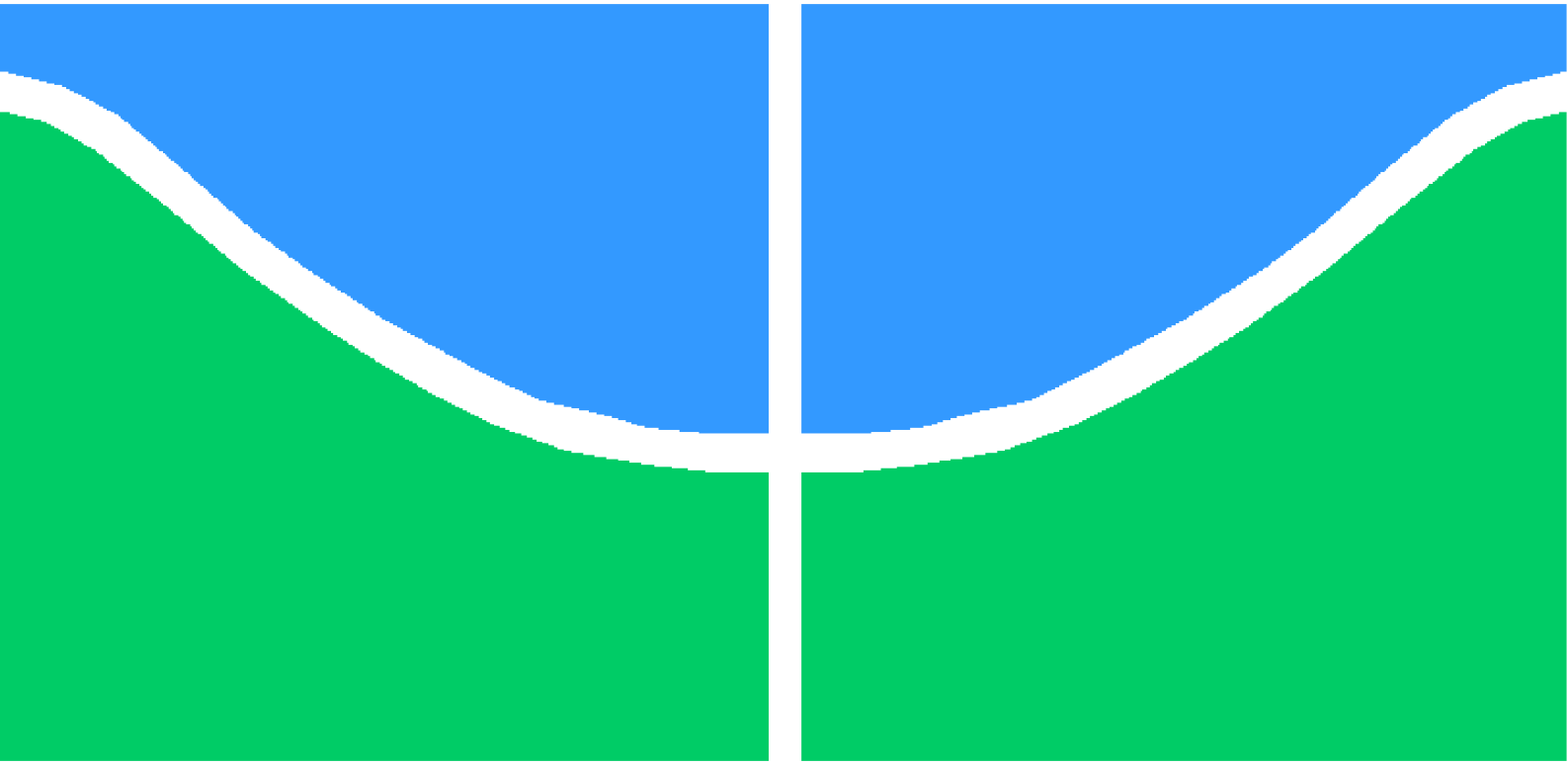

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Solução Computacional para Reconhecimento de Harmonias Musicais

Autor: José Pedro de Santana Neto
Orientador: Dr. Henrique Gomes de Moura
Coorientador: Dr. Paulo Roberto Miranda Meirelles

Brasília, DF
2015

José Pedro de Santana Neto

Solução Computacional para Reconhecimento de Harmonias Musicais

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Henrique Gomes de Moura

Coorientador: Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

2015

José Pedro de Santana Neto
Solução Computacional para Reconhecimento de Harmonias Musicais/ José Pedro de Santana Neto. – Brasília, DF, 2015-
116 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Henrique Gomes de Moura
Coorientador: Dr. Paulo Roberto Miranda Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. Reconhecimento. 2. Acordes. I. Dr. Henrique Gomes de Moura. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Solução Computacional para Reconhecimento de Harmonias Musicais

CDU 02:141:005.6

José Pedro de Santana Neto

Solução Computacional para Reconhecimento de Harmonias Musicais

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 2 de julho de 2015:

Dr. Henrique Gomes de Moura
Orientador

Dr. Paulo Roberto Miranda Meirelles
Coorientador

Dr. Fernando William Cruz
Convidado 1

Dr. Cristiano Jacques Miosso Rodrigues Mendes
Convidado 2

Brasília, DF
2015

Esse trabalho é dedicado às pessoas que quebram coisas para verem como são por dentro.

Agradecimentos

Agradeço primeiramente a Deus, inteligência criadora suprema, por permitir-me nesse mundo, vivendo, aprendendo e contemplando a beleza da natureza primordial de todas as coisas.

A minha amada e querida mãe Francisca, pela paciência, compreensão, tolerância, conselhos, carinho, dedicação, afeto, amizade, silêncio, sorrisos e um intenso amor. Meu primeiro aprendizado na vida mais puro e original de amor foi através dela. Isso me possibilitou a amar verdadeiramente o que faço e ter uma visão de vida mais profunda. Meus sinceros e eternos agradecimentos.

A meu pai Luciano, mesmo não estando presente mais, me inspirou a escolha da minha formação e me ensinou a olhar o mundo com meus próprios olhos.

A meu irmão João, companheiro e amigo de sempre. Seus conselhos e seu exemplo têm me ensinado muito a ser uma pessoa melhor.

A meu padrinho Inácio, meu segundo pai, por seus profundos conselhos sobre a vida e um exemplo para mim de homem honrado e correto.

A meu tio Antônio, por ser o padrinho da minha mãe e assumir o papel de avô na minha vida.

A minha querida tia Naíde, por ter assumido papel de avó na minha vida e ter tido um olhar único sobre minha vida.

A minha madrinha Nevinha, minhas tias Titia, Tia Marli, Tia Gracinha e Tia Bá. O amor delas é indescritível com palavras. A toda minha família pelo apoio, confiança e compressão.

A meus amigos-irmãos Thiago e Leandro, pelo companheirismo indescritível de muitos anos e apoio de sempre.

A minha amiga Marina Shinzato, pelas longas conversas e por ter sido meu ombro forte ao longo desse trabalho.

A minha amiga Anaely, pelo apoio compreensão e inspiração.

A minha amiga Ana Luisa, pelas longas conversas e incrível amizade.

A meus professores de música Boggie e Gedeão por todo conhecimento e inspiração musical.

A meu orientador professor Henrique Moura, pelo exemplo, inspiração, amizade, conselhos, apoio, confiança e investimento de longas conversas. Esse trabalho necessaria-

mente foi fruto de uma orientação em excelência.

A meu co-orientador professor Paulo Meirelles, pelo exemplo e ensinamentos valiosos e práticos sobre o mundo do software e a vida.

Aos professores Hilmer, Milene, Maria de Fátima, Cristiano e Fernando pelos valiosos ensinamentos e exemplos de profissionais-cientistas.

A equipe do LAPPIS pelo suporte e aprendizado na produção de softwares de qualidade.

A professora Suzete e a equipe do MídiaLab por todo aprendizado.

Aos meus amigos da faculdade e companheiros de disciplinas Carlos, Álvaro, Fagner, Eduardo, Wilton, João, Daniel, Matheus, Kleber, Hebert, André Guedes, David, Yeltsin, Wilker, Thaiane, Tomaz, Maxwell, Luiz Oliveira e André Mateus, pela compreensão, apoio e motivação.

Aos meus restantes amigos Luiz Matos, Fábio Costa, Daniel Bucher, Renan, Chico, Leônidas, Lucas, Nayron, Thiago Ribeiro, Marcos Ramos, Cleiton, Marcos Ronaldo, José Alisson, José Alberto, Vilmey, Yan, Igor Josafá, Guilherme Fay, Sérgio, Lucas Kanashiro, Charles Oliveira, Rodrigo, Álex, Jefferson, Alexandre, Matheus Souza, Ana Luiza e outros que esqueci de citar, pelo apoio e zueira de sempre.

E as pessoas que passaram na minha vida e influenciaram de alguma forma nesse trabalho. Meus agradecimentos.

"A vida não é uma sonata que para
realizar sua beleza tem de ser tocada até o fim,
ao contrário, a vida é um álbum de minissonatas.
Cada momento de beleza vivido e amado,
por efêmero que seja, é uma experiência completa
que está destinada à eternidade.
Um único momento de beleza e amor
justifica a vida inteira."
(Rubem Alves)

Resumo

Atualmente, a música está num patamar único no que diz respeito às várias abordagens de se contemplar e se executar e, com isso, a tecnologia vem cada vez mais sendo usada para otimizar os processos musicais. Um dos exemplos de tecnologia são sistemas automáticos de transcrição de música que auxiliam o músico, substituindo por vezes de maneira significativa partituras, tablaturas e cifras. Esse presente trabalho tem como objetivo desenvolver um protótipo de uma solução computacional para reconhecimento de harmonias musicais. Para tal fim, priorizou-se a modelagem matemática da solução: implementação da análise espectral da amostra de áudio, classificação em notas musicais, classificação em acordes com suportes a inversões, transição rítmica, reconhecimento dos padrões harmônicos ao longo do tempo, extração de tonalidade musical e o projeto do sistema solução sistematizado em engenharia de software. O desenvolvimento da solução se deu através de um método de desenvolvimento empírico, iterativo e incremental, utilizando a linguagem de programação Matlab para implementação. De fundamentos teóricos foram utilizados conceitos físicos do som, teoria musical, processamento de sinais e redes neurais artificiais. O desenvolvimento da solução permitiu o reconhecimento de acordes em tríades maiores, menores, aumentados, diminutos e invertidos em amostras isoladas de acordes gravados, transcrição automática de acordes ao longo do tempo e extração de tonalidade musical. O sistema solução final tem como requisito uma entrada de áudio de uma música tipo WAVE e duas saídas: acordes ao longo do tempo numa precisão de 1 segundo e a tonalidade da música.

Palavras-chaves: reconhecimento. acordes. música. processamento. sinais. redes. neurais. harmonia.

Abstract

Currently, the music have been in top level with regard to various approaches to behold and run. The technology is increasingly becoming too an interaction approach with the musical processes. One of the technology examples are automatic music transcription systems that help the musician, improving significantly scores, tabs and chords. This present study aims to develop a prototype of computational solution for recognition of musical harmonies. For this purpose, implementations of spectral analysis of the audio sample, classification of musical notes, chord classification with support inversion, recognition of rhythmic and harmonic transition patterns over time and extraction of musical tonalities were made. The development of the solution was through a method of empirical, iterative and incremental cicles, using Matlab programming language for implementation. Of theoretical foundations were used physical concepts of sound, music theory, signal processing and artificial neural networks. The development solution has allowed the recognition of the chord triads in larger, smaller, increased, and miniature inverted in isolated samples of recorded chords, chord automatic transcription over time and extraction of musical tone.

Key-words: recognition. chords. music. processing. signals. networks. neural.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

aum	Acorde aumentado
dim	Acorde diminuto
M	Acorde maior
m	Acorde menor
A	Lá
B	Si
C	Dó
D	Ré
E	Mi
F	Fá
G	Sol
#	Sustenido
b	Bemol
Hz	Hertz
db	Decibéis
.wav	Formato de arquivo WAVE
PCP	*Pitch Class Profile*
CTT	*Chord Type Template*
PNN	*Probabilistic Neural Network*
PDCA	*Plan Do Check Act*
PCM	*Pulse Code Modulation*
STFT	*Short Time Fourier Transform*
ISMIR	*The International Society of Music Information Retrieval*
MIREX	*Music Information Retrieval Evaluation eXchange*

Sumário

1 Introdução

1.1 Contexto

A inclinação da humanidade para a música é algo inato, essencial em todas as culturas e desde os primórdios da espécie humana vem se desenvolvendo intensamente (CRUZ, 2008). A tecnologia vem cada vez mais se tornando uma abordagem de interação com os processos musicais (THÉBERGE, 1997). Desde sintetizadores eletrônicos até afinadores programados em software, a música vem acompanhando o desenvolvimento técnico-científico.

Um bom exemplo de impacto direto da tecnologia sobre a música é o software Auto-Tune (TYRANGIEL, 2009). O software é um editor de áudio em tempo real criado pela empresa Antares Audio Technologies (ANTARES, 2014) para afinar instrumentos e vozes. Muitos cantores e artistas usam desse software para poder executar as músicas com mais afinação em apresentações e gravações.

Outro exemplo de software impactante na música é o afinador Tuner-gStrings (COHORTOR.ORG, 2014). Ele é um aplicativo da plataforma para dispositivos móveis Android que permite a afinação de quaisquer instrumentos musicais. Na loja virtual Google Play ele está com 4,6 de 5 estrelas em 155.957 avaliações e o número de instalações entre 10.000.000 e 50.000.000 no mundo inteiro [1].

Ambos softwares apresentados são ferramentas de suporte para o músico poder executar corretamente as músicas e facilitar muito trabalho que seria de natureza manual. O presente trabalho tem como objetivo apresentar um protótipo de uma solução computacional de suporte ao músico para extração de informações harmônicas objetivando a automação da percepção musical.

1.2 Problemática

Os músicos, em geral, sempre necessitaram do conhecimento de informações sobre as músicas com o intuito de serem melhor executadas. Informações do tipo de compasso, tom, harmonia, escalas utilizadas, andamento, expressões e variações de tempo. Especificamente obter a noção de harmonia e tom das músicas é de grande valor no que diz respeito a instrumentos melódicos e jazzistas (MONSON, 2009).

Normalmente as informações de harmonia e tom são inferidas na partitura e tablaturas por regras simples como a primeira nota que começa e termina a música ou como

[1] Dados levantados: https://play.google.com/store/apps/details?id=org.cohortor.gstrings

o primeiro acorde que começar e terminar. Também são utilizados noções de escalas e acidentes para inferir que tais notas realmente pertencem a um determinado tom.

Essas técnicas são efetivas se no caso houver partituras, tablaturas ou cifras.Também há a possibilidade, mas somente para quem tem um ouvido bastante treinado, de ouvir melodias e harmonias e poder extrair informações de acordes e tom. Poucas pessoas possuem essa habilidade de discernir notas, tons e harmonia apenas ouvindo o som.

Dado a problemática, sistemas automáticos de transcrição de música (KLAPURI, 2004) são perfeitamente adequados a atender as necessidades de extração de informações relevantes numa dada faixa de áudio.

No que se trata diretamente sobre harmonias e acordes, existem muitos trabalhos publicados sobre sistemas de transcrição automática de acordes. O primeiro deles foi de Fujishima (FUJISHIMA, 1999) que postulou o uso da *Short Time Fourier Transform* para a montagem do *Pitch Class Profile* (PCP) que seria um vetor contendo as energias das 12 notas musicais. Esse vetor PCP iria servir de insumo para o processo de *Chord Type Template* (CTT) que é uma multiplicação entre o PCP e os templetes registrados de acordes com o intuito de extrair energias de cada acorde. Depois desse trabalho há vários outros que podem ser citados, como, por exemplo, (KHADKEVICH; OMOLOGO, 2011), (KHADKEVICH; OMOLOGO, 2011), (HARTE, 2010), (PEETERS, 2006), (CHO, 2010), (LEE, 2006), (CHEN, 2012), (HAAS; MAGALHÃES; WIERING, 2012) e (BOULANGER, 2013), esse último se destaca como o estado da arte de transcrição automática de acordes com 93.6% de eficácia num banco de dados de músicas polifônicas e multi-instrumentais MIREX[2].

No ponto de vista também da aprendizagem musical, há uma motivação para a criação de um reconhecedor de harmonias, dado que muitos iniciantes não sabem os acordes corretos para cada posição do instrumento, como também, os ouvidos são pouco treinados. Um sistema capaz de auxiliar no reconhecimento das harmonias seria de grande ajuda para o aprendiz abstrair os padrões musicais.

Em visto do que foi exposto, um reconhecimento de harmonias musicais facilitaria a atuação do músico por informar acordes e notas. Isso é muito bom pois substituiria parcialmente o uso das partituras, tablaturas e cifras além de funcionar como um ótimo guia para solistas e improvisadores (principalmente jazzistas).

O reconhecimento de harmonias musicais, no ponto de vista computacional, é inerentemente complexo. Primeiramente deve-se achar uma forma de representação das amostras de áudio em termos de frequências, a ferramenta para esse tipo de atividade deverá fazer uma transformação da informação que está no domínio temporal para o domínio frequencial. Em seguida é preciso identificar notas musicais em meio ao espectro de

[2] http://www.music-ir.org/

frequências calculado, focando cobrir com acurácia relações de bandas de frequência com as notas em si. Tendo convertido o espectro de frequência em notas musicais é preciso definir a classificação de acordes dado um embasamento teórico-musical definido, mapeando as combinações possíveis de tríades com os respectivos acordes. E, por fim, é preciso consolidar uma solução para reconhecimento de acordes ao longo de uma música e, para tanto, deverá ser contemplado uma implementação que identifica transições rítmicas e reconhecimento de harmonias ao decorrer da música.

1.3 Objetivos

Esse trabalho tem como objetivo principal desenvolver uma solução computacional para reconhecimento de harmonias musicais.

Como objetivos específicos têm-se implementação das soluções em:

- desenvolver solução computacional para reconhecimento de acordes e suas inversões;
- desenvolver solução computacional para reconhecimento de acordes ao longo do tempo;
- desenvolver solução computacional para extração do tom da música.

1.4 Organização do Trabalho

Esse trabalho está organizado em capítulos. O Capítulo 2 apresenta um arcabouço de fundamentos teóricos físicos, musicais, de processamento de sinais e de redes neurais. O Capítulo 3 apresenta a metodologia, as técnicas utilizadas, os ciclos de desenvolvimento da solução e o projeto da solução. O Capítulo 4 apresenta resultados da solução computacional proposta. O Capítulo 5 apresenta as conclusões e evoluções futuras. Segue no final referências bibliográficas e apêndice com demais códigos da solução.

2 Fundamentos Teóricos

Neste capítulo será introduzido conceitos teóricos para a definição de axiomas e ferramentas físico-matemáticas sobre o som, musicais, de processamento de sinais e redes neurais. Será abordado primeiramente conceitos físicos no que tange a natureza do som como propagação, formação e dinâmica. Após será exposto fundamentos sobre notas musicais e harmonia. Enfim, então, será apresentado teorias computacionais de processamento de sinais e redes neurais de classificação. Tais conceitos serão importantes para delimitar soluções de manipulação do sinal de áudio a fim de extrair informações musicalmente relevantes.

2.1 Conceitos Físicos do Som

Em vista do que é exposto no trabalho (SANTOS, 2008), o som pode ser visto como uma perturbação mecânica nas moléculas do meio, uma frente de compressão variável de perfil mecânico e longitudinal com velocidade e pressão. O meio de propagação do som pode ser de diversas naturezas como por exemplo sólido, líquido e gasoso. Dessa perturbação mecânica entende-se como variação de pressão em relação ao tempo e espaço. Em vista disso, a equação diferencial que expressa o comportamento do som é a de natureza ondulatória $\mathbf{p = p(x,t)}$ descrita em 2.1.

$$\frac{\partial^2 \mathbf{p}}{\partial x^2} = \frac{1}{c^2} \cdot \frac{\partial^2 \mathbf{p}}{\partial t^2} \tag{2.1}$$

Na qual $\mathbf{p}$ é a pressão, $\mathbf{x}$ é a localização longitudinal, $\mathbf{c}$ é a velocidade do som e $\mathbf{t}$ é a localização temporal. A solução harmônica para a Equação (2.1) é definida por 2.2.

$$\mathbf{p}(\mathbf{x}, \mathbf{t}) = \mathbf{A}.exp^{j(wt-kx)} + \mathbf{B}.exp^{j(wt+kx)} \tag{2.2}$$

Em que $\mathbf{k}$ é dado por $\mathbf{w/c}$ e as constantes complexas $\mathbf{A}$ e $\mathbf{B}$ são utilizadas para condições de contorno.

Uma solução simples para a equação de onda 2.2 pode ser consolidada com a equação 2.3.

$$\mathbf{p(t)} = \mathbf{1}.cos(2.\pi.440.t) \tag{2.3}$$

Nela a variável **x** foi fixada na origem do sistema cartesiano e o comportamento da onda só está sendo analisado em relação a variável temporal **t**. Há de considerar de suma importância a fixação frequencial **w** em 440 Hz. Em termos musicais essa nota equivale ao Lá de tom puro, ou seja, nota construída artificialmente sem harmônicos somados (diferentemente dos instrumentos reais que possuem os harmônicos). Um exemplo de gráfico gerado por essa função é dado pela figura 1.

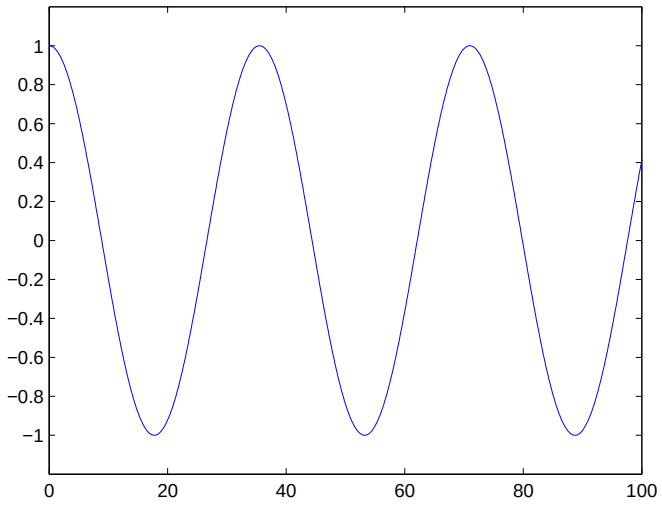

Figura 1: Função da Equação 2.3

Na forma mais teórica, um acorde, como será apresentado a seguir, é composto de no mínimo 3 ondas sonoras somadas (MED, 1996), ou seja, para que seja formado um acorde Am, por exemplo, a onda sonora deve ser expressa pela equação 2.4.

$$\mathbf{p(t)} = \mathbf{1}.cos(2.\pi.440.t) + \mathbf{0,5}.cos(2.\pi.523.t) + \mathbf{1}.cos(2.\pi.660.t) \tag{2.4}$$

Pode-se considerar que cada constante multiplicadora das função cosseno determinará a energia de uma onda sonora específica. Nesse caso as ondas sonoras de mais energia são as de 440 Hz e 600 Hz. A onda de menor energia é a de 523 Hz que possui a metade da energia das outras. Esse fato será decisivo para a detecção de acordes. É pertinente comentar que esse acorde montado é um modelo simplificado pois possuem somente tons puros (sem harmônicos somados). No contexto da solução trabalhada será considerado também acordes com harmônicos somados assim como é nos instrumentos reais. Esse fato aumenta a complexidade da solução. Em termos de representação gráfica da função, o acorde Am pode visualizado segundo a figura 2.

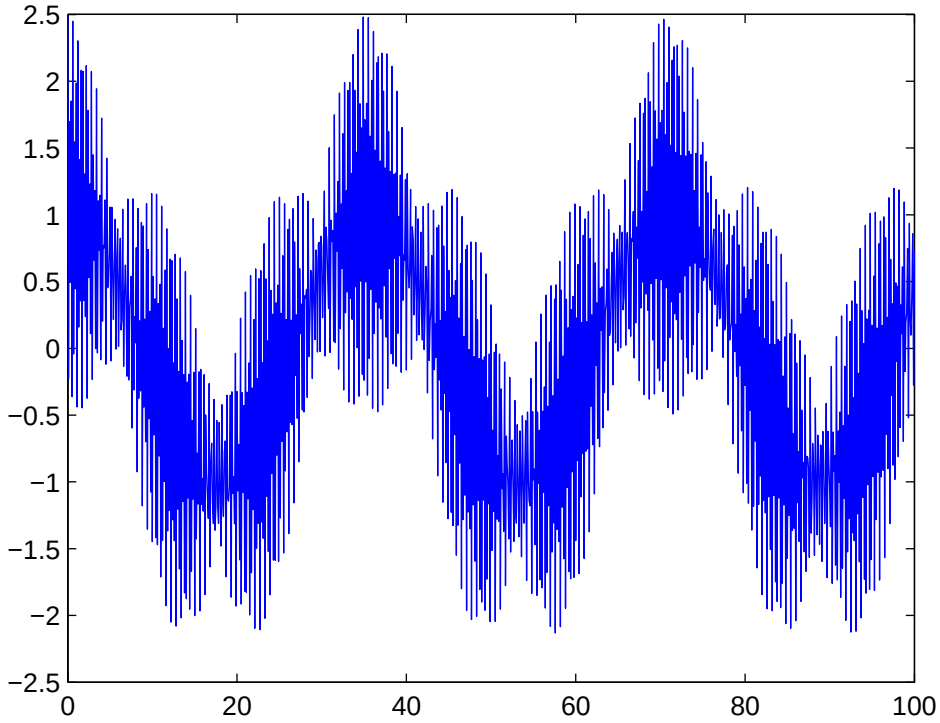

Figura 2: Função da Equação 2.4 (Am)

2.2 Conceitos Musicais

A música em si, além de ter em sua essência todas as leis físicas da ondulatória sonora, ela é uma forma de arte no que se refere a apresentação estética e do belo (WÖLFFLIN; JÚNIOR, 2000). Para a construção do belo em formas de som, há desenvolvido durante toda história da humanidade um conjunto de técnicas e metodologias bem apuradas. Nesse aspecto, a música define-se como ciência e pode ser abordada nas áreas de teoria básica, solfejo, ritmo, percepção melódica, dinâmica, harmonia, contraponto, formas musicais, instrumentos musicais, instrumentação, orquestração, arranjo, fisiologia da voz, fonética, psicologia da música, pedagogia musical, história da música, acústica musical, análise musical, composição e regência (MED, 1996).

A estrutura da arte musical em si é baseada na combinação de sons em forma simultânea e sucessiva recorrendo a ordem, equilíbrio e proporção dentro do tempo. Os principais elementos formadores da música podem ser divididos nessas categorias:

- melodia - sons dispostos em ordem sucessiva ao longo do tempo (concepção horizontal da música);
- harmonia - sons dispostos em ordem simultânea ao longo do tempo (concepção vertical da música);

- contraponto - conjunto de melodias e harmonias (concepção híbrida vertical e horizontal da música);
- ritmo - ordem e proporção em que estão dispostos as melodias e as harmonias.

O sons que formam as melodias e as harmonias possuem características principais como:

- altura - frequência das vibrações sonoras. Quanto maior frequência mais agudo o som será;
- duração - tempo de extenção do som ao longo do tempo;
- intensidade - amplitudade ou força das vibrações sonoras, conhecido como volume;
- timbre - combinação das intensidades dos harmônicos que um determinado agente sonoro.

A altura e intensidade do som são as características essenciais para a formulação dos conceitos de notas e acordes. Em altura entende-se como a divisão das frequências em 7 notas musicas - *Dó*, *Ré*, *Mi*, *Fá*, *Sol*, *Lá* e *Si*. Também essas mesmas notas possuem uma correspondente em sequência de letras alfabéticas introduzidas pelo Papa Gregório Grande - C, D, E, F, G, A e B. Normalmente essa sequência de letras são usadas para denominar acordes. Entretanto tais divisões de notas não são a menor divisão para o sistema temperado (MED, 1996). A menor divisão de notas se denomina semitom e são configurados pelos acidentes sustenidos (#) ou bemois (♭). Considerando essa divisão semitonal o sistema fica representado nessa sequência de 12 notas musicais (entre uma divisão e outra há a presença de um semitom): *Dó*, *Dó*# ou *Ré*♭, *Ré*, *Ré*# ou *Mi*♭, *Mi*, *Fá*, *Fá*# ou *Sol*♭, *Sol*, *Sol*# ou *Lá*♭, *Lá*, *Lá*# ou *Si*♭ e *Si*. Ou seguindo a denominação inglesa: C, $C\#$ ou $D\flat$, D, $D\#$ ou $E\flat$, E, F, $F\#$ ou $G\flat$, G, $G\#$ ou $A\flat$, A, $A\#$ ou $B\flat$ e B.

