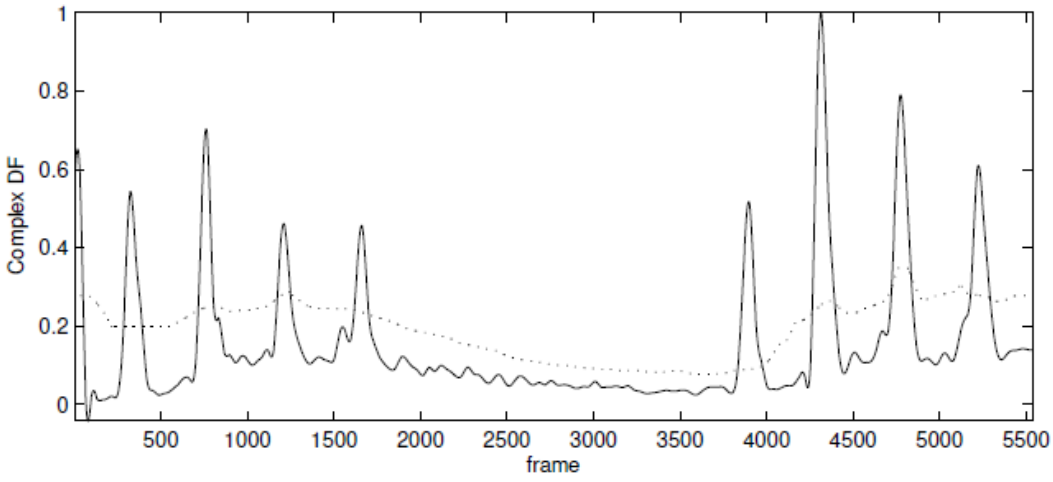
O limiar pode ser definido de forma fixa, com um valor global para toda a função de detecção, ou dinamicamente, variando ao longo da função. Quando utilizado um limiar fixo, todos os valores acima deste serão considerados picos: $d(n) \geq \delta$, sendo $d(n)$ a função de detecção e δ um valor constante (limiar). O limiar global, apesar de ser um método computacionalmente eficiente, tende a ser uma opção ineficiente na medida em que os sinais de música sofrem variações significativas ao longo do sinal (DUXBURY et al., 2003B), detectando falsos *onsets* nas passagens da música com volume intensificado e falhando em detectar *onsets* em trechos mais silenciosos (BELLO et al., 2005).

Em contrapartida, o limiar dinâmico tenta adaptar-se a tais mudanças (DINIZ, 2009), conforme pode ser observado na Figura 7, utilizando um limiar calculado a partir da função de detecção: $\tilde{\delta}(n)$. O cálculo para geração deste limiar, normalmente, dá-se através da suavização da função de detecção. Esta, por sua vez, pode ser elaborada baseada em percentis, assim, *outliers* presentes na função de detecção afetam menos os resultados se comparadas a abordagens baseadas na média:

$$\tilde{\delta}(n) = \delta + \lambda \,.\, \text{mediana}\,\{d(n-M), \ldots, |d(n+M)|\} \qquad (12)$$

em que λ é um valor constante para definição do limiar e M é o tamanho de uma janela aplicada sobre os pontos da função a fim de suavizar o sinal (BELLO et al., 2005).

Figura 7 – Limiar dinâmico calculado sobre uma função de detecção gerada utilizando o algoritmo de domínio complexo



Fonte: DUXBURY, Chris; BELLO, Juan Pablo; DAVIES, Mike; SANDLER, Mark. Complex domain onset detection for musical signals. In: PROCEEDINGS OF THE 6TH INTERNACIONAL CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFx-03), 6., 2003, Londres. **Anais**... Londres: Queen Mary University of London, 2003.

2.2 DETECÇÃO DE *OFFSETS*

Ao contrário do que acontece no caso da detecção de *onsets*, ainda existem poucos estudos na área da detecção de *offsets*, sendo normalmente apenas uma etapa dos trabalhos que a mencionam. No entanto, o trabalho de Liang et al. (2015) foi dedicado de forma exclusiva à detecção de *offsets*, servindo de embasamento para futuras pesquisas na área.

Os sinais de áudio tendem a apresentar um desvanecimento gradual na etapa de relaxamento da nota, enquanto na fase de ataque a amplitude do sinal intensifica-se

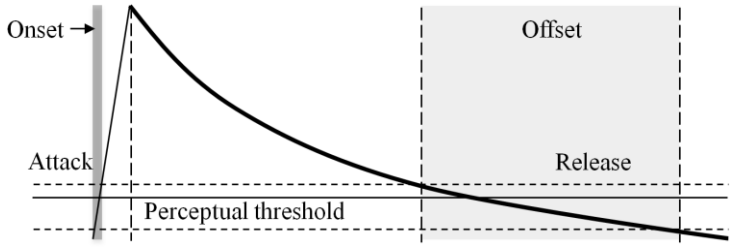
rapidamente, conforme pode ser observado na Figura 8. Desta forma, humanos tendem a identificar *offsets* em regiões diferentes do sinal, na medida em que o limiar utilizado para fazer tais detecções varia entre pessoas. Este problema também existe na detecção de *onsets*, no entanto, por ocorrer em um período mais breve, o espaçamento entre as diferentes opiniões tende a variar menos (LIANG et al., 2015).

Basicamente, existem atualmente três diferentes características de um sinal que podem ser analisadas para identificar *offsets*, sendo elas energia do sinal, fluxo do espectro e *pitch.* A análise da energia do sinal é um método relativamente simples, inclusive para a detecção de *onsets*, em que é avaliado se o envelope do sinal ultrapassa um determinado limiar. A fórmula pode ser expressa por:

$$E(n) = \sum_{k=0}^{N} |X_k(n)| \tag{13}$$

em que $X_k(n)$ representa o espectro do sinal para a janela n do sinal (LIANG et al., 2015). Esta abordagem para detecção de *offsets* é utilizada por Bello, Monti e Sandler (2000), na qual a presença de um *offset* é detectada quando a energia do sinal fica abaixo de um determinado limiar. No entanto, Liang et al. (2015) apontam que notas normalmente são sucedidas pela presença de outras notas, tornando os resultados da utilização de um limiar fixo insatisfatórios.

Figura 8 – Envelope do modelo de execução de uma nota de um violino (dedilhada)



Fonte: LIANG, Che-Yuan; SU, Li; YANG, Yi-Hsuan; LIN, Hsin-Ming. Musical offset detection of pitched instruments: the case of violin. In: PROCEEDINGS OF THE INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE (ISMIR), 16., 2015, Málaga. **Anais**... Málaga: Universidad de Málaga, 2015. p. 281-287.

A detecção de *offsets* baseada no fluxo do espectro, por sua vez, exerce o processo inverso ao método da diferença espectral para detecção de *onsets*, representada pela Equação (2). A inversão desse processo dá-se por duas formas, por meio da análise do espectro no sentido contrário ou a partir da supressão somente do fluxo positivo do espectro. A avaliação do espectro de trás para frente pode ser expressa por:

$$SD_{rc}(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X_k(n)| - (|X_k(n+3)|) \quad (14)$$

A supressão do fluxo positivo do espectro, por sua vez, substitui o termo $H(x)$ da equação por $H'(x) = (|x| - x)/2$, em que todo o fluxo positivo é desconsiderado, pois resultará em zero (LIANG et al., 2015).

A utilização do *pitch* do sinal para definição de um *offset* ocorre de duas maneiras, por meio da detecção da alteração do *pitch* ou da identificação de ausência de periodicidade no sinal, indicando que este não possui um *pitch* vinculado. A presença de uma alteração de *pitch* no sinal é uma informação precisa sobre a presença de uma transição de nota que, por sua vez, engloba tanto o *offset* da nota anterior quanto o *onset* da próxima nota. No entanto, quando uma mesma nota é executada em sequência, pode não haver alteração no *pitch*. Essa característica é uma limitação desta abordagem. A fim de detectar um *offset* em um determinado *frame* do sinal, podem ser avaliadas as seguintes condições:

$$mod_{12}(m_n - m_{n-1}) \geq 1 \ \wedge \ c_n - c_{n-1} < 0 \quad (15)$$

Na qual c_n indica o grau de periodicidade do sinal no frame n, conforme pode ser visto na seção 3.1.2, e m_n equivale à nota MIDI correspondente à determinada f_0, que pode ser calculada através da fórmula:

$$m_n = 12 \ \times \log_2(\frac{f_0[n]}{440}) \ + \ 69 \quad (16)$$

em que o numeral 12 equivale à quantidade de notas na escala musical, 440 à frequência do lá acima do dó central e 69 à nota MIDI referente ao A440. A função de módulo utilizada na primeira sentença é empregada com o intuito de prevenir erros na detecção das oitavas das notas (LIANG et al., 2015).

A detecção de *offsets* por meio da análise da confiança do *pitch* do sinal – isto é, o nível de periodicidade do sinal – baseia-se na ideia de que, no momento em que um sinal deixa de ser periódico, existe a indicação de um *offset*. Para executar tal tarefa, esta abordagem, assim como diversos outros métodos na área de transcrição de música, utiliza um limiar δ_c como base para efetuar a detecção. Duas condições devem ser verdadeiras para que um *offset* seja detectado em um *frame* n, sendo elas:

$$c_{n-1} > \delta_c \wedge c_n < \delta_c \quad (17)$$

ou seja, um *offset* é detectado sempre que o sinal oscila entre valores acima e abaixo do limiar definido (LIANG et al., 2015).

