SISTEMA DE PESQUISA DE MÚSICAS ATRAVÉS DE SOLFEJO COM FOCO EM MÚSICAS BRASILEIRAS

Alex Libório Caranha

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Luiz Wagner Pereira Biscainho

Rio de Janeiro
Setembro de 2013

SISTEMA DE PESQUISA DE MÚSICAS ATRAVÉS DE SOLFEJO COM FOCO EM MÚSICAS BRASILEIRAS

Alex Libório Caranha

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Luiz Wagner Pereira Biscainho, D.Sc.

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Marcio Nogueira de Souza, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2013

Caranha, Alex Libório

Sistema de Pesquisa de Músicas Através de Solfejo com Foco em Músicas Brasileiras/Alex Libório Caranha. – Rio de Janeiro: UFRJ/COPPE, 2013.

XX, 155 p.: il.; 29, 7cm.

Orientador: Luiz Wagner Pereira Biscainho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2013.

Referências Bibliográficas: p. 108 – 113.

1. *Query by Humming*. 2. Transcrição Musical. 3. Estimação de *Pitch*. 4. Detecção de *Onset*. 5. Comparação de melodias. I. Biscainho, Luiz Wagner Pereira. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Aos meus pais Maria Aldenice Libório Caranha e Francisco Eligier Araújo Caranha, à minha querida esposa Luciana Leite Caranha e à minha filha Isabela Leite Caranha.

Agradecimentos

Gostaria de expressar os meus sinceros agradecimentos a um vasto conjunto de pessoas, que contribuíram de diversas formas para que a realização deste trabalho fosse possível.

Em primeiro lugar agradeço a Deus, pois sem Ele todas as coisas não seriam possíveis, agradeço por guiar meu caminho sempre e por ser tão bondoso, generoso e presente em minha vida.

Agradeço aos meus pais por serem pessoas simples e batalhadoras, por terem dado tudo que esteve ao alcance para criar seus filhos da melhor maneira possível, mesmo que para isso tivessem que se abster de seus sonhos; por terem criado a mim e às minhas irmãs com os mesmos valores, com respeito ao próximo; por serem pais amorosos e por nos colocarem em primeiro lugar em suas vidas.

À minha esposa por estar ao meu lado em todos os momentos, sejam bons ou ruins, por ter sido quem mais me apoiou na decisão de sair de nossa cidade natal para fazer o mestrado na UFRJ, enquanto nem todos acreditavam que isso seria possível; pelo apoio; por saber dizer sim e não quando necessário; por ser a mulher que eu amo; por hoje termos uma família linda já com a nossa primogênita Isabela Leite Caranha, recém-chegada em nossas vidas, e por fazer de mim uma pessoa melhor todos os dias.

Às minhas irmãs Ligiane Caranha Lima e Alcielle Libório Caranha por todo incentivo e ajuda desde o início de minha jornada no mestrado; por serem exemplos de pessoas em quem pude me espelhar e por todo amor e carinho recíproco desde sempre.

À família Menezes por me acolher como um membro de sua família, mesmo sem termos qualquer laço de parentesco; por ter confiado em meu objetivo de vir à cidade do Rio de Janeiro para estudar e por atender ao pedido de meu amigo Igor Mar (sobrinho de Sra. Vera e Sr. Ivanildo) de me ajudar no início desta caminhada. Obrigado, amigo Igor Mar, Sra. Vera, Sr. Ivanildo, Sra. Gezilda, Sr. Belmiro, Sra. Cláudia, Sr. Valério, Sra. Andréa, David, Daniel, Carla e Felippe.

Ao professor Luiz Wagner Pereira Biscainho pela oportunidade de ser meu orientador; por ser alguém com quem sempre pude conversar e buscar conselhos; por ter acreditado em meu trabalho e pela forma como este trabalho foi conduzido.

À Fundação de Amparo à Pesquisa do Amazonas - FAPEAM por ter financiado o projeto no qual este trabalho foi baseado e por ter acreditado na conclusão deste, mesmo tendo seu prazo extrapolado em relação ao acordado inicialmente, devido a uma série de fatores.

Ao meu professor e amigo Waldir Sabino da Silva Júnior por todos os conselhos dados e por ser um exemplo para mim desde a graduação.

Ao meu colega Alexandre Leizor Szczupak pela grande ajuda na configuração dos equipamentos utilizados na gravação da base de dados deste trabalho e pela troca de conhecimentos de nossa área, sempre que possível.

A todos aqueles que participaram das gravações da base de dados de solfejos de músicas brasileiras construída para este trabalho, o qual dispuseram de tempo com a simples intenção de poder colaborar com as pesquisas nesta dissertação.

Aos meus amigos Thiago Brito Bezerra e Celso Guido Rolim Costa pela ajuda financeira e pelo companheirismo quando mais precisei.

Aos meus amigos do Laboratório de Processamento de Sinais - LPS da COPPE/UFRJ pela troca de conhecimentos e acima de tudo pela amizade.

Ao meu amigo Thiago Lameirão pelo incentivo e por acreditar em meu trabalho, tornando possível minha permanência no Rio de Janeiro, e aos meus amigos da empresa Vinci Partners também pelo incentivo.

Por fim, agradeço a todas as pessoas que mencionei e também as que não mencionei, por terem contribuído direta ou indiretamente para a realização deste trabalho e por terem feito parte desta etapa de minha vida que apreciei de forma completa. O meu muito obrigado a todos.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SISTEMA DE PESQUISA DE MÚSICAS ATRAVÉS DE SOLFEJO COM FOCO EM MÚSICAS BRASILEIRAS

Alex Libório Caranha

Setembro/2013

Orientador: Luiz Wagner Pereira Biscainho

Programa: Engenharia Elétrica

O grande volume de dados multimídia armazenados diariamente em diversos sistemas e *sites* em todo o mundo cria a necessidade de algoritmos de recuperação de conteúdo que aceitem como entrada material das mais diversas naturezas: texto, imagem, vídeo ou som. Tais algoritmos, além de precisarem lidar com mais de um formato de entrada, precisam ser robustos o suficiente para que tal tarefa se realize de modo eficaz e eficiente. Apresenta-se nesta dissertação o estudo e desenvolvimento de um sistema de pesquisa de músicas através de solfejo, também conhecido por *Query by Humming*, sobre uma base de dados composta exclusivamente por músicas brasileiras. São abordados e avaliados algoritmos de estimação de frequência fundamental, algoritmos de detecção de *onset* (início de notas), algoritmo de transcrição musical e algoritmos de comparação de melodia, além do próprio sistema como um todo. Como subproduto da dissertação, foi criado um banco de solfejos gravados em formato `WAV`, que tem como objetivo simular a entrada de usuários no sistema.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

QUERY BY HUMMING SYSTEM WITH FOCUS ON BRAZILIAN SONGS

Alex Libório Caranha

September/2013

Advisor: Luiz Wagner Pereira Biscainho

Department: Electrical Engineering

The large volume of multimedia data stored daily in various systems and websites around the world creates the need for algorithms for recovery of several types of content: text, image, video or sound. Such algorithms, in addition to tackling more than one input format, must be robust enough to allow that such a task is carried out effectively and efficiently. In this work, we study and develop a system for music search through solfeggio, also known as Query by Humming, over a database comprising Brazilian music exclusively. Algorithms for fundamental frequency estimation, onset detection, music transcription and melody comparison, as well as the complete system itself, are discussed and evaluated. As a by-product of the dissertation, a database of solfeggios in `WAV` format which simulates the input of users to the system was created.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Métodos de recuperação de informações de áudio, imagem e vídeo em bancos de dados serão cada vez mais requisitados, já que a consulta a esse tipo de informação se faz necessária em acervos digitais, tanto em bibliotecas quanto em *sites* de venda de discos, ou mesmo em *sites* de pesquisa de conteúdo multimídia. Para tanto, tais métodos precisam ser bem formulados de modo a realizar pesquisas eficazes e eficientes em material volumoso.

Em áudio, um método para realizar pesquisa em bases de dados musicais através da melodia é o chamado *Query by Humming*. A partir do cantarolar de usuários, capturado com o uso de um microfone, é possível extrair informações da melodia solfejada e mesmo que haja diferenças (em tonalidade, tempo, afinação e tipo de voz) na gravação em relação a um padrão previamente armazenado, é gerada pelo sistema uma lista de músicas que correspondam de forma mais aproximada à melodia informada pelo usuário. A partir dessa lista, é possível encontrar informações sobre músicas, grupos/bandas musicais, datas de gravações e compositores, entre outras.

1.1 Motivação

A motivação para a realização deste trabalho reside no fato de que sistemas de pesquisas de músicas por conteúdo (através de voz, como solfejo, assobio e canto) serão um dia tão comuns quanto sistemas de pesquisa de músicas via texto.

A tecnologia atual deve ser capaz de realizar pesquisas de conteúdo multimídia em grande quantidade de material armazenado em bancos de dados de áudio, imagem e vídeo. Os sistemas de pesquisa de músicas através de solfejos realizam essa tarefa em áudio. Tais sistemas, como [1, 2], podem ser aplicados em:

- Bibliotecas digitais de músicas, *sites* de venda de discos [3];
- Biblioteca musical pessoal, educação musical [4];

- Aparelhos celulares ou `MP3` *players* [3];
- Sistemas de seleção de músicas em *karaoke*;
- Sistema de compra e *download* de *ringtones* para celular;
- Sistemas de treinamento musical de canto — por exemplo: *Pitch Perfector* [5], *Sing and See* [6];
- Busca de plágio em músicas.

1.2 Objetivo

Neste trabalho é descrito, desenvolvido e avaliado um sistema de *Query by Humming*. São abordados métodos de estimação de frequência fundamental, métodos de detecção de *onsets* (início de notas), método de transcrição musical e métodos de comparação de melodias encontrados na literatura, atuando sobre uma base de dados formada apenas por músicas brasileiras.

1.3 Arquitetura

Em geral, sistemas de busca de músicas por canto/solfejo/assobio possuem funcionamento e arquitetura similares. Basicamente, é necessário obter o áudio-consulta (*query*) do usuário, transcrever o sinal de áudio em alguma notação simbólica — dentre as usadas, a mais popular é a `MIDI` — e por fim, utilizar algum(ns) método(s) de comparação de melodias, com a finalidade de comparar o áudio-consulta e as melodias contidas na base de dados, representadas também em notação simbólica, tendo como saída uma lista de músicas, ordenadas de forma decrescente de semelhança com a melodia fornecida.

A arquitetura geral adotada para o sistema em estudo nesta dissertação é mostrada na Figura 1.1:

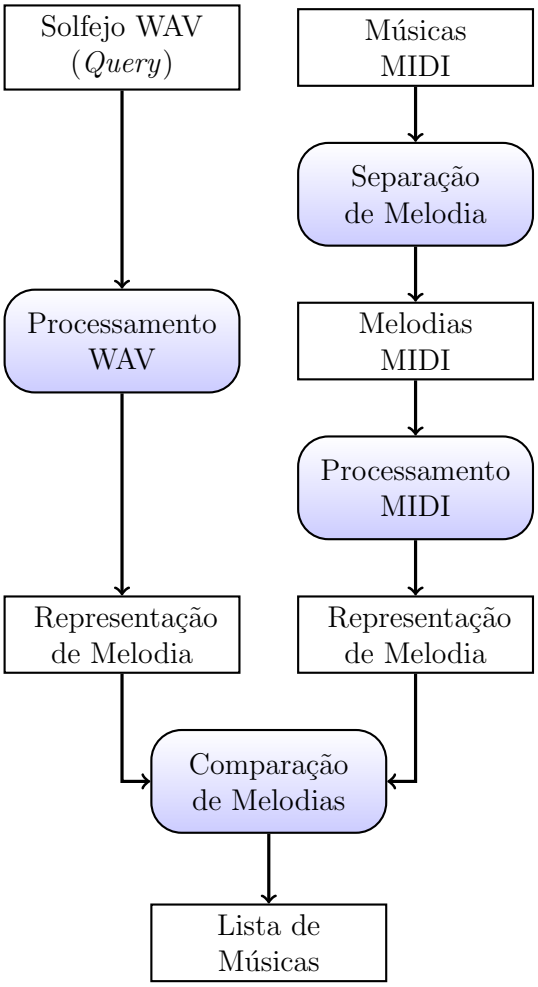


Figura 1.1: Diagrama de blocos do sistema de *Query by Humming*.

O diagrama adotado possui, em resumo, as seguintes propriedades:

- **Músicas MIDI**: Base de dados de músicas em formato `MIDI`.
- **Processamento WAV**: Ação realizada para converter o sinal de áudio `WAV` informado pelo usuário ao sistema em dois sinais, um com a representação de altura de notas (*Pitch Tracking*) em hertz e o outro com a representação de início de notas (*Onset Detection*).
- **Processamento MIDI**: Ação realizada para converter as músicas do formato `MIDI` numa notação simbólica de alto nível (**Representação de Melodia**).
- **Representação de Melodia**: Representação simbólica de sinal transcrito (tanto do áudio-consulta quanto dos arquivos em formato `MIDI` da base de dados) necessária para a realização da comparação de melodias — alguns métodos de comparação de melodias assumem que as melodias estejam em

representação simbólica diferente da `MIDI` (como em Código de Parson[1] e *pitch vectors*, dentre outras representações) para a verificação de semelhança entre as melodias.

- **Comparação de Melodias**: Adota-se algum método de comparação de melodia, com o intuito de obter as melodias mais semelhantes de acordo com o áudio-consulta.

- **Lista de Músicas**: Lista gerada pelo sistema e informada ao usuário. Esta lista contém o título das músicas ordenadas de acordo com a semelhança destas com a melodia informada pelo usuário.

1.4 Organização da Dissertação

Esta dissertação é formada por 9 capítulos, que estão organizados da seguinte forma:

O Capítulo 2 trata do modelo de produção de voz humana.