Como foi explicado na seção 2.1, o som possui características frequenciais e podem ser diferenciados entre si pela frequência de ressonância. Assim ocorrem com as notas musicais que são definidas cada uma pela frequência base na qual é ressoada. A figura 3 mostra as frequências de cada nota musical num intervalo de 16,352 Hz até 31.608,5 Hz. Na primeira linha da tabela são mostradas as oitavas [1] e a primeira coluna são as notas musicais em si numa sequência de semitons de *dó* até *si*. O cruzamento da tabela entre oitava e a nota é a frequência da nota dado aquela oitava seguido por, entre parênteses, quantidade de semitons a partir do *dó* central (destacado em azul no valor de 261,63 Hz). Um exemplo de frequência de uma nota é o *lá* de afinação (destacado em verde) no valor

[1] Oitava de uma nota, nesse contexto, é definido como o intervalo de dobro ou a metade da frequência de referência (MED, 1996).

de 440 Hz. As relações de frequências e notas musicais vão poder delimitar quais notas estão presentes num dado sinal com certas frequências específicas.

Octave → Note ↓	0	1	2	3	4	5	6	7	8	9	10
C	16.352 (−48)	32.703 (−36)	65.406 (−24)	130.81 (−12)	261.63 (±0)	523.25 (+12)	1046.5 (+24)	2093.0 (+36)	4186.0 (+48)	8372.0 (+60)	16744.0 (+72)
C♯/D♭	17.324 (−47)	34.648 (−35)	69.296 (−23)	138.59 (−11)	277.18 (+1)	554.37 (+13)	1108.7 (+25)	2217.5 (+37)	4434.9 (+49)	8869.8 (+61)	17739.7 (+73)
D	18.354 (−46)	36.708 (−34)	73.416 (−22)	146.83 (−10)	293.66 (+2)	587.33 (+14)	1174.7 (+26)	2349.3 (+38)	4698.6 (+50)	9397.3 (+62)	18794.5 (+74)
E♭/D♯	19.445 (−45)	38.891 (−33)	77.782 (−21)	155.56 (−9)	311.13 (+3)	622.25 (+15)	1244.5 (+27)	2489.0 (+39)	4978.0 (+51)	9956.1 (+63)	19912.1 (+75)
E	20.602 (−44)	41.203 (−32)	82.407 (−20)	164.81 (−8)	329.63 (+4)	659.26 (+16)	1318.5 (+28)	2637.0 (+40)	5274.0 (+52)	10548.1 (+64)	21096.2 (+76)
F	21.827 (−43)	43.654 (−31)	87.307 (−19)	174.61 (−7)	349.23 (+5)	698.46 (+17)	1396.9 (+29)	2793.8 (+41)	5587.7 (+53)	11175.3 (+65)	22350.6 (+77)
F♯/G♭	23.125 (−42)	46.249 (−30)	92.499 (−18)	185.00 (−6)	369.99 (+6)	739.99 (+18)	1480.0 (+30)	2960.0 (+42)	5919.9 (+54)	11839.8 (+66)	23679.6 (+78)
G	24.500 (−41)	48.999 (−29)	97.999 (−17)	196.00 (−5)	392.00 (+7)	783.99 (+19)	1568.0 (+31)	3136.0 (+43)	6271.9 (+55)	12543.9 (+67)	25087.7 (+79)
A♭/G♯	25.957 (−40)	51.913 (−28)	103.83 (−16)	207.65 (−4)	415.30 (+8)	830.61 (+20)	1661.2 (+32)	3322.4 (+44)	6644.9 (+56)	13289.8 (+68)	26579.5 (+80)
A	27.500 (−39)	55.000 (−27)	110.00 (−15)	220.00 (−3)	440.00 (+9)	880.00 (+21)	1760.0 (+33)	3520.0 (+45)	7040.0 (+57)	14080.0 (+69)	28160.0 (+81)
B♭/A♯	29.135 (−38)	58.270 (−26)	116.54 (−14)	233.08 (−2)	466.16 (+10)	932.33 (+22)	1864.7 (+34)	3729.3 (+46)	7458.6 (+58)	14917.2 (+70)	29834.5 (+82)
B	30.868 (−37)	61.735 (−25)	123.47 (−13)	246.94 (−1)	493.88 (+11)	987.77 (+23)	1975.5 (+35)	3951.1 (+47)	7902.1 (+59)	15804.3 (+71)	31608.5 (+83)

Figura 3: Frequências das notas musicais em Hz (LABGARAGEM, 2014)

Acordes são harmonias formadas por pelo menos uma tríade (três notas)[2] tocadas simultâneamente e são definidos por certas quantidades de semitons entre as notas (MED, 1996). Essa quantidade se denomina intervalo musical.

As 3 notas da tríade são referenciadas como tônica (a nota base do acorde), terça e quinta (MED, 1996). Para acordes maiores (M) a distância entre a tônica e a terça é de 4 semitons (terça maior) e entre a tônica e a quinta é de 7 semitons (quinta justa). Para acordes menores (m) a distância entre a tônica e a terça é de 3 semitons (terça menor) e entre a tônica e a quinta é de 7 semitons (quinta justa). Para acordes aumentados (aum) a distância entre a tônica e a terça é de 4 semitons (terça maior) e entre a tônica e a quinta é de 8 semitons (quinta aumentada). Para acordes diminutos (dim) a distância entre a tônica e a terça é de 3 semitons (terça menor) e entre a tônica e a quinta é de 6 semitons (quinta diminuta).

Exemplos de acordes maiores são:

- dó maior - CM (tríade $Dó$, Mi e Sol);
- lá maior - AM (tríade $Lá$, $Dó\#$ e Mi).

Exemplos de acordes menores são:

- mi menor - Em (tríade Mi, Sol e Si,);
- ré sustenido menor - $D\#m$ (tríade $Ré\#$, $Fá\#$ e $Lá\#$).

Exemplos de acordes aumentados são:

[2] Existem acordes mais complexos com mais de 3 notas porém o escopo desse trabalho só se delimita a tríades.

- sol aumentado - *Gaum* (tríade *Sol*, *Si* e *Ré#*);
- si aumentado - *Baum* (tríade *Si*, *Ré#* e *Sol*).

Exemplos de acordes diminutos são:

- dó sustenido diminuto - *C#dim* (tríade *Dó#*, *Mi* e *Sol*);
- lá sustenido diminuto - *A#dim* (tríade *Lá#*, *Dó#* e *Mi*).

Na teoria dos acordes também há a presença do conceito de inversões (MED, 1996). Inverter um acorde consiste em trocar de posição para uma oitava a cima a nota inferior, trocar a nota mais baixa do acorde por uma outra da mesma denominação só que com o dobro de frequência acima. Em tríades há a presença do acorde em seu estado fundamental, a primeira inversão (a terça fica sendo como a nota mais grave) e a segunda inversão (a quinta fica sendo como a nota mais grave).

Segue exemplos de acordes em estado fundamental:

- dó maior - *CM* (tríade *Dó*, *Mi* e *Sol*);
- mi menor - *Em* (tríade *Mi*, *Sol* e *Si*,);
- sol aumentado - *Gaum* (tríade *Sol*, *Si* e *Ré#*);
- lá sustenido diminuto - *A#dim* (tríade *Lá#*, *Dó#* e *Mi*).

Segue exemplos de acordes em primeira inversão:

- dó maior - *CM* (tríade *Mi*, *Sol* e *Dó*);
- mi menor - *Em* (tríade *Sol*, *Si* e *Mi*);
- sol aumentado - *Gaum* (tríade *Si*, *Ré#* e *Sol*);
- lá sustenido diminuto - *A#dim* (tríade *Dó#*, *Mi* e *Lá#*).

Segue exemplos de acordes em segunda inversão:

- dó maior - *CM* (tríade *Sol*, *Dó* e *Mi*);
- mi menor - *Em* (tríade *Si*, *Mi* e *Sol*);
- sol aumentado - *Gaum* (tríade *Ré#*, *Sol* e *Si*);
- lá sustenido diminuto - *A#dim* (tríade *Mi*, *Lá#* e *Dó#*).

2.3 Conceitos de Processamento de Sinais

O conceito de processamento de sinais está inteiramente ligado à natureza do sinal e a aplicação que normalmente se dá é de modificação ou análise. Sinal pode ser entendido como um objeto matemático, em geral uma função matemática, que descreve o comportamento de um determinado fenômeno da natureza podendo ser, entre outros, físico, químico, biológico e financeiro (OPPENHEIM; WILLSKY; NAWAB, 1983).

Da natureza do sinal abordado, é explícito de que o mesmo é de cunho físico - sinais sonoros. Com a possibilidade de se trabalhar com sinal sonoro, há em processamento de sinais a liberdade de modificar ou extrair informações relevatantes para uma dada aplicação. Nesse contexto o sinal será processado com o intuito de colher informações para serem analisadas de forma a deduzir comportamentos de ondas sonoras.

Os sinais geralmente são funções relacionadas ao tempo. Eles podem ser processados em tempo contínuo (analogicamente) ou em tempo discreto (digitalmente). O escopo desse trabalho está restringido somente ao processamento na forma digital.

Para haver processamento digital é preciso que o sinal seja descrito computacionalmente num hardware. A forma de captação de um fenômeno contínuo para o ambiente computacional se denomina processo de amostragem e quantização (DRUYVESTEYN, 1992).

Amostrar um sinal significa recolher um número determinado de amostras dado um período de tempo, ou seja, haverá uma frequência (taxa de amostragem) **F** associada a um período de tempo **T** que proporcionará um conjundo finito de amostras num intervalo temporal. A relação exposta dar-se-á por:

$$\mathbf{F_s} = \frac{1}{T} \tag{2.5}$$

Segundo o teorema de amostragem Nyquist–Shannon (UNSER, 2000), para sons musicais o mais adequado é que a taxa de amostragem seja o dobro da frequência máxima de audição do ouvido humano - 22.050 Hz, ou seja, a frequência será de 44.100 Hz. Essa grandeza significa que serão captadas 44.100 amostras de áudio a cada segundo.

Quantizar um sinal significa alocar valores digitais para os valores analógicos do eixo da ordenada, que são normalmente valores de tenões elétricas. Essa alocação está ligada diretamente às características do conversor analógico/digital. Nesse caso específico foi usado um conversor de 16 bits para a quantização do sinal. Tal fato permite a presença de 65.536 (2 elevado a 16) valores para representar as subidas e descidas da onda sonora.

O conceito de energia está totalmente ligado a muitas outras áreas e é de extrema importância por ser essencial nos fenômenos naturais (OPPENHEIM; WILLSKY; NAWAB, 1983). O aspecto energético adotado nessa presente solução será em tempo discreto que

é definido como:

$$\mathbf{E} = \sum_{t=t1}^{t2} |x[n]|^2 \tag{2.6}$$

Outro conceito relacionado que é de extrema importância é a lei de conservação de energia representada pelo teorema de Parseval. Esse teorema mostra que a energia do sinal sempre se conserva independentemente da projeção que o sinal foi submetido. Mais especificamente na transformada de fourier discreta o teorema é descrito como:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N}. \sum_{k=0}^{N-1} |X[k]|^2 \tag{2.7}$$

Tal qual N é o número total de amostras e $X(k)$ a transformada discreta de fourier.

A transformada de fourier é uma ferramenta muito importante para a realização desse trabalho. Ela permite projetar o sinal em funções de base senoidais, ou seja, é possível ver através dela quais componentes frequenciais de senóides estão presentes no sinal e qual é a energia das mesmas.

A representação da transformada de fourier em frequência discreta (DFT) é dada por:

$$X[k] = \sum_{n=0}^{N-1} x[n].exp^{-j2\pi.\frac{k}{N}.n} \tag{2.8}$$

A representação da transformada inversa de fourier em frequência discreta é dada por:

$$x[n] = \frac{1}{N}. \sum_{k=0}^{N-1} X[k].exp^{j2\pi.\frac{k}{N}.n} \tag{2.9}$$

2.4 Conceitos de Redes Neurais Artificiais

Identificar padrões num sinal nem sempre é um trabalho trivial ou até mesmo determinístico. Normalmente sinais possuem ruídos intratáveis, sua composição é complexa no sentido de haver muitas amostras para análise e, como os sinais são fenômenos naturais, facilmente são vistos como sistemas complexos (MORIN; MATOS, 2007). Determinar uma equação ou um algorítmo fixo e determinístico para classificação e processamento de sinais é bastante limitado e aderente há vários erros.

Além disso, para o reconhecimento de harmonias é preciso usar conceitos de teoria musical no que diz respeito aos acordes e notas para o reconhecimento de padrões presentes no sinal. É preciso então ter alguma forma de representar esse conhecimento musical no ponto de vista computacional.

Diante desse ambiente de incertezas e requisitos de incorporação do conhecimento musical no campo computacional, uma solução que é aderente ao contexto é o uso de redes neurais artificiais. Essa técnica, além de prover as características necessárias para

deixar a solução estável, ela modela o funcionamento neural de organismos vivos. Esse fato é muito interessante visto que surge a possibilidade de usar os mesmos mecanismos (de forma análoga) de reconhecimento de padrões sonoros do cérebro humano num sistema computacional.

Dado um especialista que possui o reconhecimento de padrões dos acordes, basta somente consolidar uma arquitetura de rede neural para receber esse conhecimento empírico de modo que o seu uso seja eficiente para classificação.

Entende-se por rede neural como "um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão para armazenar conhecimento experimental e torná-lo disponível para o uso" (S; HAYKIN, 2009).

O aprendizado empírico é uma característica essencial nas redes neurais artificiais. Sua estrutura oferece suporte para que conhecimentos adquiridos de maneira experimental (via ser humano muitas vezes) possam ser aprendidos e usados. O processo de aprendizagem da rede se chama algorítmo de aprendizado e o mesmo pode ser feito de várias formas como lei de Hebb, algorítmo de *backpropagation*, estratégias de competição e máquina de Boltzmann. Além disso é envolvido nesse processo paradígmas de aprendizado que é como o ambiente vai atuar sobre a rede neural para que ela possa aprender. Exemplos de paradígmas de aprendizado são aprendizado supervisionado, aprendizado por reforço e aprendizado não-supervisionado (ou auto-organizado) (S; HAYKIN, 2009).

Outra característica essencial de uma rede neural é a representação do conhecimento. Essa característica é referente às informações armazenadas ou a modelos utilizados por uma pessoa ou máquina com o intuito de interpretar, prever e responder de forma coerente ao mundo exterior (S; HAYKIN, 2009). Para tal representação deve-se levantar em conta quais informações serão abstraídas e tornadas explícitas e como a informação será codificada no sistema. Com o intuito de atingir os objetivos de uma boa representação do conhecimento na rede neural há um conjunto de regras sugeridas a se seguir (S; HAYKIN, 2009):

- regra 1 - entradas similares normalmente devem produzir representações similares no interior da rede e devem ser classificadas como pertencentes a mesma categoria;
- regra 2 - itens de classes diferentes devem ser representados de formas diferentes;
- regra 3 - se uma característica é importante deve-se haver um grande número de neurônios envolvidos na representação daquele item de rede;
- regra 4 - informações prévias e invariâncias devem ser incorporadas a rede para que o sistema fique simples e sem trabalho para aprender as mesmas.

Por fim outra característica importante de uma rede neural é a capacidade de generalização. Isso permite com que entradas desconhecidas possam ser classificadas e tratadas de forma coesa e coerente, fazendo com que circunstâncias críticas e imprevisíveis possam ser contornadas sem grandes prejuízos.

A unidade mínima de processamento de uma rede neural é o neurônio artificial. O modelo do neurônio artificial é representado pela figura 4.

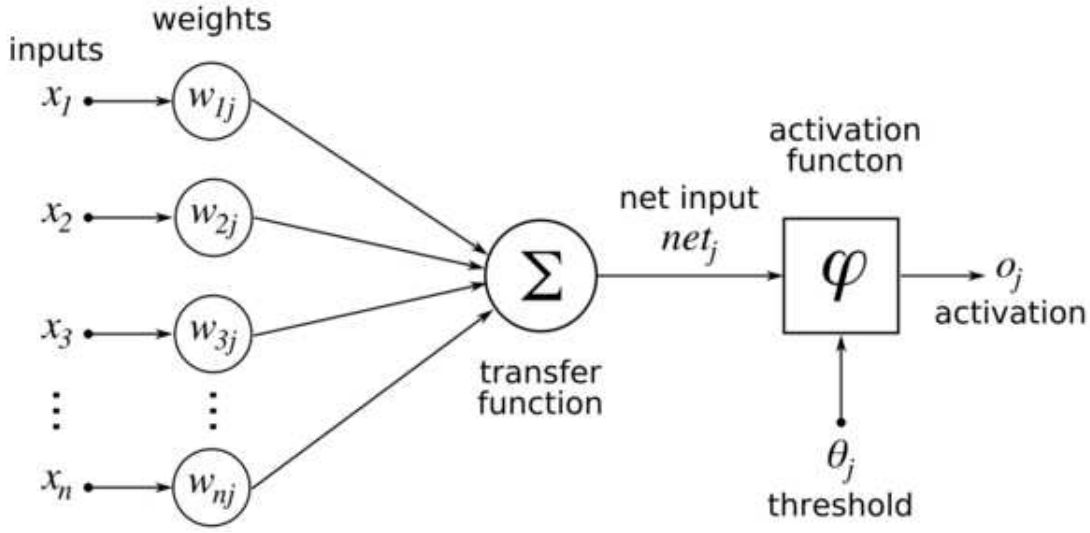

Figura 4: Modelo de um neurônio (S; HAYKIN, 2009).

Como é representado na figura 4 os conjuntos de w representam pesos sinápticos para a modulação dos sinais de entrada, ou seja, para cada entrada há uma unidade escalar que multiplicará sinais de entrada de acordo com o conhecimento aprendido. Após há um somador $\sum$ para efetuar operações de combinações lineares para a produção de um escalar no final desse processo. Por último há uma função de ativação, mais conhecido como um limiar de ativação para que a resposta possa ser propagada a outros neurônios, caso o escalar resultante do somador for maior que o limiar.

Para o presente problema, foi sugerido o uso da rede neural do tipo *Probabilistic Neural Network* (PNN) (SPECHT, 1990). Ela é inerentemente um sistema de classificação bastante simples de aprendizado não-supervisionado, ou seja, novos conhecimentos são adquiridos pela simples inserção de novos neurônios. Segue o modelo arquitetural representado na figura 5.

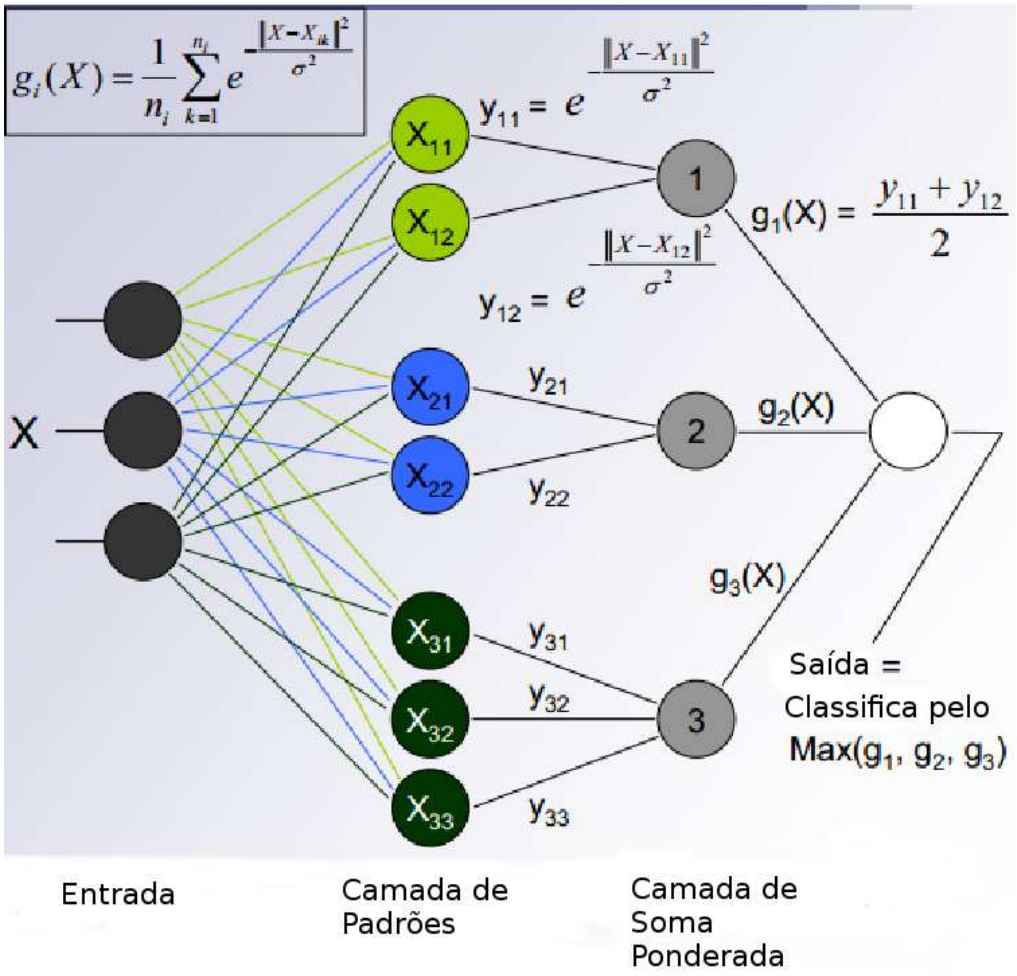

Figura 5: Modelo arquitetural da PNN (JURGEN, 2014).

Segundo o estudo (SPECHT, 1990), a rede neural PNN possui 3 camadas: a primeira, camada de padrões, é responsável por calcular, através de uma função de transferência de base gaussiana 2.10, a distância euclidiana entre a entrada X e o valor aprendido pela rede X_{nm}, ou seja, classificar a probabilidade de um indivíduo ser de uma determinada classe; a segunda, camada de soma ponderada, é responsável por realizar uma média simples 2.11 dessas probabilidades de cada classe; a terceira, camada de classificação por máximo, é responsável por extrair a classe de maior probabilidade, ou seja, pegar o valor máximo dado o conjunto de classes calculadas $\mathbf{g}$. O valor final de saída é a classificação do sinal de entrada.

$$Y_{nm} = \exp\left(|X - X_{nm}|^2\right)/(\sigma)^2 \tag{2.10}$$

$$G_n(X) = (Y_{nn} + Y_{n(n+1)})/2 \tag{2.11}$$

Dado o que foi exposto nesse capítulo, os fundamentos teóricos apresentados serão, nos capítulos seguintes, a base principal para a investigação de soluções de extrações de informações musicais num sinal de áudio. Dentro do que foi apresentado de fundamenta-

ção, os conceitos físicos do som demonstram a natureza do fenômeno trabalhado através de modelos matemáticos que delimitarão as técnicas coerentes a serem implementadas. Os conceitos musicais são as referências, em termos de conhecimento, que serão usadas para alimentar a base de conhecimento da solução computacional proposta. Os conceitos de processamento de sinais serão usados para abstrair informações releventes do sinal de áudio de tal forma que as mesmas sejam tratadas computacionalmente. Os conceitos de redes neurais artificiais, principalmente o modelo arquitetural da PNN, serão usados para prover significação às informações extraídas do processamento de áudio e classificação dado uma base de conhecimento musical.

3 Desenvolvimento da Solução

3.1 Metodologia

Dado o contexto e características do trabalho foi estabelecido um método de desenvolvimento empírico, iterativo e incremental. Na engenharia de software se destacam dois processos de controle de desenvolvimento: processo definido e processo empírico. O processo definido é constiuído de um conjunto de sub-processos rigoros nos quais possuem entradas e saídas bem definidas e repetitivas (WEISS et al., 1999). Já o processo empírico é constituído de um conjunto de sub-processos imperfeitamente definidos nos quais as entradas e saídas são imprevisíveis e não repetíveis, características essas presentes no desenvolvimento desse trabalho.

A metodologia empírica de desenvolvimento de software se embasa em três fundamentos: precisa ser transparente, visto que o máximo de variáveis devem estar visíveis para os envolvidos no projeto; dado as variáveis expostas a metodologia precisa ser frequentemente inspecionada; feito as inspeções objetivo final é adaptar de acordo com as necessidades. Esses três fundamentos visam ajustar o processo de desenvolvimento para evitar variações de produção inaceitáveis e maximizar a mesma (DYBÅ; DINGSØYR, 2008).

Para que os procedimentos da metodologia empírica possa ocorrer ela precisa ser de natureza iterativa e incremental. Iterativa e incremental, pois terá ciclos curtos de desenvolvimento e a cada ciclo terá incrementos de código. No final de cada ciclo ter-se-à como resultado parâmetros de feedback para a melhoria contínua.

Outra análise do método empírico é o embasamento no ciclo de melhoria de processos e produtos *Plan-Do-Check-Act* (PDCA) (SHEWHART, 1980). Ele é constituído em 4 fases: *Plan* - planejamento do desenvolvimento; *Do* - executar o que foi planejado; *Check* - avaliar o que foi feito; *Act* - propor melhorias para os próximos ciclos.

Com o intuito de atingir os objetivos, serão abordados questões, hipóteses e critérios relacionados às problemáticas levantadas na extração e classificação de informações do sinal de áudio. No que diz respeito as questões, segue a formulação das mesmas:

- Se cada nota é uma frequência de vibração sonora, como analisar o sinal no ponto de vista de frequências? Dessa questão, surge a seguinte hipótese:
  - A transformada de fourier pode construir o espectro de frequências do sinal. O critério para avaliar essa hipótese é:
    - Vetor com os níveis de energia relacionados a cada frequência.

- Como configurar essas informações para localizar as notas musicais? Dessa questão, surge a seguinte hipótese:
  - Dado que cada nota musical é um conjunto de frequências, realocar as energias frequenciais da transformada de fourier afim de que cada posição do vetor seja 1 unidade de frequência (Hz) pode mapear a energia de cada nota. Os critérios para avaliar essa hipótese são:
    * Vetor com os níveis de energia relacionados a cada frequência deve ter o tamanho de 22050 posições.
    * Cada posição do vetor de energia relacionados a cada frequência possui valor de 1 Hz.
- Como adicionar as próximas camadas da rede para determinação dos acordes? Dessa questão, surge a seguinte hipótese:
  - Visto que associar as frequências as notas musicas é uma tarefa muito complexa para uma solução determinística, uma rede neural de aprendizado não supervisionado do tipo $ProbabilisticNeuralNetwork$ (PNN) pode classificar um conjunto de frequências em sua respectiva nota musical. O critério para avaliar essa hipótese é:
    * Vetor com os níveis de energia relacionados a cada nota.
- Como reconhecer acordes no tempo de tal forma a saber onde eles ocorrem? Dessa questão, surge a seguinte hipótese:
  - Uma solução de reconhecimento de energias ao longo do tempo pode a determinar o rítmo. Em tese é calcular a energia do sinal, aplicar um filtro passa-baixas para suavização do sinal e aplicar um método de auto-correlação com o intuito de caracterizar picos de repetição de níveis elevados de energia. O critério para avaliar essa hipótese é:
    * Vetor com picos demonstrando a localização de ocorrência de cada acorde no tempo.
- Como ler o sinal todo e ter a visilibilidade em tempo e frequência? Dessa questão, surge a seguinte hipótese:
  - Para se ter uma resolução do sinal em tempo em frequência é preciso ter uma transformada que agregue esses dois aspectos. Pode-se testar isso com a transformada wavelets. O critério para avaliar essa hipótese é:
    * Matriz contendo às várias bandas de frequências filtradas relativas a cada notas ao longo do tempo.