3 ESTIMATIVA DA FREQUÊNCIA FUNDAMENTAL

Um determinado sinal pode ser considerado periódico quando ele se repete durante intervalos de tempos, chamados de períodos (BENETOS, 2012 apud YEH, 2008). A frequência fundamental (f_0) de um sinal, normalmente, corresponde à menor frequência da série de harmônicas do sinal (GERHARD, 2003). Sinais periódicos podem ser descritos como uma série de senoides relacionadas harmonicamente, denominadas de harmônicas ou parciais do sinal. Neste cenário, as sub-harmônicas – as harmônicas, com exceção da fundamental – do sinal apresentam-se, no domínio de frequência, como múltiplos da f_0. No entanto, instrumentos musicais não produzem sons estritamente harmônicos, sendo geralmente considerados como *quasi-periódicos*. Assim, as parciais do sinal não representam múltiplos perfeitos da fundamental (BENETOS, 2012).

A determinação da f_0 de um sinal, em sua forma mais simplista, pode ser executada por meio da análise da representação da pressão do ar – correspondente às amplitudes da onda em um sinal sonoro – ao longo do tempo, e a partir da forma desta onda, determinar a f_0 do sinal (GERHARD, 2003).

Pitch, por sua vez, pode ser definido como "atributo de percepção que permite ordenar diferentes sons em uma escala de frequências" (BENETOS, 2012 apud KLAPURI, 2006). Na medida em que o *pitch* está relacionado com a f_0, muitas pesquisas apresentam métodos para detecção deste quando, na realidade, está sendo feita somente a extração da f_0. Métodos de extração do *pitch* devem gerar, preferencialmente, resultados em uma escala de *pitch* e não de frequência, como normalmente são encontrados (GERHARD, 2003).

3.1 MÉTODOS

3.1.1 Função de autocorrelação

O grau de similaridade entre duas ondas é a sua correlação. As ondas são comparadas em diferentes momentos, a fim de verificar sua similitude. A autocorrelação (ACF), por sua vez, refere-se ao nível de similaridade do próprio sinal consigo, considerando atrasos de tempos (*lags*) diferentes (GERHARD, 2003) ao somar os resultados das multiplicações entre todos os elementos do sinal com os elementos do sinal deslocados no tempo. Ressalta-se que, para *lag* igual a zero, as duas ondas são exatamente iguais, pois está sendo comparada com ela mesma,

no mesmo espaço de tempo (MCLEOD, 2008). Este trata-se de “um método simples, rápido e confiável” (BELLO; MONTI; SANDLER, 2000, p. 2).

Assim como alguns dos métodos de detecção de *onsets* mencionados neste trabalho, o método da detecção de f_0 através da autocorrelação também se baseia na STFT, em que se aplica o cálculo da autocorrelação em cada janela desta (BELLO; MONTI; SANDLER, 2000). Isso se deve, primordialmente, ao fato de que a tendência é de que a frequência de uma nota não se mantenha estável e que, constantemente, sofra oscilações. Devido a isto, a função de autocorrelação deve ser aplicada em pequenas janelas, em que a frequência sofra pouca variação. A janela, no entanto, deve ter no mínimo duas vezes o tamanho do período da onda, para que seja possível aplicar a autocorrelação de forma bem-sucedida (MCLEOD, 2008).

A função da autocorrelação pode ser expressa por:

$$ACF(\tau) = \sum_{n=0}^{N-\tau-1} x_n x_{n+\tau} \tag{18}$$

para a qual $0 \leq \tau < W$, em que x_n é o sinal, N equivale ao tamanho do sinal e τ representa o *lag*. A partir da geração da função de autocorrelação, é possível extrair a f_0 através do primeiro intenso pico presente nesta (BENETOS, 2012).

Segundo McLeod (2008), é possível utilizar funções de janelamento sobre o sinal, a fim de suavizá-lo antes da geração da função de autocorrelação, de tal forma que sejam amenizados os “efeitos das pontas”, causados ao encontrar correlação no início e no final do sinal. A janela de Hamming normalmente é utilizada para este fim e pode ser representada conforme:

$$w_n = 0.53836 - 0.46164\, cos(\frac{2\pi n}{W-1}) \tag{19}$$

em que W equivale ao tamanho da janela de Hamming, que deve conter a mesma quantidade de coeficientes que o sinal para poder ser aplicada. A função da autocorrelação com janelamento é definida por:

$$ACF_{win}(\tau) = \sum_{n=0}^{N-\tau-1} x_n w_n x_{n+\tau} w_{n+\tau} \tag{20}$$

A função de autocorrelação apresenta bons resultados na detecção de sinais periódicos, no entanto, sinais de música tendem a sofrer variações na frequência e amplitude ao longo do tempo. Portanto, não são representadas por um sinal periódico perfeito, impactando diretamente nos resultados do método (MCLEOD, 2008). Além disso, o método da autocorrelação pode identificar incorretamente a oitava de uma determinada nota, na medida em que este método pode considerar como f_0 um determinado pico de uma harmônica da fundamental (GERHARD, 2003).

3.1.2 YIN

Proposto por Alan de Cheveigné e Hideki Kawahara, o método YIN apresenta melhorias no método da função de autocorrelação a fim de tentar solucionar o principal problema deste – a dificuldade em determinar a f_0, pois a função apresenta picos em suas sub-harmônicas (GERHARD, 2003 apud CHEVEIGNÉ; KAWAHARA, 2002).

O método YIN baseia-se na função da diferença quadrática, uma variação da função de autocorrelação, que objetiva minimizar a diferença entre o sinal e a si próprio deslocado no tempo, o inverso do que faz a autocorrelação (GERHARD, 2003). Assim, esta função busca medir a diferença entre um sinal e a si mesmo deslocado no tempo (TAVARES; BARBEDO; LOPES, 2007). A função da diferença pode ser expressa pela fórmula:

$$SDF(\tau) = \sum_{n=1}^{N} (x_n - x_{n+\tau})^2 \tag{21}$$

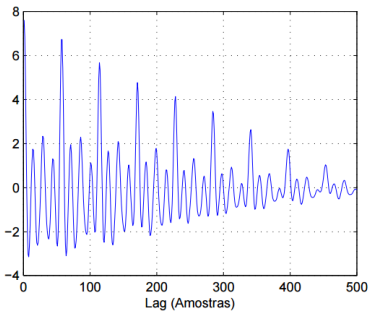
em que, assim como na função da autocorrelação, τ representa o *lag* para comparação, N equivale ao tamanho do sinal e x_n é o sinal (BENETOS, 2012). Quanto menor o valor de $SDF(\tau)$ maior será a confiança de que a f_0 do sinal esteja em $1/\tau$. Desta forma, caso este valor seja muito alto, considera-se que este sinal não possui *pitch* vinculado. O cálculo da f_0 é expresso por:

$$f_0 = \begin{cases} 0, & 1 - SDF(\tau) \leq \delta_c \\ (\arg_\tau \min SDF(\tau))^{-1}, & 1 - SDF(\tau) > \delta_c \end{cases} \tag{22}$$

em que $\arg_\tau \min SDF(\tau)$ representa a posição do menor valor da função da diferença. A confiança do *pitch* é expressa por $c_n = 1 - \min SDF(\tau)$ e representa a probabilidade de que o sinal seja periódico e possua um *pitch* atrelado (LIANG et al., 2015).

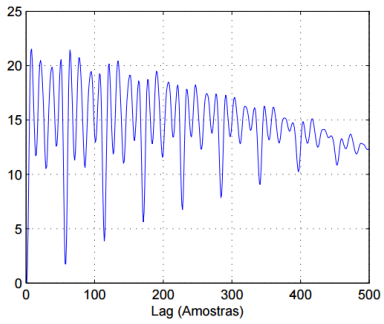
Na Figura 9 é possível visualizar a função da autocorrelação e compará-la com a função da diferença quadrática, ilustrada na Figura 10:

Figura 9 – Função de autocorrelação



Fonte: MONTEIRO, Lucas Lago. **Síntese aditiva aplicada à transferência de timbre em vocalise**. 2012. 102 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

Figura 10 – Função da diferença quadrática



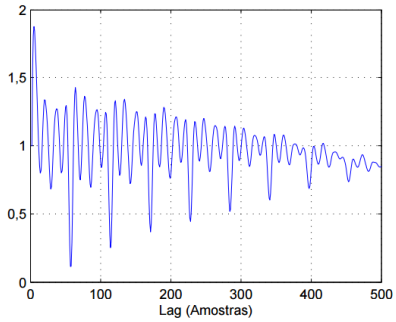
FONTE: MONTEIRO, Lucas Lago. **Síntese aditiva aplicada à transferência de timbre em vocalise**. 2012. 102 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

No entanto, o método da diferença quadrática ainda apresenta problemas na identificação da f_0 em casos em que detecta valores muito altos erroneamente. Portanto, foi proposta a função da média normalizada acumulada da diferença quadrática (Figura 11), sendo esta uma variação da fórmula da diferença quadrática, que pode ser representada por:

$$SDF'(\tau) = \begin{cases} 1, & \tau = 0 \\ SDF(\tau) \Big/ \frac{1}{\tau} \sum_{n=1}^{\tau} SDF(n), & \tau \neq 0 \end{cases} \quad (23)$$

para a qual, quando o *lag* for diferente de zero, divide-se o valor da diferença quadrática pela média da soma das diferenças quadráticas até o *lag* atual a fim de reduzir estes erros (CHEVEIGNÉ; KAWAHARA, 2002).