O Capítulo 3 detalha como foi preparada a base de dados de referência de músicas brasileiras utilizada neste trabalho. Esta base compreende o conjunto de músicas em formato `MIDI` e um conjunto de solfejos em `WAV` representando as entradas de solfejos dos usuários no sistema.

O Capítulo 4 apresenta estudo, implementação e avaliação de algoritmos de estimação de frequência fundamental. Etapas de pré e pós-processamento são abordadas de modo a prover melhorias ao sinal de processamento e de saída (*pitch tracking*) dos algoritmos analisados.

O Capítulo 5 apresenta estudo, implementação e avaliação de algoritmos de detecção de eventos. Etapas de pré e pós-processamento são abordadas de modo a prover melhorias ao sinal de processamento e de saída (*onset detection*) dos algoritmos analisados.

O Capítulo 6 apresenta estudo e implementação de algoritmo de representação melódica (notação base para a etapa de comparação melódica).

O Capítulo 7 apresenta estudo e implementação de algoritmos de comparação de melodias.

O Capítulo 8 apresenta a avaliação do sistema através da comparação de melodias adotando a base de solfejos no processo e a posição das músicas numa lista retornada pela aplicação.

Por fim, o Capítulo 9 apresenta as conclusões deste trabalho, suas contribuições e sugestões para a sua continuidade.

[1] Código simplificado de representação de melodias em caracteres. Utiliza os caracteres: 'U'-*Up*, 'D'-*Down* e 'R'-*Repeat* para subida, descida e repetição de notas, respectivamente, numa melodia.

Capítulo 2

Modelo de produção de voz

De modo a realizar corretamente o estudo sobre a comparação de melodias solfejadas e as etapas intermediárias neste processo, é interessante que se conheça o objeto principal de estudo desta dissertação: a voz humana. Assim, neste capítulo será abordada como acontece a produção de voz no ser humano de modo a introduzir o leitor ao tema.

2.1 Produção da voz humana

A voz humana é uma onda acústica produzida por um conjunto de órgãos, músculos, ligamentos e ossos. O aparelho fonador humano[1] é formado em parte pelos aparelhos digestivo e respiratório. Na Figura 2.1 são mostrados todos os órgãos que compõem o aparelho fonador.

O aparelho fonador é composto pelas seguintes partes:

- **Produtor**: Os pulmões, brônquios e traqueia fornecem a corrente de ar que pressiona a laringe, sendo estes órgãos do sistema respiratório;
- **Vibrador**: Na laringe produz-se o som fundamental, pois esta abriga as pregas vocais;
- **Ressonador**: A faringe, boca e a cavidade nasal são responsáveis pela amplificação do som, também conhecida por ressonância.
- **Articulador**: Lábios, língua, palato mole, palato duro e mandíbula moldam os sons, transformando-os em orais e nasais.

Devido ao conjunto das partes acima citadas ser único em cada ser humano, o sinal de voz produzido por ele contém propriedades acústicas distintas. Os hábitos de falar de cada locutor também colaboram para este fato.

[1] Conjunto de órgãos responsáveis pela formação dos fonemas.

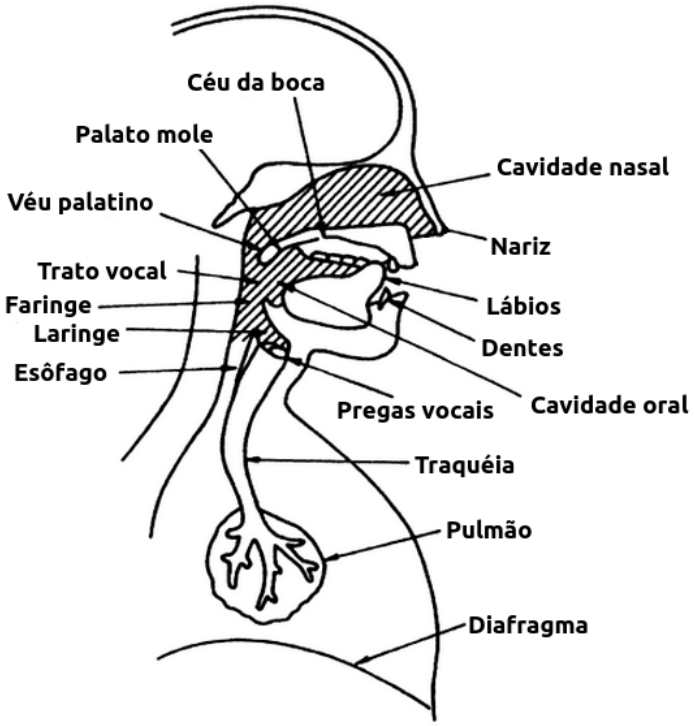


Figura 2.1: Diagrama esquemático do aparelho fonador humano, adaptado de FURUI [7].

Num modelo extremamente simplificado, a voz é produzida da seguinte forma: Quando o ar expelido pelos pulmões penetra na traqueia e alcança a laringe, onde, ao atravessar a glote[2], encontra seu primeiro obstáculo. A glote pode encontrar-se como aberta ou fechada. No caso de estar aberta, o ar força a passagem através das pregas, fazendo-as vibrar e produzindo o som com a característica das articulações sonoras. Em caso contrário, as pregas vocais, por estarem relaxadas, deixam o ar escapar sem que se haja qualquer vibração da laringe, e as articulações produzidas recebem o nome de articulações surdas. Sendo assim, a forma de onda da voz pode ser considerada como uma convolução entre a excitação e a resposta ao impulso de um filtro linear, como mostra a Figura 2.2.

Ao sair da laringe, a corrente de ar entra na cavidade da faringe, que lhe oferece dois caminhos possíveis para o exterior: o canal bucal (que determina o som oral) e o nasal (que determina o som nasal).

[2]A glote (conhecida também por pomo de Adão) é a abertura entre duas pregas musculares das paredes superiores da laringe.

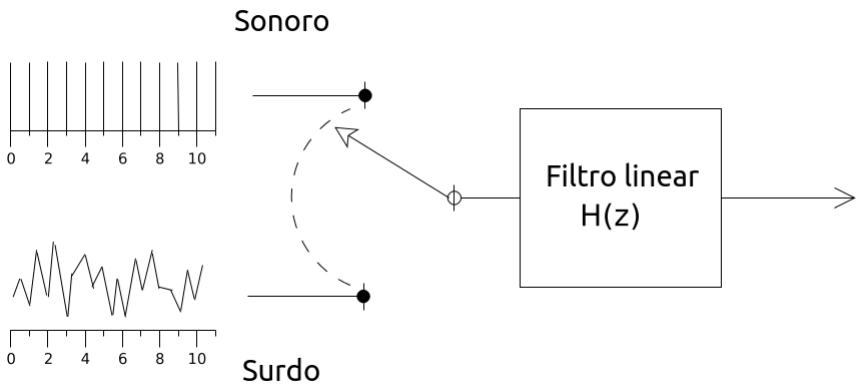


Figura 2.2: Modelo linear de produção de voz, adaptado de LÓPEZ *et al.* [8].

2.2 Características de um sinal de voz

O sinal de voz conduz diversas informações que podem ser classificadas como parâmetros de alto e de baixo nível, de acordo com suas características, como segue:

- **Alto nível**: Características usadas por seres humanos para distinção de uma pessoa de outra, como sotaque, maneirismos e conteúdo de fala.
- **Baixo nível**: Características usadas pelo cérebro humano e por sistemas de computador para diversas finalidades, não somente para reconhecimento de voz. Exemplo de características: ritmo, frequência, timbre e magnitude espectral.

Capítulo 3

Bases de dados

Neste capítulo são apresentados os parâmetros considerados na composição das bases de dados do sistema de pesquisa implementado. Tais parâmetros descrevem os critérios para as seguintes finalidades:

- Escolha das músicas;
- Formato das músicas;
- Obtenção das músicas;
- Ferramentas utilizadas na edição.

3.1 Escolha das músicas

Diversos trabalhos encontrados na literatura utilizam bases de dados de músicas norte-americanas, alemãs, chinesas, irlandesas, dentre outras nacionalidades, ou de grupo musical, como em [9], onde foram utilizadas cinquenta músicas da banda de rock britânica *The Beatles*. Por este motivo e por até então não se ter notícia de uma base de dados de músicas brasileiras, optou-se por criar uma base de dados com esta característica.

Um sistema de *Query by Humming* deve ser capaz de pesquisar as músicas mais populares de uma região (no caso, o Brasil) — ou tempo (últimos vinte anos seria uma boa meta). Como selecionar uma lista de músicas que satisfaça esses critérios? A listagem de músicas feita pela Revista Rolling Stone [10] de título: "As 100 Maiores Músicas Brasileiras" foi a escolhida para compor a base de dados, por conter músicas brasileiras, na maior parte popular, e de diferentes épocas. Devido ao fato de a lista de músicas não suprir completamente o que se procurava, foram adicionadas a essa lista 10 músicas de cunho popular, entre elas músicas infantis, de festas populares e hinos, resultando em 110 músicas no total. A lista final conta com 52 músicas selecionadas a partir destas, privilegiando sua popularidade. A lista

das músicas da base de dados pode ser encontrada na Tabela A.1 do Apêndice A, ordenada decrescentemente pelo total de gravações das músicas.

3.2 Formato utilizado

Em muitos trabalhos de *Query by Humming*, opta-se por extrair dados de *pitch* em forma de vetores (*pitch vectors*) contendo a evolução no tempo de alturas das notas de melodias, armazenando-os em uma base de dados. Para isso ser possível, três caminhos podem ser seguidos:

1. Obter ou criar uma base de dados contendo apenas a melodia de músicas em arquivos de áudio — de preferência sem compressão; em formato `WAV`, por exemplo;

2. Construir um sistema para extração da melodia de arquivos de áudio polifônico, também em formato sem compressão, como o item anterior.

3. Obter ou criar uma base de dados em formato `MIDI` (monofônico ou polifônico). A identificação da trilha[1] que corresponde à linha de voz cantada — comumente usada como linha de melodia em base de dados — é simples de ser encontrada e manuseada em arquivos neste formato.

Dos itens citados, o mais encontrado em trabalhos nesta área é o terceiro, devido à facilidade de extração de *pitch vectors* de arquivos `MIDI` (embora arquivos `MIDI` não sejam propriamente gravações como arquivos `WAV`, `MP3` e `OGG`, dentre diversos outros), devido ao tamanho do arquivo — que comparado com outros formatos é mínimo (cerca de alguns *kilobytes* em média) — e por esse tipo de arquivo ser popular e amplamente aceito por músicos e terceiros, profissionais e amadores.

Outro ponto importante: a extração da melodia de material polifônico é uma área de intenso estudo. A separação da melodia do restante do áudio ainda não é garantida com perfeição nos *softwares* atuais, sendo este um ponto desfavorável à opção pela confecção da base de dados pretendida através deste caminho. O primeiro caminho identificado foi invalidado por não haver uma base de dados somente de melodias de músicas brasileiras, em formato sem compressão, até este momento disponível na literatura. O caminho escolhido foi, portanto, o terceiro: decidiu-se criar uma base de dados composta de músicas no formato `MIDI`.

[1] Arquivos no formato `MIDI` possuem as informações de cada instrumento organizadas em trilhas, junto com seus eventos (pausas, notas e tempos, dentre outros).

3.3 Obtenção das músicas

Como já mencionado, músicas em formato `MIDI` são facilmente encontradas na internet. Este foi o ponto de partida para a obtenção das músicas para a base de dados. Uma vez encontradas as músicas, foram selecionados trechos, um por música, de modo a guardar na base de dados apenas o trecho mais popular, facilmente identificável pelas pessoas. O critério utilizado para isso foi em geral escolher o refrão ou as primeiras frases cantadas de cada música, correspondendo a 30 segundos, em média.

Por mais que músicas em formato `MIDI` sejam criadas e editadas por profissionais e terceiros, não se pode garantir a exata transcrição realizada por seus criadores. Sendo assim, para garantir sua confiabilidade em relação à versão idealizada pelo compositor ou mais popularmente difundida, cada música passou pelas seguintes etapas:

1. Pesquisa e obtenção da música no formato `MIDI` na internet que contivesse a trilha da melodia. Nesta etapa foi perseguida a trilha de voz cantada na música.

2. Edição de possíveis erros na melodia quanto às notas, pausas e tempo identificados a partir da melodia da versão de estúdio da música. Esta etapa contou com a ajuda de um profissional em transcrição musical. Quando necessário, este criou integralmente o arquivo a partir da edição de uma partitura.

3. Quando necessária, a remoção de todas as trilhas do arquivo `MIDI`, com exceção da trilha da melodia.

4. Quando necessária, a remoção de todos os trechos, com exceção do trecho selecionado da música.

O resultado da etapa 1 corresponde aos arquivos que compõem a base de **Músicas MIDI**. As etapas 2, 3 e 4 correspondem ao processamento chamado de **Separação de Melodia** realizado nesta base de dados, gerando como resultado os arquivos de melodia que compõem a base de **Melodias MIDI**.

3.4 Base de Dados de Solfejos

Nesta seção são apresentadas as etapas avaliadas na composição da Base de Dados de Solfejos do sistema *Query By Humming* desenvolvido.

3.4.1 Por que solfejos?

Existem alguns ramos a seguir num sistema de pesquisa de músicas. São estes:

- *Query by Singing*: Consulta através do canto;
- *Query by Whistling*: Consulta através de assobio;
- *Query by Humming*: Consulta através de solfejo.

Devido a o fato de cantarolar/solfejar ser uma opção bastante simples de se recordar uma música e seu emprego ser realizado com certa naturalidade pelas pessoas, optou-se por realizar um sistema de pesquisa de músicas através de solfejos. Outro fator importante: o fato de a estimação de altura e a identificação de eventos musicais (como notas) serem mais facilmente reconhecidas no sinal de áudio de solfejo que em canto, isso facilita a tarefa de transcrição, necessária em sistemas de pesquisa de música por conteúdo, como os citados acima. E por último, assobiar é uma ação que nem todas as pessoas são aptas a realizar.