- Como extrair o tom da música? Dessa questão, surge a seguinte hipótese:
  - Para extrair o tom da música é preciso somar a energia das notas totais ao longo da música e em seguida extrair o acorde mais provável. O critério para avaliar essa hipótese é:
    - Extração de um acorde maior ou menor ao final do processo.
- Como trabalhar com a frequência de aparecimento das notas? Dessa questão, surge a seguinte hipótese:
  - Se binarizar com 1 e 0 o mapa de notas no tempo a soma das notas será unitária, equivalente a frequência. O critério para avaliar essa hipótese é:
    - A extração automática dos acordes deverá ser determinado pela quantidade de notas tocadas e não pela energia das notas tocadas.
- Em relação aos acordes transitórios, como corrigir? Dessa questão, surge a seguinte hipótese:
  - Se ao deslocar a janela de tamanho de 1 segundo em passos de 0.2 segundos e calcular o espectro de frequência de cada passo pode-se fazer a média do acorde de cada tempo e poder cancelar os acordes transitórios. O critério para avaliar essa hipótese é:
    - Os acordes transitórios devem ser cancelados.
- Como extrair a nota mais grave (baixo) de cada período do tempo? Dessa questão, surge a seguinte hipótese:
  - Com o mapa de notas binarizado é possível extrair o baixo pegando a primeira posição com valor 1 de cada tempo. O critério para avaliar essa hipótese é:
    - Posição de valor 1 mais grave extraídas do conjunto de energias das notas.
- Como incluir a nota mais grave (baixo) de cada período do tempo aos acordes aumentados e invertidos? Dessa questão, surge a seguinte hipótese:
  - Pode-se referenciar um acorde invertido ou aumentado apartir da combinação de acordes fundamentais e baixos extraídos para cada período respectivo. O critério para avaliar essa hipótese é:
    - Extração de todas as possibilidades de acordes em tríades reconhecidas com sucesso.

3.1.1 Linguagem de Programação

Existem algumas ferramentas para muitas linguagens de programação que focam processamento de sinais. Uma delas é uma biblioteca em C++ desenvolvida por David Weenink baseada no processomento Mel Frequency Cepstral Coefficients (MFCC) [1]. Ela dá suporte a extração de dados de arquivos, cálculo da transformada de fourier (FFT), seu respectivo espectro de energia e cálculo da transformada de cosseno. Uma outra vantagem do uso dela é a linguagem de programação C++ que é bastante rápida em relação ao Java, Ruby e Python. Porém ela não dá suporte a estruturas de álgebra linear e cálculos estatísticos como operação de correlação. Processamento de sinais e redes neurais demanda também muito esforço no uso de matrizes e suas respectivas operações, fato esse torna o uso da linguagem C++ e da biblioteca citada desvantajosos.

Em vista das necessidades expostas, a linguagem de programação/ferramental escolhida para o trabalho é o Matlab [2]. Matlab é um software científico para computação numérica. Essa escolha foi feita primeiramente por tratar de ser um software voltado para aplicações científicas e para os axiomas da álgebra linear. Além disso, essa plataforma possui um conjunto de ferramentas para visualização de gráficos, pacotes de fórmulas matemáticas pré-programadas e estruturas de dados voltadas para a análise numérica e matricial. Ela possui uma característica de ser interpretada e dinamicamente tipada.

3.2 Técnicas Utilizadas para Desenvolvimento do Sistema-Solução

Nesta seção serão apresentadas as técnicas usadas para o desenvolvimento da solução, no intuito de reconhecer e extrair informações de harmonias em músicas comportadas em arquivos de áudio. A figura 6 mostra o diagrama de atividades em alto nível da modelagem matemática prototipada computacionalmente.

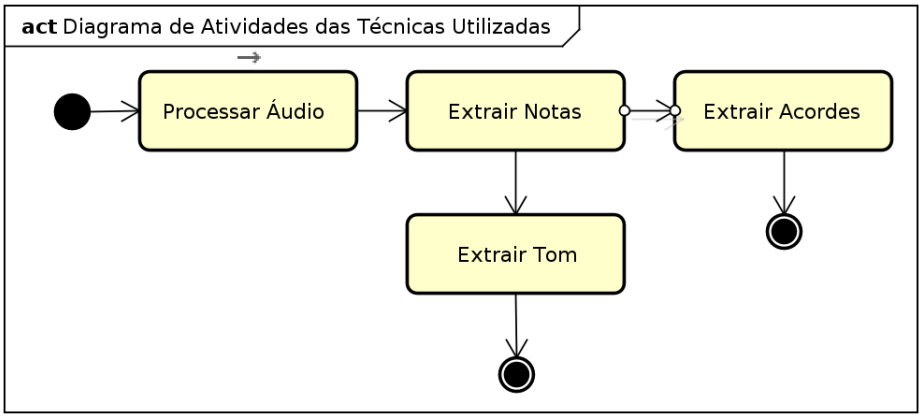

Figura 6: Diagrama de Atividades das Técnicas Utilizadas

[1] http://kaldi.sourceforge.net/feat.html#feat_mfcc
[2] http://www.mathworks.com/

No início do processo, há como entrada um arquivo de áudio do tipo WAVE e, no final, duas saídas que são os acordes ao longo do tempo e o tom da música. O fluxo para a extração de tonalidade musical é definido, em sequência, por processar o áudio, extrair notas e extrair tom. O fluxo para a extração de acordes ao longo do tempo é definido, em sequência, por processar o áudio, extrair notas e extrair acordes. As atividades são definidas em:

1. **Processar Áudio**: composta pelos procedimentos 1, 2 e 3, é responsável por segmentar e extrair informações relevantes relacionadas a frequências do sinal de áudio;

2. **Extrair Notas**: composta pelos procedimentos 4 e 5, é responsável por extrair informações de notas musicais dado o espectro de frequência. Essa atividade está relacionada a camada de reconhecimento de notas musicais da rede neural;

3. **Extrair Tom**: composta pelos procedimentos 7, é responsável por extrair o tom da música. Essa atividade está relacionada a camada de reconhecimento de acordes musicais da rede neural;

4. **Extrair Acordes**: composta pelos procedimentos 6, 8, 9 e 10, é responsável por os acordes ao longo do tempo. Essa atividade está relacionada a camada de reconhecimento de acordes musicais da rede neural.

A seguir será explicado cada um dos procedimentos que formam as atividades apresentadas. Ao final será apresentado um diagrama de atividades da interação dos procedimentos em conjunto.

3.2.1 Procedimento 1: Separar Janelas de 1 Segundo em 5 Partes

Após o sinal ser carregado num vetor de audio *stereo*, ele deverá ser transformado num do tipo mono. Sinal mono de áudio é aquele com somente um canal. Isso é necessário para que o processamento não fosse redundante. Não agregaria valor nesse caso processar um sinal de duplo canal sendo que a fonte emissora de ondas sonoras é comum para ambos. Após essa conversão, cada segundo do sinal é repartido em 5 partes de tamanhos iguais a 1 segundo, porém deslocadas numa escala 0.2 segundos. Esse processo é para a segmentar áudio no intuito de achar o acorde mais provável num intervalo de tempo de 1 segundo, fazendo com que acordes de transição ou ruidosos sejam suprimidos. A figura 7 ilustra o processo descrito e o código do mesmo está presente na secção A.1 dos apêndices.

Como é mostrado na figura 7, a cada segundo do sinal são criadas 5 janelas (A, B, C, D e E), de tamanhos iguais a 1 segundo, deslocadas numa escala de 0.2 segundos entre si. A primeira janela (A) é a mesma em sua posição original e as próximas janelas são recortadas do sinal a partir de um deslocamento de 0.2 segundos assim como é mostrado

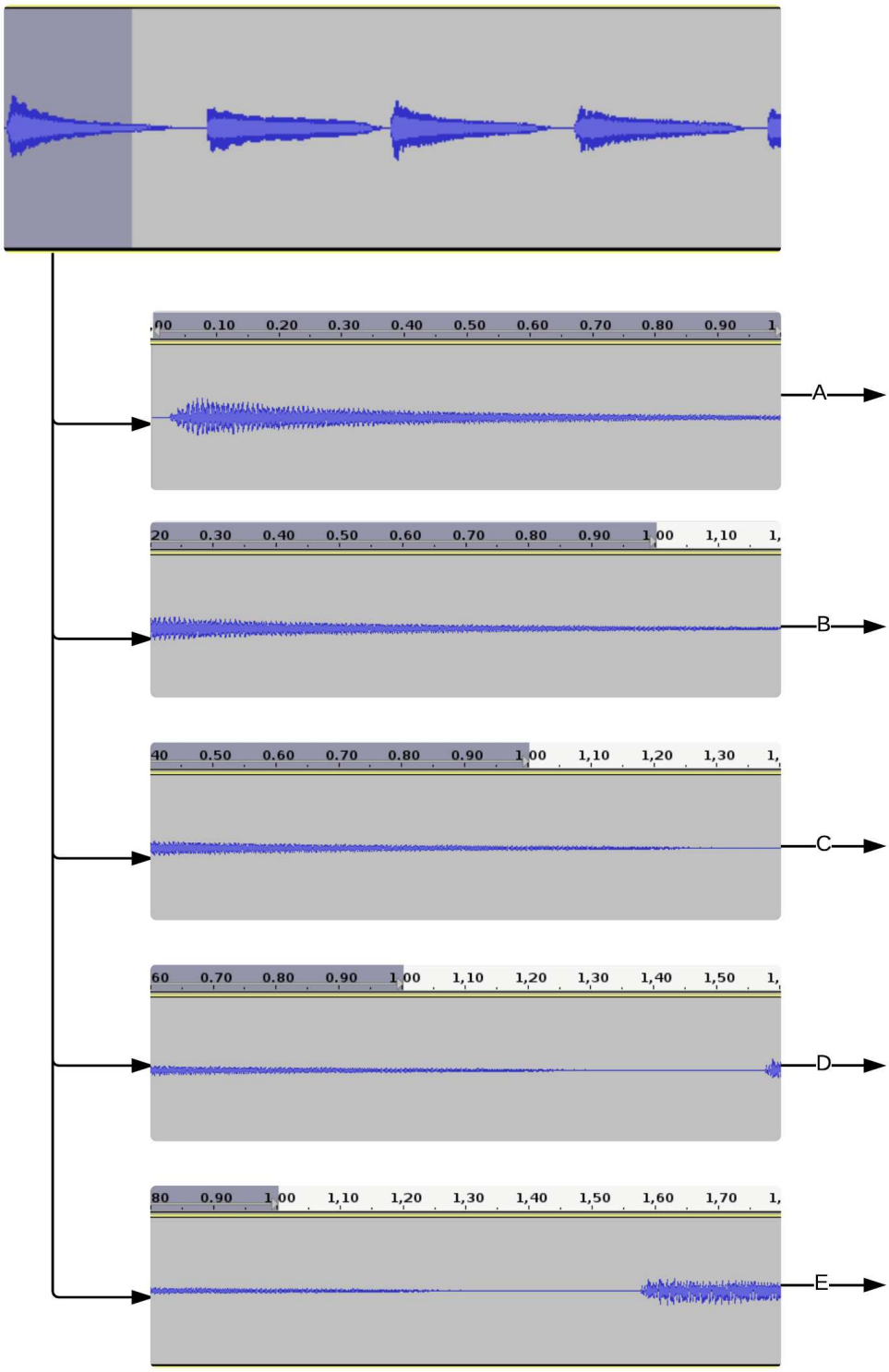

Figura 7: Segmentação de áudio - janelas de 1 segundo em 5 partes deslocadas.

nas equações em 3.1, 3.2, 3.3 e 3.4.

$$A(t) = B(t + 0.2) \tag{3.1}$$

$$B(t) = C(t + 0.2) \tag{3.2}$$

$$C(t) = D(t + 0.2) \tag{3.3}$$

$$D(t) = E(t + 0.2) \tag{3.4}$$

3.2.2 Procedimento 2: Aplicar Janelas de Blackman

Dado que o contexto da solução se deu por *Short Time Fourier Transform*, é preciso minimizar as distorções oriundas dos janelamentos. Para tal foi proposto a multiplicação de cada parte das janelas ao longo do tempo por uma janela de blackman, uma do tipo gaussiana abordada nos trabalhos (CABRAL, 2005) e (THOSHKAHNA, 2011). A figura 8 ilustra o processo descrito e o código do mesmo está presente na secção A.2 dos apêndices.

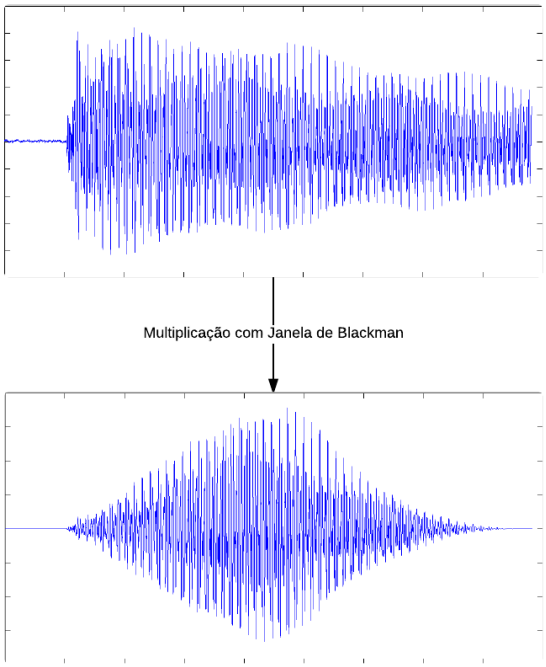

Figura 8: Multiplicação do sinal pela janela de blackman.

3.2.3 Procedimento 3: Calcular o Espectro de Frequências

O passo seguinte é adquirir os espectros de frequências oriundos do cálculo da transformada discreta de fourier. O cálculo será feito para cada uma das 5 partes de janelas. A figura 9 ilustra o processo descrito e o código do mesmo está presente na secção A.3 dos apêndices.

Primeiramente é alocado uma variável para comportar os 5 espectros de frequência, um para cada parte da janela. Depois os sinais passam por uma transformação de subamostragem na qual são eliminadas informações de altas frequências a partir de 1500 Hz. É feito o cálculo do módulo da transformada de frourier e esse mesmo vetor passar por uma reorganização de *slots* de tal forma que cada *slot* comporta-se 1 unidade de Hz.

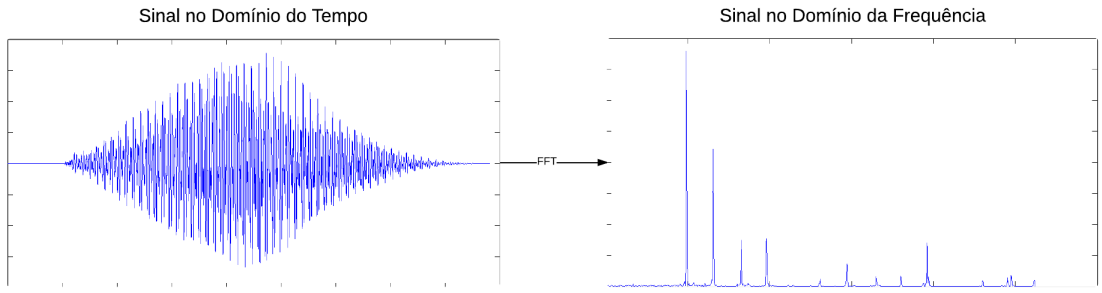

Figura 9: Transformação do domínio temporal para o domínio frequencial.

3.2.4 Procedimento 4: Adquirir Energias das Notas

Nesse procedimento cada espectro de frequência é correlacionado com conjunto de notas musicais dado um conjunto de frequências que nelas estão presentes. A figura 10 ilustra o processo descrito e o código do mesmo está presente na secção A.4 dos apêndices.

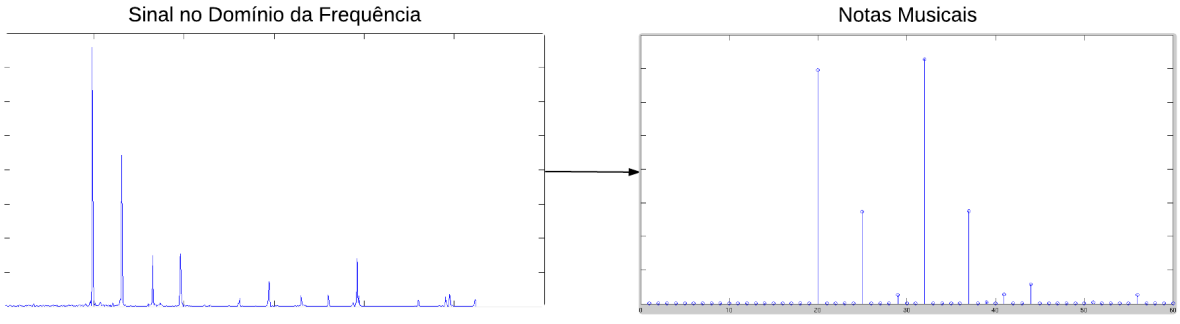

Figura 10: Cálculo das energias de cada nota.

A primeira atividade é carregar a base de dados de notas musicais originando o retorno de um matriz 60 notas por 1500 frequências. Então cada conjunto de notas relacionados às partes de janela serão correlacionados a partir de uma operação de multiplicação e, por fim, é feita a soma dos quadrados dos termos. Ao final uma matriz de notas por tempo é construída para cada uma das 5 partes.

3.2.5 Procedimento 5: Binarizar Energia das Notas

Esse passo compreende o processo de limiarização das energias das notas em 0's ou 1's de tal forma que se possa detectas as notas tocadas (1) ou não (0). Ao final desse processo espera-se conjuntos de energias de notas musicais em somente dois valores - 1 ou 0. A figura 11 ilustra o processo descrito e o código do mesmo está presente na secção A.5 dos apêndices.

No começo do procedimento é destacado um laço para cada uma das partes das janelas. No meio do procedimento destaca-se com operação de realocação do valor 0 para

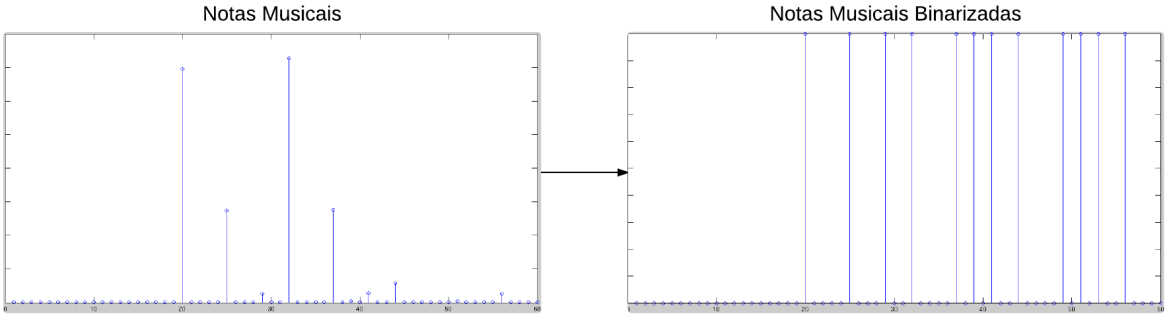

Figura 11: Binarização das notas.

valores de energia menores que 180% do valor máximo do conjunto de notas e 1 se for caso ao contrário. Por fim cada conjunto de notas são realocados em células.

3.2.6 Procedimento 6: Extrair Baixos

Com o intuito de determinar acordes com inversões e discernir os que são de natureza aumentada, acoplou-se no sistema um componente desenvolvido para a extração das notas mais graves numa dada janela de tempo. A figura 12 ilustra o processo descrito e o código do mesmo está presente na secção A.6 dos apêndices.

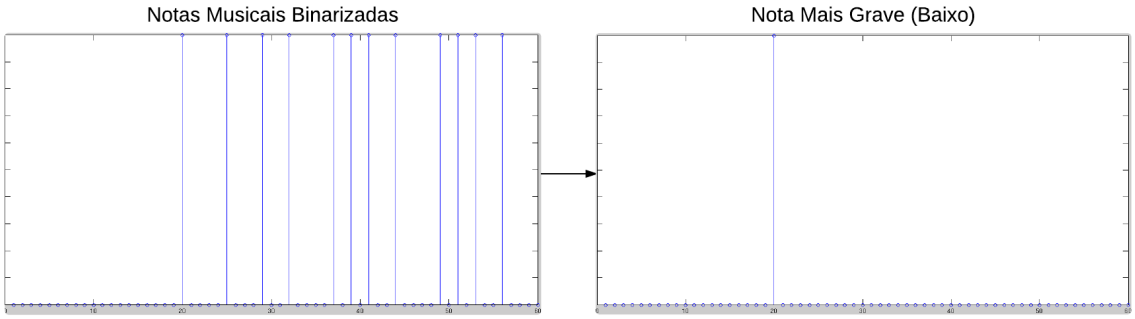

Figura 12: Extração da nota mais grave.

No procedimento verificamos que os primeiros passos são de atribuição de variáveis em relação as partes das janelas. Depois cada uma dessas partes serão analisadas quanto as notas mais recorrentes com a função **mode**. Dado essa análise os baixos são extraidos com a primeira ocorrência de 1, dado que as notas estão binarizadas.

3.2.7 Procedimento 7: Extrair Tonalidade

A extração de tonalidade é um módulo do sistema que possui como entrada o conjunto de notas binarizadas das partes de janela. A saída é o tom da música tocado baseando-se em acordes fundamentais maiores e menores. A figura 13 ilustra o processo descrito e o código do mesmo está presente na secção A.7 dos apêndices.

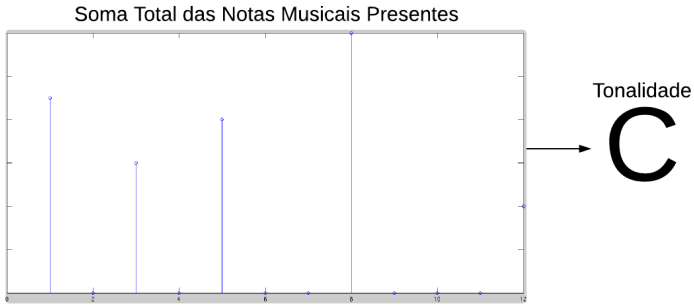

Figura 13: Extração da tonalidade da música.

No início desse processo há declaração nominal dos acordes em tipo string. Após o conjunto de notas em relação são somadas, cada uma na sua respectiva frequência, para gerar um vetor que totaliza a soma das frequências tocadas ao longo de todo áudio. O procedimento seguinte, focando extrair um acorde desse vetor de notas ao longo de todo áudio, é utilizado uma correlação do mesmo com uma base dados carregada de notas pelos respectivos acordes. Ao final cada acorde da base de dados terá sua energia correspondente e, ao extrair o máximo das energias, é adquirido o acorde tom da música.

3.2.8 Procedimento 8: Extrair Acordes Fundamentais

A extração de acordes fundamentais é um módulo do sistema que possui como entrada o conjunto de notas binarizadas das partes de janela. A saída é um conjunto de acordes fundamentais ao longo do tempo. A figura 14 ilustra o processo descrito e o código do mesmo está presente na secção A.8 dos apêndices.

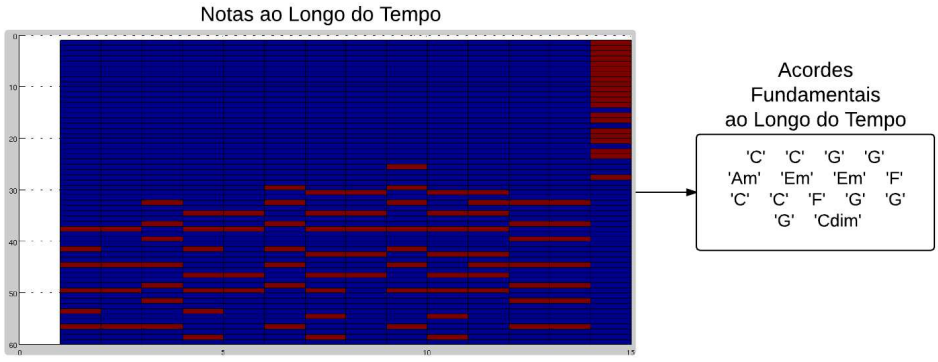

Figura 14: Extração de acordes fundamentais a cada segundo.

No início desse processo há o carregamento da base de dados de acordes em relação as notas musicais. Após o conjunto de notas são somadas em relação às respectivas oitavas gerando vetor de somente 12 posições. Esse mesmo vetor é submetido então a um processo de correlação aos acordes derivados da base de dados. Ao final cada acorde da base de dados terá sua energia correspondente e, ao extrair o máximo das energias, é adquirido os acordes fundamentais ao longo do tempo.

3.2.9 Procedimento 9: Extrair Acorde Recorrente das Partes

A extração de acorde recorrente é um módulo do sistema que possui como entrada o conjunto de acordes das partes de janela. A saída são acordes fundamentais ao longo do tempo com eliminação das 5 partes. A figura 15 ilustra o processo descrito e o código do mesmo está presente na secção A.9 dos apêndices.

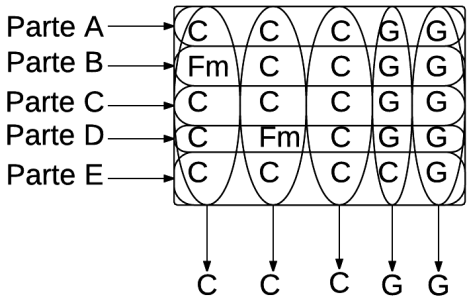

Figura 15: Aquisição do acorde mais recorrente dado as 5 partes deslocadas.

No início desse processo há a atribuição de variáveis a cada uma das 5 partes. Após o conjunto das partes em acordes é submetido a função **mode** para extrair o acorde mais recorrente dentro de uma dada faixa de tempo do áudio.

3.2.10 Procedimento 10: Extrair Acordes com Inversões

Da última etapa do sistema, a extração de acordes com inversões tem como finalidade formar acordes invertidos a partir da entrada dos acordes fundamentais e os baixos. A figura 16 ilustra o processo descrito e o código do mesmo está presente na secção A.10 dos apêndices.

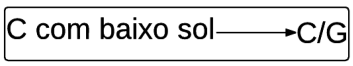

Figura 16: A construção de acordes invertidos e aumentados.

No início desse processo há o carregamento de um vetor de acordes, cada um com uma nomeclatura de acorde. Após é construído pares de acordes e baixos para mapear os acordes tocados dentro do vetor de acordes nominais. Ao final do processo as células de acordes e baixos identificados são referenciados dentro do vetor de acordes nominais com inversões, através das células de pares de acordes e baixos.

3.2.11 Modelo Matemático do Protótipo

Dado os procedimentos apresentados que consolidam o modelo matemático do protótipo proposto, a figura 17 mostra a interação deles em conjunto.

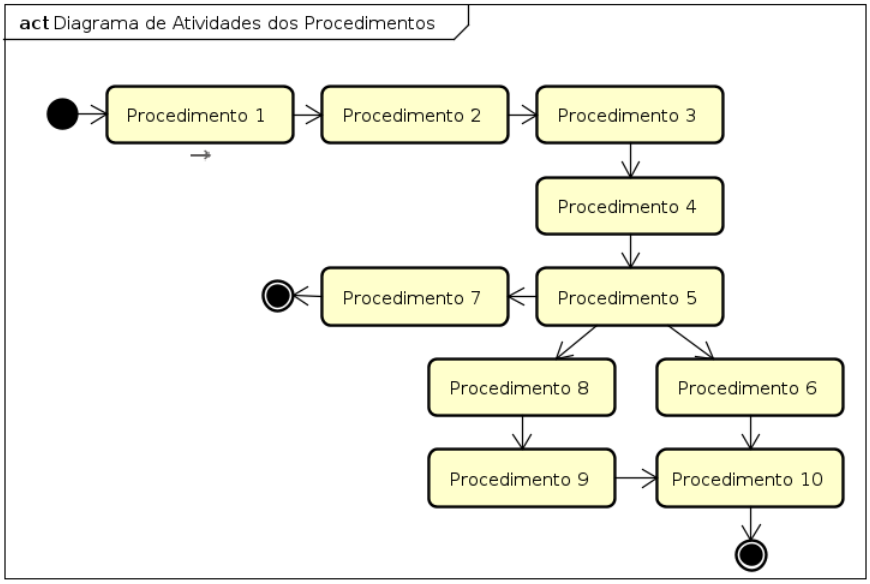

Figura 17: Diagrama de Atividades dos Procedimentos

O fluxo apresentado na figura 17 tem como entrada uma música em arquivo áudio e duas saídas: sequência de acordes a cada 1 segundo num vetor de *strings* e uma tonalidade da música numa *string*.