Figura 11 – Função da média normalizada acumulada da diferença quadrática



Fonte: MONTEIRO, Lucas Lago. **Síntese aditiva aplicada à transferência de timbre em vocalise**. 2012. 102 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

Ainda, o método de YIN apresenta propostas de melhorias no método, tais como a definição de um limiar para reduzir erros de oitavas, onde que as sub-harmônicas do sinal estão em foças mais profundas do que a própria f_0. Sugere, ainda, aplicar uma interpolação parabólica, a fim de contornar o erro causado quando a f_0 não é múltipla da taxa de amostragem do sinal (MONTEIRO, 2012). Desta forma, este método atende sua proposta, sendo conceitualmente simples e com ótima precisão na detecção da f_0 (KLAPURI, 2004).

4 PROTÓTIPO

As primeiras etapas deste trabalho analisaram os principais métodos utilizados na transcrição de áudios monofônicos, a fim de tornar possível a seleção de um método para detectar *onsets*, um método para detectar *offsets* e um terceiro para identificar a frequência fundamental de um som e, assim, detectar a sua nota. Ao longo deste estudo, um protótipo foi desenvolvido a partir das técnicas estudadas.

A linguagem de programação Python foi utilizada no desenvolvimento deste protótipo, na medida em que possui uma vasta biblioteca de APIs que facilitam a manipulação de sinais, assim como o trabalho com arquivos de música, especificamente com arquivos WAV, utilizados no caso deste protótipo. Para manipulação de vetores e cálculo de expressões algébricas, a biblioteca NumPy[1] foi utilizada. A interpretação de arquivos de áudio foi feita através do uso da biblioteca SciPy[2], abstraindo a complexidade da extração dos metadados destes arquivos.

Os gráficos apresentados neste estudo foram plotados através da Matplotlib – uma biblioteca de geração de gráficos. Por fim, foi necessário manipular arquivos MIDI. Para tal, a biblioteca Mido[3] foi utilizada, o que permitiu interpretar arquivos pré-existentes para análise.

4.1 MANIPULAÇÃO DE ARQUIVOS DE ÁUDIO

O protótipo deve ser capaz de analisar os arquivos de áudio e extrair destes as informações inerentes ao processo de transcrição automática de música. Para este estudo, foi necessário detectar a taxa de amostragem do áudio, além dos dados que representam o sinal em si. Somente áudios no formato de arquivos WAV foram permitidos como entrada, pois este não possui compactação dos dados, como ocorre nos arquivos com extensão MP3. O protótipo trabalha com arquivos que utilizam um único canal, ou seja, que utilizam um sistema monaural.

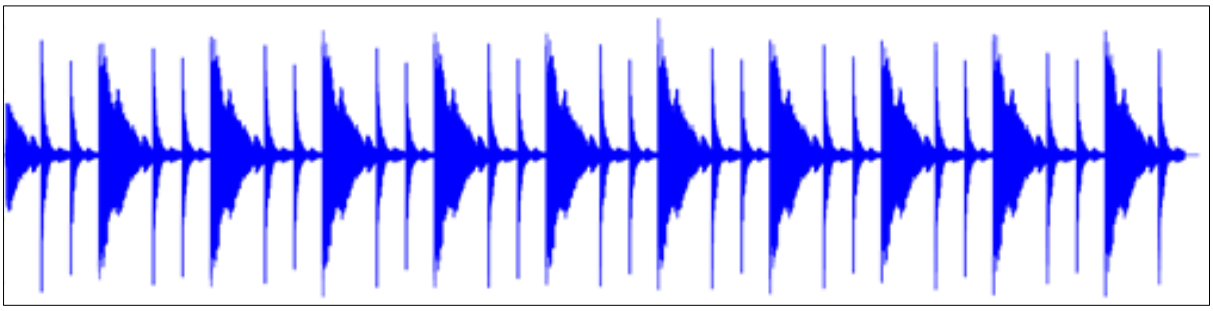
Os arquivos WAV são compostos por diversos trechos, denominados de *chuncks*. O cabeçalho do arquivo contém informações como quantidade de canais, taxa de amostragem e quantidade de bits por amostragem. A área de dados, por sua vez, é composta pelas informações

[1] Disponível em http://www.numpy.org/
[2] Disponível em https://www.scipy.org/
[3] Disponível em https://mido.readthedocs.io/

necessárias para representar o áudio no formato de onda, ou seja, armazena a amplitude ao longo do tempo.

A Figura 12 representa o sinal de um áudio de dez segundos do instrumento triângulo. O áudio analisado possui taxa de amostragem de 44100 Hz, informação também extraída do arquivo WAV.

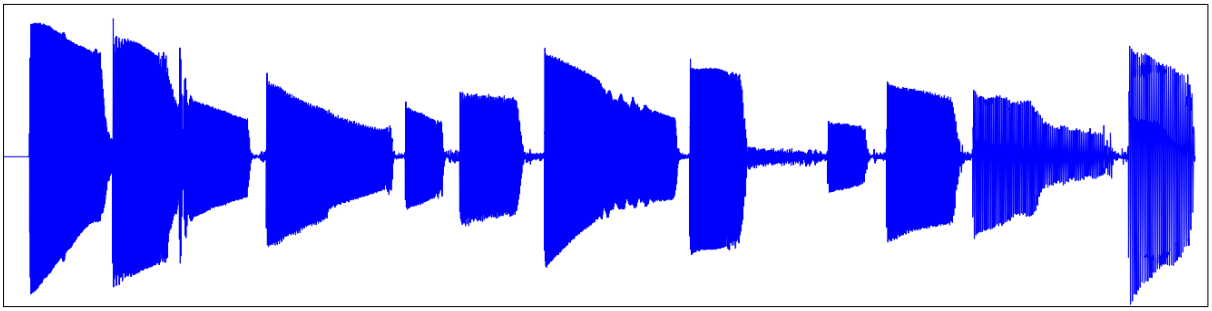
Figura 12 – Formato de onda de um áudio de um triângulo



Fonte: do autor

A Figura 13, por sua vez, apresenta, em formato de onda o áudio de uma guitarra durante um período de quatro segundos. Ao contrário do exemplo acima, neste existem diversos *legatos*, ou seja, notas que se interligam umas nas outras, sem existir um intervalo de tempo entre ambas.

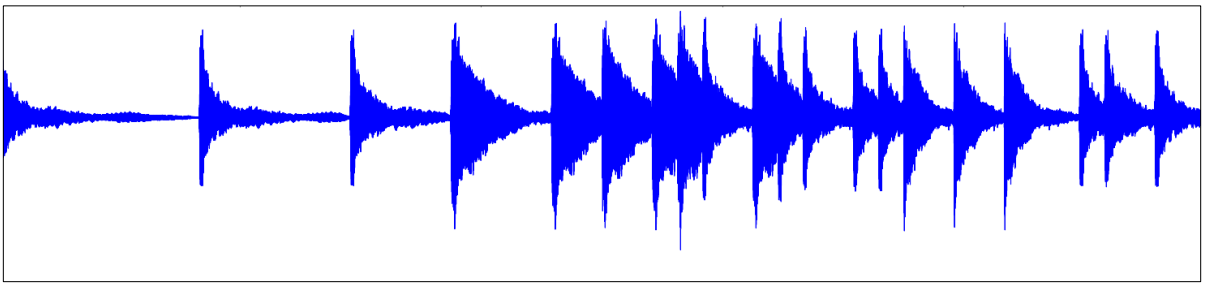
Figura 13 – Formato de onda de áudio de guitarra com *legatos*



Fonte: do autor

Divergindo dos dois exemplos acima, a Figura 14 representa o sinal de um áudio sintetizado, gerado a partir da conversão de um arquivo MIDI para um arquivo WAV. Áudios sintetizados apresentam uma quantidade menor de harmônicas que sons reais, ou seja, o áudio não consegue ser fielmente representado (BELLO et al., 2004). Esta característica influencia diretamente no processo de transcrição automática, na medida em que existem menos informações com as quais o software deve ser capaz de lidar, simplificando o processo.

Figura 14 – Formato de onda de áudio sintetizado



Fonte: do autor

Neste trabalho, foi utilizada a biblioteca SciPy, que permite extrair a área de dados de arquivos WAV, assim como a taxa de amostragem deste.

4.2 DETECÇÃO DE EVENTOS

Na segunda etapa do desenvolvimento do protótipo, foram implementadas três funções no software: a detecção de *onsets*, a detecção de *offsets* e a detecção de picos, utilizada por ambas as funções anteriores. A seguir, estão descritas, detalhadamente, as implementações feitas.