3.4.2 Qualidade das gravações

Os solfejos dos usuários neste trabalho são gravados em formato `WAV` com taxa de amostragem de 48.000 amostras por segundo, em PCM[2] (*Pulse Code Modulation*) de 16 bits, em um só canal, ou seja, mono. Tal configuração foi escolhida sabendo que o aumento de qualidade em um arquivo de áudio de baixa qualidade não é possível de ser realizada, já o contrário sim. Esta configuração é próxima da melhor qualidade possível de gravação de áudio nos equipamentos utilizados, e permite, se desejado, a realização de testes com qualidade reduzida.

3.4.3 Gravador de solfejos

Para realizar as gravações de solfejos, fez-se necessária a criação de um aplicativo de auxílio aos usuários/participantes neste processo. Este aplicativo possui as seguintes etapas:

1. Escolha das músicas e gravação de solfejos com acompanhamento de áudio: Nesta etapa, o aplicativo apresenta uma lista de 10 músicas escolhidas aleatoriamente da base de dados. O usuário deve então solfejar o trecho indicado de cada música, podendo solicitar a mudança também aleatória de uma música por outra pelo programa. Uma vez escolhida uma música, o usuário tem a

[2]Formato de áudio digital desenvolvido na década de 70. Utilizado no sistema telefônico e em CDs (*Compact Discs*), dentre outros.

opção de ouvir o trecho que deve ser solfejado, sendo este áudio apresentado em formato `WAV` ou `MIDI`, junto com a letra da melodia cantada. Para isso, foi necessário gerar um conjunto de trechos `WAV` baseados na base de dados de **Melodias MIDI**. Estes arquivos `WAV` foram obtidos através de *sites* e editados de modo a conter apenas o trecho da música correspondente ao trecho presente no arquivo `MIDI` de cada música. Caso o usuário não consiga acompanhar o áudio durante a gravação, este pode gravar novamente o solfejo quantas vezes achar necessário.

2. Gravação de solfejos sem acompanhamento: Após realizar a primeira etapa de gravações, o usuário deve gravar novamente os solfejos, mas desta vez sem acompanhamento algum, ou seja, *a cappella*[3]. O aplicativo apresenta a lista com as mesmas músicas da etapa 1, porém em ordem embaralhada, de modo a permitir que o usuário realize o solfejo da maneira que recorda a música e não por lembrar de como gravou na etapa anterior. O usuário é livre para solfejar no tempo e na tonalidade em que se sentir mais à vontade. Assim como na primeira etapa, o usuário pode gravar o solfejo para cada música quantas vezes quiser.

A linguagem de programação escolhida para criação do aplicativo foi a linguagem `JAVA` devido a esta ser multiplataforma, ser *software* livre sob os termos da *GNU General Public License* (GPL), ser orientada a objetos, ser segura e possuir vasto conjunto de bibliotecas e material na internet.

A Figura 3.1(a) mostra uma lista de músicas da etapa 1 a serem solfejadas pelo usuário. A Figura 3.1(b) mostra a letra da primeira música, que serve como referência ao usuário na gravação do solfejo.

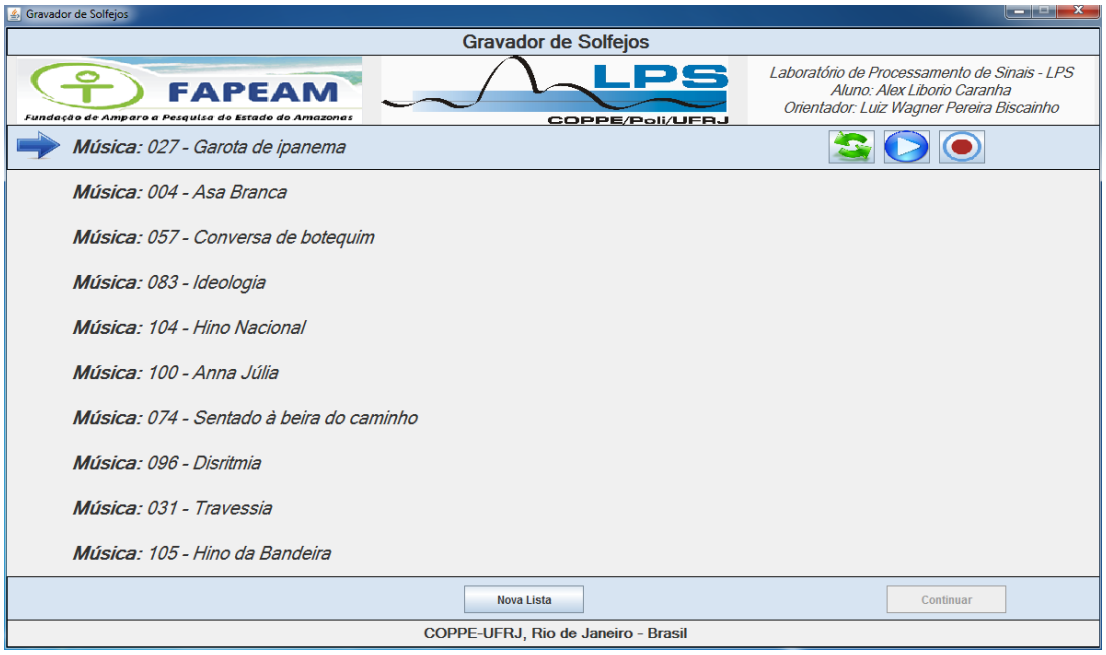
A Figura 3.2(a) mostra a mesma lista de músicas da etapa 1, mas desta vez embaralhada. A Figura 3.2(b) mostra a letra da primeira música, que serve como referência ao usuário na gravação do solfejo, assim como na etapa 1.

O aplicativo foi criado com a intenção de o usuário poder gravar o solfejo sem a intervenção de outra(s) pessoa(s). Por este motivo, a interface do programa foi pensada de modo a ajudar o usuário a realizar tal tarefa, com informações sendo mostradas em cada botão e com figuras sugestivas e de fácil assimilação.

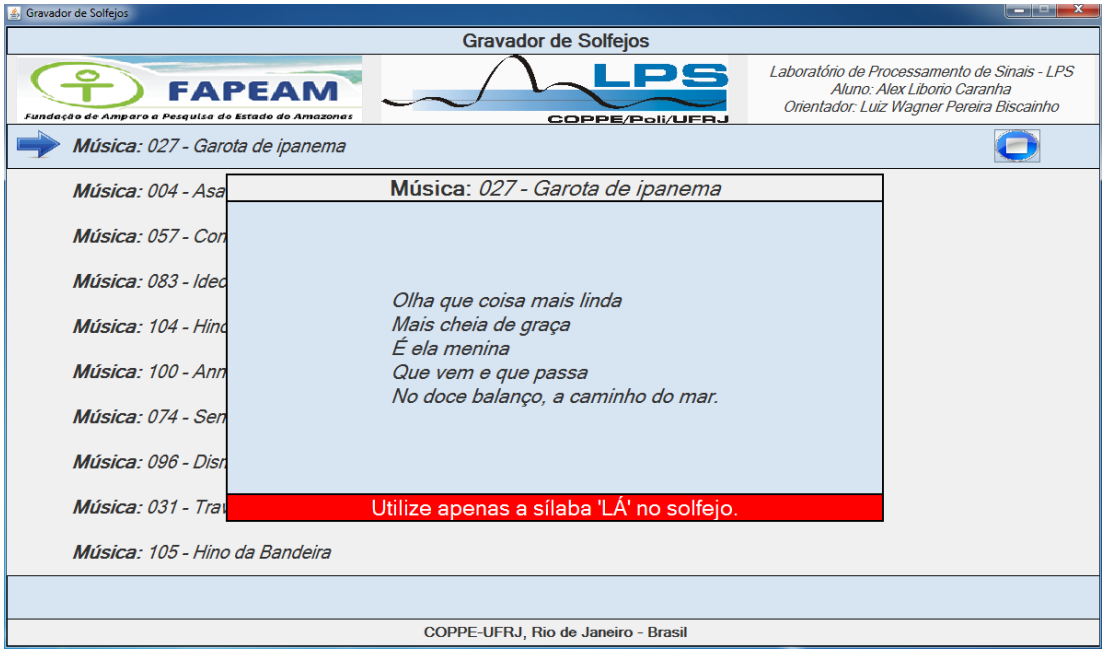
A tela inicial do programa solicita a criação da ficha do usuário que realizará as gravações. A ficha possui duas importantes funções. A primeira, para possível levantamento estatístico, pois são levantadas informações como idade, sexo, cidade, estado e país de origem. A segunda, para servir de controle no caso de um usuário desejar participar mais de uma vez das gravações; neste caso, o aplicativo elimina

[3]De origem italiana, o termo descreve o ato de cantar sem acompanhamento instrumental.

da lista as músicas que já foram solfejadas pelo usuário, de modo a não existir mais de um solfejo para uma música de um mesmo usuário na base de solfejos.

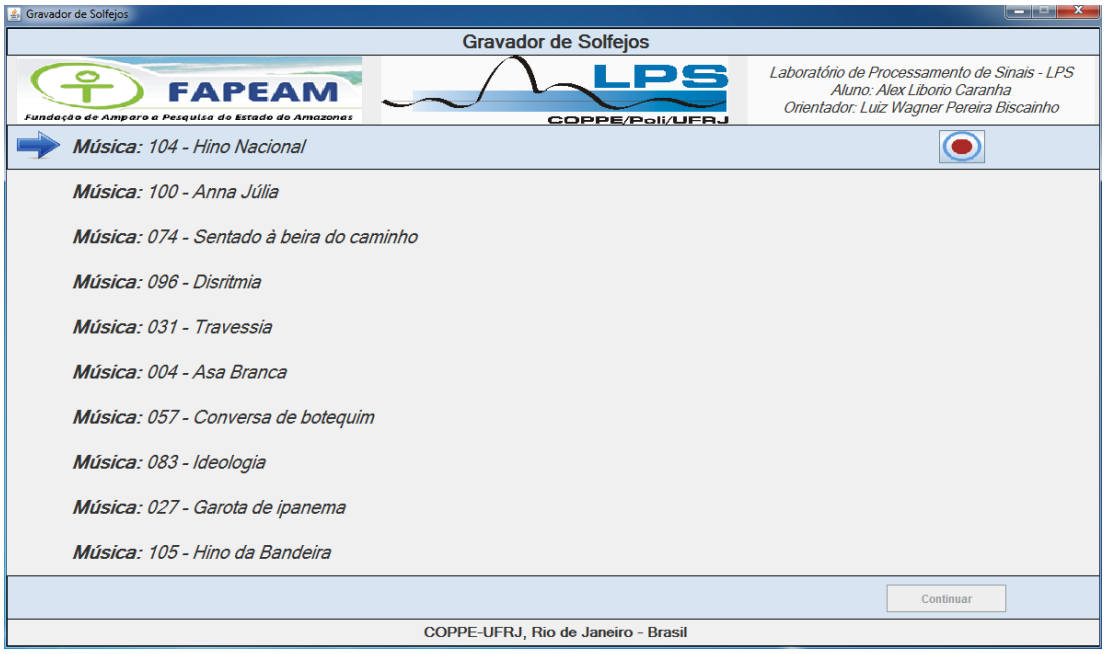


(a) Lista de Músicas

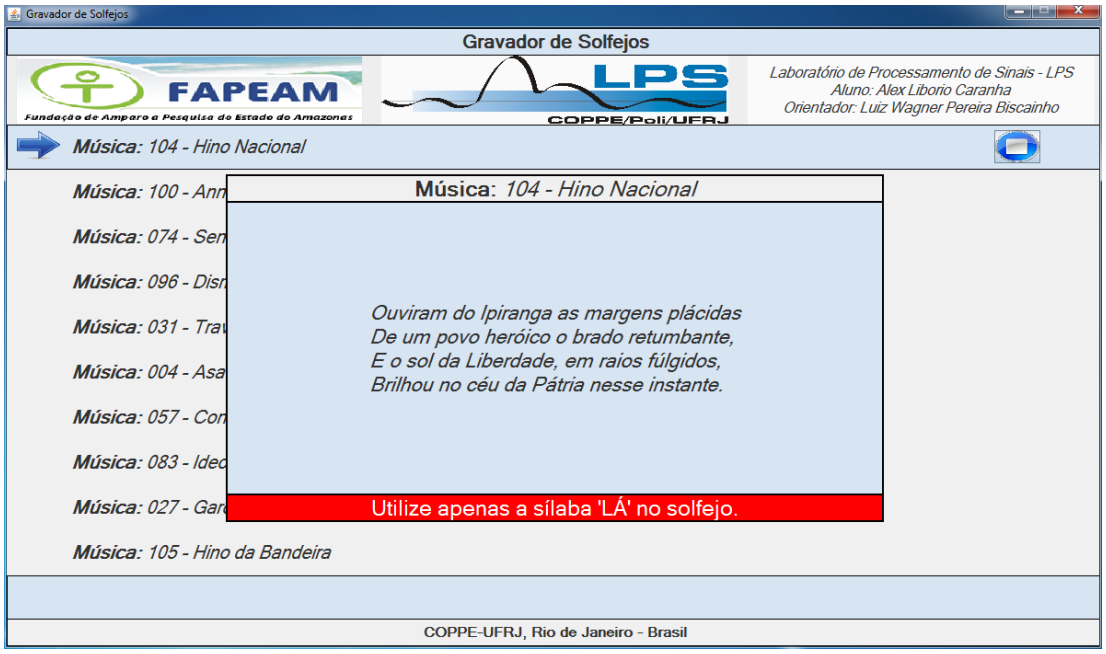


(b) Letra do trecho a ser solfejado

Figura 3.1: Imagens da etapa 1 do Gravador de Solfejos: O usuário escolhe e grava os solfejos procurando seguir o acompanhamento (arquivo `MIDI` ou `WAV`).



(a) Lista de Músicas



(b) Letra do trecho a ser solfejado

Figura 3.2: Imagens da etapa 2 do Gravador de Solfejos: O usuário grava os solfejos novamente, mas desta vez, sem acompanhamento algum.