Dar-se-á, para o processo de extração de acordes ao longo do tempo, os seguintes procedimentos:

- **1 - Separar Janelas de 1 Segundo em 5 Partes**: cada 1 segundo do áudio-música é segmentado em 5 partes gerando uma estrutura de 5 janelas de 1 segundo;
- **2 - Aplicar Janelas de Blackman**: as janelas de áudio são submetidas a uma multiplicação com janelas de Blackman;
- **3 - Calcular o Espectro de Frequências**: para cada janela, são calculados espectros de frequência;
- **4 - Adquirir Energias das Notas**: dado os espectros de frequência, há o cálculo das energias das notas via camada de rede neural;
- **5 - Binarizar Energia das Notas**: as energias das notas musicais são binarizadas (1 ou 0), pois para o sistema só é preciso saber se as notas ocorreram e não a intensidade que elas ocorreram;
- **6 - Extrair Baixos**: os baixos de cada segundo são extraídos;

- **8 - Extrair Acordes Fundamentais**: os acordes fundamentais são extraídos via camada de rede neural;
- **9 - Extrair Acorde Recorrente das Partes**: os acordes fundamentais mais recorrentes são selecionados para cada 1 segundo de tempo;
- **10 - Extrair Acordes com Inversões**: os acordes fundamentais são combinados com os baixos extraídos para gerar os acordes invertidos e aumentados.

Dar-se-á, para o processo de extração de tonalidade musical, os seguintes procedimentos:

- **1 - Separar Janelas de 1 Segundo em 5 Partes**: cada 1 segundo do áudio-música é segmentado em 5 partes gerando uma estrutura de 5 janelas de 1 segundo;
- **2 - Aplicar Janelas de Blackman**: as janelas de áudio são submetidas a uma multiplicação com janelas de Blackman;
- **3 - Calcular o Espectro de Frequências**: para cada janela, são calculados espectros de frequência;
- **4 - Adquirir Energias das Notas**: dado os espectros de frequência, há o cálculo das energias das notas via camada de rede neural;
- **5 - Binarizar Energia das Notas**: as energias das notas musicais são binarizadas (1 ou 0), pois para o sistema só é preciso saber se as notas ocorreram e não a intensidade que elas ocorreram;
- **7 - Extrair Tonalidade**: todas as energias unitárias das notas ao longo da música são somadas gerando um vetor de ocorrência das notas. Dado esse vetor é calculado, usando a camada da rede neural de reconhecimento de acordes, o tom mais apropriado.

Em vista do que foi exposto, na seção seguinte é mostrado o desenvolvimento da aplicação das técnicas e procedimentos mostrados anteriormente. Também será discutido como cada procedimento obteve resultados satisfatórios.

3.3 Ciclos de Desenvolvimento

Nesta parte do trabalho serão descritos os ciclos de desenvolvimento para a construção do sistema-solução. Os detalhes e o código completo podem ser encontrados no repositório *github* [3].

[3] https://github.com/josepedro/TCC

3.3.1 Estrutura do Ciclo

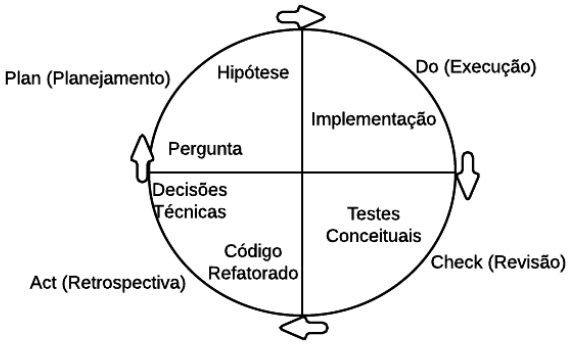

Figura 18: Modelo de ciclo de desenvolvimento adaptado.

Fases do ciclo de desenvolvimento:

- **Pergunta:** No início de cada ciclo é feito uma pergunta a ser respondida que se adere aos objetivos do trabalho.
- **Hipótese:** A partir dessa pergunta é feita hipóteses que possam responder. Vale ressaltar que a hipótese aqui referenciada é uma afirmação a ser verificada e validada e, portanto, difere da hipótese de uma questão de pesquisa;
- **Implementação:** As hipóteses são pensadas, construídas e implementadas num script;
- **Testes Conceituais:** Cada hipótese implementada é testada conforme a teoria usada;
- **Retrospectiva:** Dado os resultados dos testes conceituais e código refatorado, é feitto uma avaliação do que foi produzido e decisões técnicas são tomadas.

3.3.2 Ciclo 1: Espectro de Frequências

- **Pergunta:** Para saber da harmonia da música é preciso saber as notas dela. Se cada nota é uma frequência de vibração sonora, como analisar o sinal no ponto de vista de frequências? É possível?
- **Hipótese:** A Transformada de Fourier pode construir o espectro de frequências do sinal.
- **Implementação:**
  - o sinal é convertido para mono;

  - o sinal é normalizado;
  - é calculado a transformada de fourier do sinal;
  - o valor absoluto da transformada de fourier é calculado;
  - o valor absoluto da transformada de fourier é normalizado.

```
1 som = som(1:length(som));
2 som = som/max(som);
3 t = fft(som);
4 SINAL=sqrt(t.*conj(t));
5 SINAL=SINAL/max(SINAL);
```

- **Testes Conceituais:** Testes foram feitos para ver se os picos de frequência correspondem ao sinal de entrada. O resultado foi positivo e os picos representam a energia das frequências;
- **Retrospectiva:** A Transformada de Fourier, em específico a $FastFourierTransform$, realmente produz resultados satisfatórios em determinar o espectro de energia da frequências. Mas as informações não estão configuradas para localizar as notas musicais.

3.3.3 Ciclo 2: Realocação das Frequências em 1 Unidade de Hz

- **Pergunta:** Como configurar essas informações para localizar as notas musicais?
- **Hipótese:** Dado que cada nota musical é um conjunto de frequências, realocar as energias frequenciais da transformada de fourier a fim de que cada posição do vetor seja 1 unidedade de frequência (Hz) pode mapear a energia de cada nota.
- **Implementação:**
  - o valor da taxa de amostragem é definido;
  - um vetor de frequências é definido dado o tamanho do sinal;
  - é calculado o valor absoluto e normalizado da transformada de fourier do sinal;
  - a estrutura de laço recalcula as componentes da transformada de fourier de tal forma que cada posição do vetor seja 1 unidade em Hz de frequência.

```
1 fs = 44100;
2 f = (0:length(som)-1)*fs/length(som);
3 freq = f(1:round(length(f)/2));
4 SOM = abs(fft(som));
```

```
SOM = SOM/max(SOM);
SOM = SOM(1:round(length(f)/2));
l = 1;
j = 0;
i = 1;
SOMA = 0;
while (i<length(freq))
    if (round(freq(i)) == round(freq(i+1)))
        SOMA = SOM(i+1) + SOMA;
        j = j + 1;
    else
        respfreq(l) = SOMA/(j+1);
        j = 0;
        SOMA = SOM(i+1);
        l = l+1;
    end
    i = i+1;
end
l = 0; j = 0; i = 0;
```

- **Testes Conceituais:** De fato as notas musicais foram localizadas com mais facilidade em determinados grupos de frequências. Tal ordenamento de frequências resultou num vetor de 22050 posições, independentemente do tamanho da amostra.

- **Retrospectiva:** A estratégia de realocar as energias decimais das frequências numa unidade de frequência se adere corretamente ao objetivo de encontrar as notas musicais. Entretanto as frequências não estão associadas as notas musicais.

3.3.4 Ciclo 3: Frequências e Notas Musicais

- **Pergunta:** Dado um conjunto de frequências, como associar essas as notas musicais?

- **Hipótese:** Visto que associar as frequências as notas musicas é uma tarefa muito complexa para uma solução determinística, uma rede neural de aprendizado não supervisionado do tipo *Probabilistic Neural Network* (PNN) pode classificar um conjunto de frequências em sua respectiva nota musical.

- **Implementação:**

  – uma matriz de base de dados com as notas musicais e seus respectivos pesos é carregada;

  – um laço é executado para calcular a probabilidade (energia) de cada nota musical da saída de cada neurônio.

```
%BASE DE DADOS DE NOTAS MUSICAIS DA REDE NEURAL
%NOTAS
notas(12,22050) = 0; %matriz das notas
%Do grave
notas(1,61) = 0.1;
notas(1,62) = 0.2;
notas(1,63) = 0.4;
notas(1,64) = 0.6;
notas(1,65) = 0.8;
notas(1,66) = 1;
notas(1,67) = 0.8;
notas(1,68) = 0.6;
notas(1,69) = 0.4;
notas(1,70) = 0.2;
notas(1,71) = 0.1;
.
.
.
i = 1; %contador para andar ao longo do vetor
b = 0.15; %sensibilidade da rede
while (i <= 12)
      %S1(i) = exp(-(norm(rfeq - notas(i,:))*b));
      %correlacao = corrcoef(rfeq,notas(i,:));
      %S1(i) = correlacao(1,2);
      S1(i) = sum(abs(rfeq.* notas(i,:)));

    i = i + 1;
end
```

- **Testes Conceituais:** Foram testadas 3 funções de transferência do neurônio. A primeira função de transferência - a exponencial da subtração dos valores - não obteve resultados satisfatórios pois para notas adjacentes as mesmas eram confundidas pela rede. Esse fato se dá pelo retorno de baixa magnitude da subtração de valores. A segunda função de transferência - a correlação dos valores - caracterizou as notas. Porém a operação de subtração da média faz com que a energia final seja baixa, além de requisitar mais operações. A terceira função de transferência - a multiplicação dos valores - foi caraterizou as notas e se mostrou rápida, em relação as funções de transferência anteriores, pois envolve poucas operações matématicas.

- **Retrospectiva:** A rede neural PNN foi bastante classificou corretamente as frequências em termos de notas musicais. Porém as notas musicais não estão associadas a acordes musicais.

3.3.5 Ciclo 4: Determinação dos Acordes Fundamentais

- **Pergunta:** Como adicionar as próximas camadas da rede para determinação dos acordes?
- **Hipótese:** Para poder mapear as notas é preciso adicionar mais 2 camadas. Uma camada para classificação de acordes, dado um conjunto de notas musicais e a outra para classificação de um acorde dado os conjuntos possíveis de acordes.
- **Implementação:**
  - uma matriz de base de dados com acordes musicais e seus respectivos pesos é carregada;
  - um laço é executado para calcular a probabilidade (energia) de cada acorde musical da saída de cada neurônio.

```
% BASE DE DADOS PARA ACORDES

%------------------------------
BD(12,48) = 0;
%------------------------------

afin1 = 0; afin2 = 0;

%C
%CM
BD(12,1) = afin1;
BD(1,1) = 1; %baixo
BD(2,1) = afin2;
BD(4,1) = afin1;
BD(5,1) = 1; %terca
BD(6,1) = afin2;
BD(7,1) = afin1;
BD(8,1) = 1; %quinta
BD(9,1) = afin2;

.
.
.
%(430 LINHAS DE ACORDES)

%RADIAL BASIS LAYER para BD notas

while (i <= 48)
    S2(i) = sum(abs(S1.* BD(i,:)));
    i = i + 1;
```

```
31 end
```

- **Testes Conceituais:** Foram testadas todas as possibilidades de acordes e os que não foram evetivamente reconhecido foram as inversões e os acordes aumentados.
- **Retrospectiva:** De certo o acerto não foi total pois falta implementar uma camada que reconheça inversões.

3.3.6 Ciclo 5: Detecção de Transições Rítmicas

- **Pergunta:** Como reconhecer acordes no tempo de tal forma a saber onde eles ocorrem?
- **Hipótese:** Uma solução de reconhecimento de energias ao longo do tempo pode determinar o rítmo. Em tese é calcular a energia do sinal, aplicar um filtro passa-baixas para suavização do sinal e aplicar um método de auto-correlação com o intuito de caracterizar picos de repetição de níveis elevados de energia.
- **Implementação:**
  - um sinal de áudio é carregado;
  - é calculado o valor absoluto do sinal de áudio;
  - o valor absoluto do sinal de áudio é filtrado por um passa-baixas;
  - é feito uma subamostragem de 22050 no sinal;
  - o sinal é autocorrelacionado para gerar os picos de energia correlacionadas.

```
1      % opening the file
2      file = open(file_path);
3      file.data = file.data(1 : file.fs*4);
4      bpm_music = abs(file.data);
5      %filtering the pulses of minor energy
6      signal_filtered = filter_signal(bpm_music);
7      % Building array with means movies
8      %signal_pulses = decrease_resolution(signal_filtered, file.fs, 1000);
9      signal_pulses = downsample(signal_filtered, 22050);
10     % Beginnnig the correlation
11     array_correlation = correlate_moments(signal_pulses);
```

- **Testes Conceituais:** Para um caso específico a solução funcionou, porém os outros casos ela não se aderiu.

- **Retrospectiva:** De certo modo detectar onde acorde ocorre no sinal não agrega valor para o escopo desse trabalho pois os níveis de energia são muito variáveis e não há um padrão como para a detecção.

3.3.7 Ciclo 6: Aplicação da Transformada Wavelets

- **Pergunta:** Como ler o sinal todo e ter a visilibilidade dele em tempo e frequência?
- **Hipótese:** Para se ter uma resolução completa do sinal em tempo em frequência é preciso de ter uma transformada que agregue esses dois aspectos. Poder testar isso com a transformada wavelets.
- **Implementação:**
  - são definidos critérios de parada da iteração de filtros da transformada wavelets;
  - são definidos iterações de filtros passa-baixas do tipo daubechies 45;
  - são definidos iterações de filtros passa-altas do tipo daubechies 45.

```
1  function [signal, imin, imax, iterations, energy] = ...
2  tree_iterator(signal, mini, maxi, imin, imax, iterations, energy)
3
4  if iterations == 0
5      if mini >= 1 && maxi <= 22050
6        [signal, imin, imax, iterations, energy] = ...
7        tree_iterator(signal, mini, maxi, 1, 22050, 1, 0);
8      else
9        imin = 1;
10       imax = 22050;
11       return;
12 end
13 elseif iterations > 0
14     imean = (imax - imin)/2 + imin;
15     %Low
16     if mini >= imin && maxi <= imean
17       [h0, h1] = wfilters('db45');
18       [signal, y1] = decomposition_1level_qmf(h0, h1, signal);
19       energy = sum(abs(signal)) + energy;
20       iterations = iterations + 1;
21       imax = imean;
22       [signal, imin, imax, iterations, energy] = ...
23       tree_iterator(signal, mini, maxi, imin, ...
24       imax, iterations, energy);
25     %High
26     elseif mini >= imean && maxi <= imax
27       [h0, h1] = wfilters('db45');
```

```
28        [y0, signal] = decomposition_1level_qmf(h0, h1, signal);
29        energy = sum(abs(signal)) + energy;
30        iterations = iterations + 1;
31        imin = imean;
32        [signal, imin, imax, iterations, energy] = ...
33        tree_iterator(signal, mini, maxi, ...
34        imin, imax, iterations, energy);
35      else
36        return;
37      end
38  end
```

- **Testes Conceituais:** A solução falhou num teste muito simples. Ao submeter um sinal puro numa frequencia determinada e constante o banco de filtros wavelets distorcia o sinal, deslocando a fase do sinal para frequencias adjacentes da original.
- **Retrospectiva:** Dado a barreira técnica de deslocamento de fase do sinal, ainda não foi encontrado uma solução de resolução tempo-frequência.

3.3.8 Ciclo 7: Aplicação da Transformada de Fourier Janelada

- **Pergunta:** Como ler o sinal todo e ter a visilibilidade dele em tempo e frequência?
- **Hipótese:** Para se ter uma resolução completa do sinal em tempo em frequência é preciso de ter uma transformada que agregue esses dois aspectos. Poder testar isso com a $Short\ Fourier\ Transform$ (Transformada de Fourier Janelada).
- **Implementação:**
  - são carregados as bases de dados de notas e acordes musicais;
  - o tempo total da música é definido;
  - um laço, que irá percorrer segundo a segundo a música, é executado calculando a STFT em cada segundo.

```
1  function [notes_time, chords_time, chord_pitch] = DA3(signal, fs)
2
3  load_notes;
4  load_chords;
5  .
6  .
7  .
8  % begin to analyse music
9  time_seconds_total = fix((length(signal)/fs));
10 notes_time(time_seconds_total, 60) = 0;
```

```
chords_time = {};
for time = 1:time_seconds_total
    signal_time = signal(1+((time-1)*fs):time*fs);
    window = blackman(length(signal_time));
    signal_time = window'.*signal_time;
    signal_time = downsample(signal_time, 21);
    fs_time = fs/21;
    module_fft = abs(fft(signal_time));
    respfreq(1:fs_time) = 0;
    window_mean = length(signal_time)/fs_time;
    for frequency = 1:fs_time
        respfreq(frequency) = sum(module_fft( ...
        1+((frequency-1)*window_mean):frequency* ...
        window_mean))/window_mean;
    end
    respfreq = respfreq(1:fix(length(respfreq)/2));
    for note = 1:60
        notes_time(time, note) = sum(respfreq.* ...
        notes(note,:));
    end
    energy_chords(1:48) = 0;
    for chord = 1:48
        energy_chords(chord) = sum(notes_time(time, :) ...
        .*chords(chord,:));
    end
    chords_time{time} = dictionary_chords{ ...
    find(energy_chords==max(energy_chords))};
end
notes_energy_total = notes_time(1,:);
for time = 2:time_seconds_total
    notes_energy_total = notes_energy_total + notes_time(time,:);
end
energy_chords(1:48) = 0;
for chord = 1:48
    energy_chords(chord) = sum(notes_energy_total.*chords(chord,:));
end
chord_pitch = dictionary_chords{find(energy_chords== ...
max(energy_chords))};
```

- **Testes Conceituais:** A solução da transformada de fourier janelada foi testada com acordes de violão e piano ao longo do tempo e o resultado foi satisfatório exceto para acordes de transição.

- **Retrospectiva:** A solução da transformada de fourier janelada se encaixou bem no conjunto.

3.3.9 Ciclo 8: Extração de Tonalidade

- **Pergunta:** Como extrair o tom da música?
- **Hipótese:** Para extrair o tom da música é preciso somar a energia das notas totais ao longo da música e em seguida extrair o acorde mais provável.
- **Implementação:**
  - todas as ocorrências de notas da música são somadas;
  - as oitavas das notas também são somadas;
  - é calculado o tom mais provável para a música.

```
1  function [chord_pitch, chord_pitch_number] = ...
2  get_chord_pitch(notes_time, time_seconds_total, chords)
3
4    dictionary_chords = ...
5  .
6  .
7  .
8    notes_energy_total(60) = 0;
9    for note = 1:60
10     notes_energy_total(note) = sum([notes_time(:,note)]);
11   end
12
13   % discover tone music
14   notes_energy_tone(12) = 0;
15   for note = 1:12
16     notes_energy_tone(note) = notes_energy_total(note) ...
17      + notes_energy_total(note + 12) ...
18       + notes_energy_total(note + 2*12) + ...
19       notes_energy_total(note + 3*12) ...
20         + notes_energy_total(note + 4*12);
21   end
22
23   % find chord tone
24   load_chords_tone;
25   chords_tone(48) = 0;
26   for chord = 1:48
27     chords_tone(chord) = sum((notes_energy_tone.* ...
28     chords_tone_mask(:, chord)'.^2));
29   end
30
31   chord_pitch_number = find(chords_tone==max(chords_tone));
32   chord_pitch = dictionary_chords{chord_pitch_number};
```

- **Testes Conceituais:** A solução foi testada com sequencia de acordes do violão e as vezes o tom não é o certo.
- **Retrospectiva:** A quantidade de energia está atrapalhando a extração do tom. O tom é definido como a frequência de aparecimento das notas ao longo da música.

3.3.10 Ciclo 9: Binarização de Energia das Notas

- **Pergunta:** Como trabalhar com a frequencia de aparecimento das notas?
- **Hipótese:** Se binarizar com 1 e 0 o mapa de notas no tempo a soma das notas será unitária, equivalente a frequência.
- **Implementação:**
  - o primeiro laço define qual parte será binarizada;
  - o segundo laço define o instante de tempo que será binarizado;
  - o terceiro laço define as notas que serão binarizadas.

```
% binarize set of notes
    for set = 1:5
        notes_time = set_of_notes_time{set};

        for time = 1:time_seconds_total
            for note = 1:60
                if notes_time(time, note) < max(max(notes_time))/180
                    notes_time(time, note) = 0;
                else
                    notes_time(time, note) = 1;
                end
            end
        end

        set_of_notes_time{set} = notes_time;
    end
```

- **Testes Conceituais:** A solução foi testada e verificada com picos somente de 1 e vales somente de 0.
- **Retrospectiva:** Com essa binarização o tom da música foi efetivamente corrigido.

3.3.11 Ciclo 10: Segmentação de Áudio

- **Pergunta:** Em relação aos acordes transitórios, como corrigir?

- **Hipótese:** Se ao deslocar a janela de tamanho de 1 segundo em passos de 0.2 segundos e calcular o espectro de frequência de cada passo pode-se fazer a média do acorde de cada tempo e poder cancelar os acordes transitórios.
- **Implementação:**
  - a parte A é definida com 0 segundos de deslocamento;
  - a parte B é definida com 0.2 segundos de deslocamento;
  - a parte C é definida com 0.4 segundos de deslocamento;
  - a parte D é definida com 0.6 segundos de deslocamento;
  - a parte E é definida com 0.8 segundos de deslocamento;
  - as 5 partes são associadas a uma estrutura de célula.

```
1  function set_of_windows_signals = ...
2  build_window_short_fft(signal, time, fs)
3    signal = [signal(:)];
4
5      % part A
6      time_start_A = round(1+((time-1)*fs));
7      time_end_A = round(time*fs);
8      signal_time_A = signal(time_start_A:time_end_A);
9      window = blackman(length(signal_time_A));
10     signal_time_A = window.*signal_time_A;
11
12     % part B (displacement = + 0.2 seconds)
13     time_start_B = round(1+((time-1)*fs+0.2*fs));
14     time_end_B = round((time+0.2)*fs);
15     if time_start_B < length(signal) && time_end_B <= length(signal)
16         signal_time_B = signal(time_start_B:time_end_B);
17         window = blackman(length(signal_time_B));
18         signal_time_B = window.*signal_time_B;
19     else
20         signal_time_B(length(signal)) = 0;
21     end
22
23     % part C (displacement = + 0.4 seconds)
24     time_start_C = round(1+((time-1)*fs+0.4*fs));
25     time_end_C = round((time+0.4)*fs);
26     if time_start_C < length(signal) && time_end_C <= length(signal)
27         signal_time_C = signal(time_start_C:time_end_C);
28         window = blackman(length(signal_time_C));
29         signal_time_C = window.*signal_time_C;
30     else
31         signal_time_C(length(signal)) = 0;
```

```
32      end
33
34      % part D (displacement = + 0.6 seconds)
35      time_start_D = round(1+((time-1)*fs+0.6*fs));
36      time_end_D = round((time+0.6)*fs);
37      if time_start_D < length(signal) && time_end_D <= length(signal)
38          signal_time_D = signal(time_start_D:time_end_D);
39          window = blackman(length(signal_time_D));
40          signal_time_D = window.*signal_time_D;
41      else
42          signal_time_D(length(signal)) = 0;
43      end
44
45      % part E (displacement = + 0.8 seconds)
46      time_start_E = round(1+((time-1)*fs+0.8*fs));
47      time_end_E = round((time+0.8)*fs);
48      if time_start_E < length(signal) && time_end_E <= length(signal)
49          signal_time_E = signal(time_start_E:time_end_E);
50          window = blackman(length(signal_time_E));
51          signal_time_E = window.*signal_time_E;
52      else
53          signal_time_E(length(signal)) = 0;
54      end
55
56      set_of_windows_signals = {};
57      if length(signal_time_A) == length(signal_time_B) && ...
58          length(signal_time_A) == length(signal_time_C) && ...
59           length(signal_time_A) == length(signal_time_D) &&  ...
60            length(signal_time_A) == length(signal_time_E)
61          set_of_windows_signals{1} = signal_time_A;
62          set_of_windows_signals{2} = signal_time_B;
63          set_of_windows_signals{3} = signal_time_C;
64          set_of_windows_signals{4} = signal_time_D;
65          set_of_windows_signals{5} = signal_time_E;
66      else
67          set_of_windows_signals{1} = signal_time_A;
68          set_of_windows_signals{2} = signal_time_A;
69          set_of_windows_signals{3} = signal_time_A;
70          set_of_windows_signals{4} = signal_time_A;
71          set_of_windows_signals{5} = signal_time_A;
72      end
```

- **Testes Conceituais:** Os testes foram feitos em series de acordes tocados no violão e de fato os acordes transitórios desapareceram.
- **Retrospectiva:** Com essa correção os acordes maiores, menores e diminutos estão sendo reconhecidos corretamente.

3.3.12 Ciclo 11: Extração de Baixos

- **Pergunta:** Como extrair a nota mais grave (baixo) de cada período do tempo?
- **Hipótese:** Com o mapa de notas binarizado é possível extrair o baixo pegando a primeira posição com valor 1 de cada tempo.
- **Implementação:**
  - são definidas as partes de janela em suas respectivas variáveis;
  - são extraídos os valores de notas mais recorrentes em cada instante de tempo;
  - são extraídos os baixos dado o primeiro maior valor máximo;
  - é calculado o módulo da posição do baixo para que o mesmo sempre fique nas posições mais iniciais do vetor.

```
1  function bass_time = get_bass(set_of_notes_time)
2
3    notes_time_A = set_of_notes_time{1};
4    notes_time_B = set_of_notes_time{2};
5    notes_time_C = set_of_notes_time{3};
6    notes_time_D = set_of_notes_time{4};
7    notes_time_E = set_of_notes_time{5};
8
9    total_seconds = length(notes_time_A(:,1));
10   notes_time(total_seconds, 60) = 0;
11   for time = 1:total_seconds
12     for note = 1:60
13       notes_to_analyse = [notes_time_A(time, note) ...
14        notes_time_B(time, note) ...
15      notes_time_C(time, note) ...
16      notes_time_D(time, note) notes_time_E(time, note)];
17       notes_time(time, note) = mode(notes_to_analyse);
18     end
19   end
20
21     bass_time(1:total_seconds) = 0;
22   for time = 1:total_seconds
23     maxs = find(notes_time(time,:)==max(notes_time(time,:)));
24     bass_time(time) = maxs(1);
25   end
26
27   for bass = 1:length(bass_time)
28     bass_time(bass) = mod(bass_time(bass) - 1, 12) + 1;
29   end
30
```

```
31 end
```

- **Testes Conceituais:** Os testes foram feitos com acordes e de fato ele reconhece as notas mais graves.
- **Retrospectiva:** Dado os baixos definidos, os acordes com inversões e aumentados não estão incluídos.

3.3.13 Ciclo 12: Extração de Acordes Invertidos e Aumentados

- **Pergunta:** Como incluir a nota mais grave (baixo) de cada período do tempo aos acordes aumentados e invertidos?
- **Hipótese:** Pode-se referenciar um acorde invertido ou aumentado apartir da combinação de acordes fundamentais e baixos extraídos para cada período respectivo.
- **Implementação:**
  - acordes fundamentais e baixos são associados;
  - dado uma associação de acordes fundamentais e baixos os mesmos são referenciados como string num vetor de dicionário de acordes.

```
1  function chords_with_bass = ...
2  get_chords_bass(chords_number, bass_time)
3  dictionary_chords = ...
4  .
5  .
6  .
7   % build chords with bass to translate to dictionary
8   chords_with_bass_number = {};
9   chord_iterator = 1;
10  for chord = 1:48
11   for bass = 1:12
12     chords_with_bass_number{chord_iterator} = [chord, bass];
13     chord_iterator = chord_iterator + 1;
14   end
15  end
16  chords_with_bass = {};
17  for time = 1:length(chords_number)
18    for chord = 1:length(chords_with_bass_number)
19     peer_chord = chords_with_bass_number{chord};
20     if peer_chord(1) == chords_number(time) && ...
21     peer_chord(2) == bass_time(time)
22       chords_with_bass{time} = dictionary_chords{chord};
```

```
23          end
24        end
25      end
26    end
```

- **Testes Conceituais:** Todas as possibilidades de acordes foram reconhecidos com sucesso.
- **Retrospectiva:** Tal hipótese resolveu definitivamente a problemática de acordes invertidos e aumentados identificados de forma incorreta.

3.4 Protótipo da Solução Proposta

O protótipo da solução proposta codificada em MATLAB se encontra no repositório do *github*[4]. A figura 19 mostra a página principal do repositório.