4.2.1 Detecção de *onsets*

Nas etapas prévias deste estudo, foram analisadas as principais abordagens de detecção de *onsets* no processo de transcrição de música monofônica. Conforme descrito anteriormente, a principal etapa no processo de detecção de *onsets* é a geração da função de detecção, que, posteriormente, será utilizada para definir os pontos do áudio onde iniciam as execuções das notas.

No desenvolvimento do protótipo, dois métodos utilizados na geração da função de detecção foram implementados para analisar seus resultados. São eles: a geração da função de detecção através da diferença espectral e a geração desta através do desvio de fase do sinal.

A primeira abordagem utilizada no processo de geração da função de detecção foi aplicada com o uso da diferença espectral. Este método baseia-se no princípio de que inícios de execuções de notas apresentam variações em espectros adjacentes do sinal. Para medir esta variação, deve ser analisada a diferença da magnitude ao longo dos canais.

Este método, assim como outros analisados neste trabalho, utiliza o procedimento da Transformada de Fourier de curto tempo para segregar o sinal em diversas janelas, aplicado sobre o sinal após a análise do conteúdo do arquivo de música. Para isso, os dados do arquivo são separados em janelas de determinado tamanho, definido através de parâmetro, e, sobre cada janela, é aplicada a Transformada rápida de Fourier para calcular a Transformada Discreta de Fourier, que permite visualizar o sinal no domínio de frequência do sinal.

O segundo método utilizado neste trabalho para gerar a função de detecção foi o método do desvio de fase. Assim como o método da diferença espectral, também é baseado no domínio de frequência do sinal para calcular a diferença entre a fase de duas janelas em sequência. Desta forma, pontos com grande variância no desvio da fase entre janelas indicam pontos com possíveis indicadores de inícios de notas.

4.2.1.1 Diferença espectral

A partir do cálculo da STFT sobre o sinal, torna-se possível calcular a diferença entre dois espectros sequenciais do sinal, com o intuito de gerar um fluxo que represente a diferença espectral ao longo do sinal, sendo esta, por sua vez, a função de detecção deste sinal. Para calcular este fluxo, os espectros do sinal são processados iterativamente, sempre sendo comparados com o espectro diretamente anterior. Esta diferença, portanto, é adicionada ao fluxo espectral do sinal.

A diferença entre dois espectros é medida através da soma da diferença de todos os *bins* dos dois espectros em evidência, conforme pode ser visualizado na Tabela 1:

Tabela 1 – Exemplo de cálculo de diferença espectral para dois espectros

Bins do espectro X	*Bins* do espectro X+1	Diferença espectral
0,029541916	0,051240577	0,021698661
0,016871861	0,120628928	0,103757067
0,129862669	0,133251655	0,003388986
0,02265725	0,003200291	0,019456959
0,015639681	0,002279173	0,013360508
0,0143437	0,001922668	0,012421032
Total		0,174083213

Fonte: do autor

Assim, o valor total da somatória das diferenças, conforme observado na Tabela 1, representa o instante X na função de detecção, variando através das janelas da STFT.

Desenvolvida neste protótipo, a classe *SpectralFlux* tem como responsabilidade aplicar o conceito da fórmula citada sobre os dados recebidos (entrada) para retornar os instantes nos quais ocorrem os *onsets*. O método *getevents* segrega os dados em janelas de tamanho *windowsize* por meio do método *frames*, que recebe o tamanho da janela e retorna uma matriz dos dados da música. Após esta etapa, para cada janela é aplicada a FFT, possibilitando a manipulação dos dados no domínio de frequência, o que permite que sejam feitos os comparativos de diferença de fluxo.

Assim, para cada janela do espectro, são somadas as diferenças entre todos os seus *bins*. Tal tarefa fica sob responsabilidade da função *__getflux*, que itera sobre todos os *bins* de duas janelas adjacentes, somando as diferenças de magnitude encontradas, ou seja, valores negativos são considerados de forma positiva. Assim, o método *getevents* itera sobre as janelas do espectro, calculando a diferença espectral de cada uma destas, e insere em um vetor, que representa a função de detecção desta abordagem.

Figura 15 – Código-fonte do método de diferença espectral do protótipo

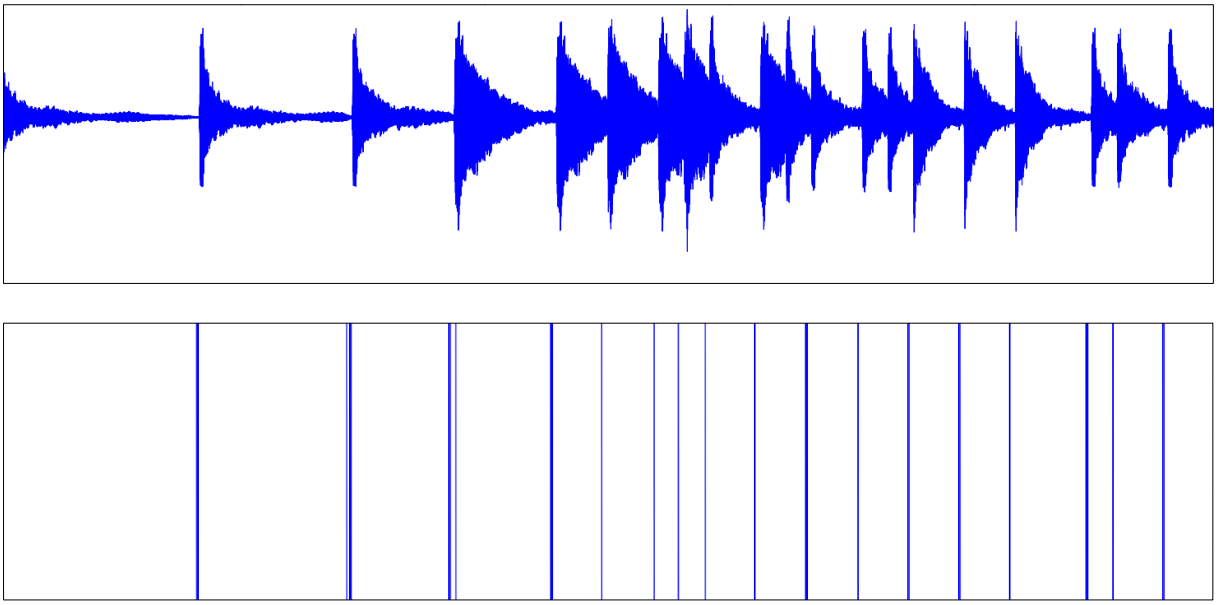
```
def getevents(self, data, windowsize=1024):
    # Retorna espectro dos frames do sinal
    spectrum = [f.fft() for f in data.frames(windowsize)]
    # Calcula o fluxo espectral
    spectralflux = []
    for i in range(1, len(spectrum)):
        flux = self.__getflux(spectrum[i], spectrum[i - 1])
        spectralflux.append(flux)
    …


def __getflux(self, currspectrum, prevspectrum):
    flux = 0
    for bin in range(0, len(currspectrum)):
        # Calcula a diferença do fluxo em módulo
        dflux = abs(currspectrum[bin]) - abs(prevspectrum[bin])
        if dflux < 0:
            dflux *= -1
        flux += dflux
    return flux
```

Fonte: do autor

A partir deste ponto, a função de detecção está gerada e já pode ser utilizada pelas rotinas de detecção de picos para extração dos *onsets*. A Figura 16 apresenta o resultado da aplicação do método de detecção de *onsets* através deste método e do método de detecção de picos (abordado no subcapítulo 4.2.2) acima de um limiar global:

Figura 16 – Formato de onda de áudio sintetizado e *onsets* detectados do trecho



Fonte: do autor

Assim, pode ser observado que este método apresenta algumas detecções incorretas, tanto na falha da detecção quanto na detecção de múltiplos eventos de *onset* para um único início de nota. Porém, ressalta-se que, neste cenário, o método de detecção de picos também pode estar influenciando no resultado apresentado, pois a abordagem utilizada para detectar os picos na função de detecção aponta quais são os *onsets* extraídos da função.

4.2.1.2 Desvio de fase

O método de detecção de *onsets* por meio da geração de função de detecção a partir do desvio de fase, assim como o método da diferença espectral, utiliza a STFT em seu estágio inicial. No entanto, a diferença em relação ao método da diferença espectral está na grandeza que é analisada. Enquanto o procedimento da diferença espectral analisa as variâncias entre as magnitudes do espectro, o desvio de fase analisa unicamente a diferença entre as fases de duas janelas adjacentes.

Assim como na diferença espectral, no desvio de fase a diferença também é calculada a partir da soma da diferença de todos os *bins* de duas janelas em análise. A Tabela 2 exemplifica o cálculo do mesmo trecho de áudio utilizado para elaboração da Tabela 1.