3.4.4 Ambiente e equipamentos utilizados

As gravações de solfejos foram realizadas nas dependências do LPS (Laboratório de Processamento de Sinais) da UFRJ, em duas salas. A primeira, acusticamente isolada, própria para gravações de áudio, acomoda: o usuário que fará as gravações de solfejos, 1 par de fones de ouvido, 1 microfone e cabos de som. Foi adicionado

um *mouse wireless* nesta sala, para o usuário operar o aplicativo **Gravador de solfejos** à distância, uma vez que optou-se por não deixar o computador/*notebook* na mesma sala, evitando assim qualquer tipo de ruído causado por este equipamento. A segunda sala abriga: 1 interface de áudio M-Audio, 1 mesa de som Alesis Multimix8, 1 *notebook*, cabos de som, 1 par de fones de ouvido, 1 microfone e um instrutor para auxiliar o usuário no uso do aplicativo.

Os equipamentos utilizados foram os seguintes:

- **M-Audio - Fast Track Pro**: É uma interface de áudio *usb* (*Universal Serial Bus*) compatível com os programas mais conhecidos do mercado. Possui recurso *phantom power* para as entradas de microfone. Através da conexão *usb*, é possível o acesso a todas as entradas e saídas necessárias na gravação de microfones, guitarras, etc., assim como `MIDI`.

Figura 3.3: Fast Track Pro da M-Audio

  Dentre as características da interface, mostrada na Figura 3.3, destacam-se:

  - 2 entradas frontais mic/instrumento com preamps (Neutrik XLR/ 1/4"TRS);
  - Entrada (1/4"TRS) para efeito externo;
  - 2 saídas balanceadas (1/4"TRS) e 4 saídas não balanceadas (RCA);
  - Monitoração direta via *hardware* de baixíssima latência;
  - Saída de fones de ouvido (1/4"TRS) com controle de volume;
  - Seletor de fonte A/B para fones de ouvido;
  - Controle de entrada e saída para monitoração direta.

- **Alesis - MultiMix8**: É um dispositivo interface de gravação de áudio e *mixer* multicanal. Não somente é possível gravar cada canal independentemente, como também gravar a mistura destes.

  Dentre as características do dispositivo, mostrado na Figura 3.4, destacam-se:

  - Interface de áudio USB integrada;

Figura 3.4: MultiMix8 da Alesis

  - 4 entradas de alto ganho para microfone ou linha com *phantom power*;
  - 3 bandas de equalização por canal;
  - Entradas: 2 pares de entradas estéreo;
  - Nível de saída: fornece controle sobre a saída separada da sala de controle (*control room*);
  - Compatível com Mac OS X, Windows XP (32 bits) ou Vista (32 bits).

- **Sennheiser HD 265**: Par de fones de ouvido profissional que oferece um alto nível de atenuação de ruído de fundo para aplicações de monitoramento profissional e amador.

Figura 3.5: Fone de ouvido Sennheiser HD 265

Dentre as características do dispositivo, mostrado na Figura 3.5, destacam-se:

  - Impedância: 150 ohms;
  - Sensibilidade dos auscultadores: 106 dB;
  - Frequência dos auscultadores: 10 - 25.000 Hz;
  - Comprimento do cabo: 3 m;
  - Peso do produto: 215 g;
  - Conector: $3,5/6,3$ mm estéreo.

- **TSI 650 SW**: Microfone de som limpo, de alta projeção, possui captação com a máxima projeção e um dos menores níveis de distorção do mercado.

Figura 3.6: Microfone TSI 650 SW

Dentre as características do microfone, mostrado na Figura 3.6, destacam-se:

  - Resposta de frequência de 50 Hz a 15 kHz;
  - Baixa impedância: 250 ohms;
  - Nível de potência -52.1 dB sendo 0 dB igual a 1 mw / 10 μbar.
  - Certificação ISO 9002.

- **Shure Model 81**: É um microfone condensador unidirecional de alta qualidade projetado para estúdio de gravação, transmissão e reforço de som. Sua ampla resposta de frequência, as características de baixo ruído e baixa suscetibilidade a RF o tornaram um padrão para aplicações que envolvam instrumentos acústicos.

  Dentre as características do microfone, mostrado na Figura 3.7, destacam-se:

  - Resposta de frequência de 20 Hz a 20 kHz;
  - Baixo ruído e nível de corte de saída alta;
  - Baixa distorção sobre uma ampla gama de impedâncias de carga;
  - Baixa suscetibilidade a RF;
  - Impedância de saída avaliada em 150 ohms (85 ohms real);

Figura 3.7: Microfone Shure Model 81

3.4.5 Outras características

- A base de dados de solfejos conta com um total de 1.195 gravações por 43 pessoas.
- As gravações foram realizadas de agosto a dezembro do ano de 2011.
- Nas Tabelas A.1 e A.2 do Apêndice A, é possível encontrar a quantidade de gravações por música, de acordo com o tipo de acompanhamento que o usuário utilizou, sendo a primeira referente às gravações e a segunda às gravações que apresentaram alguma saturação (para referência):
  - **Tipo 1**: Acompanhamento de piano (`MIDI`);
  - **Tipo 2**: Acompanhamento de música (`WAVE`);
  - **Tipo 3**: Sem acompanhamento (*a cappella*).

Capítulo 4

Algoritmos de estimação de frequência fundamental

4.1 Introdução

A etapa de estimação de frequência fundamental é uma das etapas mais importantes em sistemas de Transcrição Musical Automática - TMA, pois tem como objetivo a localização da altura das notas presentes no sinal. Sendo necessária a transcrição musical do sinal de áudio informado pelo usuário em sistemas de *Query by Humming*, esta etapa deve receber a mesma importância em sistemas dessa natureza.

Para sinais perfeitamente harmônicos, a frequência fundamental ou f_0 corresponde ao 1º harmônico localizável (se presente) em um trecho de sinal de áudio monofônico com altura definida. Por corresponder ao espaçamento entre cada duas frequências da série harmônica, esta frequência é responsável pela percepção de altura de notas, enquanto que o balanço de energia entre todos os harmônicos responde pela composição do timbre[1] da fonte sonora. O uso da palavra *pitch* (altura percebida em Hz) referindo-se à frequência fundamental é normalmente encontrado na literatura, uma vez que para sinais harmônicos o *pitch* e a frequência fundamental são equivalentes, ainda que não haja esta componente. Segundo PARK [11], *pitch* é um aspecto perceptivo do som originado por características periódicas ou quase periódicas deste, correspondendo à altura percebida[2]. A frequência fundamental f_0 e o período fundamental T_0 possuem a seguinte relação:

$$f_0 = \frac{1}{T_0} \tag{4.1}$$

Os algoritmos de estimação de frequência fundamental devem traçar, então, a

[1] A natureza das fontes sonoras é caracterizada pelo timbre — o reconhecimento de instrumentos através do som só é possível graças a seus timbres, que os caracterizam.

[2] É mais rigoroso definir *pitch* como a altura percebida do som numa dada intensidade (40 dB SPL)

altura das notas presentes ao longo do tempo em um sinal de áudio monofônico. Isso é possível apenas se o algoritmo encontrar alguma periodicidade no trecho avaliado do sinal. Deste modo, os algoritmos devem discriminar se o som presente no sinal é sonoro ou não-sonoro — lembrando que existem outras definições, como oclusivos sonoros e fricativos sonoros, que possuem características tanto dos sonoros quanto dos não-sonoros. Neste trabalho, a estimação da altura das notas é realizada em sinais de solfejo realizados pelos usuários, utilizando a sílaba "lá". Como os sinais gravados nem sempre estão livres de ruído, interferências diversas ou mesmo mau uso do microfone — devido ao fato de algumas pessoas cantarem com o microfone muito próximo à boca, causando saturação do sinal — normalmente é utilizado algum pré-processamento antes da execução dos algoritmos de estimação de *pitch*, para eliminar características indesejadas do sinal, através de filtros ou heurísticas.

Muitos trabalhos realizam no pré-processamento a reamostragem do sinal de voz, deixando-o com qualidade inferior à original. Dessa forma, espera-se que a velocidade de execução destes algoritmos aumente, tendo em vista a redução dos dados a serem processados por eles. Outro motivo para a reamostragem é testar/medir a eficiência dos algoritmos em sinais de qualidade igual à dos sinais de áudio em sistemas de telefonia, com frequência de amostragem fixada em 8.000 Hz. Esta taxa de amostragem é mais que suficiente para a realização do *pitch tracking*, uma vez que a concentração da maior parte de energia encontra-se abaixo de 4.000 Hz, o que se pode observar facilmente no espectrograma de um sinal de voz cantada.

Tendo como base a partitura mostrada na Figura 4.1, considere os arquivos em formato `WAV` mostrados na Figura 4.2, sintetizado como um piano e pelo *software* Vocaloid[3], respectivamente, como sinal de entrada para os algoritmos apresentados neste capítulo, de modo a exemplificá-los.

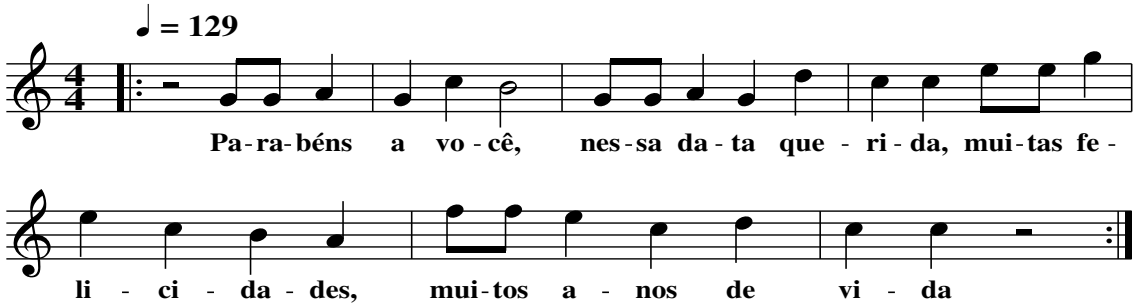
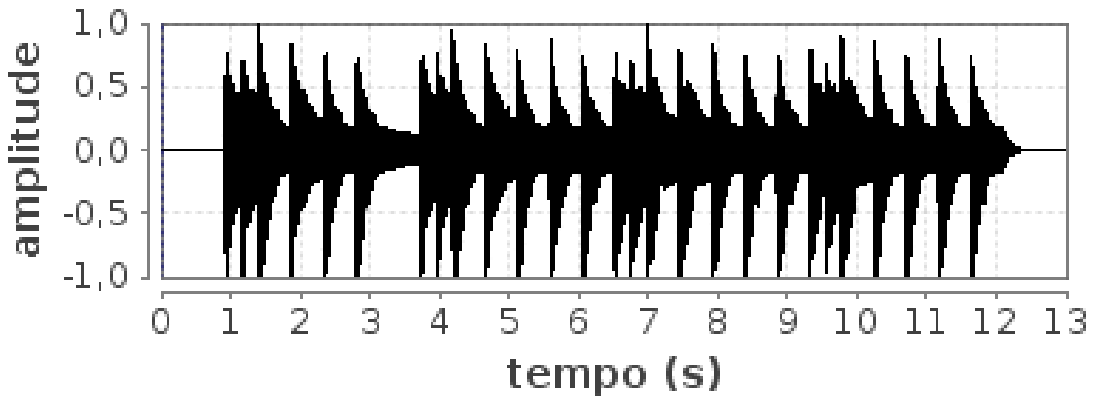
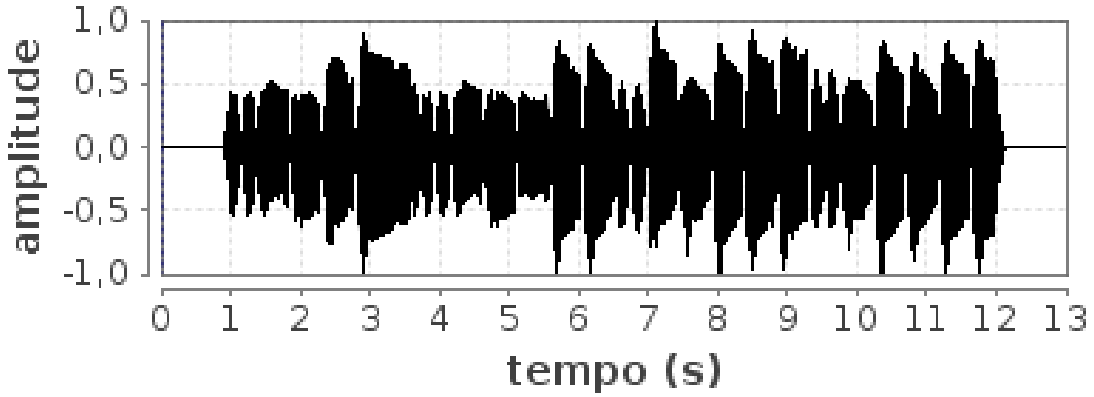


Figura 4.1: Partitura da música: "Parabéns a você".

[3] *Software* sintetizador de voz. Mais informações a respeito deste *software* são encontradas na seção 4.3 - Avaliação.



(a) Sinal gerado por um sintetizador de piano.



(b) Sinal gerado por um sintetizador de voz utilizando apenas a sílaba: "lá".

Figura 4.2: Ambos os sinais (a) e (b) estão amostrados em 8.000 Hz e são baseados na partitura mostrada na Figura 4.1.

A Figura 4.3 mostra o *pitch tracking* obtido pela frequência convertida de cada nota em formato `MIDI` para Hz, tendo como base a partitura da Figura 4.1. Essa conversão foi obtida através da Equação (4.2):

$$f(m) = 440.(\sqrt[12]{2})^{m-69}, \tag{4.2}$$

onde m representa a nota em formato `MIDI` e f a frequência em Hz. Tal equação utiliza a escala temperada ocidental em sua formação, que define 12 notas por oitava, onde a razão entre notas contíguas vale $\sqrt[12]{2}$. A nota Lá - 440 Hz, conhecida por A4 ou Lá da 4ª oitava, corresponde à nota 69 em `MIDI`.