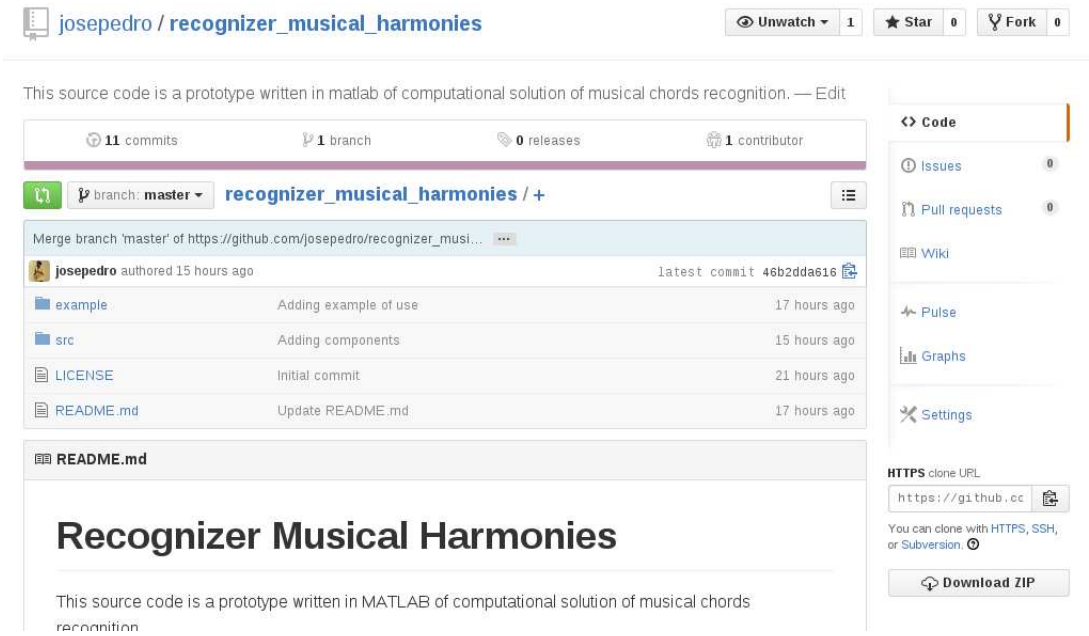

Figura 19: Página principal do repositório.

A estrutura de pastas na raiz do projeto está da seguinte forma:

- **example**: pasta contendo amostras de música em arquivo WAVE e um *script* contendo um exemplo de uso do protótipo;
- **src**: pasta contendo os *scripts* modularizados do protótipo ;
- **LICENSE**: arquivo de texto contendo o tipo de licença que é do tipo *Apache License*, *Version* 2.0;

[4] https://github.com/josepedro/recognizer_musical_harmonies

- **README**: arquivo de texto contendo informações iniciais e instruções para executar o *script* de exemplo de uso.

A seguinte estrutura está presente dentro da pasta **src**:

- **audio_processing**: módulo para processar o sinal de áudio. É formado pelos scripts:
  - **build_window_short_fft.m**: *script* resultado referente aos ciclos de desenvolvimento 7 (subseção 3.3.8) e 10 (subseção 3.3.11);
  - **get_frequency_spectrum.m**: *script* resultado referente aos ciclos de desenvolvimento 1 (subseção 3.3.2), 2 (subseção 3.3.3) e 7 (subseção 3.3.8).
- **chords_extractor**: módulo para extrair acordes ao longo do tempo. É formado pelos scripts:
  - **analyse_set_of_chords.m**: *script* resultado referente ao ciclo de desenvolvimento 10 (subseção 3.3.11);
  - **get_chords_bass.m**: *script* resultado referente ao ciclo de desenvolvimento 12 (subseção 3.3.13);
  - **get_set_of_chords_time.m**: *script* resultado referente ao ciclo de desenvolvimento 4 (subseção 3.3.5);
  - **load_chords_tone.m**: *script* resultado referente ao ciclo de desenvolvimento 4 (subseção 3.3.5);
  - **load_dictionary_chords.m**: *script* resultado referente ao ciclo de desenvolvimento 12 (subseção 3.3.13).
- **key_extractor**: módulo para extrair tonalidade musical. É formado pelo script:
  - **get_chord_pitch.m**: *script* resultado referente ao ciclo de desenvolvimento 8 (subseção 3.3.9).
- **notes_extractor**: módulo para extrair notas musicais ao longo do tempo. É formado pelos scripts:
  - **binarize_notes.m**: *script* resultado referente ao ciclo de desenvolvimento 9 (subseção 3.3.10);
  - **get_bass.m**: *script* resultado referente ao ciclo de desenvolvimento 11 (subseção 3.3.12);
  - **get_energy_notes.m**: *script* resultado referente ao ciclo de desenvolvimento 3 (subseção 3.3.4);

  - **load_notes.m**: *script* resultado referente ao ciclo de desenvolvimento 3 (subseção 3.3.4).

- **main.m**: *script* principal que irá executar os módulos e foi formado incrementalmente no andamento dos ciclos citados.

A figura 20 mostra o diagrama de componentes dos *scripts* que formam o protótipo da solução proposta. Pode-se observar que o *script* principal *main.m* é responsável por fazer chamadas de todos os *scripts* exceto os que estão relacionados a alocação de constantes e dados para a rede neural, exemplos desses *scripts* são **load_notes.m**, **load_chords_tone.m** e **load_dictionary_chords.m**.

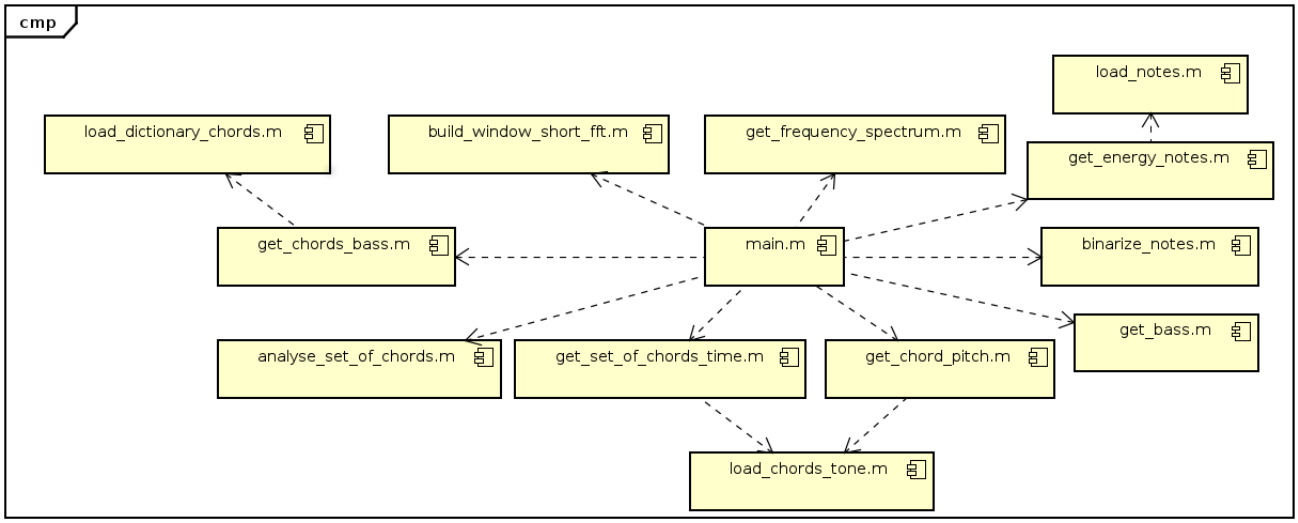

Figura 20: Diagrama de componentes do protótipo da solução proposta.

3.5 Projeto da Solução Computacional

3.5.1 Escopo

O projeto da solução computacional do protótipo apresentado está inserido no contexto de uma ferramenta computacional para extração de informações harmônicas numa música, contida inicialmente num arquivo de áudio $WAVE$. Essa ferramenta deverá ser implementada em linguagem de programação Python, pois a mesma possui suporte multiparadigma, bibliotecas específicas para operações matriciais, métodos numéricos e processamento de sinais, além ser de fácil suporte, manutenção e evolução.

3.5.2 Requisitos Funcionais

Dado o contexto de extração de informações harmônicas numa música em arquivo $WAVE$, os requisitos funcionais são:

- Extrair acordes (maiores, menores, aumentados, diminutos e invertidos) ao longo do tempo com 1 segundo de precisão e dispor ao usuário uma *string* com os acordes;
- Extrair tonalidade musical e dispor ao usuário uma *string* com o tom da música.

3.5.3 Modelagem do Projeto

A figura 21 mostra o diagrama de componentes dos módulos básicos de domínio da solução. São 5 componentes básicos de domínio:

- **main.py**: componente responsável por iniciar o processo, configurar dependências e executar os outros módulos;
- **Processador de Áudio**: componente responsável por processar o áudio e extrair informações de domínio da frequência necessárias para identificação de notas;
- **Extrator de Notas**: componente responsável por extrair notas musicais ao longo do tempo numa precisão de 1 segundo, dado as informações de espectro de frequência do áudio;
- **Extrator de Tom**: componente responsável por extrair tom da música, dada as notas extraídas;
- **Extrator de Acordes**: componente responsável por extrair acordes (maiores, menores, aumentados, diminutos e invertidos), ao longo da música, numa precisão de 1 segundo.

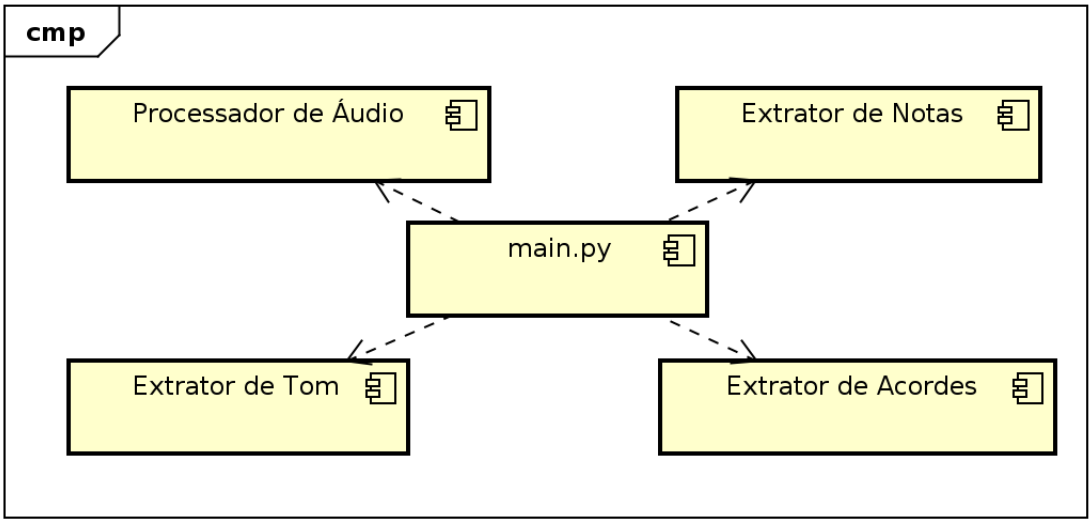

Figura 21: Diagrama de componentes básicos de domínio.

No ponto de vista de scprits e dependências da linguagem Python, a figura 22 mostra o diagrama de componentes dos mesmos.

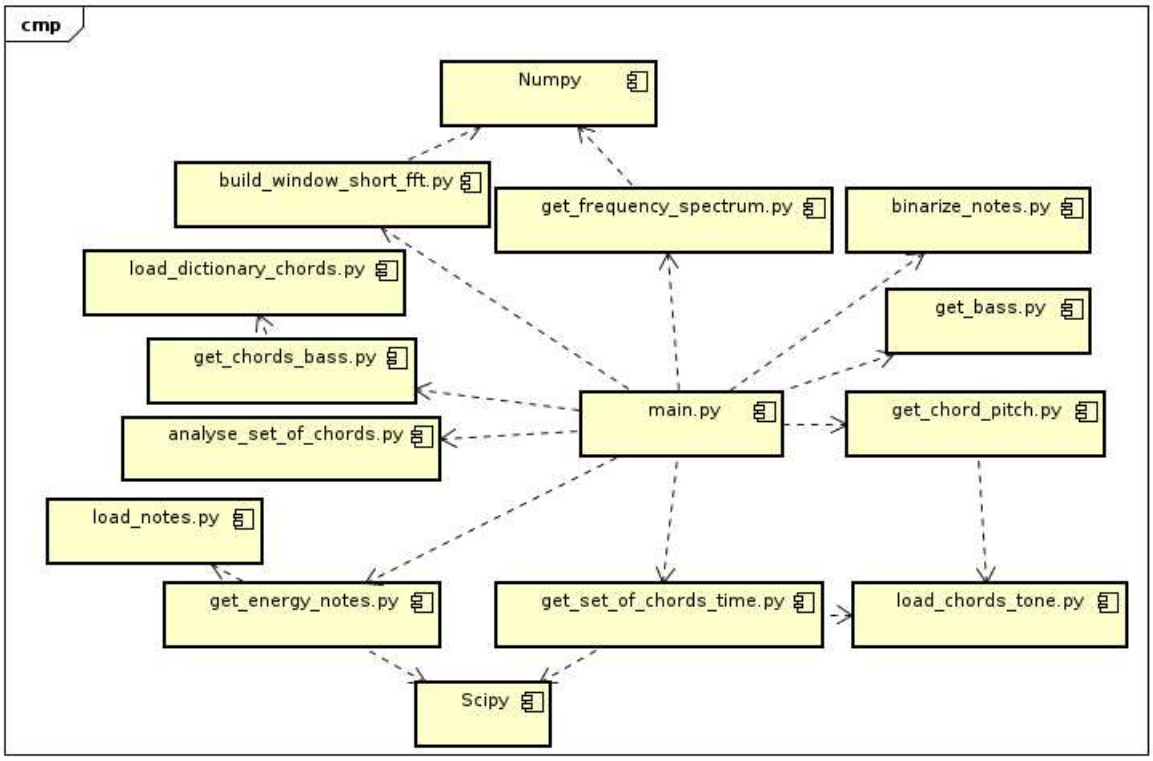

Figura 22: Diagrama de componentes básicos de domínio.

Para que o projeto seja viável, a solução proposta necessita de 2 bibliotecas escritas em Python que são:

- **Numpy**[5]: biblioteca para cálculos científicos fundamentais em Python. Possui um objeto matriz N-dimensional e vastas operações de álgebra linear. Para esse contexto essa biblioteca será usada para as operações de janelamento de blackman ($numpy.blackman$) e transformada de fourier ($numpy.fft$);

- **Scipy**[6]: biblioteca para cálculos matemáticos, estatísticos e probabilísticos. Ela é mais abrangente que Numpy porém é focada para cálculos matemáticos básicos. Nela serão usadas as funções de correlação linear de pearson ($scipy.stats.pearsonr$).

Os *scripts* próprios do projeto seguem a mesma estrutura e funcionamento dos expostos na seção 3.4. Porém vale ressaltar que cabe adaptações para que o objeto matemático-computacional alocado pela biblioteca Numpy possa ser adequadamente processado.

Em visto do que foi desenvolvido nesse capítulo, foi apresentado a metodologia de execução do presente trabalho, o desenvolvimento do protótipo do sistema-solução como um todo, o andamento em ciclos de execução de desenvolvimento da solução proposta, a

[5] Mais informações sobre essa biblioteca podem ser encontradas em http://www.numpy.org/
[6] Mais informações sobre essa biblioteca podem ser encontradas em http://www.scipy.org/

apresentação do protótipo da solução em MATLAB e o projeto em software da solução computacional em Python. Vale ressaltar que cada módulo do sistema-solução desenvolvido gera resultados no que tange o processamento de áudio, que serão apresentados no capítulo seguinte.

4 Resultados

Em vista dos procedimentos teóricos aliados a uma solução computacional, obteve-se os seguintes resultados:

- resposta em frequência;
- sugestão de notas;
- sugestão de acordes;
- detecção de transições rítmicas;
- implementação da transformada wavelets;
- transcrição de notas ao longo do tempo;
- transcrição automática de acordes ao longo do tempo;
- extração da tonalidade;
- exemplo de uso.

4.1 Resposta em Frequência e Sugestões de Notas e Acordes

Para a demonstração dos resultados dos procedimentos 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.2.8, 3.2.9 e 3.2.10 e ciclos de desenvolvimento 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.8, 3.3.10, 3.3.12 e 3.3.13 foram feitos experimentos[1] com todas as possibilidades de reconhecimento de acordes proporcionados pelo sistema. Todavia serão detalhados e comentados somente 4 pois para os outros equivalem as mesmas considerações. O resumo dos resultados dos outros acordes estará presente na tabela que se segue logo após.

4.1.1 Pré-condições dos Experimentos

No que tange às pré-condições foram levados em conta:

- teclado yamaha E413 com som de piano para a execução dos acordes;
- somente tríades (3 notas) tocadas;

[1] Nesse trabalho contextualizado na computação musical chama-se de experimento o procedimento de verificar saídas da solução computacional dada um conjunto de entradas. Para tal há condições pré-determinadas de ambiente acústico e forma de processamento para que os mesmos possam ser realizados por terceiros.

- o software Audacity [2] foi utilizado para gravação;
- o microfone convencional interno do *notebook* foi utilizado para aquisição dos sinais de áudio;
- o ruído de fundo estava com uma grandeza por volta de 45 db;
- a taxa de amostragem do sinal foi configurada em 44100 Hz;
- gravação do áudio no formato de arquivo .wav em codificação 16 pcm.

As tríades de acordes foram executadas com base na nota central $C4$ que possui o valor de aproximadamente 261,6 Hz. A figura 23 ilustra as regiões e limites usados.

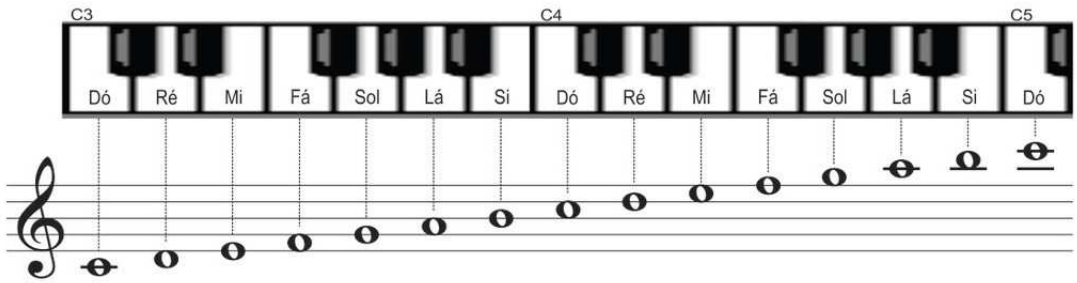

Figura 23: Teclado ilustrativo para execução dos acordes (DOZOL, 2014).

O processo de execução do experimento foi dividido em 4 etapas. A primeira relativa a gravação do acorde tocado no teclado via microfone convencional interno do *notebook*. A segunda é a exportação do som no formato de arquivo .wav pelo software audacity. A terceira etapa é a introdução do arquivo na entrada do sistema de detecção de acordes. A última atividade é a classificação do arquivo digital num acorde. A figura 24 ilustra o processo esquematizado.

Figura 24: Processo ilustrativo da execução dos experimentos.

[2] http://www.audacity.sourceforge.net

4.2 Experimento 1 - Acorde DM

Nesse experimento foi tocado a tríade *Ré* (baixo e tônica), *Fá#* e *Lá* equivalente ao acorde DM. A tríade foi tocada ao mesmo tempo e com a mesma força para todas as notas.

Segue os gráficos resultantes:

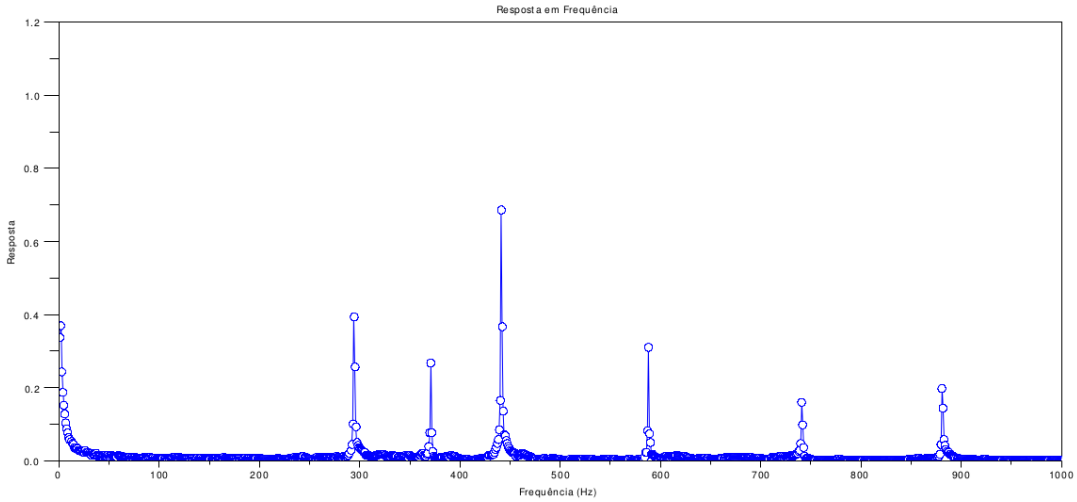

Figura 25: Gráfico da resposta em frequência para a gravação do acorde DM

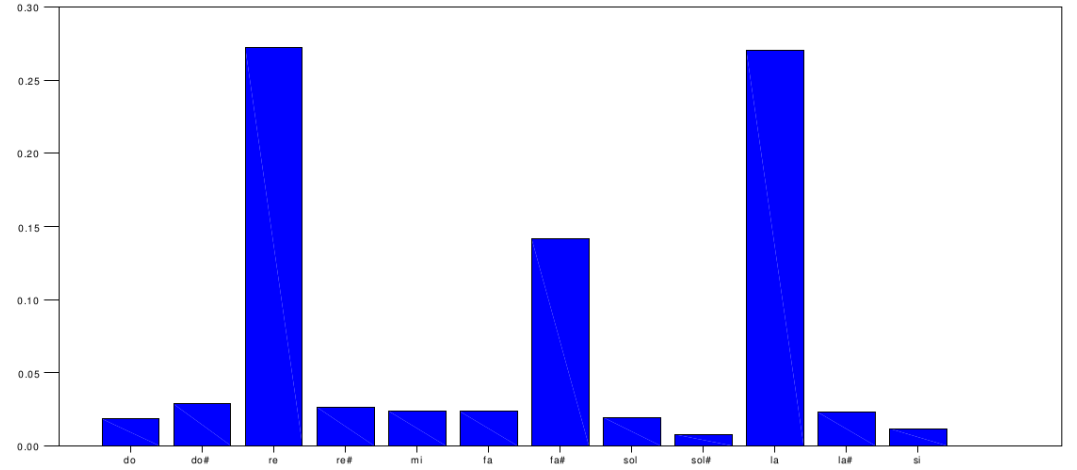

Figura 26: Gráfico de sugestão de notas para a gravação do acorde DM

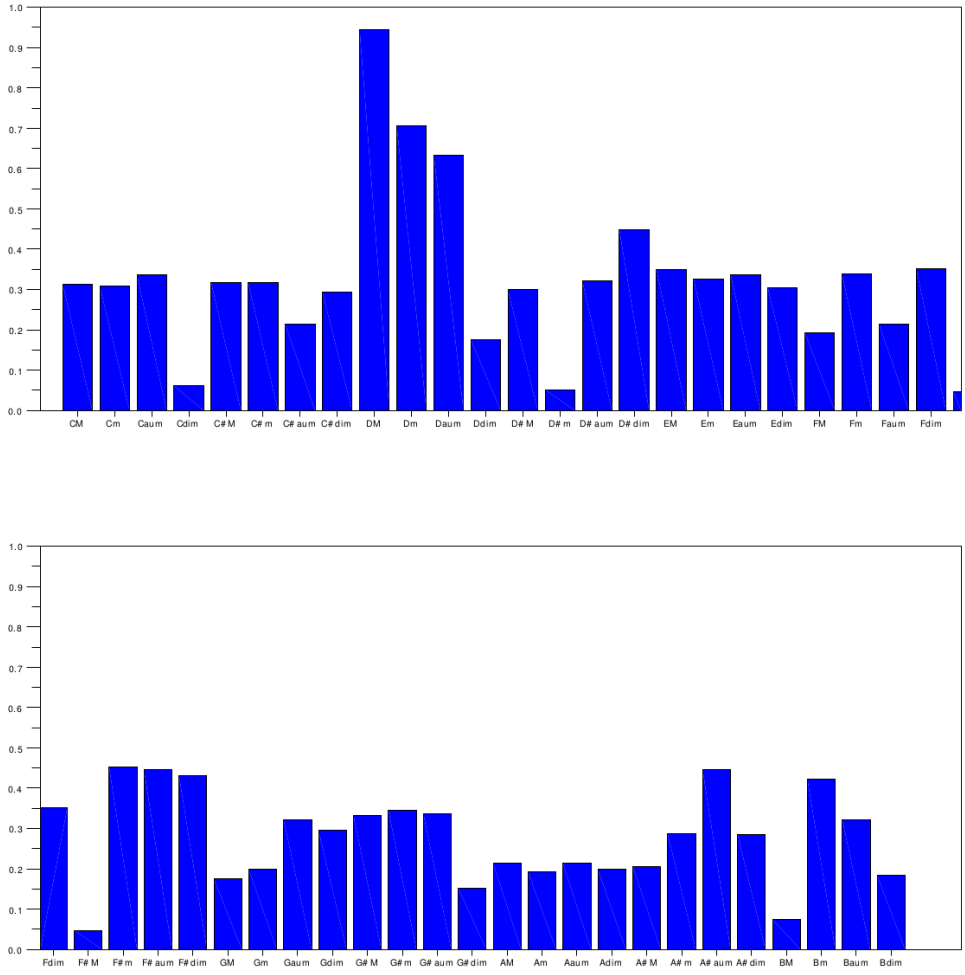

Figura 27: Gráficos de sugestão de acordes a gravação do acorde DM

Do resultado da primeira camada de processamento é gerado o gráfico da Figura 16. Esse gráfico diz respeito a natureza da composição do sinal em senoides em termos de transformada de fourier. O primeiro pico, no valor de 294 Hz, é relativo a nota *Ré*. O segundo pico, no valor de 371 Hz, é relativo a nota *Fá#*. O terceiro pico, no valor de 441 Hz, é relativo a nota *Lá*. Os picos seguintes são relativos aos harmônicos dessas três notas.

Do resultado da segunda camada de processamento é gerado gráfico da Figura 17. É possível perceber nele que as notas *Ré*, *Fá#* e *Lá* são as que mais possuem energia ou, no ponto de vista de sugestão, as mais sugeridas. De certa forma um dos fatores que contribuiram das notas *Ré* e *Lá* ser de maiores energias foi devido a presença dos harmônicos.

Do resultado da terceira camada de processamento são gerados os gráficos da Figura 18. Essa camada é relativa ao resultados das sugestões de acordes musicais. É

perceptível ver a presença sugestão mais alta no acorde DM.

4.2.1 Experimento 2 - Acorde Dm

Nesse experimento foi tocado a tríade *Ré* (baixo e tônica), *Fá* e *Lá* equivalente ao acorde Dm. A tríade foi tocada ao mesmo tempo e com a mesma força para todas as notas.

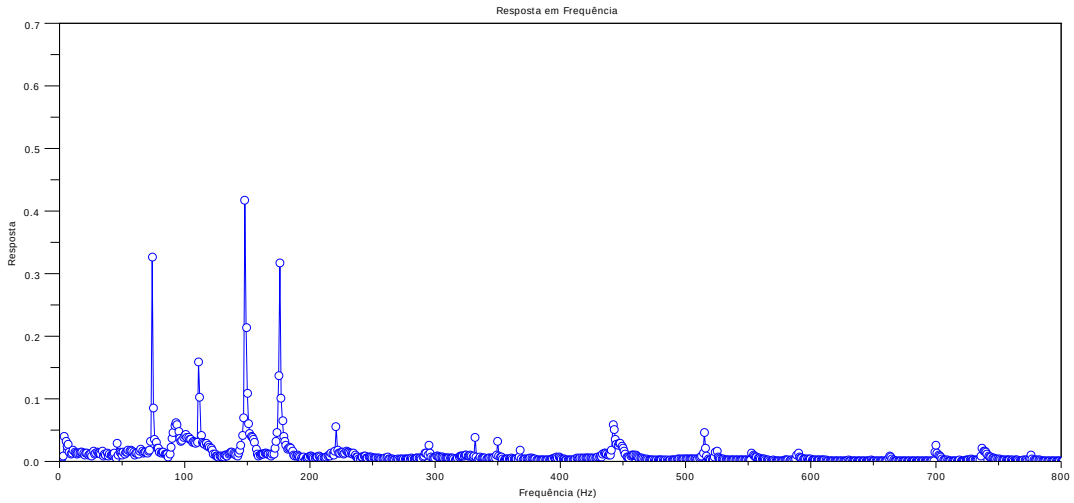

Figura 28: Gráfico da resposta em frequência para a gravação do acorde Dm.