Tabela 2 – Exemplo de cálculo de desvio de fase para dois espectros

Fase dos *Bins* do espectro X	Fase dos *Bins* do espectro X+1	Desvio de fase
3,14159265359	0,00000000000	3,141592654
0,25374496802	-1,67270216935	1,926447137
2,81334503390	-1,20493514056	4,018280174
2,52631409434	-0,67638266354	3,202696758
2,83027486794	-0,33021795792	3,160492826
3,14159265359	0,00000000000	3,141592654
Total		18,5911022

Fonte: do autor

Assim como no método da diferença espectral, a diferença entre as duas janelas é o valor adicionado à função de detecção. O protótipo desenvolvido implementa o método *getevents* para efetuar tal cálculo:

Figura 17 – Código-fonte do método de desvio de fase do protótipo

```
def getevents(self, data, size=1024, windowsize=25):
    # Retorna espectro dos frames do sinal
    spectrum = [f.fft() for f in data.frames(size)]
    # Calcula desvio de fase
    phasedeviation = []
    for i in range(1, len(spectrum)):
        deviation = self.__getflux(spectrum[i], spectrum[i - 1])
        phasedeviation.append(deviation)
    …
```

Fonte: do autor

O método *__getflux*, utilizado pelo método *getevents* no trecho de código-fonte da Figura 17, é responsável – assim como no cálculo do fluxo espectral – por calcular a diferença de fase de todos os *bins* presentes nas janelas. Conforme pode ser observado na Figura 18, a comparação das fases dá-se através da diferença dos ângulos de dois *bins*, sendo somada para cada *bin* da janela:

Figura 18 – Código-fonte do cálculo do fluxo de uma janela no método de desvio de fase

```
def __getflux(self, currspectrum, prevspectrum):
    flux = 0
    for bin in range(0, len(currspectrum)):
        currphase = np.angle(currspectrum[bin])
        prevphase = np.angle(prevspectrum[bin])
        diff = currphase - prevphase
        if diff < 0:
            diff *= -1
        flux += diff
    return flux
```

Fonte: do autor

Portanto, o resultado do método *getevents* é a função de detecção deste método, seguindo a mesma abordagem adotada para a detecção através da diferença espectral. Seguindo este mesmo modelo, após esta etapa, a função já pode ser analisada para detecção de picos a fim de detectar *onsets*.

4.2.2 Detecção de *offsets*

A detecção de *offsets* desenvolvida neste protótipo aplica abordagens semelhantes aos métodos para detecção de *onsets*, conforme Liang et al. (2015). Para isso, a diferença espectral, também utilizada neste protótipo, foi empregada para detectar os eventos de *offsets*. Esta etapa utiliza a rotina já descrita na seção 4.2.1. No entanto, antes de aplicar a fórmula sobre a função, os dados foram invertidos para ser possível aplicar o método da diferença espectral, com o intuito de detectar uma variação inversa, ou seja, momentos em que o sinal sofre desvanecimento.

Assim, foi desenvolvido o método *getoffsets*, responsável por retornar os *offsets* do áudio através do uso da diferença espectral. Este método, assim como o método *getevents* da classe *SpectralFlux*, os dados do áudio são recebidos via parâmetro. No entanto, nesta função, os dados são invertidos antes de serem passados para a função de geração da função de detecção, para que esta retorne diferenças de variação nas amplitudes no sentido inverso. A implementação pode ser observada na Figura 19.

Figura 19 – Código-fonte do método de diferença espectral na detecção de *offsets*

```
def getoffsets(data, size=1024, windowsize=25):
    spectralflux = SpectralFlux()
    datareversed = data.copy()
    datareversed = datareversed[::-1]
    return spectralflux.getevents(datareversed, size, windowsize)
```

Fonte: do autor

4.2.3 Detecção de picos

As seções 4.2.1 e 4.2.2 descreveram o processo desenvolvido neste protótipo para detectar *onsets* e *offsets*. No caso deste trabalho, para ambos os processos, o resultado final de tais processos é uma função de detecção. No caso da detecção de *onsets*, os valores da função de detecção são comparados com um determinado limiar – definido nesta etapa do processo – a fim de detectar estes *onsets*, enquanto na detecção de *offsets*, nos métodos que também utilizam a função de detecção, tais valores representam os *offsets* no áudio.

No desenvolvimento deste protótipo, foi utilizado o limiar fixo, que compreende um único limiar global para todo o sinal. Assim, o método responsável por detectar os picos de uma determinada função de detecção deve comparar tal limiar com o maior valor de uma janela. Por fim, todos os valores acima deste limiar representam momentos de troca de estado no sinal. Ou seja, representam um *onset* ou *offset* de uma nota.

O método *getpeaksbyaverage*, implementado neste protótipo, é o responsável por fazer tal processamento neste trabalho, conforme pode ser observado na Figura 20.

Figura 20 – Código-fonte do método de detecção dos picos a partir da média da função de detecção

```
def getpeaksbyaverage(self):
    …
    # Calcula média da função de detecção
    average = np.average(data)
    # Busca picos (valores acima da média) na função de detecção
    peaks = []
    while i < len(data):
        # Retorna maior valor da janela
        max = data[i:j].max()
        # Adiciona à lista de picos os valores acima da média
        if max > average:
            pos = data[i:j].argmax()
            if (i + pos) not in peaks:
                peaks.append(i + pos)
        i += size
        j += size
    return peaks
```

Fonte: do autor

Considerando que este método retorna os picos encontrados ao longo da função de detecção, é possível utilizar tais valores para definir os instantes em que eventos de *onsets* e *offsets* ocorrem.

4.3 ESTIMATIVA DA FREQUÊNCIA FUNDAMENTAL

O último processamento feito pelo protótipo a partir da análise dos dados do áudio é a detecção da frequência fundamental ao longo do sinal, a fim de identificar as notas presentes no sinal. Para executar tal tarefa, o método de detecção através da função de autocorrelação foi utilizado. Conforme foi discutido na seção 3.1.2, esta abordagem compara a onda com ela mesma deslocada no tempo. Assim, é possível calcular o nível de similitude da onda ao longo dela mesma, tornando possível detectar a frequência fundamental a partir da extração do ponto com maior grau de similaridade diferente de zero, em que a onda é igual, pois está sendo comparada consigo sem deslocamento no tempo.

Para isso, foi implementado o método *detectpitch*, que recebe os dados a serem analisados via parâmetros e retorna um vetor com as notas – já convertidas para a escala MIDI

– ao longo do sinal. Após este processamento, este vetor é cruzado com os dados das detecções de *onsets* e *offsets*. O método *detectpitch* pode ser visualizado na Figura 21.

Figura 21 – Código-fonte do método de detecção da nota por meio da autocorrelação

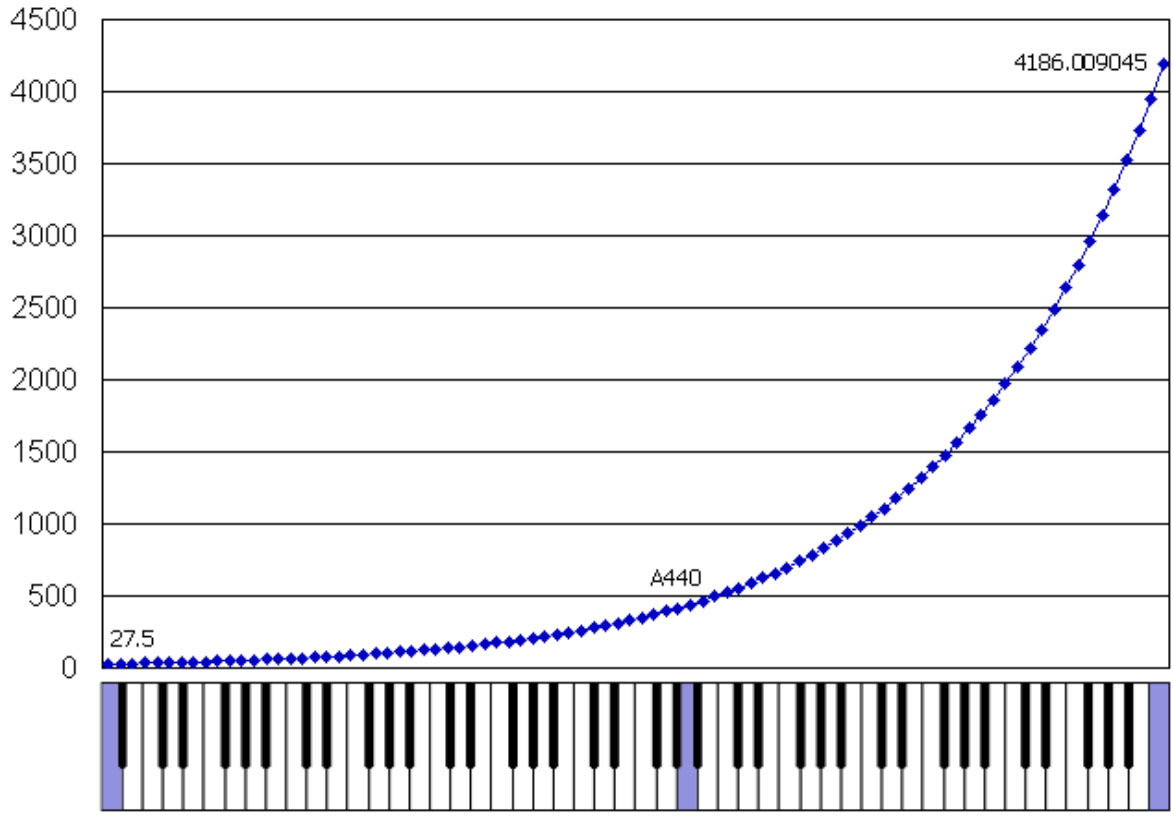
```
def detectpitch(data, minHz = 82, maxHz = 330):
    # Calcula quantidade mínima e máxima dos períodos da onda para
comportar as frequências
    maxperiodsamples = np.floor(data.samplerate / minHz)
    minperiodsamples = np.floor(data.samplerate / maxHz)
    # Calcula quantidade de janelas e dados analisados
    qtywindows = len(data) // maxperiodsamples
    qtysamples = int(qtywindows * maxperiodsamples)
    # Percorre as janelas do áudio
    result = []
    for p in range(0, qtysamples, int(maxperiodsamples)):
        # Extrai dados da janela corrente
        window = data[p:p + maxperiodsamples]
        # Aucorrelaciona os dados da janela corrente
        xcorr = np.correlate(window, window, mode='full')
        # Busca ponto de maior similitude
        max = np.argmax(xcorr[maxperiodsamples +
minperiodsamples:(maxperiodsamples * 2) + 1])
        max += minperiodsamples
        # Converte ponto máximo para frequência
        pitch = (max / data.samplerate)
        f0 = (1 / pitch)
        # Converte frequência para nota MIDI
        midiNote = Midi.freqtonote(f0)
        for i in range(0, int(maxperiodsamples)):
            result.append(midiNote)
    return result
```