O sinal mostrado na Figura 4.3 pode ser identificado como o caso ideal de *pitch tracking*, uma vez que não apresenta qualquer perturbação ocasionada por diferentes timbres, inexatidão quanto à afinação, vibratos ou qualquer outra característica encontrada em sinais gerados por instrumentos musicais em geral, assim como na

própria voz cantada. Vale ressaltar que o *pitch tracking* esperado para o piano é diferente do esperado para a voz, pois o primeiro tem ataque abrupto e frequência fundamental fixa, e o segundo tem ataque suave e algum vibrato. Portanto, ambos podem ser diferentes do que aparece na Figura 4.3.

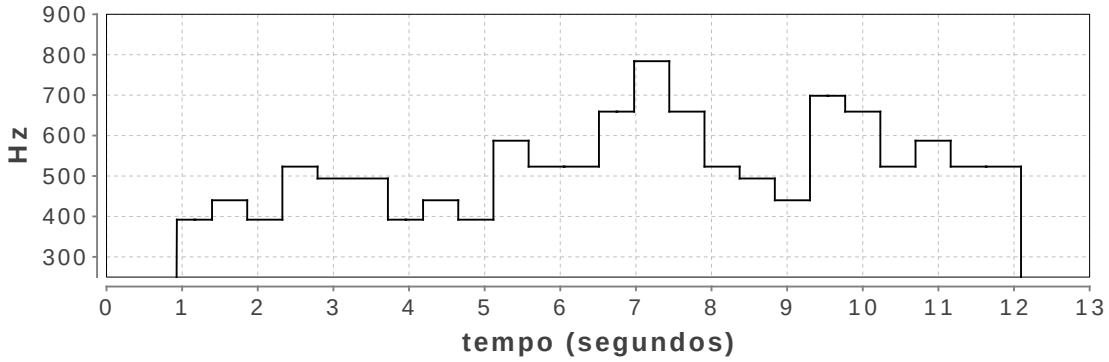


Figura 4.3: *Pitch tracking* obtido pela conversão de notas do formato MIDI para Hz da música "Parabéns a você", tendo como base a partitura da Figura 4.1, gerado a uma taxa de 100 amostras por segundo.

Os parâmetros para o processamento de voz utilizados nos algoritmos de estimação de *pitch* foram os seguintes:

- **Frequência de amostragem**: A frequência de amostragem adotada para os sinais de áudio foi de 8.000 Hz. Com isso, os algoritmos serão processados mais rapidamente do que em Frequências de amostragem maiores.

- **Largura do bloco de análise**: O bloco de análise (*frame*) deve ser de largura suficiente para que os algoritmos consigam estimar com sucesso a menor e a maior frequências desejadas. Segundo HSU *et al.* [12], o intervalo de 80 a 800 Hz engloba a maior parte de *pitches* de voz cantada. LUENGO *et al.* [13] utilizaram como intervalo de *pitch* os limites 35 a 500 Hz. KLAPURI e DAVY [14] utilizaram como intervalo os limites 65 a 2.100 Hz. Os limites de estimação de *pitch* adotados neste trabalho equivalem a C2 e B5, que em *Hertz* correspondem a 65,4064 e 987,767 Hz, respectivamente. Com isso, os períodos ficam limitados entre 1/65,4064 $\approx$ 15 ms e 1/987,767 $\approx$ 1,0 ms. A largura utilizada deve, então, ser de no mínimo 15 ms para que os algoritmos consigam estimar as frequências nesse intervalo desejado. Segundo LÓPEZ *et al.* [8], blocos de largura entre 20 e 30 ms são adequados para a maioria das aplicações. Testes empíricos mostraram que uma largura de blocos de 25 ms apresentava resultados satisfatórios para os algoritmos abordados neste capítulo.

- **Salto entre blocos**: O salto entre blocos de análise é estipulado segundo o número de estimações de f_0 por segundo desejado. Estima-se que 100 medições de f_0 por segundo seja um número adequado para esse tipo de aplicação. Assim, o salto entre blocos é obtido seguindo o raciocínio abaixo:

$$\text{salto} = \frac{1\text{ s} - (\text{largura do bloco de análise})}{99} = \frac{1000\text{ ms} - 25\text{ ms}}{99} \approx 9,85\text{ms}.$$

A Figura 4.4 mostra o diagrama de etapas frequentemente utilizado na estimação de frequência fundamental.

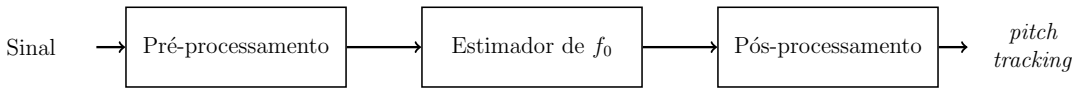


Figura 4.4: Diagrama de blocos de um estimador de f_0.

4.2 Pré-processamento

A etapa de pré-processamento tem como objetivo melhorar o desempenho do algoritmo de estimação de f_0. O pré-processamento empregado obedece a seguinte ordem:

1. **Reamostragem**: Cada sinal de áudio em formato `WAV` é reamostrado a 8.000 Hz. Isso é realizado a partir de decimações e interpolações do sinal. No *software* Matlab, por exemplo, utiliza-se a função de nome *resample*, com os seguintes parâmetros:

   - **x**: representa o sinal que será reamostrado;
   - **p**: número inteiro que indica a quantidade de pontos que se deseja interpolar;
   - **q**: número inteiro que indica a quantidade de pontos que se deseja decimar.

   Em suma, a frequência de amostragem será multiplicada pelo fator $\frac{p}{q}$ e o tamanho do sinal será o menor inteiro resultante da multiplicação do tamanho do sinal original pelo produto deste fator. Com isso, uma forma de reamostrar um sinal x, de frequência de amostragem $f_s = 48.000$ Hz para 8.000 Hz segue: *sinal = resample (x, 1, 6);*

2. **Detecção de voz**: De modo a diminuir o tempo total de processamento do algoritmo de estimação de *pitch*, foi utilizado o algoritmo proposto por TADJIKOV e AHMADI [15], que verifica a existência de voz a partir de um limiar baseado na energia do sinal presente em cada *frame*. Os *frames* que possuem energia inferior ao limiar indicam a inexistência de voz e são simplesmente ignorados pelas demais etapas do estimador. A energia em cada *frame* é calculada pela Equação (4.3):

$$E(k) = \sum_{n=(k-1).N+1}^{k.N} |x[n]|^2, \tag{4.3}$$

onde $x[n]$ corresponde ao sinal de entrada em forma de onda, k o índice do *frame* atual e N a largura do *frame* em amostras.

Após calcular a energia em cada *frame*, calcula-se o limiar de detecção de voz pela Equação (4.4):

$$\text{limiar} = \frac{\max E}{100}, \tag{4.4}$$

onde $\max E$ indica a maior energia calculada considerando todos os *frames*.

4.3 Estimadores de frequência fundamental

4.3.1 Autocorrelação - ACF

O estimador ACF (*Autocorrelation Function*) baseia-se na medida de similaridade entre as amostras de um sinal, através da soma do produto destas, espaçadas de τ. Existem diversas abordagens para este estimador:

1. *Convolução Discreta*: São utilizadas N multiplicações para compor $\text{ACF}_1(x)[\tau]$, onde N equivale à largura do *frame* x e τ ao índice de atraso discreto.

$$\text{ACF}_1(x)[\tau] = \sum_{n=0}^{N-1} x[n].x[\tau + n]. \tag{4.5}$$

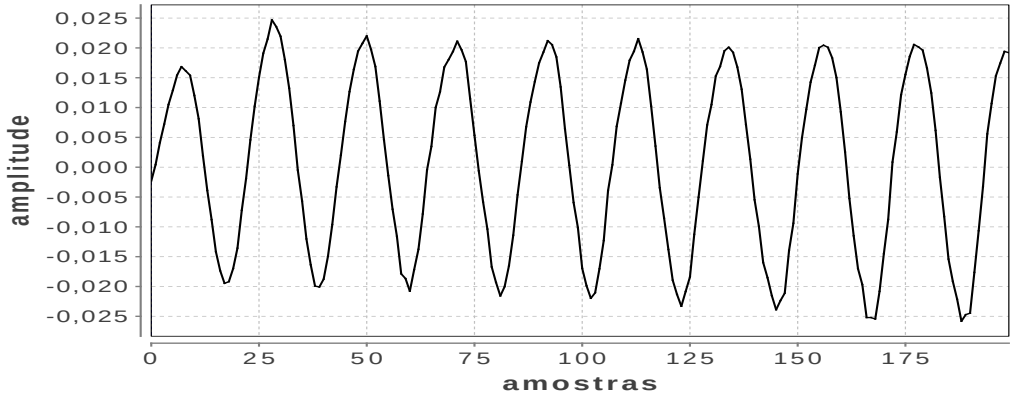
2. *FFT (Fast Fourier Transform)*: Multiplica-se o *frame* x por uma janela de *Hamming* w. Representa-se o *frame* resultante no domínio da frequência através da FFT. Realiza-se o produto deste *frame* no espectro pelo seu conjugado. Por fim, representa-se o *frame* resultante no domínio do tempo, conforme equação a seguir:

$$\mathrm{ACF}_2(x) = \mathrm{IFFT}(Z.Z^*), \tag{4.6}$$

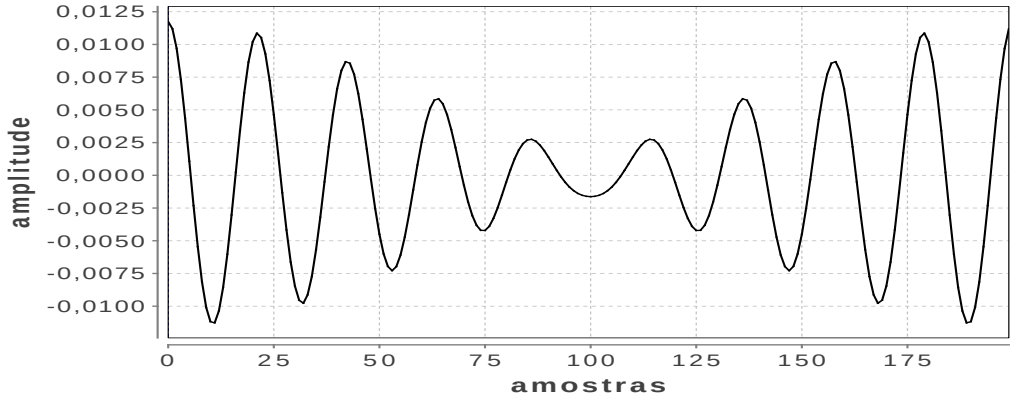
$$Z = \mathrm{FFT}(x.w). \tag{4.7}$$

A segunda abordagem foi a adotada na implementação deste algoritmo nesta seção, por despender menos processamento computacional quando comparada à primeira.

A Figura 4.5 mostra o gráfico de um *frame* e de sua autocorrelação gerada a partir da Equação (4.6).



(a)



(b)

Figura 4.5: Gráfico de um *frame* gerado a partir de um sinal de solfejo amostrado em 8.000 Hz (a), e sua autocorrelação (b) gerada a partir da Equação (4.6).

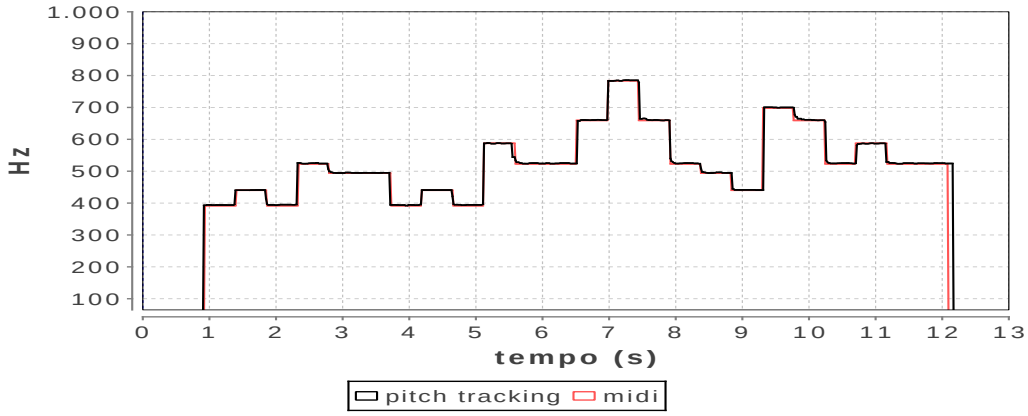
Dentre as diversas formas possíveis de estimar a frequência fundamental de um sinal através deste método, a utilizada neste trabalho corresponde a encontrar o maior pico que não o da origem do sinal de autocorrelação dentro da faixa de frequência de 65 a 1000 Hz, realizar uma aproximação parabólica de 3 pontos neste pico, de

modo a não abranger apenas os pontos múltiplos da frequência de amostragem, e por fim, calcular a frequência fundamental pela Equação (4.8).