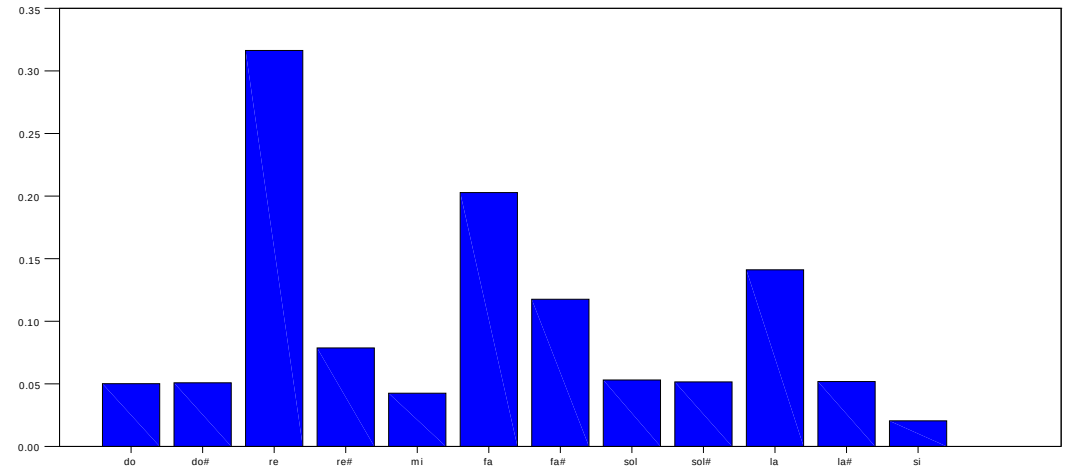

Figura 29: Gráfico de sugestão de notas para a gravação do acorde Dm.

Do resultado da primeira camada de processamento é gerado o gráfico da figura 28. Esse gráfico diz respeito a natureza da composição do sinal em senoides em termos

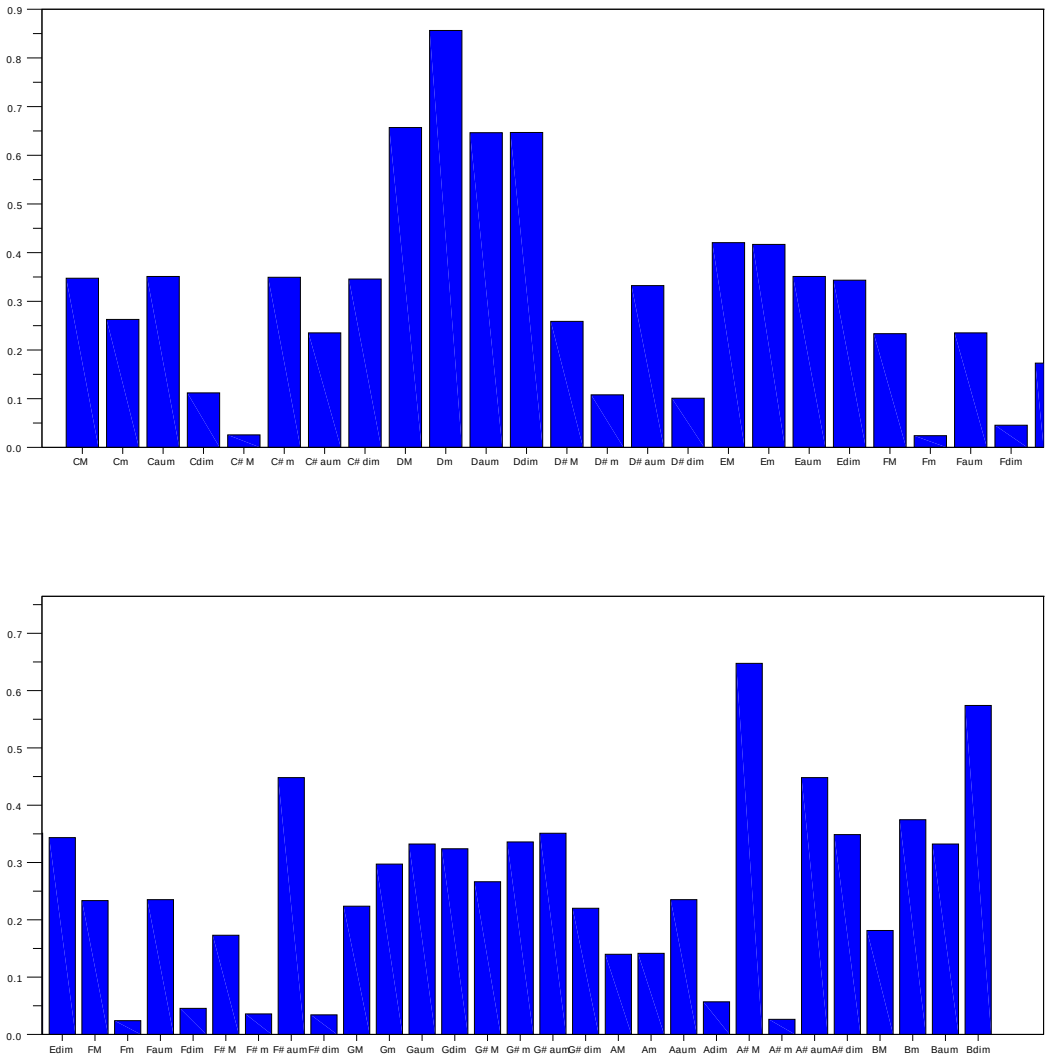

Figura 30: Gráficos de sugestão de acordes a gravação do acorde *Dm*.

de transformada de fourier. O primeiro pico, no valor de 294 Hz, é relativo a nota *Ré*. O segundo pico, no valor de 350 Hz, é relativo a nota *Fá*. O terceiro pico, no valor de 441 Hz, é relativo a nota *Lá*. Os picos seguintes são relativos aos harmônicos dessas três notas.

Do resultado da segunda camada de processamento é gerado gráfico da figura 29. É possível perceber nele que as notas *Ré*, *Fá* e *Lá* são as que mais possuem energia ou, no ponto de vista de sugestão, as mais sugeridas. De certa forma um dos fatores que contribuiram das notas *Ré* e *Lá* ser de maiores energias foi devido a presença dos harmônicos.

Do resultado da terceira camada de processamento são gerados os gráficos da figura 30. Essa camada é relativa ao resultados das sugestões de acordes musicais. É perceptível

ver a presença de altas sugestões nos acordes Dm, $A\#M$ e $Bdim$. A maior sugestão, ocasionando em resultado correto, foi no acorde Dm. As sugestões altas nos acordes $A\#M$ e $Bdim$ se deram devido à presença de duas notas de alta energia que formam esses acordes.

4.2.2 Experimento 3 - Acorde $Ddim$

Nesse experimento foi tocado a tríade $Ré$ (baixo e tônica), $Fá$ e $Sol\#$ equivalente ao acorde Ddim. A tríade foi tocada ao mesmo tempo e com a mesma força para todas as notas.

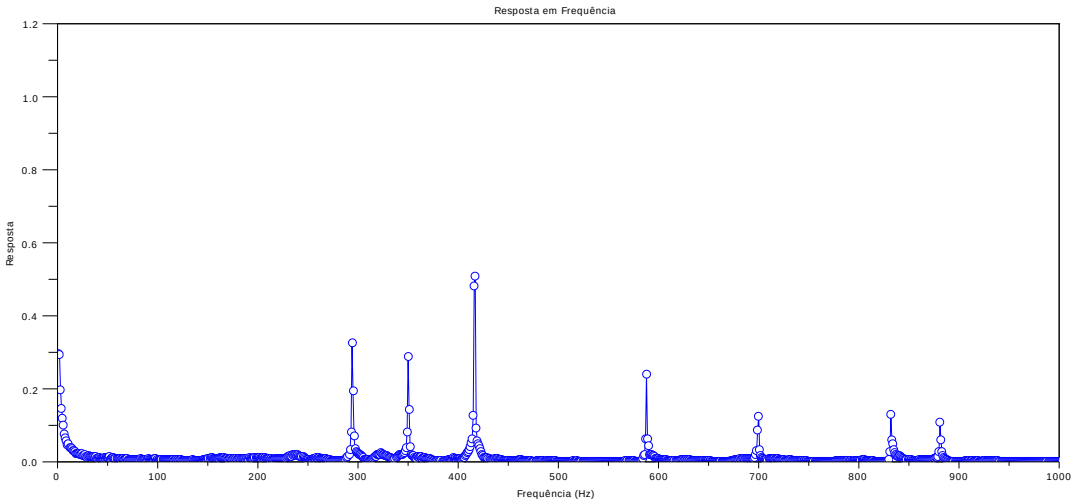

Figura 31: Gráfico da resposta em frequência para a gravação do acorde $Ddim$.

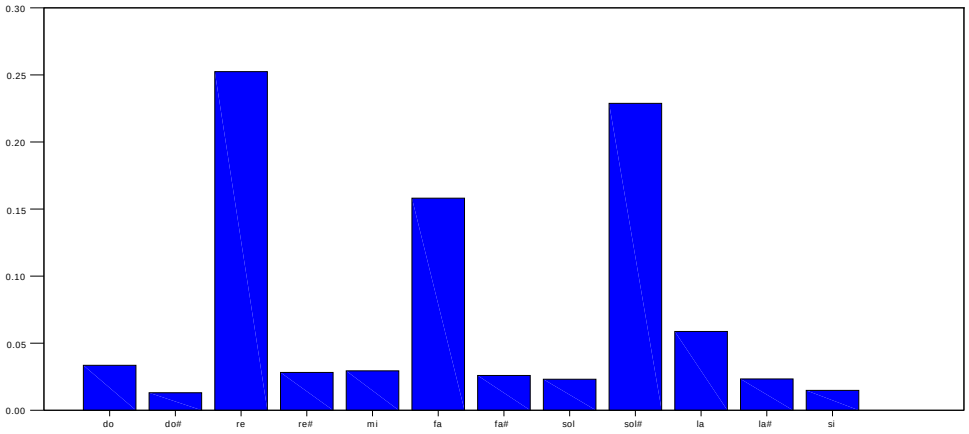

Figura 32: Gráfico de sugestão de notas para a gravação do acorde $Ddim$.

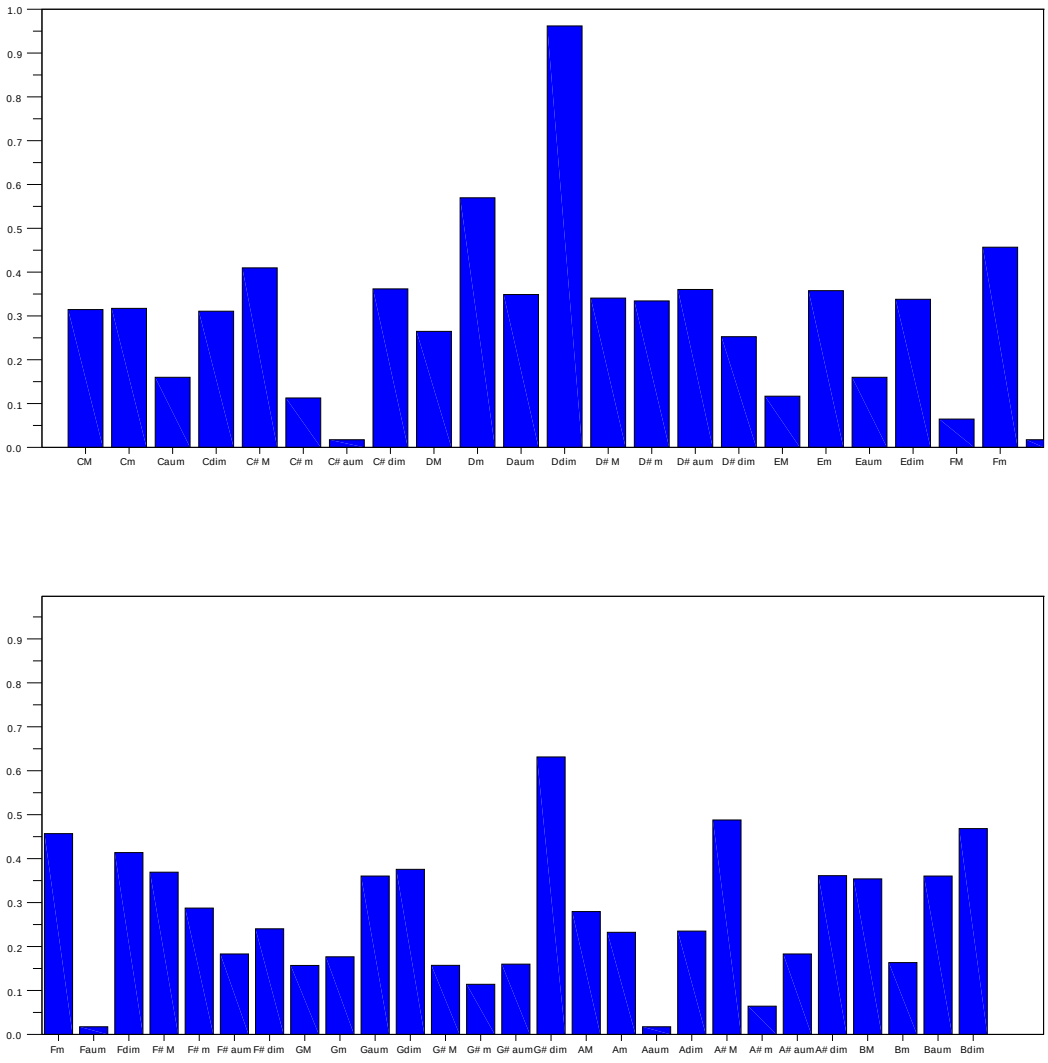

Figura 33: Gráficos de sugestão de acordes a gravação do acorde *Ddim*.

Do resultado da primeira camada de processamento é gerado o gráfico da figura 31. Esse gráfico diz respeito a natureza da composição do sinal em senoides em termos de transformada de fourier. O primeiro pico, no valor de 294 Hz, é relativo a nota *Ré*. O segundo pico, no valor de 350 Hz, é relativo a nota *Fá*. O terceiro pico, no valor de 417 Hz, é relativo a nota *Sol#*. Os picos seguintes são relativos aos harmônicos dessas três notas.

Do resultado da segunda camada de processamento é gerado gráfico da figura 32. É possível perceber nele que as notas *Ré*, *Fá* e *Sol#* são as que mais possuem energia ou, no ponto de vista de sugestão, as mais sugeridas.

Do resultado da terceira camada de processamento são gerados os gráficos da figura 33. Essa camada é relativa ao resultados das sugestões de acordes musicais. É perceptível

ver a presença de altas sugestões nos acordes *Ddim*, *Dm* e *G#dim*. A maior sugestão, ocasionando em resultado correto, foi no acorde *Ddim*. As sugestões altas nos acordes *Dm* e *G#dim* se deram devido à presença de duas notas de alta energia que formam esses acordes.

4.2.3 Experimento 4 - Acorde *Daum*

Nesse experimento foi tocado a tríade *R*é (baixo e tônica), *F*á# e *L*á# equivalente ao acorde *Daum*. A tríade foi tocada ao mesmo tempo e com a mesma força para todas as notas.

Segue os gráficos resultantes:

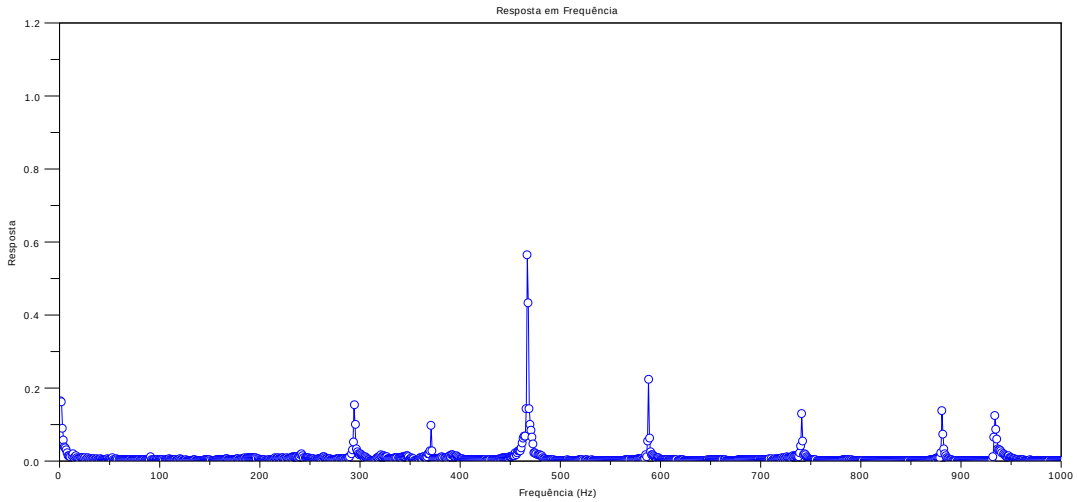

Figura 34: Gráfico da resposta em frequência para a gravação do acorde *Daum*.

Do resultado da primeira camada de processamento é gerado o gráfico da figura 34. Esse gráfico diz respeito a natureza da composição do sinal em senoides em termos de transformada de fourier. O primeiro pico, no valor de 294 Hz, é relativo a nota *R*é. O segundo pico, no valor de 371 Hz, é relativo a nota *F*á#. O terceiro pico, no valor de 467 Hz, é relativo a nota *L*á#. Os picos seguintes são relativos aos harmônicos dessas três notas.

Do resultado da segunda camada de processamento é gerado gráfico da figura 35. É possível perceber nele que as notas *R*é, *F*á# e *L*á# são as que mais possuem energia ou, no ponto de vista de sugestão, as mais sugeridas.

Do resultado da terceira camada de processamento são gerados os gráficos da figura 36. Essa camada é relativa aos resultados das sugestões de acordes musicais. É perceptível a alta sugestão dos acordes *Daum*, *F#aum* e *A#aum* com a mesma quantidade de

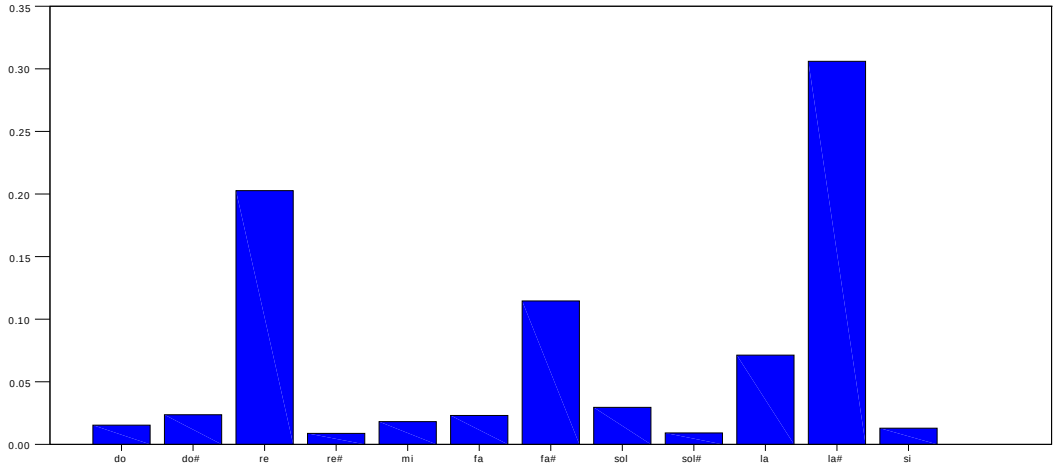

Figura 35: Gráfico de sugestão de notas para a gravação do acorde *Daum*.

energia. Isso é devido às notas comporem os mesmos acordes, diferenciando um do outro somente pela nota mais grave da tríade. Visto que o sistema possui o módulo de extração de baixos, a solução indicou que o acorde tocado foi *Daum*, visto que, como é mostrado no gráfico do espectro de frequências, a nota *Ré* é a nota mais grave da tríade.

A tabela 1 mostra os resultados do sistema dado todas as combinações dos conjuntos de acordes de tríades possíveis. Esses resultados foram gerados pelo script disponível no repositório desse trabalho [3].

[3] https://github.com/josepedro/TCC

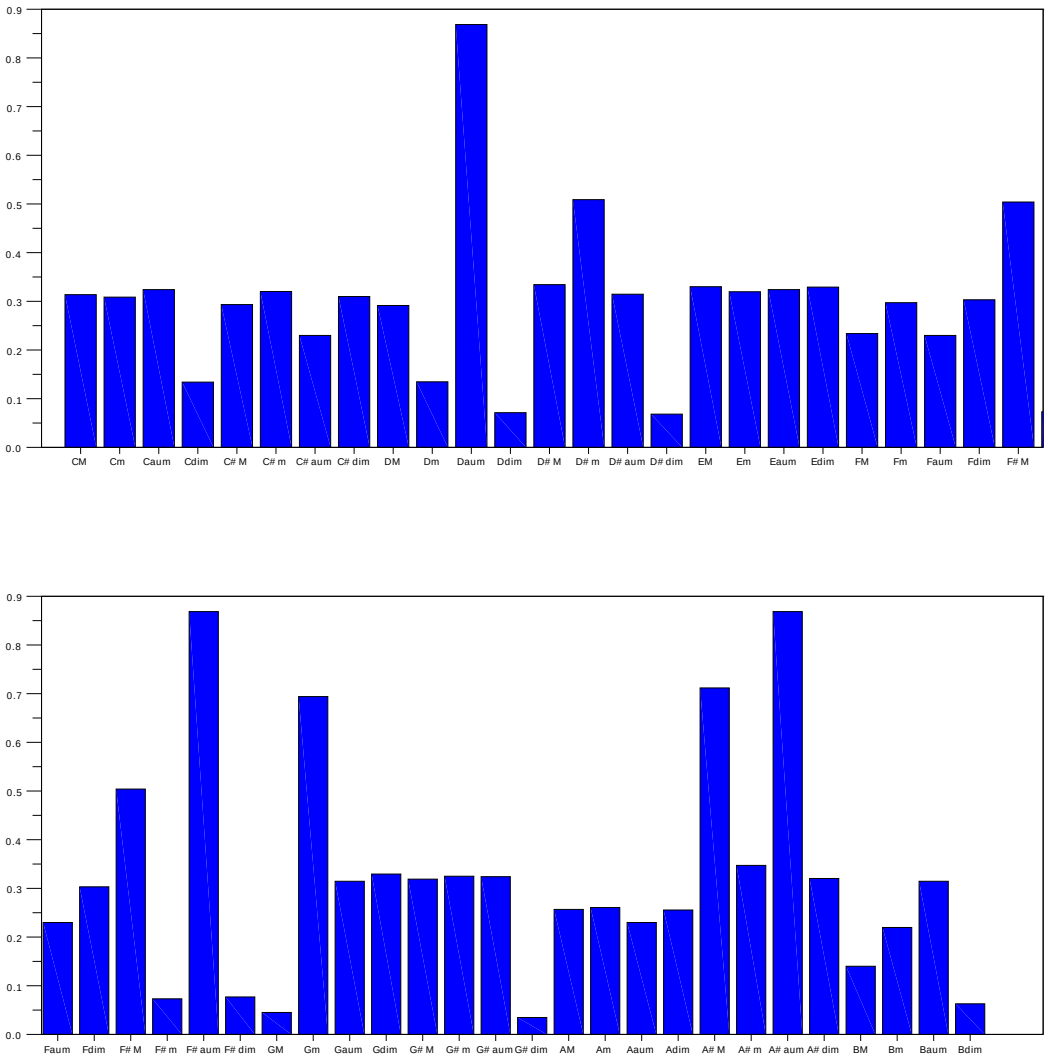

Figura 36: Gráficos de sugestão de acordes a gravação do acorde *Daum*.

Em vista do que foi apresentado de resultados em amostras separadas de acordes gravados (explicitados na tabela 1), o sistema reconhece, com 100% de acerto, todas as possibilidades numa tríade de notas - acordes maiores, menores, aumentados, diminutos e invertidos. Esses resultados são os mesmos que o primeiro trabalho na área (FUJISHIMA, 1999) chegou, porém, utilizando outro método de reconhecimento de acordes.

4.3 Detecção de Transições Rítmicas

Com o intuito de detectar acordes ao longo do tempo, foi pensado um algoritmo baseado em correlação de níveis de energia ao longo do tempo abordado no ciclo de desenvolvimento 3.3.6 que delimitaria, em teoria, número de ocorrências e instante de ocorrência de acordes. Com essa delimitação seria possível ajustar a janela de análise em

-	Acorde Fundamental	Quinta Invertida	Terça Invertida
C	C	C/G	C/E
Cm	Cm	Cm/G	Cm/D#
Caum	Caum	G#aum	Eaum
Cdim	Cdim	Cdim/F#	Cdim/D#
C#	C#	C#/G#	C#/F
C#m	C#m	C#m/G#	C#m/E
C#aum	C#aum	Aaum	Faum
C#dim	C#dim	C#dim/G	C#dim/E
D	D	D/A	D/F#
Dm	Dm	Dm/A	Dm/F
Daum	Daum	A#aum	F#aum
Ddim	Ddim	Ddim/G#	Ddim/F
D#	D#	D#/A#	D#/G
D#m	D#m	D#m/A#	D#m/F#
D#aum	D#aum	Baum	Gaum
D#dim	D#dim	D#dim/A	D#dim/F#
E	E	E/B	E/G#
Em	Em	Em/B	Em/G
Eaum	Eaum	Caum	G#aum
Edim	Edim	Edim/A#	Edim/G
FM	F	F/C	F/A
Fm	Fm	Fm/C	Fm/G#
Faum	Faum	C#aum	Aaum
Fdim	Fdim	Fdim/B	Fdim/G#
F#	F#	F#/C#	F#/A#
F#m	F#m	F#m/C#	F#m/A
F#aum	F#aum	Daum	A#aum
F#dim	F#dim	F#dim/C	F#dim/A
G	G	G/D	G/B
Gm	Gm	Gm/D	Gm/A#
Gaum	Gaum	D#aum	Baum
Gdim	Gdim	Gdim/C#	Gdim/A#
G#	G#	G#/D#	G#/C
G#m	G#m	G#m/D#	G#m/B
G#aum	G#aum	Eaum	Caum
G#dim	G#dim	G#dim/D	G#dim/B
A	A	A/E	A/C#
Am	Am	Am/E	Am/C
Aaum	Aaum	Faum	C#aum
Adim	Adim	Adim/D#	Adim/C
A#	A#	A#/F	A#/D
A#m	A#m	A#m/F	A#m/C#
A#aum	A#aum	F#aum	Daum
A#dim	A#dim	A#dim/E	A#dim/C#
B	B	B/F#	B/D#
Bm	Bm	Bm/F#	Bm/D
Baum	Baum	Gaum	D#aum
Bdim	Bdim	Bdim/F	Bdim/D

Tabela 1: Tabela de resultados dado os acordes tocados com inversões.

frequência nos limites energéticos. A figura 37 ilustra um exemplo de acordes tocados ao longo do tempo.

Na figura acima é perceptível observar, pelos níveis de energia, 8 acordes tocados. Para validar a solução, o gráfico de picos de transição rítmica deverá apresentar 8 picos ao longo do tempo, destacando, além do número de picos, a localidade coesa de cada pico.

A figura 38 mostra o gráfico de picos de transição rítmica e pode-se observar

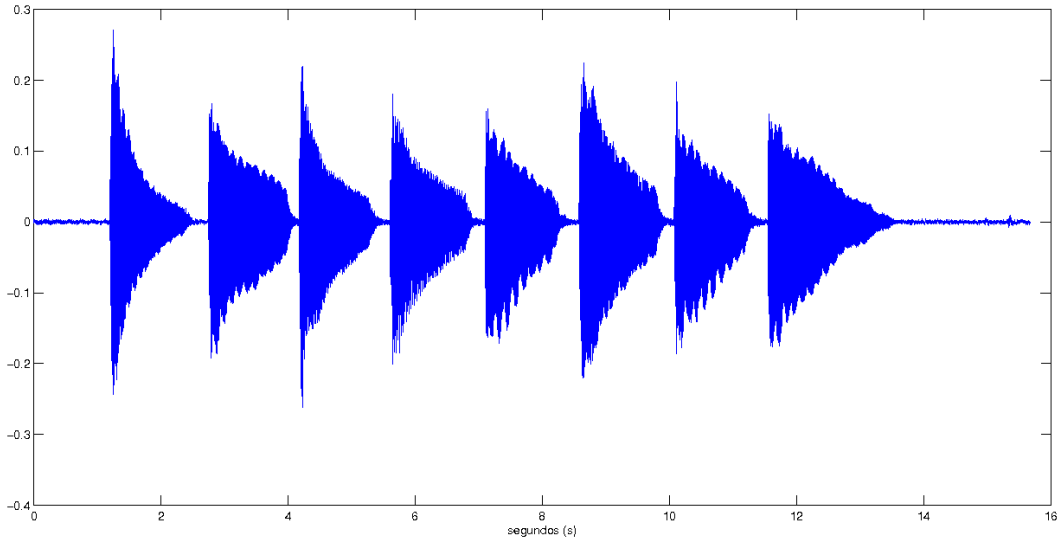

Figura 37: Acordes tocados ao longo do tempo.