Fonte: do autor

Este método recebe via parâmetro, além dos dados do áudio, a frequência mínima e a máxima a serem consideradas na extração. Tais informações são necessárias para definir o tamanho da janela no processo de detecção do *pitch* através da autocorrelação, que deve possuir o dobro do tamanho do período da onda para ser possível aplicar este procedimento de forma correta.

Conforme pode ser observado na Figura 21, os valores padrão das frequências mínima e máxima são, consecutivamente, 82Hz e 330Hz, pois trata-se do intervalo de frequências que um violão pode alcançar normalmente. No caso do piano, estes valores alteram para 27Hz e 4186Hz, sendo estas as notas mais graves e as mais agudas alcançadas pelos dois instrumentos. A Figura 22 demonstra todas as frequências alcançadas por um piano:

Figura 22 – Frequências de um piano de 88 teclas



Fonte: adaptado de Caprani (2011)

Após detectar a frequência de cada trecho do áudio, o método detectpitch converte a frequência detectada para a escala MIDI utilizando a função freqtonote da classe Midi. Esta função aplica a Equação 16. A implementação pode ser observada na Figura 23.

Figura 23 – Código-fonte para conversão de frequências em notas na escala MIDI

```
class Midi:
    NOTE_A440 = 69
    FREQ_A440 = 440
    QTY_SEMITONS = 12
    @staticmethod
    def freqtonote(freq):
        note = Midi.NOTE_A440 + Midi.QTY_SEMITONS *
               math.log(freq / Midi.FREQ_A440, 2)
        return int(note)
```

Fonte: do autor

O Quadro 1 apresenta a conversão de todas as notas presentes na escala MIDI, com sua frequência, oitava, nome da nota e identificador na escala MIDI.

Quadro 1 – Conversão de frequências para escala MIDI

Nota	MIDI	Hz	Nota	MIDI	Hz	Nota	MIDI	Hz	Nota	MIDI	Hz
C	0	8.1	G#1	32	51.9	E4	64	329.6	C 7	96	2093
C#	1	8.6	A1	33	55.0	F4	65	349.2	C# 7	97	2217
D	2	9.1	A#1	34	58.2	F#4	66	369.9	D 7	98	2349
D#	3	9.7	B1	35	61.7	G4	67	391.9	D# 7	99	2489
E	4	10.3	C2	36	65.4	G#4	68	415.3	E 7	100	2637
F	5	10.9	C#2	37	69.2	A4	69	440.0	F 7	101	2793
F#	6	11.5	D2	38	73.4	A#4	70	466.1	F# 7	102	2960
G	7	12.2	D#2	39	77.7	B4	71	493.8	G 7	103	3136
G#	8	12.9	E2	40	82.4	C5	72	523.2	G# 7	104	3322
A	9	13.7	F2	41	87.3	C#5	73	554.3	A 7	105	3520
A#	10	14.5	F#2	42	92.4	D5	74	587.3	A# 7	106	3729
B	11	15.4	G2	43	97.9	D#5	75	622.2	B 7	107	3951
C0	12	16.3	G#2	44	103.8	E5	76	659.2	C 8	108	4186
C#0	13	17.3	A2	45	110.0	F5	77	698.4	C# 8	109	4434
D0	14	18.3	A#2	46	116.5	F#5	78	739.9	D 8	110	4698
D#0	15	19.4	B2	47	123.4	G5	79	783.9	D# 8	111	4978
E0	16	20.6	C3	48	130.8	G#5	80	830.6	E 8	112	5274
F0	17	21.8	C#3	49	138.5	A5	81	880.0	F 8	113	5587
F#0	18	23.1	D3	50	146.8	A#5	82	932.3	F# 8	114	5919
G0	19	24.4	D#3	51	155.5	B5	83	987.7	G 8	115	6271
G#0	20	25.9	E3	52	164.8	C6	84	1046	G# 8	116	6644
A0	21	27.5	F3	53	174.6	C#6	85	1108	A 8	117	7040
A#0	22	29.1	F#3	54	184.9	D6	86	1174	A# 8	118	7458
B0	23	30.8	G3	55	195.9	D#6	87	1244	B 8	119	7902
C1	24	32.7	G#3	56	207.6	E6	88	1318	C 9	120	8372

Quadro 1 – Conversão de frequências para escala MIDI

(continuação)

Nota	MIDI	Hz	Nota	MIDI	Hz	Nota	MIDI	Hz	Nota	MIDI	Hz
C#1	25	34.6	A3	57	220.0	F6	89	1396	C# 9	121	8869
D1	26	36.7	A#3	58	233.0	F#6	90	1480	D 9	122	9397
D#1	27	38.8	B3	59	246.9	G6	91	1568	D# 9	123	9956
E1	28	41.2	C4	60	261.6	G#6	92	1661	E 9	124	10548
F1	29	43.6	C#4	61	277.1	A6	93	1760	F 9	125	11175
F#1	30	46.2	D4	62	293.6	A#6	94	1864	F# 9	126	11839
G1	31	48.9	D#4	63	311.1	B6	95	1975	G 9	127	12543

Fonte: adaptado de Brice (2002)

Destaca-se nesta tabela o A440, na medida que é a base para o cálculo da nota na escala MIDI. Portanto, esta foi a abordagem utilizada neste protótipo para estimar as frequências presentes na música.

5 RESULTADOS

A partir do desenvolvimento do protótipo, os métodos utilizados foram submetidos a testes para validá-los – e também para validar a implementação como um todo. Desta forma, foram analisadas as taxas de acerto na detecção de *onsets*, *offsets* e detecção das notas. Este capítulo descreve o procedimento adotado para executar tais validações.

A primeira etapa consistiu em converter áudios monofônicos no formato MIDI para formato WAV – formato interpretado pelo protótipo – seguida pela análise dos eventos no arquivo MIDI. Os eventos (*onsets, offsets* e notas) são armazenados em mensagens dentro dos arquivos MIDI e podem ser visualizados a partir da utilização da biblioteca Mido, conforme a Figura 24:

Figura 24 – Modelo de visualização dos eventos extraídos de arquivo MIDI

```
{Message} note_on channel=0 note=72 velocity=96 time=0
{Message} note_off channel=0 note=72 velocity=64 time=228
{Message} note_on channel=0 note=74 velocity=96 time=0
{Message} note_off channel=0 note=74 velocity=64 time=228
```

Fonte: do autor

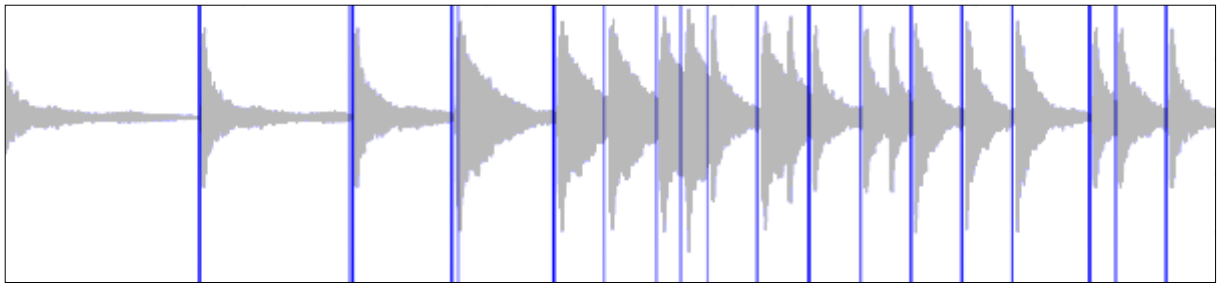
onde mensagens *note_on* são eventos de *onset,* enquanto mensagens de *note_off* são os eventos de *offset*. Ambas as mensagens carregam a nota que está sendo iniciada ou finalizada, assim como a velocidade da execução de tal evento, apesar de tal aspecto não ter sido abordado neste trabalho. Desta forma, foi possível comparar os eventos originais do áudio em relação aos detectados pelo protótipo apresentado.