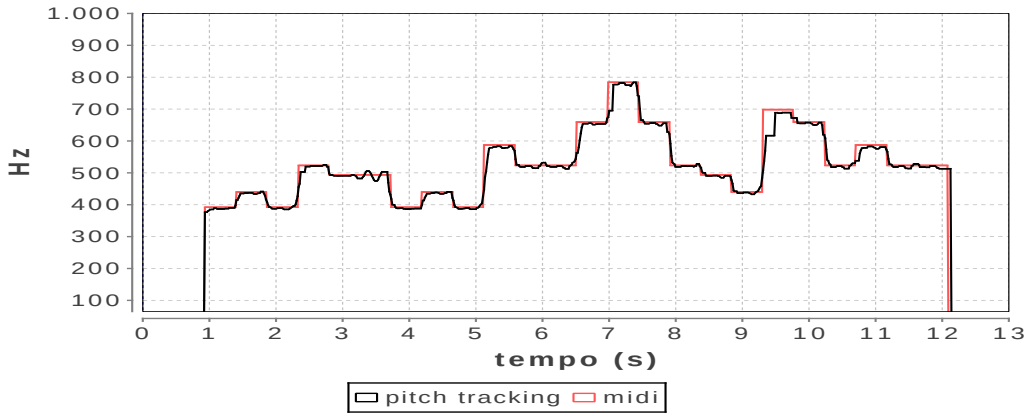
$$f_0(n) = \frac{f_s}{p_x}, \tag{4.8}$$

onde f_s representa a frequência de amostragem do sinal e p_x a abscissa do ponto aproximado pela parábola que marca o período em amostras do n-ésimo *frame*. Com o uso da aproximação parabólica, busca-se eliminar um ponto negativo deste estimador: a baixa resolução.

Os *pitch trackings* dos sinais mostrados nas Figuras 4.2(a) e 4.2(b) a partir do estimador ACF são mostrados na Figura 4.6 juntos com o *pitch tracking* ideal (sinal de referência em formato `MIDI` da Figura 4.3).



(a)



(b)

Figura 4.6: Gráfico de *pitch tracking* gerado pelo estimador ACF tendo como entrada: (a) um sinal de piano (Figura 4.2(a)) e (b) um sinal de solfejo (Figura 4.2(b)). Ambos os sinais são mostrados com o sinal de referência `MIDI` da Figura 4.3.

4.3.2 *Harmonic Product Spectrum* - HPS

Proposto por SCHROEDER em 1968 [16], o algoritmo HPS utiliza a estrutura dos harmônicos presentes no sinal para estimar a frequência fundamental de acordo com a equação abaixo:

$$p = \arg\max_f \prod_{k=1}^{n} |X[k.f]|, \tag{4.9}$$

onde p é o *pitch* estimado, $X[k, f]$ é o espectro decimado por k de um *frame* do sinal e n é o número de harmônicos adotado. Desta forma, estima-se o *pitch* como a frequência f que maximiza o produto das réplicas decimadas de X. Uma abordagem equivalente consiste em estimar o *pitch* como a frequência f que maximiza a soma do logaritmo das réplicas decimadas de X. Esta abordagem, entretanto, não foi utilizada na implementação deste algoritmo neste trabalho.

A idéia deste algoritmo é fazer com que a frequência fundamental se sobressaia em relação às outras frequências. Isso acontece a cada iteração do método ao realizar o produto das amostras de um *frame* no espectro em sua versão original com suas réplicas, pois sendo as parciais harmônicas múltiplos da frequência desejada, estas tendem a ser suprimidas enquanto que a frequência fundamental é destacada. Isto o torna também resistente a ruído aditivo e multiplicativo.

As etapas seguidas na implementação deste estimador foram as seguintes:

1. Para cada *frame*, realiza-se a multiplicação deste por uma janela de *Hamming*, definida segundo a equação abaixo:

$$w(n) = 0{,}54 - 0{,}46\left(\frac{2\pi.n}{N}\right), 0 \leq n \leq N, \tag{4.10}$$

   onde o comprimento da janela é de $(N + 1)$ pontos.

2. Representa-se o *frame* no domínio da frequência através do cálculo da FFT.

3. Geram-se réplicas do espectro do *frame* em questão por meio de decimações deste por 1, 2 e 3. Multiplicam-se, ponto-a-ponto, sua versão original por suas réplicas.

4. Procura-se o pico de maior amplitude no sinal resultante da etapa anterior e realiza-se neste pico uma aproximação por parábolas de 3 pontos. O vértice da parábola aproximada deverá indicar a localização da frequência fundamental ressaltada em relação às demais frequências.

5. Calcula-se a frequência fundamental através da equação abaixo:

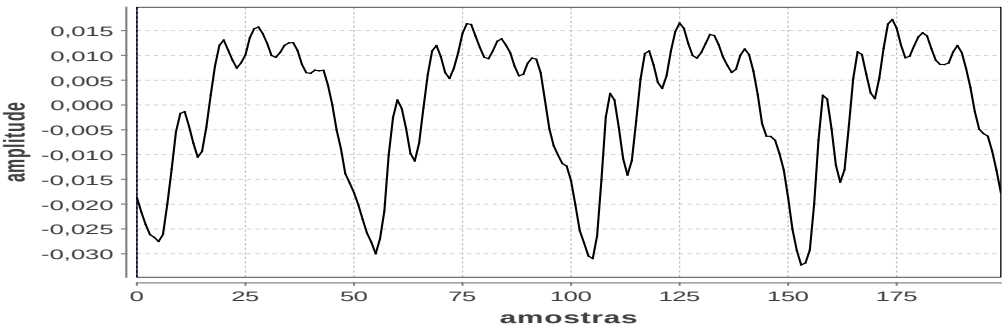
$$f_0(n) = \frac{f_s.P_x}{L}, \tag{4.11}$$

onde, para o n-ésimo *frame*, f_s corresponde à frequência de amostragem do sinal, P_x à abscissa do vértice da parábola aproximada em amostras e L ao comprimento da FFT também em amostras.

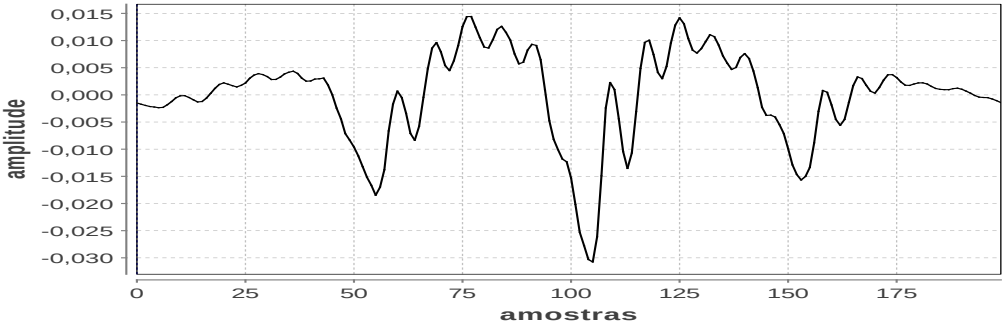
As Figuras 4.7 e 4.8 mostram o gráfico de um *frame*, *frame* multiplicado pela janela de *Hamming*, representação do espectro deste sinal e o sinal resultante do produto do espectro do sinal por suas réplicas, ressaltando a frequência fundamental, respectivamente.

Um problema comum neste algoritmo é a ocorrência de erros de oitava, em grande parte 1 oitava acima da frequência correta. Uma possível solução para este problema é seguir a regra proposta em [17]: Se o segundo maior pico corresponde aproximadamente à metade da frequência do pico escolhido e a relação entre suas amplitudes está acima de um determinado limiar ($0 < \text{limiar} < 1$), então seleciona-se o pico de menor frequência para indicar o *pitch* do *frame* em questão.

Os *pitch trackings* dos sinais mostrados nas Figuras 4.2(a) e 4.2(b) a partir do estimador HPS são mostrados na Figura 4.9 juntos com o *pitch tracking* ideal (sinal referência em formato `MIDI` da Figura 4.3).

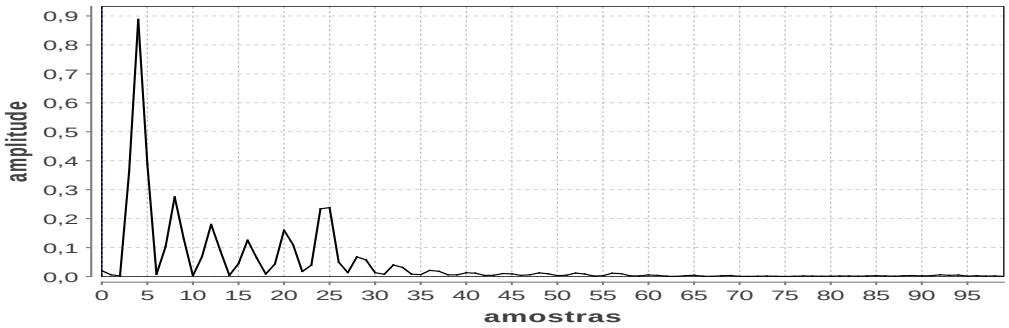


(a)

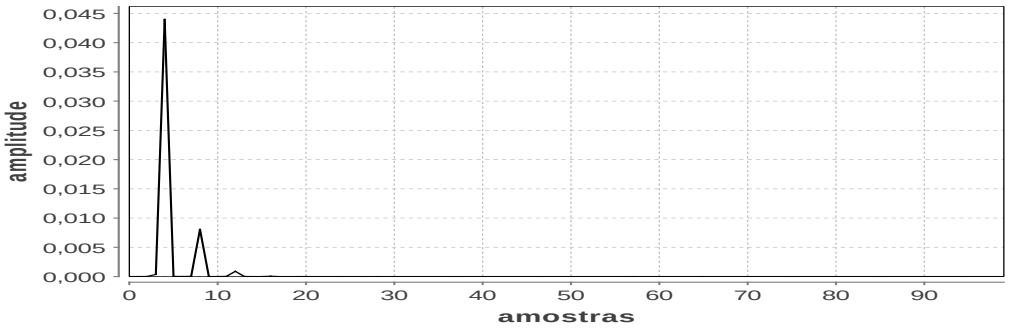


(b)

Figura 4.7: (a) *Frame* de um sinal de solfejo amostrado em 8.000 Hz, (b) *frame* multiplicado por uma janela de *Hamming*.



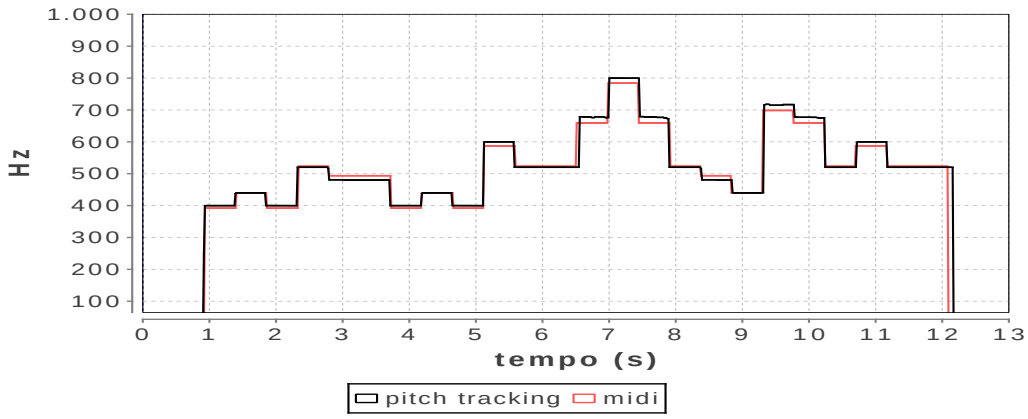
(a)



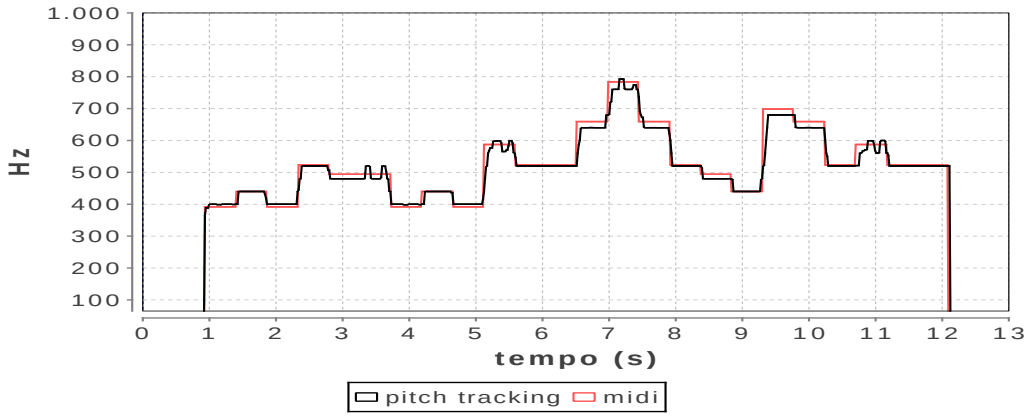
(b)

Figura 4.8: (a) representação no espectro do sinal da Figura 4.7(b), e (b) sinal resultante da multiplicação ponto-a-ponto do espectro do sinal mostrado em (a) por suas réplicas.

A resolução deste algoritmo dependente diretamente do número de amostras da FFT usada para calcular o espectro do sinal, sendo este um fator negativo do estimador.



(a)



(b)

Figura 4.9: Gráfico de *pitch tracking* gerado pelo estimador HPS tendo como entrada: (a) um sinal de piano (Figura 4.2(a)) e (b) um sinal de solfejo (Figura 4.2(b)). Ambos os sinais são mostrados com o sinal de referência `MIDI` da Figura 4.3.

4.3.3 Análise Cepstral - CEPS

Baseado no modelo mais simples de produção de sons de fala pelo ser humano, segundo o qual sopros de ar emitidos pelos pulmões, passando pelas pregas vocais (fonte vocal), ao atravessarem o trato vocal produzem os sinais de voz, como mostra o modelo da Figura 4.10, o algoritmo conhecido por Análise Cepstral, ou simplesmente *Cepstrum*[4], combina os domínios do tempo e da frequência para estimar a frequência fundamental em um sinal monofônico.

[4]Derivado da inversão das 4 primeiras letras de "spectrum".

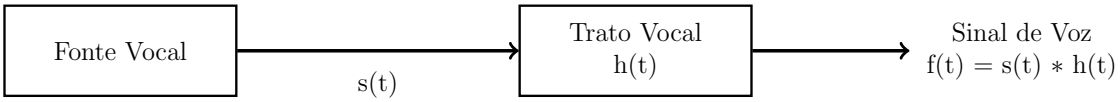


Figura 4.10: Diagrama de blocos do estimador de f_0 CEPS.