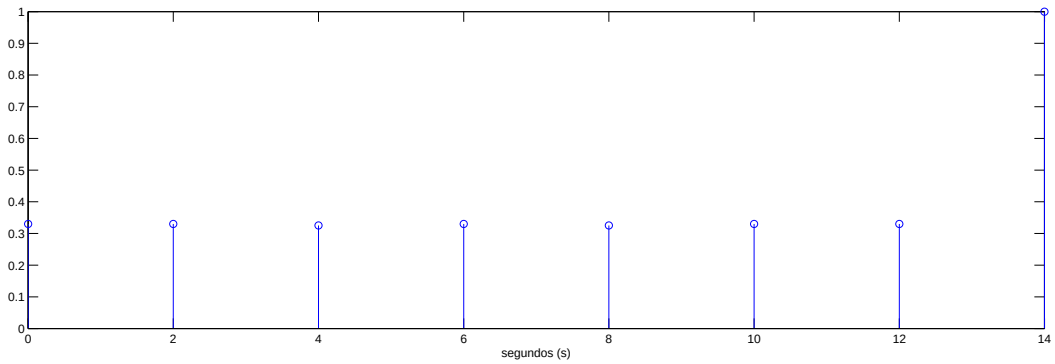

Figura 38: Gráfico de picos de transição rítmica.

dela que a quantidade de picos é coerente com a quantidade encontrada na figura 37, entretanto os instantes das localidades dos picos não estão coerentes. Um exemplo desse fato é o último acorde da figura 37 ser tocado as proximidades de 12 segundos e, na figura 38, o pulso do mesmo acorde estar localizado as proximidades de 14 segundos.

4.4 Implementação da Transformada Wavelets

Com o intuito de se localizar notas musicais ao longo do tempo, foi levantado uma hipótese do uso da transformada wavelets no ciclo de desenvolvimento 3.3.7. Esse tipo de técnica de processamento de sinais é de significativa importância de ser considerada pois a mesma possui definição em escala tempo e frequência. A escala tempo-frequência pode ser adaptada para que cada saída do banco de filtros (técnica para a implementação da transformada wavelets) pertencer a uma característica frequencial de uma nota musical.

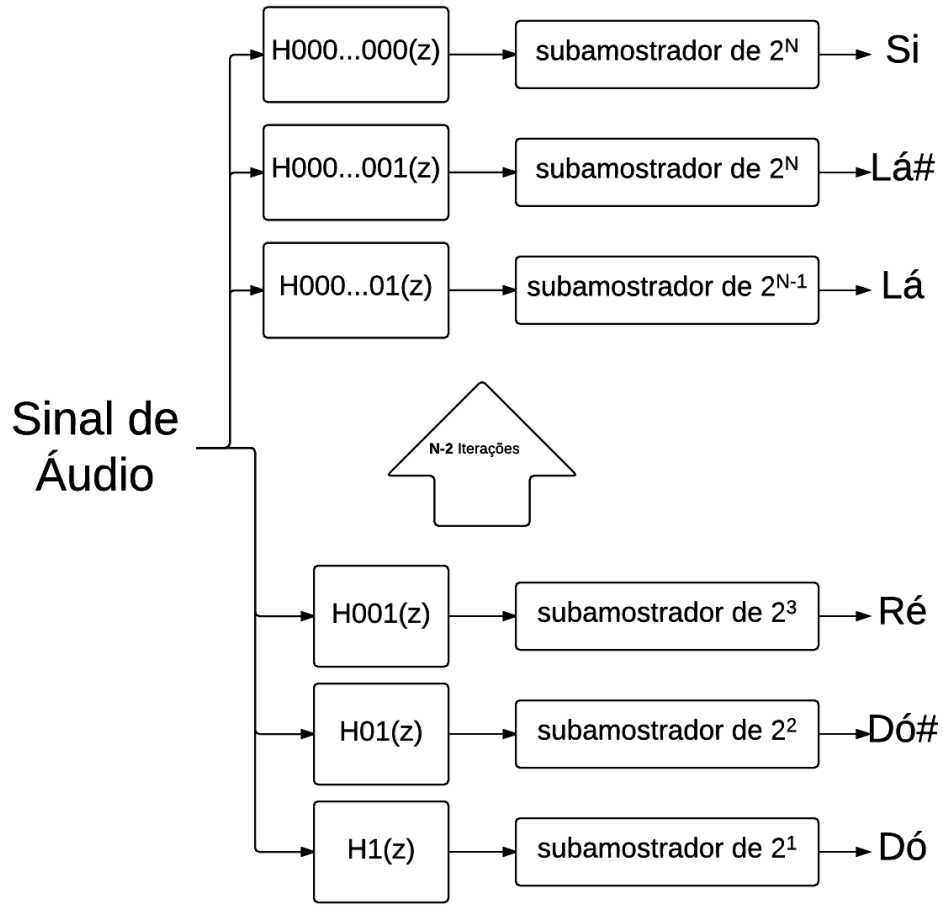

Figura 39: Hipótese ilustrativa do banco de filtros da transformada wavelets.

Tendo em vista a figura 39, a hipótese é usar as iterações do banco de filtros para localizar cada nota musical nas saídas dos mesmos. Para a validação dessa hipótese usou-se wavelets da família *daubechies* devido a alta correlação das mesmas com a natureza das ondas sonoras. Foi então verificado a viabilidade de se iterar os filtros até chegarem a resolução de uma nota musical (faixa restrita de frequências). A figura 40 mostra o gráfico da resposta em frequência de um sinal puro de 440 Hz, oriundo da saída **H001(z)**, com o banco de filtros iterado em 3 vezes.

De acordo com a figura 40 é visível que em 3 iterações no banco de filtro pode se identificar a frequência em integridade de 440 Hz pelo nível de energia máximo. Porém é preciso iterar mais vezes o banco de filtros para poder isolar somente uma nota para cada saída do mesmo. A figura 41 mostra o gráfico da resposta em frequência do mesmo sinal puro de 440 Hz, oriundo da saída **H000001(z)**, com o banco de filtros iterado em 10 vezes.

Como mostra a figura 41, o sinal de 440 Hz na entrada do banco de filtros foi defasado para 249.4 Hz. Esse fato não corrobora com o isolamento das frequências para cada saída do banco de filtros, pois, com o aumento da iteração abordado, o sinal sofreu

uma defasagem.

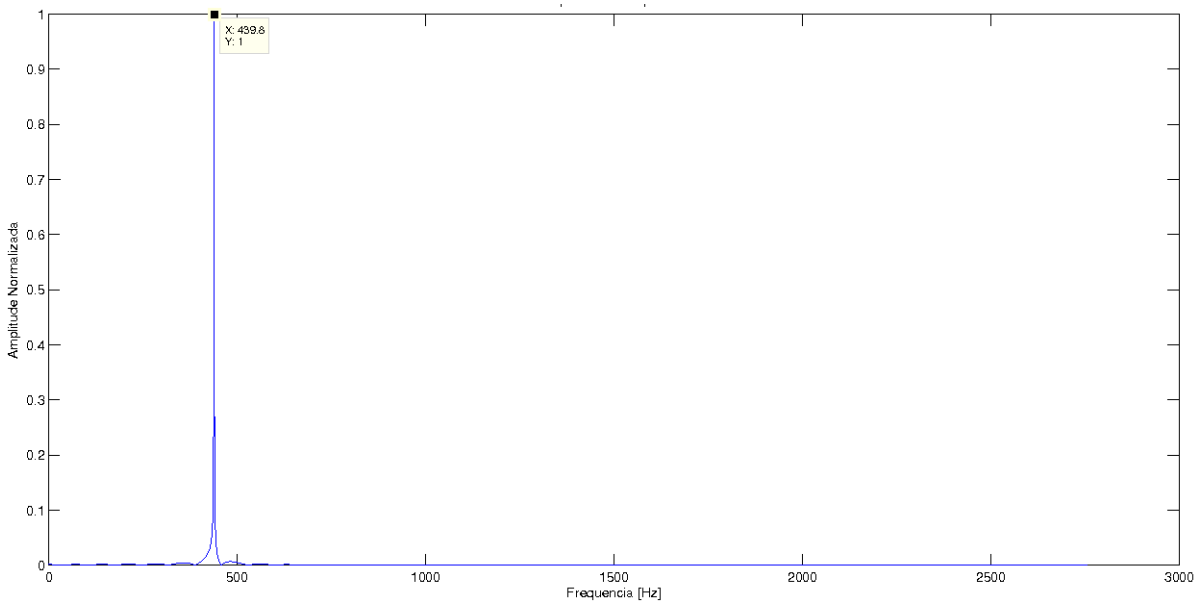

Figura 40: Espectro de sinal puro de 440 Hz em 3 iterações no banco de filtros.

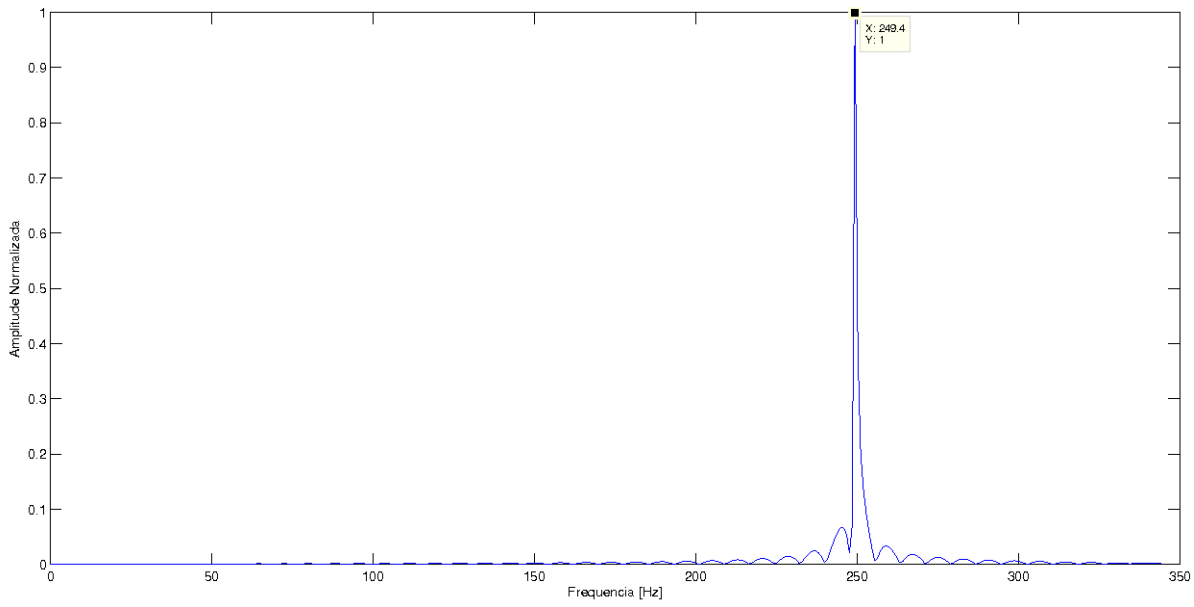

Figura 41: Espectro de sinal puro de 440 Hz em 10 iterações no banco de filtros.

4.5 Transcrição de Notas ao Longo do Tempo

Dado os módulos implementados nos procedimentos 3.2.2, 3.2.3, 3.2.4 e 3.2.5 e os resultados dos ciclos de desenvolvimento 3.3.2, 3.3.3, 3.3.4, 3.3.8 e 3.3.10, é consolidado os resultados da transcrição automática de notas ao longo do tempo. A figura 42 mostra o gráfico binário de transcrição de notas de uma escala cromática, gerada a partir de um áudio de tons puros variando de 260 Hz até 520 Hz.

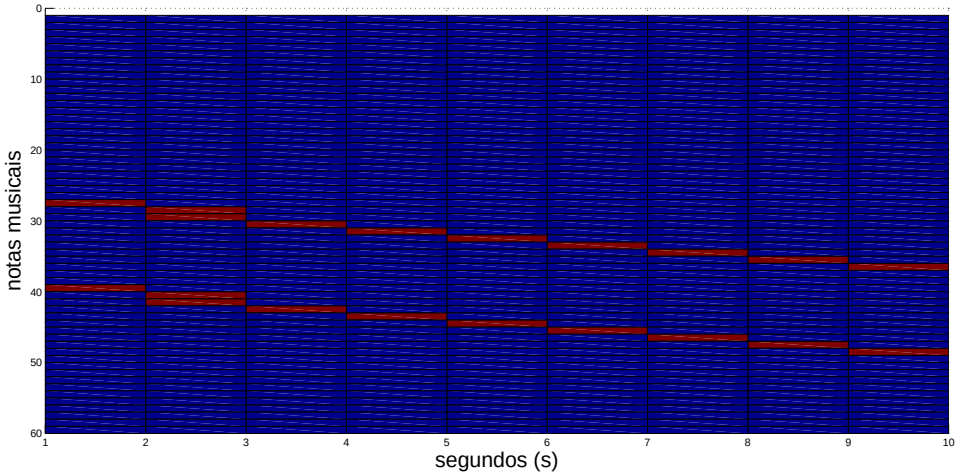

Figura 42: Transcrição de notas de um áudio de tons puros (260 Hz até 520 Hz).

A figura 42 mostrado uma escala cromática de notas musicais de duração de 1 segundo, explicitando os semitons em vermelho de cada nota. Não só a nota base foi reconhecida mas também o primeiro harmônico da mesma. Esse fato auxilia no reconhecimento de acordes visto que os harmônicos são prepoderantes e decivos para distinguir semitons.

No que diz respeito aos aspectos de reconhecimento de notas gravadas a partir de um instrumento real, violão e piano respectivamente, as figuras 43 e 44 mostram o gráfico binário de transcrição de notas de duração de 1 segundo de uma escala cromática no mesmo intervalo de frequências do caso anterior.

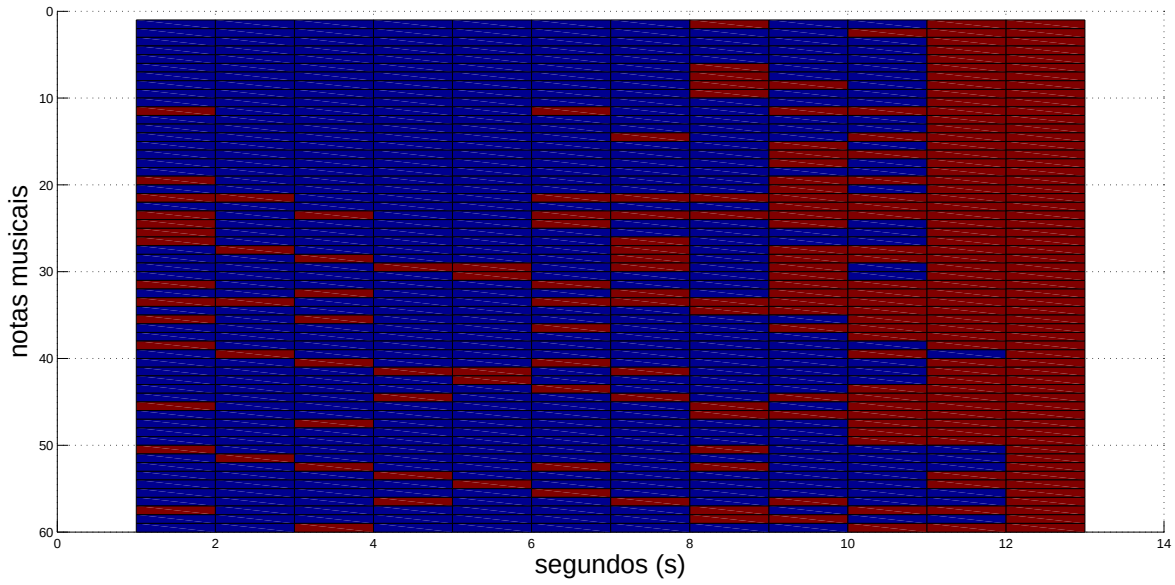

Figura 43: Transcrição de notas de um áudio de tons de violão (260 Hz até 520 Hz).

A figura 43 mostra uma escala cromática gradual de semitons de violão junto com notas de ruídos de fundo. Cada nota e cada ruído de fundo gera um harmônico ocasionando em notas que não estão presentes efetivamente no áudio.

A figura 43 mostra, com mais clareza em relação ao violão, uma escala gradual de semitons de piano. Cada nota gerou também seus respectivos harmônicos.

Mesmo nos contextos de erros de ruídos de fundos mostrados nas figuras anteriores, a rede neural minimiza-os através de padrões de tríades oriundos da teoria musical de acordes. Essa mapa de notas ao longo do tempo

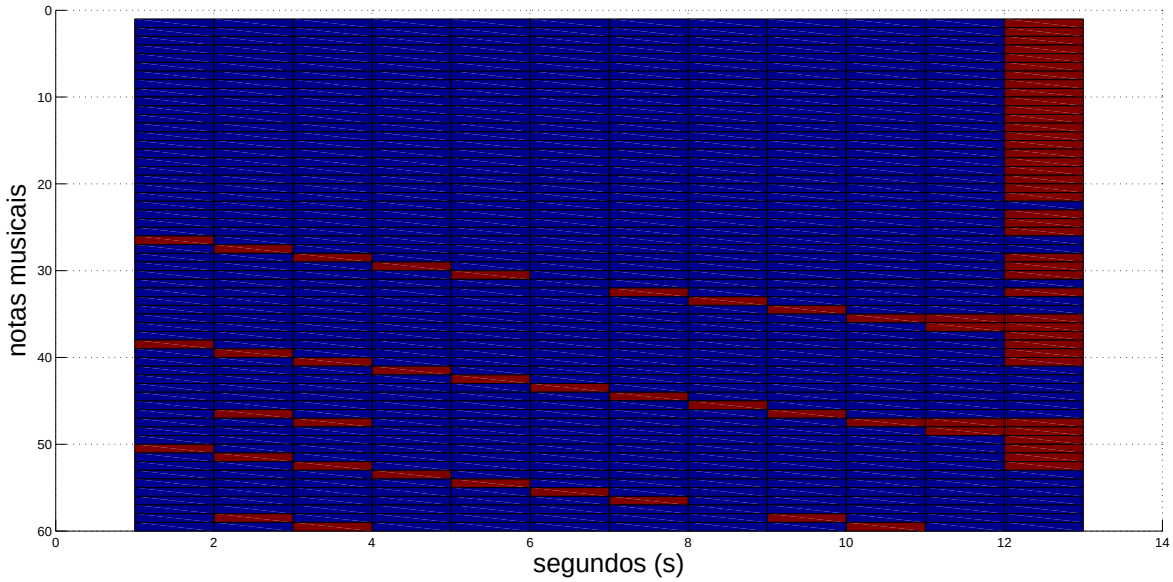

Figura 44: Transcrição de notas de um áudio de tons de piano (260 Hz até 520 Hz).

4.6 Transcrição Automática de Acordes ao Longo do Tempo

Tendo como referência os módulos implementados nos procedimentos 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.2.8, 3.2.9 e 3.2.10 e os resultados dos ciclos de desenvolvimento 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.8, 3.3.10, 3.3.10, 3.3.11, 3.3.12 e 3.3.13, foram consolidados os resultados da transcrição automática de acordes ao longo do tempo. Dois experimentos foram feitos testando a sequência de acordes tocados no piano e no violão.

A figura 45 mostra o sinal de áudio do piano e contém 10 acordes tocados ao longo de aproximadamente 20 sengudos e cada acorde foi executado aproximadamente durante 2 segundos. A tabela 2 mostra o momento, acordes tocados e os acordes reconhecidos pelo sistema.

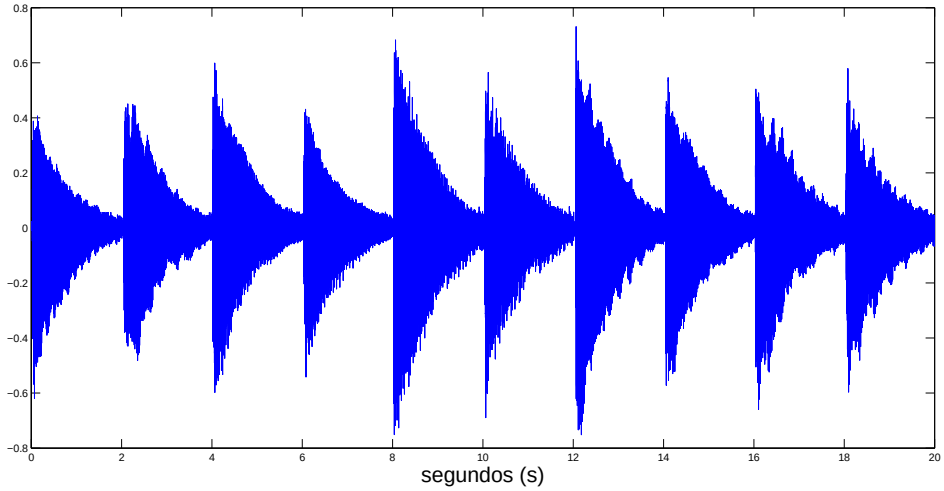

Figura 45: Gráfico do sinal de áudio do piano.

Tempo (segundos)	Acorde Tocado	Acorde Reconhecido
1	C	C
2	C	C/G (errado)
3	C/G	C/G
4	C/G	C/G
5	Am	Am
6	Am	Am
7	Am/C	Am/C
8	Am/C	Em (errado)
9	Em	Em
10	Em	Em
11	Em/B	Em/G (errado)
12	Em/B	Em (errado)
13	F	F
14	F	F
15	F/A	F/A
16	F/A	F (errado)
17	G	G
18	G	G
19	G/B	G/B
20	G/B	G/B

Tabela 2: Tabela de acordes tocados e acordes reconhecidos no piano.

Da tabela 2 pode-se ver os acordes que a solução computacional errou totalizando em 5 acordes. Dentre 20 acordes tocados em sequência o sistema acertou 15, ocasionando em 75% de acertos.

Também foi feito o mesmo experimento com o sinal de áudio de violão, sendo representado pela figura 46. Como foi feito também com o piano, são no total 10 acordes, cada um com duração de 2 segundos totalizando num sinal de 20 segundos aproximadamente.

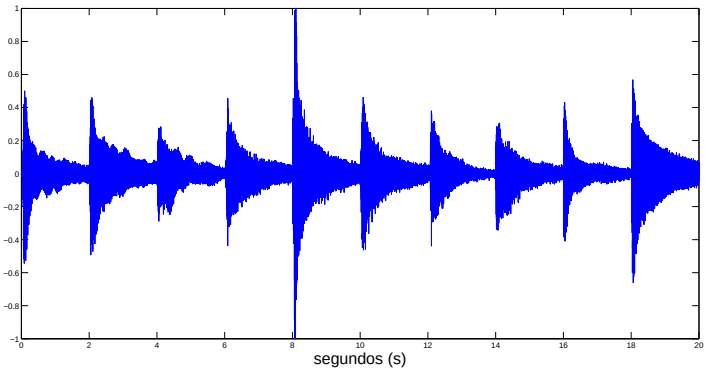

Figura 46: Gráfico do sinal de áudio do violão.

Também é evidenciado na tabela 3 os acordes tocados e os acordes reconhecidos no violão. O sistema reconheceu incorretamente 10 acordes de 20, totalizando 50% de acertos no violão.

Tempo (segundos)	Acorde Tocado	Acorde Reconhecido
1	C	C
2	C	C
3	C/G	C/G
4	C/G	C/G
5	Am	A (errado)
6	Am	C (errado)
7	Am/C	C (errado)
8	Am/C	C (errado)
9	Em	Em/B (errado)
10	Em	E/B (errado)
11	Em/B	Em/B
12	Em/B	E/B (errado)
13	F	F
14	F	F/A (errado)
15	F/A	Aaum (errado)
16	F/A	G (errado)
17	G	G
18	G	G
19	G/B	G/B
20	G/B	G/B

Tabela 3: Tabela de acordes tocados e acordes reconhecidos no violão.

Dado as análises apresentadas nas amostras de áudio, o sistema obteve 75% de acertos para acordes de piano e 50% para acordes de violão, ambos tocados ao longo do tempo. Os resultados de reconhecimento de acordes foram abaixos dos expostos em amostras separadas de acordes apresentados em 4.1.1 devido a dificuldade, parcialmente mitigada no processo segmentaçao de áudio apresentado em 3.2.1, em detectar a ocorrência de acordes ao longo do tempo. Músicas completas não foram testadas pois o presente trabalho se limitou somente no reconhecimento de acordes ao longo do tempo em contextos polifônicos e mono-instrumentais. É plausível de afirmação que o resultado foi comprome-

tido pelo fato do sistema não ser constituído de um módulo de detecção de acordes não só ao longo do tempo, mas, dado um conjunto de possibilidades harmônicas, resultados em que o acorde se comportaria como totalmente não harmônico, ou seja, ruído ou silêncio.

4.7 Extração da Tonalidade

A extração da tonalidade das músicas foi consolidada de acordo com os procedimentos 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.2.6 e 3.2.7 e ciclos de desenvolvimento 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.9 e 3.3.10. Para validar a extração do tom da música foi feito 6 experimentos com 4 músicas completas e diferentes e 2 sequências de acordes iguais porém com instrumentos diferentes (sequências essas oriundas do caso abordado de extração de acordes ao longo do tempo). A tabela 4 mostra o resultados dos experimentos em relação ao tom de fato da música[4] e tom reconhecimento pelo sistema.

Musica	Tom da Música	Tom Reconhecido
Sequência de Acordes no Piano	C	C
Sequência de Acordes no Violão	C	Em (errado)
The Beatles - Oh Darling	A	A
Deep Purple - Smoke on the Water	Gm	G (errado)
Legião Urbana - Eduardo e Mônica	E	E
The Beatles - Yesterday	F	F

Tabela 4: Tabela de extração de tonalidade das músicas.

Diante do que é mostrado na tabela 4, o sistema-solução errou 2 músicas de 6 músicas no total, ocasionando em 67% de acertos globais. Esse resultado é razoável pois o sistema-solução conseguiu, além de identificar tonalidade de música mono-instrumental, identificar tonalidades de músicas multi-instrumentais. Devido a dependência dessa funcionalidade com o módulo de reconhecimento de acordes ao longo do tempo, o resultado poderia melhorar utilizando aprimorações por regras da teoria musical.

4.8 Exemplo de Uso

Um exemplo de uso, apresentado também no arquivo README do repositório *github*[5], pode ser executado a partir do *script* **example.m** presente na pasta **example** e o resultado pode ser conferido na figura 47.

[4] Os tons das músicas que não foram tocadas manualmente foram tirados do site http://www.cifraclub.com.br/

[5] https://github.com/josepedro/recognizer_musical_harmonies

```
>> cd ..
>> cd recognizer_musical_harmonies/
>> cd example/
>> example
Warning: WAVREAD will be removed in a future release. Use AUDIOREAD instead.
> In wavread at 62
  In main at 7
  In example at 6
Chords over time:
  Columns 1 through 14

    'C'    'C/G'    'C/G'    'C/G'    'Am'    'Am'    'Am/C'    'Em'    'Em'    'Em'    'Em/G'    'Em'    'F'    'F'

  Columns 15 through 20

    'F/A'    'F'    'G'    'G'    'G/B'    'G/B'

Chord key:
C
>>
```

Figura 47: Exemplo de uso.

Esse capítulo teve como objetivo expor os resultados obtidos através do desenvolvimento do protótipo de uma solução computacional para reconhecimento de harmonias musicais. Os resultados bem sucedidos, em conjunto, formam a funcionalidade do sistema como um todo.

O primeiro resultado foi a resposta em frequência e sugestões de acordes musicais debatido na seção 4.1. Esses resultados mostram que, através do espectro de frequência calculado via transformada de fourier, as notas e os acordes estão sendo corretamente reconhecidos em amostras de áudio separadas para cada acorde. Esse fato resultou em 100% de acertos para acordes maiores, menores, aumentados, diminutos e invertidos.