Para analisar o processo transcrição, foi convertido um arquivo de formato MIDI para WAV, utilizando o piano como instrumento para execução, com 40 *onsets* e 40 *offsets*. Na sequência, foram comparados os *onsets* detectados pelo protótipo com os originais do áudio, com a garantia de que estavam sincronizados, ou seja, de que não houve variação no momento de detecção. Foram verificados, ainda, *onsets* detectados que não estavam presentes nos áudios, assim como *onsets* que estavam presentes e que o protótipo não foi capaz de detectar. Esta mesma abordagem foi utilizada na análise dos resultados da detecção de *offsets*. Às detecções corretas dá-se o nome de Verdadeiro Positivo (VP). Os eventos detectados não existentes no original recebem o nome de Falso Positivo (FP), enquanto os não detectados são chamados de Falso Negativo (FN).

O método da diferença espectral utilizado na detecção de *onsets* apresentou uma taxa de 90% para VP, 33% de FP e somente 10% para FN. Bello et al. (2005) obtiveram taxas entre 80,4% e 87,1% de VP na detecção de *onsets* através deste mesmo método. No trabalho de Bello et al. (2005) a taxa de FP variou entre 1,6% e 8,6%. Esta diferença entre os resultados obtidos pelo protótipo proposto e os obtidos por Bello et al. (2005) deve-se, primordialmente, à não utilização de pré-processamento e pós-processamento na montagem de função de detecção. O reflexo apresentado pela não utilização destes métodos demonstra a importância destas etapas, na medida em que são capazes de suavizar o sinal, removendo oscilações encontradas no áudio. Assim, por não ter sido implementado, múltiplos *onsets* foram detectados com intervalos de milissegundos de diferença para um único *onset* no áudio original. Por este motivo, este trabalho apresentou um alto percentual de FP.

A Figura 25 compara a amplitude do sinal ao longo do tempo com os momentos de *onsets* detectados pelo protótipo. É possível observar, na figura, este comportamento de falsas detecções, em que os *onsets* são identificados pelas linhas azuis e sua intensidade indica a quantidade de eventos detectados neste instante.

Figura 25 –Visualização dos *onsets* detectados em comparativo com onda do sinal



Fonte: do autor

A análise da detecção de *offsets* por meio do método de diferença espectral apresentou resultados bastante divergentes dos *onsets*, demonstrando que tal método não pode ser utilizado por uma ferramenta que deseja transcrever uma música de forma automática, pois suas taxas de acerto são muito pequenas. A taxa de VP foi de 45%, valor inferior ao obtido na detecção de *onsets* com este método (em seu modo reverso, no entanto). As taxas de FP e FN também apresentam resultados piores em relação à análise da extração de *onsets*, apresentando taxas de 54,5% e 45%, respectivamente.

Na detecção de *offsets*, o método da diferença espectral não apresentou bons resultados. Ao contrário do que acontece com os *onsets* – em que o sinal sofre uma oscilação significativa em um curto período –, o desvanecimento da nota ocorre de forma mais suave e

durante um período maior de tempo. Desta forma, torna mais difícil a detecção de um instante com uma variação relevante no fluxo do espectro de duas janelas em sequência. Outro fator observado que influencia negativamente o desempenho deste método é a instabilidade do sinal ao longo da etapa transiente de uma nota, fazendo com que oscilações do sinal gerem, incorretamente, eventos de *offsets* para as rotinas de detecção.

Além disso, foram analisadas as taxas de acerto ao extrair as notas do áudio a partir da detecção de *onsets* e *offsets*. Ou seja, para cada evento detectado, a nota deste determinado trecho foi obtida e comparada com o áudio original. O método da autocorrelação apresentou um resultado relativamente interessante, com uma taxa de acerto de aproximadamente 70%, considerando somente os pontos que detectaram corretamente *onsets* e *offsets*, para evitar que pontos deslocados no tempo interferissem no resultado.

Conforme pode ser observado nos resultados obtidos, o protótipo foi capaz de detectar corretamente um alto percentual de *onsets* e também apresentou boas taxas de acerto ao estimar as frequências presentes no sinal. Contudo, a detecção de *offsets* apresentou resultados abaixo do esperado, dificultando o uso desta abordagem em cenários reais. Nas considerações finais deste trabalho, a seguir, são apresentadas novas situações que podem estudadas a fim de melhorar os resultados obtidos e solucionar os problemas detectados.

CONSIDERAÇÕES FINAIS

A transcrição automática de áudios vem crescendo ao longo dos últimos anos, sendo objeto de estudo de diversas pesquisas. Assim, este trabalho apresentou um estudo sobre o processo de transcrição automática para áudios monofônicos. Para isso, as principais etapas deste processo foram analisadas, sendo elas: detecção de *onsets*, detecção de *offsets* e estimativa da frequência fundamental, a fim de detectar quais são as notas presentes no sinal.

A detecção de *onsets* caracteriza – em diversos métodos de transcrição de áudio – a principal etapa deste processo, na medida em que é responsável por elaborar a função de detecção utilizada pela própria rotina e, posteriormente, na extração das notas do sinal. Este trabalho apresentou diferentes abordagens para a detecção de *onsets* e suas principais características. Contudo, o foco do trabalho manteve-se na função da diferença espectral, uma vez que esta abordagem pôde ser utilizada na etapa de detecção de *offsets*.

Ao contrário da detecção de *onsets*, a detecção de *offsets* ainda não é assunto de uma quantidade significativa de estudos exclusivamente dedicados ao tema, por ser uma etapa menos crítica no processo. No entanto, a fim de desenvolver ferramentas capazes de transcrever uma música fielmente, este aspecto também deve ser estudado de forma mais incisiva. A abordagem utilizada neste trabalho foi apresentada por Liang et al. (2015) e consiste em aplicar métodos conhecidos, utilizados na detecção de *onsets*, ao sinal de trás para frente ou sobre o sinal invertido. Esta abordagem baseia-se no princípio de que *offsets*, assim como os *onsets*, tendem a causar oscilações na distribuição de energia do sinal. Por esta razão, o presente trabalho utilizou o método da diferença espectral como método para detecção de *offsets*, o que não trouxe os resultados positivos esperados.

O desenvolvimento de um protótipo serviu como ferramenta para validação das abordagens estudadas neste trabalho. Assim, permitiu que os dados detectados fossem cruzados com dados conhecidos para analisar a validade dos métodos utilizados. O método da diferença espectral para a detecção de *onsets* apresentou taxas de VP de aproximadamente 90%, porém apresentou falhas ao detectar múltiplos *onsets* para um único *onset* real. Tal problema ocorre, principalmente, pelas oscilações do sinal ao longo do ataque de uma nota. As etapas de pré-processamento e pós-processamento da geração da função de detecção podem corrigir este problema, na medida em que têm como objetivo suavizar o sinal e a função de detecção para a etapa de detecção de picos.

Na detecção de *offsets*, o método da diferença espectral não foi capaz de repetir as mesmas taxas alcançadas na detecção de *onsets*, apresentando taxas de VP de 45%. Ao contrário da etapa de ataque de uma nota¸ em que o sinal tende a sofrer alterações em um intervalo curto de tempo, a etapa de desvanecimento da nota tende a ser mais suave. Assim, o método da diferença espectral não foi capaz de diferenciar tais eventos do restante do sinal. Esta etapa ainda encontrou problemas em virtude de que o ataque de uma nota sofre muitas oscilações antes de sua estabilização, o que implica em quedas na distribuição de energia do sinal, fazendo com que o protótipo detectasse *offsets* no período de *onset* de uma nota. Desta forma, avalia-se válida a abordagem utilizada por trabalhos como o de Bello, Monti e Sandler (2000), em que os *offsets* são detectados a partir da queda da energia do sinal após a detecção de um *onset*, ou seja, está diretamente ligado ao *onset*, diminuindo a possibilidade de erro nesta abordagem.

O protótipo implementou o método da autocorrelação para detectar as notas presentes no sinal no momento dos *onsets* e dos *offsets* detectados pelo próprio protótipo. Assim, os resultados desta etapa foram analisados a partir dos VP encontrados nos demais métodos, atingindo taxas de VP de 70%. O tamanho da janela utilizada no método da autocorrelação influencia diretamente este processo, de tal forma que notas podem ser incorretamente identificadas por não estarem presentes em uma única janela ou por existirem múltiplas notas dentro de uma única janela. O trabalho de Tavares, Barbedo e Lopes (2007) estuda abordagens para utilização de janelas de tamanho variável – o que permitiria que tais problemas fossem solucionados – e poderia ser posteriormente estudado.

O formato adotado por este trabalho para transcrição automática de música também pode ser revisto, utilizando abordagens como a utilizada no trabalho de Trevilatto Jr, Barbedo e Lopes (2005), no qual a extração das notas presentes ao longo de todo o sinal norteia o restante do processo. Estes autores apontam que *onsets* e *offsets* podem ser detectados a partir da mudança da nota detectada ao longo de múltiplas janelas de determinado tamanho, ou seja, um *onset* é detectado a partir da detecção da mesma nota após análise de quatro janelas consecutivas. Na abordagem proposta no presente trabalho, no entanto, a extração das notas se faz necessária somente nos momentos em que eventos de *onsets* ou *offsets* foram detectados.