Segundo NOLL [18], o efeito do trato vocal é especificado pela resposta ao impulso $h(t)$ e o sinal de voz $f(t)$ pela convolução de $s(t)$ e $h(t)$. Se $S(\omega)$ corresponde ao espectro da fonte vocal e $H(\omega)$ à função de transferência do trato vocal, então, o espectro do sinal de fala corresponde ao produto de $S(\omega)$ por $H(\omega)$, como mostram as equações a seguir:

$$f(t) = s(t) * h(t), \tag{4.12}$$

$$F(\omega) = S(\omega).H(\omega), \tag{4.13}$$

onde $*$ simboliza a convolução.

Sabendo que sinais de fala são quasi-periódicos para sinais de fala vozeados, e que a potência espectral de $|F(\omega)|^2$ consiste de harmônicos espaçados de T^{-1} Hz, onde T representa o período em segundos, a forma mais natural de obter este período na potência espectral é através da Transformada de Fourier do espectro, onde os picos correspondem ao período procurado. A função de autocorrelação do sinal original no tempo é como o espectro de potência espectral é mais comumente conhecido. Esta função de autocorrelação $r(\tau)$ é descrita matematicamente como segue [18]:

$$r(\tau) \equiv \mathfrak{F}^{-1}[|F(\omega)|^2] \tag{4.14}$$

$$= \mathfrak{F}^{-1}[|S(\omega)|^2.|H(\omega)|^2] \tag{4.15}$$

$$= r_s(\tau) * r_h(\tau), \tag{4.16}$$

onde $\mathfrak{F}^{-1}$ representa a Transformada de Fourier inversa e $r_s(\tau)$ e $r_h(\tau)$ são funções de autocorrelação de $s(t)$ e $h(t)$, respectivamente.

Em alguns casos, o resultado desta representação fornece picos largos e em outros múltiplos picos, fazendo com que esta abordagem seja, em geral, não satisfatória para a estimação de *pitch*. Uma solução para isto é fazer com que os efeitos da fonte vocal e do trato vocal sejam facilmente identificados e separados, ou tornados quase independentes [18]. A transformada de Fourier do logaritmo da potência espectral realiza essa separação, através de sua propriedade que relaciona o logaritmo do

produto à soma dos logaritmos:

$$log|F(\omega)|^2 = \log[|S(\omega)|^2.|H(\omega)|^2] \tag{4.17}$$

$$= \log|S(\omega)^2| + \log|H(\omega)|^2. \tag{4.18}$$

A Transformada Inversa de Fourier do logaritmo da potência espectral preserva essa propriedade de adição, tornando os efeitos da fonte vocal e do trato vocal aditivos, representados no domínio do tempo [19]:

$$\mathfrak{F}^{-1}[log|F(\omega)|^2] = \mathfrak{F}^{-1}[log|S(\omega)^2|] + \mathfrak{F}^{-1}[log|H(\omega)^2|]. \tag{4.19}$$

O efeito do trato vocal é produzir uma "ondulação" de baixas frequências no logaritmo do espectro, enquanto que a periodicidade da fonte vocal uma "ondulação" de altas frequências também no logaritmo do espectro.

As etapas seguidas na implementação deste estimador foram, então, as seguintes:

1. Para cada *frame*, realiza-se o produto deste por uma janela de *Hamming*, definida pela Equação (4.10).

2. Representa-se o *frame* no domínio da frequência através do cálculo da FFT.

3. Calcula-se o logaritmo da potência espectral (Figura 4.11).

4. Calcula-se a Transformada Inversa de Fourier do logaritmo da potência espectral do sinal.

5. Por fim, estima-se a frequência fundamental pelo maior pico do sinal da etapa anterior aplicando uma aproximação por parábola de 3 pontos. Assim a f_0 é encontrada pela Equação (4.8) apresentada no estimador ACF.

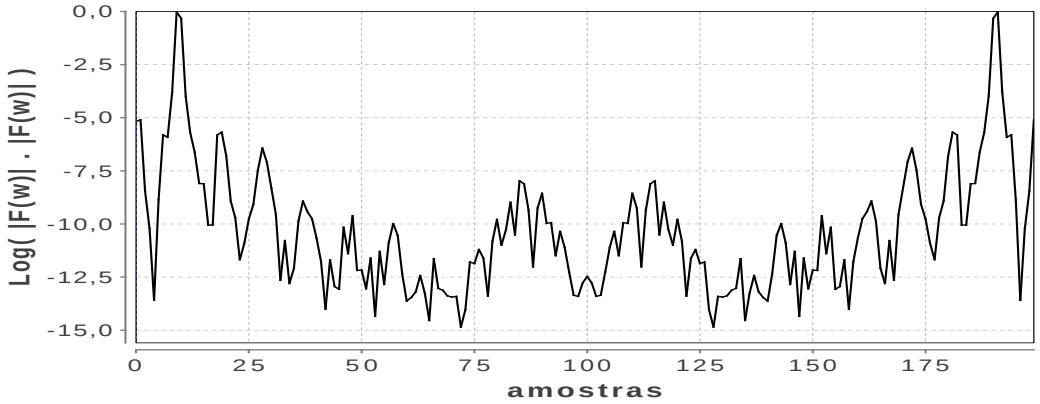
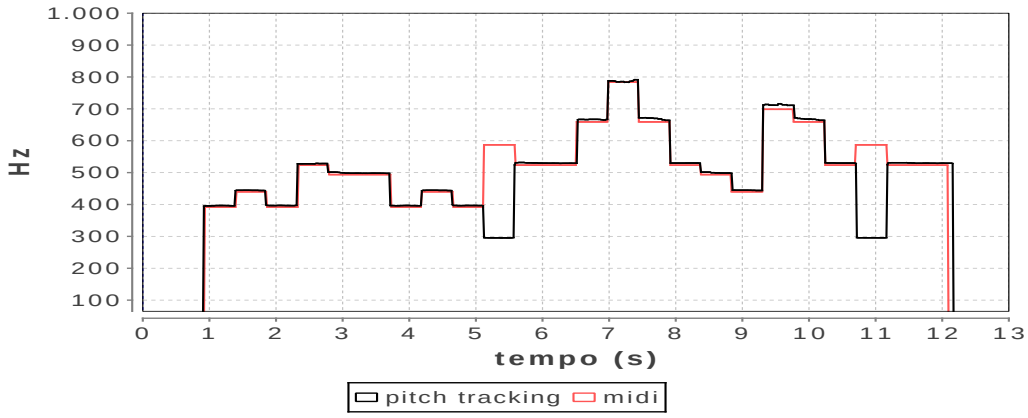
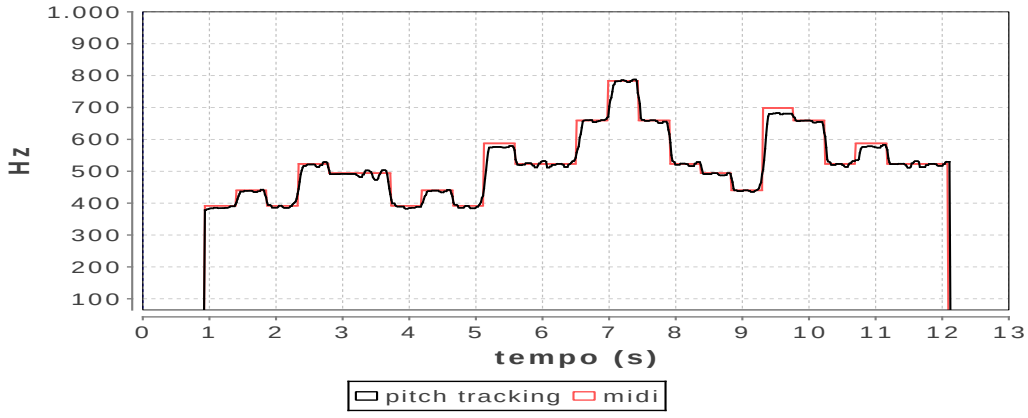


Figura 4.11: Logaritmo da potência espectral de um *frame* extraído do sinal de solfejo 4.2(b).

Os *pitch trackings* dos sinais mostrados nas Figuras 4.2(a) e 4.2(b) a partir do estimador CEPS são mostrados na Figura 4.12 juntos com o *pitch tracking* ideal (sinal referência em formato `MIDI` da Figura 4.3).



(a)



(b)

Figura 4.12: Gráfico de *pitch tracking* gerado pelo estimador CEPS tendo como entrada: (a) um sinal de piano (Figura 4.2(a)) e (b) um sinal de solfejo (Figura 4.2(b)). Ambos os sinais são mostrados com o sinal de referência `MIDI` da Figura 4.3.

4.3.4 YIN

Criado em 2002 por CHEVEIGNÉ e KAWAHARA, este estimador baseia-se no método de autocorrelação ACF [20]. Os passos que compõem o estimador são os seguintes:

1. **Função Diferença - FD**

   Consideremos inicialmente um sinal $x[t]$, discreto e periódico, de período T:

$$x[t] - x[t+T] = 0, \forall t. \tag{4.20}$$

Para o somatório do quadrado das diferenças das amostras espaçadas de T, em uma janela de W amostras de $x[t]$, a equação abaixo também é verdadeira:

$$\sum_{j=t+1}^{t+W} (x[j] - x[j+T])^2 = 0. \tag{4.21}$$

Deste modo, quando T é desconhecido, podemos estimar seu valor pela equação diferença:

$$d(t,\tau) = \sum_{j=t+1}^{t+W} (x[j] - x[j+\tau])^2. \tag{4.22}$$

O período será o menor valor de τ que torna a Equação (4.22) igual a zero.

Devido ao fato de sinais de voz cantada não serem puramente periódicos, a função diferença realizada nesse sinal apresenta diversos vales (ver Figura 4.13).

2. **Função Diferença Normalizada - FDN**

   Uma forma de se obter o período T através da **Função Diferença** em sinais de voz cantada é encontrar o valor mais próximo de zero nesta função. Por outro lado, este valor poderá ser referente a um harmônico e com isso, o período estimado poderia ser um múltiplo do verdadeiro período T. Com a intenção de facilitar a solução deste problema, é proposta a **Função Diferença Normalizada**:

$$d'(t,\tau) = \begin{cases} 1, & \text{se } \tau = 0, \\ \frac{d(t,\tau)}{[(\frac{1}{\tau}) \sum_{j=1}^{\tau} d(t,j)]} & \text{em caso contrário.} \end{cases} \tag{4.23}$$

   A função $d'(t,\tau)$ possui a vantagem de eliminar o limite superior do intervalo de busca e normalizar os dados para o próximo passo do estimador [20].

3. **Limiar Absoluto**

   Observe na Figura 4.13 que a **Função Diferença Normalizada** possui diversas "ondulações". Os pontos da função relevantes para a estimação do período T são aqueles que produzem valores mais próximos de zero. Para facilitar o encontro destes pontos, um **Limiar Absoluto** é empregado.

   O menor valor de τ que resulta em um valor abaixo deste limiar, sendo este um vale, é o período procurado. Caso nenhum seja encontrado, o mínimo global é escolhido.

   Segundo [20], a proporção de potência aperiódica tolerada dentro de um sinal não puramente periódico pode ser uma interpretação do limiar neste caso.

4. **Interpolação Parabólica**

   Quando o período de um sinal é múltiplo do período de amostragem deste, os passos anteriores encontram o período exato. Como não é possível prever/afirmar que isso será verdade sempre, até o passo 3 este estimador encontrará um período com uma margem de erro ocasionada por esse fato. Uma forma de atacar este problema é a **interpolação parabólica**.

   Cada vale encontrado abaixo do limiar passa por um processo chamado *fitting*, que nada mais é do que a aproximação de pontos por uma função, neste caso uma parábola. A abscissa do vértice que tem como ordenada o menor valor entre os vértices equivale ao período T procurado.

   No exemplo mostrado na Figura 4.13, o período do sinal é de $\frac{21{,}1182}{f_s} = 0{,}002639775$ segundo, sendo f_s a frequência de amostragem do sinal em 8.000 Hz. A frequência fundamental equivale, então, a $\frac{1}{0{,}002639775} \approx 378{,}82$ Hz.

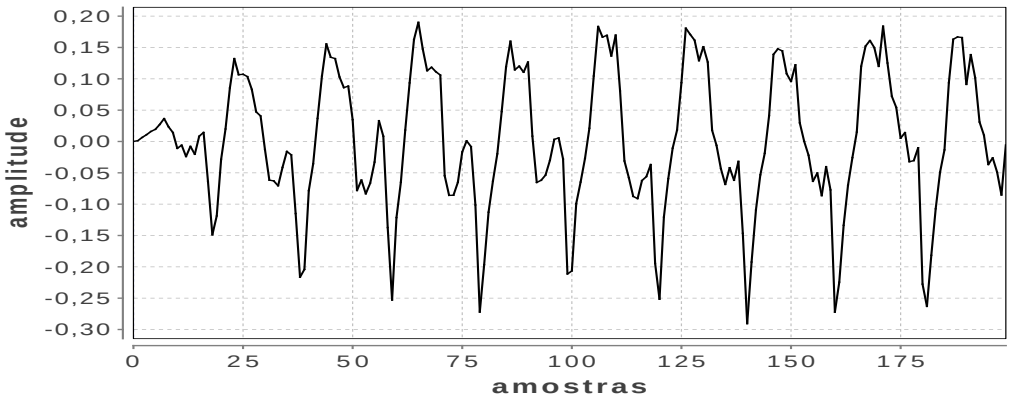
5. **Melhor Estimativa Local**

   Nesta etapa é realizada uma pesquisa, tendo como base cada índice de tempo t, pelo valor mínimo de $d'(\theta, T_\theta)$, estando θ no intervalo $[t - \frac{T_{\max}}{2}, t + \frac{T_{\max}}{2}]$. O período estimado no tempo θ é indicado por T_θ, e $T_{\max}$ é o maior período esperado.