O segundo resultado, debatido na seção 4.3, foi a detecção de transições rítmicas. O resultado deu insatisfatório pois a hipótese não mostra a localização no tempo dos picos coerentemente com os ataques dos acordes tocados. Pode-se inferir que essa hipótese pode ser usada para mensurar números de acordes numa dada janela de tempo. Tal resultado não foi implementado no sistema-solução.

O terceiro resultado, debatido na seção 4.4, foi a implementação da transformada wavelets para a visualização das bandas de frequência ao longo do tempo. Para que as saídas dos filtros sejam cada nota musical é preciso de iterar os mesmos. Foi constatado que em 10 iterações o banco de filtros distorce o sinal, inviabilizando a implementação dessa técnica no sistema-solução.

O quarto resultado, debatido na seção 4.5, foi a transcrição de notas ao longo do tempo. A transcrição de notas foi satisfatória visto que para tons puros reconhce as notas ao longo do tempo. Para sons instrumentais como de violão e piano, mesmo que parcialmente, a hipótese implementada também reconhece notas. Tal resultado foi implementado no sistema.

O quinto resultado, debatido na seção 4.6, foi a transcrição de acordes ao longo do tempo. A transcrição de acordes foi parcialmente satisfatória visto que, para acordes de violão e piano, obteve 50% e 75% respectivamente. Tal resultado foi implementado no

sistema.

O sexto resultado, debatido na seção 4.7, foi a extração de tonalidade da música. Tal processo obteve 67% de acertos globais, demonstrando que foi satisfatório o resultado pois também foram submetidos para análise, no sistema-solução, músicas polifônicas e multi-instrumentais. Tal resultado foi implementado no sistema-solução.

Também há o resultado do exemplo de uso na seção 4.8, tal resultado demonstra a execução do *script* de exemplo no repositório e as informações de saída, acordes ao longo do tempo com precisão de 1 segundo e tonalidade da música.

Há de considerar como resultado o método de avaliação da solução computacional. Esse método se embasa nas entradas submetidas e as saídas. Dessa forma uma verificação de acertos e erros é realizada a fim de estimar uma porcentagem resultados satisfatórios do protótipo computacional. A forma mais adequada, na comunidade científica, de avaliar uma solução de computação musical é o MIREX[6]. *Music Information Retrieval Evaluation eXchange* (MIREX) é um processo de avaliação de algorítmos e soluções computacionais para computação musical. Tal processo é aberto todo ano com datas pré-definidas para cada tema (como por exemplo *Audio Fingerprinting*, *Audio Chord Estimation* e *Audio Melody Extraction*). Esse processo é realizado pelo laboratório *Music Information Retrieval Systems Evaluation Laboratory* (IMIRSEL) e, após avaliação da solução, a mesma pode ser comparada com as outras de forma objetiva. Esse trabalho não foi possível de ser avaliado devido ao prazo do processo de avaliação.

De 6 resultados, 1 foi totalmente sucedido, 2 não foram sucedidos e 3 foram parcialmente sucedidos. Os resultados sucedidos foram implementados no protótipo sistema-solução e delimitaram funcionalidades e limitações que serão discutidas no capítulo seguinte.

[6] http://www.music-ir.org/mirex/wiki/MIREX_HOME

5 Conclusões

Desde os primórdios da humanidade, a música vem se desenvolvendo de forma intensa, acoplando vários elementos da cultura corrente aos seus processos de audição, estruturação, composição e execução. Um tipo específico desses elementos são as tecnologias computacionais que, ao interferirem nos processos musicais, otimizam e catalizam as várias formas de se interagir com a música. O produto desse trabalho, então, é um protótipo de uma tecnologia computacional para reconhecimento e extração de informações harmônicas, objetivando a automação da percepção musical.

No ponto de vista da automação do reconhecimento e extração de harmonias musicais, o músico muitas vezes não tem acesso a partituras ou cifras para executar músicas e esse fato se intensifica bastante quando é um estudante iniciante que, muitas vezes, necessita de um auxiliador técnico para orientação. Nesse trabalho foram elaborados e implementados os processos computacionais de reconhecimento de acordes musicais e extração de tonalidades musicais, informações harmônicas essas que são de grande valia para amenizar os problemas citados.

Para a consolidação do protótipo do sistema-solução, objetivou-se desenvolver soluções computacionais para reconhecimento de acordes e suas inversões, reconhecimento de acordes ao longo do tempo e extração do tom da música. O primeiro objetivo foi totalmente alcançado visto que o sistema reconheceu, em amostras separadas de acordes gravados, 100% das possibilidades de tríades (maiores, menores, aumentadas, diminutas e invertidas). O segundo objetivo foi parcialmente alcançado visto que o sistema reconheceu a sequência de acordes num áudio completo tanto para piano com 75% de acertos, tanto para violão com 50% de acertos. O terceiro objetivo foi parcialmente alcançado visto que a solução reconheceu 67% corretamente o tom de todas as amostras de músicas corretamente, porém vale ressaltar que músicas completas, polifônicas e multi-instrumentais formaram parte das amostras testadas e esse fato significa um grande ganho do protótipo sistema-solução visto que até então os objetivos anteriores se limitaram somente a músicas mono-instrumentais. Esse fato corrobora na afirmação de que o sistema-solução pode obter resultados coerentes em situações mais complexas na extração de tonalidade.

Também é passível de consideração as implementações não satisfatórias para o contexto do problema e melhoria do protótipo sistema-solução. A primeira implementação é a detecção de transições rítmicas, que não foi satisfatória para localizar a ocorrência de acordes ao longo do tempo, devido aos picos de detecção estarem localizados em lugares incoerentes em relação aos locais de execução dos acordes. Pode-se concluir que a detecção de transições rítmicas somente é satisfatória para extrair a quantidade de acordes numa

determinada janela de tempo. A segunda implementação é a transformada wavelets que não foi adequada para filtrar frequências específicas de notas musicais devido às distorções nos sinais de saída ocasionadas pelas iterações no banco de filtros.

Esse trabalho atingiu resultados congruentes no que é apresentado no primeiro trabalho da área de computação musical sobre o assunto tratado (FUJISHIMA, 1999). Todavia, além de atingir resultados semelhantes no que diz respeito a reconhecimento de acordes em amostras gravadas separadamente, o sistema-solução desenvolvido provê parcialmente o reconhecimento de acordes ao longo do tempo num áudio completo e extração de tonalidade em músicas completas, inclusive, multi-instrumentais.

5.1 Ameaças e Limitações

Em vista das ameaças e limitações do trabalho, a análise exposta para o reconhecimento de acordes em amostras gravadas separadamente foi feita somente para o piano, essa análise não foi feita para outros instrumentos musicais. Espera-se que, para outros instrumentos musicais, possa ter alguns resultados divergentes, requerindo novas implementações no sistema-solução a fim de atingir a adaptação a outros instrumentos musicais.

Há ameaças e limitações também no reconhecimento de acordes ao longo do tempo. Na análise feita, o mesmo não atingiu 100% de acertos pois a implementação de segmentação de áudio não detecta o acorde ao longo do tempo, ela somente sugere uma estimativa de um acorde numa determinada janela de tempo sem considerar a presença ou não de um acorde, ou seja, ruídos de fundo são caracterizados como acordes.

No que diz respeito a extração de tonalidade musical, foi mostrado resultados insatisfatórios tanto para músicas completas multi-instrumentais como para músicas mono-instrumentais (caso do violão). Compreende-se de que ruídos de fundo ocasionam interferências no espectro de frequência, resultando em informações incoerentes para a interpretação da rede neural.

Também há de se questionar as metodologias de avaliação e análise do sistema-solução como um todo. O método quantitativo foi baseado em acertos e erros nos resultados finais do protótipo numa base de músicas relativamente pequena.

5.2 Desdobramentos

Ao decorrer desse trabalho, especificamente ao tratar de uma solução matemática para a substituição da transformada de fourier janelada através da hipótese das transformadas wavelets (seção 4.4), houve um desdobramento no trabalho. Como o uso das

transformadas wavelets, nesse contexto, não foi satisfatório, foi investigado uma nova adaptação dessa transformada para o problema de identificação de notas musicais.

No que diz respeito aos fundamentos da álgebra linear, a transformada de fourier e as demais, inclusive wavelets, possui uma função primordial que é projetar em bases a informação a ser processada e tratada. A transformada de fourier e wavelets projetam os sinais em bases ortonormais fazendo com que o produto final dessa operação seja coeficientes energéticos distribuídos ao longo de cada componente da base. Na transformada wavelets a operação que faz com que o sinal seja projetado em bases wavelets é a convolução. Dado que projetar o sinal em bases de famílias wavelets não proporcionou resultados satisfatórios, pensou-se na projeção do sinal em bases de notas musicais, ou seja, projetar o sinal, através da operação de convolução, em senos frequências bem definidas somados com seus respectivos harmônicos. Ao final, para analisar se determinadas notas estão presentes no sinal, é plausível de se calcular a energia do sinal para cada nota. Ao final do processo um conjunto de energias de notas serão calculadas, explicitando assim, quais notas foram realmente tocadas.

Há resultados empíricos satisfatórios dessa estratégia de projeção de notas musicais mas tal método carece de uma metodologia rigorosa através de prova matemática e comparações com métodos do estado da arte da computação musical. Esse desdobramento foi submetido ao congresso ISMIR 2015[1] e, por causa desses pontos citados, não foi aceito e ainda está sendo trabalhado.

5.3 Trabalhos Futuros

Há de se considerar, em trabalhos futuros, análises e testes do protótipo da solução desenvolvida com outros instrumentos musicais. Tais sugestões podem revelar novas hipóteses a serem discutidas e implementadas a fim de melhorar o sistema-solução e simplificar alguns processos.

Também deve-se revisar, discutir e reimplementar o procedimento de segmentação de áudio, pois o mesmo possui falhas não detecção de acordes no momento em que ocorre. Um método mais adequado na segmentação de áudio poderá oferecer melhoridas substanciais no reconhecimento de acordes ao longo do tempo. Para auxiliar essa implementação é plausível de se considerar a inserção de conhecimento na rede neural em casos que não é acorde, fazendo com que o classificador tenha a capacidade de apontar ruídos em geral.

Outra característica que poderia ser implementada e atribuída ao protótipo sistema-solução é um módulo de pré-processamento de áudio, objetivando a amenização de ruídos mesmo contendo cortes em alguns harmônicos do instrumento tocado. Quando mais o áudio se aproximar a tons puros, melhor o sistema-solução irá classificar o áudio.

[1] http://ismir2015.uma.es/

Em relação às metodologias de avaliação e análise do sistema-solução como um todo, sugere-se o uso das ferramentas disponíveis em MIREX[2]. Essas ferramentas oferecem um suporte mais sistemático e criterioso na avaliação de sistemas de recuperação musical no geral. As soluções de computação musical de estado da arte são avalidas por esse modo e, fazendo o uso abrangente, pode-se comparar com mais fidedignidade esse e mais outros trabalhos no campo da computação musical.

Como trabalho futuro também há a necessidade de se implementar o projeto proposto, em Python, do protótipo de solução computacional apresentado nesse trabalho. A implementação desse trabalho num projeto de solução de engenharia de software oferecerá ambiente propício para boa manutenibilidade e evolução do sistema. Vários contribuidores poderão trabalhar no sistema visto que há poucas soluções de código aberto sobre as funcionalidades de extração de acordes e tonalidades musicais.

[2] http://www.music-ir.org/mirex/wiki/MIREX_HOME

Referências

ANTARES. *Auto-Tune 7*. 2014. Disponível em: <http://www.antarestech.com/products/detail.php?product=Auto-Tune_7_1>. Citado na página 25.

BOULANGER. Audio chord recognition with recurrent neural networks.http://ismir.net/. In: *ISMIR*. [S.l.: s.n.], 2013. p. 335–340. Citado na página 26.

CABRAL. Automatic x traditional descriptor extraction: The case of chord recognition. In: *Proceedings of the 6th international conference on music information retrieval (ISMIR'2005), London*. [S.l.: s.n.], 2005. Citado na página 47.

CHEN. Chord recognition using duration-explicit hidden markov models. In: CITESEER. *ISMIR*. [S.l.], 2012. p. 445–450. Citado na página 26.

CHO. Exploring common variations in state of the art chord recognition systems. In: CITESEER. *Proceedings of the Sound and Music Computing Conference (SMC)*. [S.l.], 2010. p. 1–8. Citado na página 26.

COHORTOR.ORG. *Tuner-gStrings Free*. 2014. Disponível em: <https://play.google.com/store/apps/details?id=org.cohortor.gstrings>. Citado na página 25.

CRUZ, F. W. *Necessidades de informação musical de usuários não especializados*. Tese (Doutorado) — UNIVERSIDADE DE BRASÍLIA, 2008. Citado na página 25.

DOZOL. 2014. Disponível em: <http://www.adrianodozol.blogspot.com.br>. Citado 2 vezes nas páginas 15 e 76.

DRUYVESTEYN. *Coder for incorporating an auxiliary information signal in a digital audio signal, decoder for recovering such signals from the combined signal, and record carrier having such combined signal recorded thereon*. [S.l.]: Google Patents, 1992. US Patent 5,161,210. Citado na página 35.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. *Information and software technology*, Elsevier, v. 50, n. 9, p. 833–859, 2008. Citado na página 41.

FUJISHIMA, T. Realtime chord recognition of musical sound: A system using common lisp music. In: *Proc. ICMC*. [S.l.: s.n.], 1999. v. 1999, p. 464–467. Citado 3 vezes nas páginas 26, 85 e 100.

HAAS, B. de; MAGALHÃES, J. P.; WIERING, F. Improving audio chord transcription by exploiting harmonic and metric knowledge. In: CITESEER. *ISMIR*. [S.l.], 2012. p. 295–300. Citado na página 26.

HARTE, C. *Towards automatic extraction of harmony information from music signals*. Tese (Doutorado) — Department of Electronic Engineering, Queen Mary, University of London, 2010. Citado na página 26.

JURGEN. *Jurgen*. 2014. Disponível em: <http://www.wi.hs-wismar.de/~cleve/>. Citado 2 vezes nas páginas 15 e 39.

KHADKEVICH, M.; OMOLOGO, M. Time-frequency reassigned features for automatic chord recognition. In: IEEE. *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. [S.l.], 2011. p. 181–184. Citado na página 26.

KLAPURI, A. P. Automatic music transcription as we know it today. *Journal of New Music Research*, Taylor & Francis, v. 33, n. 3, p. 269–282, 2004. Citado na página 26.

LABGARAGEM. 2014. Disponível em: <http://labdegaragem.com/profiles/blogs/hino-de-times-de-futebol-de-sao-paulo-tocados-pelo-arduino>. Citado 2 vezes nas páginas 15 e 33.

LEE, K. Automatic chord recognition from audio using enhanced pitch class profile. In: *Proc. of the International Computer Music Conference*. [S.l.: s.n.], 2006. Citado na página 26.

MED, B. Teoria da música. 4ª edição revista e ampliada. *Brasília-DF, Musimed*, 1996. Citado 5 vezes nas páginas 30, 31, 32, 33 e 34.

MONSON, I. *Saying something: Jazz improvisation and interaction*. [S.l.]: University of Chicago Press, 2009. Citado na página 25.

MORIN, E.; MATOS, D. *Introdução ao pensamento complexo*. [S.l.]: Sulina Porto Alegre, 2007. Citado na página 36.

OPPENHEIM, A. V.; WILLSKY, A. S.; NAWAB, S. H. *Signals and systems*. [S.l.]: Prentice-Hall Englewood Cliffs, NJ, 1983. Citado na página 35.

PEETERS, G. Chroma-based estimation of musical key from audio-signal analysis. In: *ISMIR*. [S.l.: s.n.], 2006. p. 115–120. Citado na página 26.

S, S.; HAYKIN. *Neural networks and learning machines*. [S.l.]: Pearson Education Upper Saddle River, 2009. Citado 3 vezes nas páginas 15, 37 e 38.

SANTOS, M. Caracterização de fontes sonoras e aplicação na auralização de ambientes. Florianópolis, SC, 2008. Citado na página 29.

SHEWHART, W. *Economic Control of Quality of Manufactured Product/50th Anniversary Commemorative Issue. Millwauki: American Society for Quality, 1980*. [S.l.], 1980. Citado na página 41.

SPECHT, D. F. Probabilistic neural networks. *Neural networks*, Elsevier, v. 3, n. 1, p. 109–118, 1990. Citado 2 vezes nas páginas 38 e 39.

THÉBERGE, P. *Any Sound You Can Make: Making Music/Consuming Technology*. [S.l.]: University Press of New England, 1997. Citado na página 25.

THOSHKAHNA. A transient detection algorithm for audio using iterative analysis of stft. In: *ISMIR*. [S.l.: s.n.], 2011. p. 203–208. Citado na página 47.

TYRANGIEL, J. Auto-tune: Why pop music sounds perfect. *Time Magazine*, p. 1877372–3, 2009. Citado na página 25.

UNSER, M. Sampling-50 years after shannon. *Proceedings of the IEEE*, IEEE, v. 88, n. 4, p. 569–587, 2000. Citado na página 35.

WEISS, D. M. et al. Software product-line engineering: a family-based software development process. Addison-Wesley Professional; Har/Cdr edition, 1999. Citado na página 41.

WÖLFFLIN, H.; JÚNIOR, J. A. *Conceitos fundamentais da história da arte: o problema da evolução dos estilos na arte mais recente*. [S.l.]: Martins Fontes, 2000. Citado na página 31.

Apêndices

APÊNDICE A – Primeiro Apêndice

Esse apêndice diz respeito aos códigos feitos na plataforma MATLAB.

A.1 Código do Procedimento 1

```
% get total seconds of time to mensure the length of music
signal = signal(:,1);
time_seconds_total = fix((length(signal)/fs));

% preparing struct to allocate notes in time
set_of_notes_time = {};
for set = 1:5
    notes_time(time_seconds_total, 60) = 0;
    set_of_notes_time{set} = notes_time;
end
```

A.2 Código do Procedimento 2

```

function set_of_windows_signals = build_window_short_fft(signal, time, fs)
    signal = [signal(:)];

    % part A
    time_start_A = round(1+((time-1)*fs));
    time_end_A = round(time*fs);
    signal_time_A = signal(time_start_A:time_end_A);
    signal_time_A = blackman(length(signal_time_A)).*signal_time_A;

    % part B (displacement = + 0.2 seconds)
    time_start_B = round(1+((time-1)*fs+0.2*fs));
    time_end_B = round((time+0.2)*fs);
    if time_start_B < length(signal) && time_end_B <= length(signal)
        signal_time_B = signal(time_start_B:time_end_B);
        signal_time_B = blackman(length(signal_time_B)).*signal_time_B;
    else
        signal_time_B(length(signal)) = 0;
    end

```

```
21      % part C (displacement = + 0.4 seconds)
22      time_start_C = round(1+((time-1)*fs+0.4*fs));
23      time_end_C = round((time+0.4)*fs);
24      if time_start_C < length(signal) && time_end_C <= length(signal)
25          signal_time_C = signal(time_start_C:time_end_C);
26          signal_time_C = blackman(length(signal_time_C)).*signal_time_C;
27      else
28          signal_time_C(length(signal)) = 0;
29      end
30
31      % part D (displacement = + 0.6 seconds)
32      time_start_D = round(1+((time-1)*fs+0.6*fs));
33      time_end_D = round((time+0.6)*fs);
34      if time_start_D < length(signal) && time_end_D <= length(signal)
35          signal_time_D = signal(time_start_D:time_end_D);
36          signal_time_D = blackman(length(signal_time_D)).*signal_time_D;
37      else
38          signal_time_D(length(signal)) = 0;
39      end
40
41      % part E (displacement = + 0.8 seconds)
42      time_start_E = round(1+((time-1)*fs+0.8*fs));
43      time_end_E = round((time+0.8)*fs);
44      if time_start_E < length(signal) && time_end_E <= length(signal)
45          signal_time_E = signal(time_start_E:time_end_E);
46          signal_time_E = blackman(length(signal_time_E)).*signal_time_E;
47      else
48          signal_time_E(length(signal)) = 0;
49      end
50
51      set_of_windows_signals = {};
52      if length(signal_time_A) == length(signal_time_B) && ...
53          length(signal_time_A) == length(signal_time_C) && ...
54           length(signal_time_A) == length(signal_time_D) &&  ...
55            length(signal_time_A) == length(signal_time_E)
56          set_of_windows_signals{1} = signal_time_A;
57          set_of_windows_signals{2} = signal_time_B;
58          set_of_windows_signals{3} = signal_time_C;
59          set_of_windows_signals{4} = signal_time_D;
60          set_of_windows_signals{5} = signal_time_E;
61      else
62          set_of_windows_signals{1} = signal_time_A;
63          set_of_windows_signals{2} = signal_time_A;
64          set_of_windows_signals{3} = signal_time_A;
65          set_of_windows_signals{4} = signal_time_A;
66          set_of_windows_signals{5} = signal_time_A;
67      end
```

A.3 Código do Procedimento 3

```
% get frequency spectrum
function set_of_spectrums = get_frequency_spectrum( ...
set_of_windows_signals, sampling)

    % allocate struct to spectrum
    set_of_spectrums = {};
    sampling = sampling/21;

    for part_signal_iterator = 1:5
        % make downsample to put frequency max in 1050 Hz
        signal = downsample(set_of_windows_signals{ ...
part_signal_iterator}, 21);
        % doing fourier transform
        frequencies=(0:length(signal)-1)*sampling/length(signal);
        module_fft = abs(fft(signal));
        f_round = round(frequencies);
        frequencies_energy(max(f_round)) = 0;
        for slot = 2:length(f_round)
            frequencies_energy(f_round(slot)) = module_fft(slot);
        end
        frequency_spectrum_part = frequencies_energy(1:fix(end/2));
        set_of_spectrums{part_signal_iterator} = frequency_spectrum_part;
    end
```

A.4 Código do procedimento 4

```
function set_of_notes_time = get_energy_notes(set_of_spectrums, ...
set_of_notes_time, time)

    % load data notes
    load_notes;

    for set = 1:5
        respfreq = set_of_spectrums{set};
        notes_time = set_of_notes_time{set};

        % this case works in one case
        respfreq = [respfreq zeros(1, length(notes(1,:)) - ...
length(respfreq))];
        for note = 1:60
            notes_time(time, note) = sum((respfreq.*notes(note,:)).^2);
```

```
16          end
17
18          set_of_notes_time{set} = notes_time;
19      end
```

A.5 Código do procedimento 5

```
1  % binarize set of notes
2      for set = 1:5
3          notes_time = set_of_notes_time{set};
4
5          for time = 1:time_seconds_total
6              for note = 1:60
7                  if notes_time(time, note) < max(max(notes_time))/180
8                      notes_time(time, note) = 0;
9                  else
10                     notes_time(time, note) = 1;
11                 end
12             end
13         end
14
15         set_of_notes_time{set} = notes_time;
16     end
```

A.6 Código do procedimento 6

```
1  function bass_time = get_bass(set_of_notes_time)
2
3      notes_time_A = set_of_notes_time{1};
4      notes_time_B = set_of_notes_time{2};
5      notes_time_C = set_of_notes_time{3};
6      notes_time_D = set_of_notes_time{4};
7      notes_time_E = set_of_notes_time{5};
8
9      total_seconds = length(notes_time_A(:,1));
10     notes_time(total_seconds, 60) = 0;
11     for time = 1:total_seconds
12         for note = 1:60
13             notes_to_analyse = [notes_time_A(time, note) ...
14              notes_time_B(time, note) ...
15           notes_time_C(time, note) ...
16            notes_time_D(time, note) ...
```

```
17              notes_time_E(time, note)];
18               notes_time(time, note) = mode(notes_to_analyse);
19          end
20      end
21
22      bass_time(1:total_seconds) = 0;
23      for time = 1:total_seconds
24          maxs = find(notes_time(time,:)==max(notes_time(time,:)));
25          bass_time(time) = maxs(1);
26      end
27
28      for bass = 1:length(bass_time)
29          bass_time(bass) = mod(bass_time(bass) - 1, 12) + 1;
30      end
31
32  end
```

A.7 Código do Procedimento 7

```
1   function [chord_pitch, chord_pitch_number] = ...
2   get_chord_pitch(notes_time, time_seconds_total, chords)
3
4       dictionary_chords = { 'C', 'Cm', 'Caum', 'Cdim', ...
5        'C#', 'C#m', 'C#aum', 'C#dim', 'D', 'Dm', 'Daum', 'Ddim', ...
6        'Eb', 'Ebm', 'Ebaum', 'Ebdim', 'E', 'Em', 'Eaum', 'Edim', ...
7        'F', 'Fm', 'Faum', 'Fdim', 'F#', 'F#m', 'F#aum', 'F#dim', ...
8        'G', 'Gm', 'Gaum', 'Gdim', 'G#', 'G#m', 'G#aum', 'G#dim', ...
9        'A', 'Am', 'Aaum', 'Adim', 'Bb', 'Bbm', 'Bbaum', 'Bbdim', ...
10       'B', 'Bm', 'Baum', 'Bdim' };
11
12      notes_energy_total(60) = 0;
13      for note = 1:60
14          notes_energy_total(note) = sum([notes_time(:,note)]);
15      end
16
17      % discover tone music
18      notes_energy_tone(12) = 0;
19      for note = 1:12
20          notes_energy_tone(note) = notes_energy_total(note) + ...
21           notes_energy_total(note + 12) ...
22              + notes_energy_total(note + 2*12) + ...
23               notes_energy_total(note + 3*12) ...
24                  + notes_energy_total(note + 4*12);
25      end
```

```
26
27      % find chord tone
28      load_chords_tone;
29      chords_tone(48) = 0;
30      for chord = 1:48
31          chords_tone(chord) = sum((notes_energy_tone.* ...
32               chords_tone_mask(:, chord)'.^2));
33      end
34
35      chord_pitch_number = find(chords_tone==max(chords_tone));
36      chord_pitch = dictionary_chords{chord_pitch_number};
```

A.8 Código do Procedimento 8

```
1  function set_of_chords_time = get_set_of_chords_time(set_of_notes_time)
2      load_chords_tone;
3
4      set_of_chords_time = {};
5      for set = 1:5
6          notes_time = set_of_notes_time{set};
7          time_total = length(notes_time(:,1));
8          chords_time(1:time_total) = 0;
9
10         for time = 1:time_total
11
12             notes_energy_tone(12) = 0;
13             for note = 1:12
14                 notes_energy_tone(note) = notes_time(time, note) + ...
15                 notes_time(time, note + 12) ...
16                     + notes_time(time, note + 2*12) + ...
17                      notes_time(time, note + 3*12) ...
18                         + notes_time(time, note + 4*12);
19             end
20
21             energy_chords(1:48) = 0;
22             for chord = 1:48
23                 energy_chords(chord) = sum((notes_energy_tone.* ...
24                         chords_tone_mask(:, chord)').^2);
25             end
26
27             max_chord = find(energy_chords==max(energy_chords));
28
29             chords_time(time) = max_chord(1);
30         end
```

```
31
32          set_of_chords_time{set} = chords_time;
33      end
34  end
```

A.9 Código do Procedimento 9

```
1  function chords = analyse_set_of_chords(set_of_chords)
2
3      set_of_chords_A = set_of_chords{1};
4      set_of_chords_B = set_of_chords{2};
5      set_of_chords_C = set_of_chords{3};
6      set_of_chords_D = set_of_chords{4};
7      set_of_chords_E = set_of_chords{5};
8
9      total_seconds = length(set_of_chords_A);
10     chords(1:total_seconds) = 0;
11     for time = 1:total_seconds
12         chords_to_analyse = [set_of_chords_A(time) ...
13          set_of_chords_B(time) ...
14          set_of_chords_C(time) set_of_chords_D(time) ...
15           set_of_chords_E(time)];
16         chords(time) = mode(chords_to_analyse);
17     end
18 end
```

A.10 Código do procedimento 10

```
1  function chords_with_bass = get_chords_bass(chords_number, bass_time)
2
3   load_dictionary_chords;
4
5   % build chords with bass to translate to dictionary
6   chords_with_bass_number = {};
7   chord_iterator = 1;
8   for chord = 1:48
9      for bass = 1:12
10         chords_with_bass_number{chord_iterator} = [chord, bass];
11         chord_iterator = chord_iterator + 1;
12     end
13  end
14
```

```
15  %------------------------------------------------------------------
16  chords_with_bass = {};
17  for time = 1:length(chords_number)
18      for chord = 1:length(chords_with_bass_number)
19         peer_chord = chords_with_bass_number{chord};
20         if peer_chord(1) == chords_number(time) && ...
21             peer_chord(2) == bass_time(time)
22             chords_with_bass{time} = dictionary_chords{chord};
23         end
24      end
25  end
26
27  end
```