Neste trabalho foram abordados métodos capazes de detectar os principais eventos de uma música, porém diversas outras características inerentes a uma música não foram analisadas, como a presença de *legatos*, *glissando* e velocidade da música, entre outras. Assim, novos estudos podem abordar novos métodos que sejam capazes de extrair tais informações de sinais de áudio, permitindo que seja possível transcrever a música de uma forma mais fiel.

O aprimoramento dos algoritmos existentes e o desenvolvimento de novos – a fim de tornar possível transcrever áudios polifônicos – são parte de um processo de evolução natural que já está em progresso, na qual estudos surgem com esta vertente. Assim, futuros trabalhos podem estudar tais técnicas e propor métodos para aprimorar o processo de transcrição automática de música. As aplicações para ferramentas capazes de transcrever áudios polifônicos crescem ainda mais se comparadas a áudios monofônicos, aumentando a importância de seu avanço.

Referências Bibliográficas

BELLO, Juan Pablo. **Towards the Automated Analysis of Simple Polyphonic Music**: A Knowledge-based Approach. 2003. 239 f. Tese (Doutorado em Engenharia Eletrônica), University of London, Queen Mary, 2003.

BELLO, Juan Pablo; DAUDET, Laurent; ABDALLAH, Samer; DUXBURY, Chris; DAVIES, Mike; SANDLER, Mark B. A Tutorial on Onset Detection in Music Signals. **IEEE Transactions on speech and audio processing**, v. 14, n. 5, p. 1035-1047, set. 2005.

BELLO, Juan Pablo; MONTI, Giuliano; SANDLER, Mark. Techniques for Automatic Music Transcription. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MUSIC INFORMATION RETRIEVAL (ISMIR), 1., 2000, Plymouth. **Anais**... Plymouth: UMass, 2000.

BENETOS, Emmanouil. **Automatic Transcription of Polyphonic Music Exploiting Temporal Evolution**. 2012. 216 f. Tese (Doutorado em Engenharia Eletrônica), University of London, Queen Mary, 2012.

BENETOS, Emmanouil; DIXON, Simon; GIANNOULIS, Dimitrios; KIRCHHOFF, Holger; KLAPURI, Anssi. Automatic music transcription: breaking the glass ceiling. In: PROCEEDINGS OF THE INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE (ISMIR), 13., 2012, Porto. **Anais**... Porto: INESC TEC, 2012. p. 379-384.

_____. Automatic music transcription: challenges and future directions. **Journal of Intelligent Information Systems**, v. 41, n. 3, p. 407-434, dez. 2013.

BRICE, Richard. **Midi Notes and Frequencies.** 2002. Disponível em: <http://www.richardbrice.net/midi_notes.htm>. Acesso em: 15 nov. 2016.

CAPRANI, Ole. **Notes, Midi Numbers and Note Frequencies.** 2011. Disponível em: <http://www.cs.au.dk/~dsound/DigitalAudio.dir/MidiAndFrequencies/MidiAndNoteFrequencies.html>. Acesso em: 15 nov. 2016.

CHEVEIGNÉ, Alain de; KAWAHARA, Hideki. YIN, a fundamental frequency estimator for speech and music. Journal of the Acoustical Society of America (JASA), v. 111, n. 4, p. 1917–1930, abr. 2002.

COLLINS, Nick. A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions. In: 118TH CONVENTION OF THE AUDIO ENGINEERING SOCIETY, 118., 2005, Barcelona. **Anais**… Barcelona: 2005.

_____. Using a pitch detector for onset detection. In: PROCEEDINGS OF 6TH INTERNATIONAL CONFERENCE ON MUSIC INFORMATION RETRIEVAL (ISMIR), 6., 2005, Londres. **Anais**… Londres: University of London, 2005. p. 100-106.

CUADRA, Patricio de la; MASTER, Aaron; SAPP, Craig. Efficient Pitch Detection Techniques for Interactive Music. In: PROCEEDINGS OF THE INTERNACIONAL

COMPUTER MUSIC CONFERENCE (ICMC), 27., 2001, Havana. **Anais**… Havana: 2001. p. 87-90.

DINIZ, Felipe Castello da Costa Betrão. **Transcrição musical automática usando representação frequencial eficiente por banco de filtros de alta**. 2009. 170 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-graduação em Engenharia Elétrica, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, 2009.

DIXON, Simon. Onset detection revisited. In: PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFx-06), 9., 2006, Montreal. **Anais**… Montreal: McGill University, 2006. p. 133-137.

_____. Simple Spectrum-Based Onset Detection. In: PROCEEDINGS OF MUSIC INFORMATION RETRIEVAL EVALUATION EXCHANGE (MIREX), 2., 2006, Illinois. **Anais**… Illinois: IMIRSEL, 2006. p. 62-66.

DUXBURY, Chris; BELLO, Juan Pablo; DAVIES, Mike; SANDLER, Mark. A combined phase and amplitude based approach to onset detection for audio segmentation. In: PROCEEDINGS 4TH EUROPEAN WORKSHOP ON IMAGE ANALYSIS FOR MULTIMEDIA INTERACTIVE SERVICES (WIAMIS-03), 4., 2003, Londres. **Anais**… Londres: Queen Mary University of London, 2003. p. 275-280.

_____. Complex domain onset detection for musical signals. In: PROCEEDINGS OF THE 6TH INTERNACIONAL CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFx-03), 6., 2003, Londres. **Anais**... Londres: Queen Mary University of London, 2003.

DZIUBIŃSKI, Marek; KOSTEK, Bozena. High Accuracy and Octave Error Immune Pitch Detection Algorithms. **Archives of Acoustics**, v. 29, n. 1, p. 1-21, 2004.

ELLIS, Daniel P. W. Extracting information from music audio. **Communications of the ACM**, v. 49, n. 8, ago. 2006.

GERHARD, David. **Pitch Extraction and Fundamental Frequency:** History and Current Techniques. Canada: University of Regina, 2003. 23 p. ISBN 0-7731-0455-0.

KLAPURI, Anssi. Automatic transcription of music. In: PROCEEDINGS OF THE STOCKHOLM MUSIC ACOUSTICS CONFERENCE (SMAC 03), 3., 2003, Suécia. **Anais**… Suécia: Tampere University of Technology, 2003.

_____. **Signal Processing Methods for the Automatic Transcription of Music**. 2004. 112 f. Tese (Doutorado em Tecnologia), Tampere University of Technology, Tampere, 2004.

LIANG, Che-Yuan; SU, Li; YANG, Yi-Hsuan; LIN, Hsin-Ming. Musical offset detection of pitched instruments: the case of violin. In: PROCEEDINGS OF THE INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE (ISMIR), 16., 2015, Málaga. **Anais**... Málaga: Universidad de Málaga, 2015. p. 281-287.

MCLEOD, Philip. **Fast, Accurate Pitch Detection Tools for Music Analysis**. 2008. 181 f. Tese (Doutorado em Ciência da Computação), University of Otago, Dunedin, 2008.

MONTEIRO, Lucas Lago. **Síntese aditiva aplicada à transferência de timbre em vocalise**. 2012. 102 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico**: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico. 2. ed. Novo Hamburgo: Universidade Feevale, 2013. 276 p.

SANTINI, Rose Marie; SOUZA, Rosali Fernandez de. Recuperação da informação de música e a ciência da informação: tendências e desafios de pesquisa. In: ENCONTRO NACIONAL DE PESQUISA EM CIÊNCIA DA INFORMAÇÃO, 8., 2007, Florianópolis. **Anais**... Salvador: UFBA, 2007.

TAVARES, Tiago Fernandes; BARBEDO, Jayme Garcia Arnal; LOPES, Amauri. Transcrição Automática de Sinais de Áudio Monofônico Baseada em Quadros de Tamanho Variável. In: V Congresso de Engenharia de Áudio, 2007, São Paulo. **Anais**... São Paulo, 2007. p. 47-50.

TREVILATTO Jr., Narciso, BARBEDO, Jayme Garcia Arnal; LOPES, Amauri. Transcrição Automática de Sinais de Áudio Monofônico. In: 10 Simpósio Brasileiro de Computação Musical, 2005, São Paulo. **Anais**... São Paulo: UNICAMP, vol. 1, p. 291-294.

ZHOU, Ruohua. **Feature extraction of musical content for automatic music transcription**. 2006. 169 f. Dissertação (Mestrado em Engenharia) – Academia Chinesa de Ciência, École polytechnique fédérale de Lausanne, Lausana, 2006.

ZHOU, Ruohua; REISS, Joshua. Music onset detection combining energy-based and pitch-based approaches. In: PROCEEDINGS OF MIREX AS PART OF 8TH INTERNACIONAL CONFERENCE ON MUSIC INFORMATION RETRIEVAL (ISMIR), 8., Viena. **Anais**… Viena: University of Technology, 2007.

ZORNDORF, Nathan; CARREON, Kristine. **Monophonic Pitch Recognition**. 2013. 84 p. Trabalho de Conclusão de Curso (Engenharia Elétrica) – Curso de Engenharia Elétrica, California Polytechnic State University, San Luis Obispo, 2013.