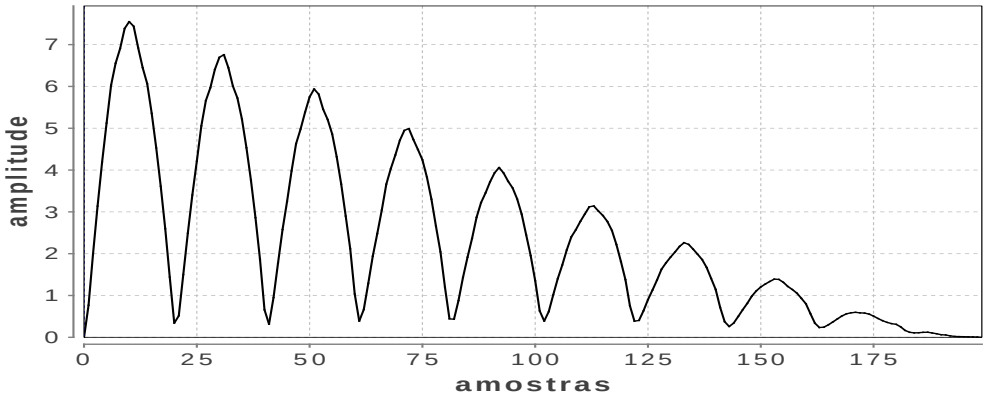
   Após a estimativa inicial, o algoritmo é aplicado novamente para o intervalo citado, de modo a obter a estimativa final do período T.

Apesar de o estimador possuir 5 passos bem definidos, neste trabalho foi desconsiderado o passo 5, devido a este despender muito tempo no processamento e não produzir uma melhora tão significativa na estimação de *pitch*.

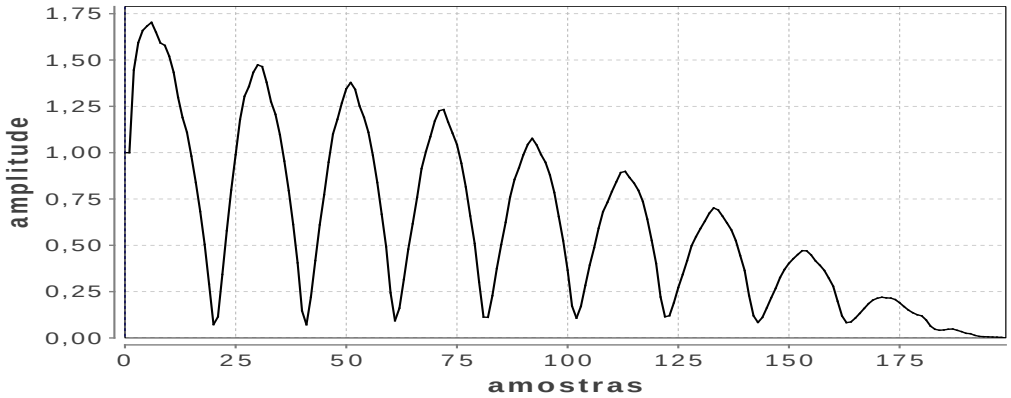
Os *pitch trackings* dos sinais mostrados nas Figuras 4.2(a) e 4.2(b) a partir do estimador YIN são mostrados na Figura 4.14 juntos com o *pitch tracking* ideal (sinal referência em formato `MIDI` da Figura 4.3).



(a)

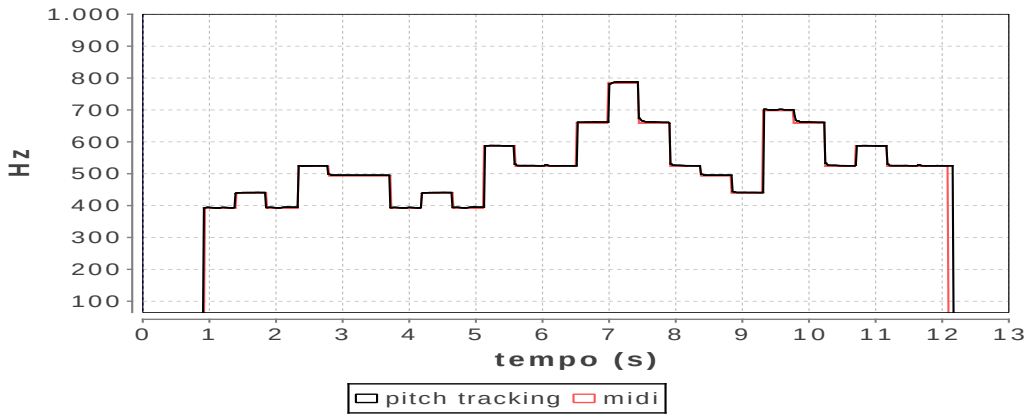


(b)

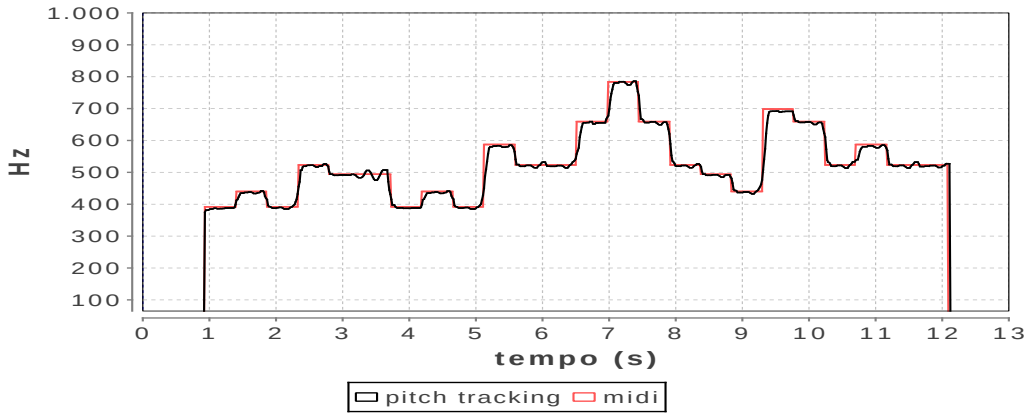


(c)

Figura 4.13: Gráfico de um *frame* gerado a partir de um sinal de solfejo amostrado em 8.000 Hz (a); (b) Função diferença do sinal; (c) Função diferença normalizada.



(a)



(b)

Figura 4.14: Gráfico de *pitch tracking* gerado pelo estimador YIN tendo como entrada: (a) um sinal de piano (Figura 4.2(a)) e (b) um sinal de solfejo (Figura 4.2(b)). Ambos os sinais são mostrados com o sinal referência `MIDI` da Figura 4.3.

4.4 Pós-processamento

A etapa de pós-processamento tem como objetivo realizar certos ajustes no sinal de saída dos estimadores de f_0. Devido à característica ruidosa e/ou a erros impulsivos nos sinais de *pitch tracking*, é comum a utilização de técnicas de suavização nestes sinais com o intuito de eliminar tais comportamentos. Outro ajuste bastante útil é a eliminação de trechos de curta duração, que possam ser considerados impossíveis de serem emitidos pelo ser humano.

O pós-processamento empregado consiste em suavizar o sinal e eliminar alguns trechos cuja duração esteja abaixo de um limiar.

No primeiro caso, a suavização empregada utiliza-se de um filtro digital não-linear conhecido por filtro de mediana, comumente utilizado em processamento de imagens, devido a este filtro apresentar melhores resultados na eliminação de ruído

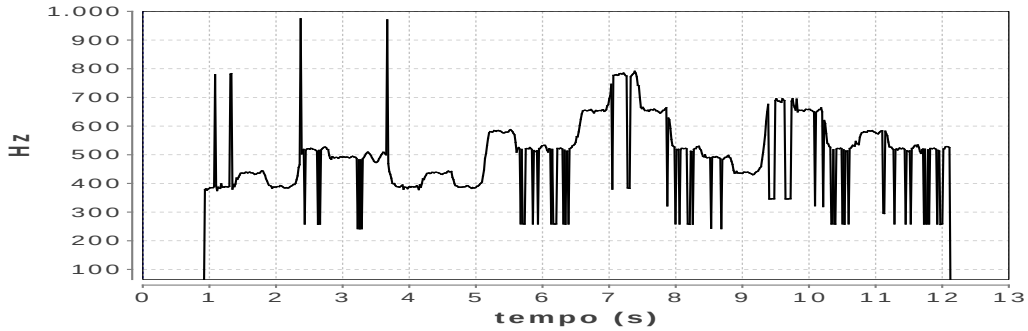
impulsivo sem alterar muito o formato do sinal, quando comparado ao filtro de média. O filtro de mediana realiza para cada ponto do sinal um agrupamento de n elementos em um *buffer*, normalmente sendo n um número ímpar, e após a ordenação dos elementos, sendo $n-1$ vizinhos do ponto em questão, atribui-se a este ponto a mediana deste conjunto de elementos do *buffer*. O número de elementos escolhido para essa suavização foi de 7 pontos e sua escolha se deu via testes empíricos nos sinais de *pitch tracking* gerados pelos algoritmos implementados nesta seção.

No segundo caso, são buscados trechos cuja duração esteja abaixo de 150 ms com a finalidade de eliminá-los do sinal. Para isso, calcula-se a derivada do sinal, e para os pares de amostras cuja amplitude em módulo esteja acima de 100 Hz, verifica-se a duração em segundos do intervalo compreendido pelo par de amostras. No caso de estar abaixo do limiar (150 ms), as amostras limitadas pelo par têm sua amplitude substituída pela da amostra que antecede o trecho em questão.

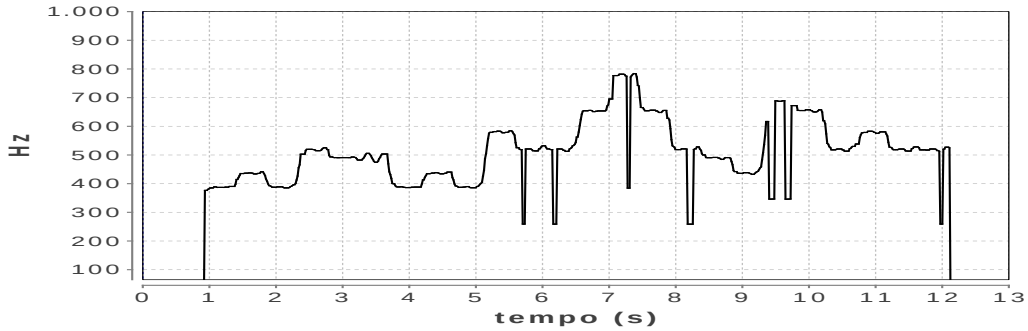
A Figura 4.15 mostra um sinal de *pitch tracking* de solfejo sem qualquer ajuste, com o ajuste pelo filtro de mediana, com a derivada deste sinal filtrado e com o ajuste pela duração de trechos mais curtos que 150 ms.

Uma consideração extra empregada foi de certificar-se que o *pitch tracking* gerado estivesse dentro da faixa em *Hertz* desejada (65 a 1.000 Hz) obedecendo a seguinte regra:

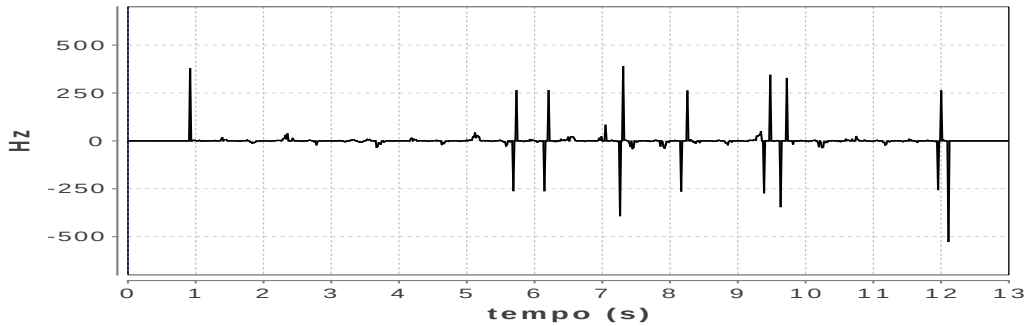
- *Pitch* acima de 1.000 Hz: Divide-se a amplitude da amostra por 2 até que o valor esteja abaixo de 1.000 Hz. Desta forma, o *pitch* é diminuído de oitava.
- *Pitch* abaixo de 65 Hz: Zera-se a amplitude da amostra.



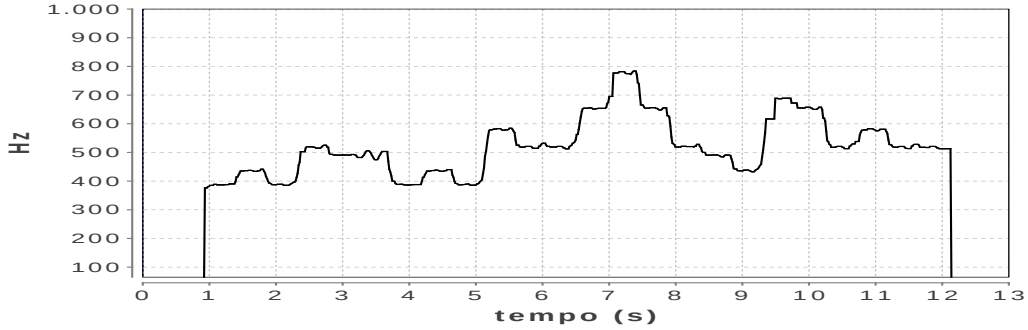
(a)



(b)



(c)



(d)

Figura 4.15: (a) Sinal de solfejo gerado pelo estimador ACF, (b) sinal com suavização pelo filtro de mediana, (c) Derivada do sinal suavizado pela mediana e (d) sinal com ajuste pela duração de trechos mais curtos que 150 ms.

4.5 Avaliação

De modo a permitir a escolha de um dos algoritmos para ser utilizado como estimador de f_0 na aplicação desenvolvida nesta dissertação, avaliou-se o desempenho dos mesmos de acordo com algumas das classes apresentadas por KOTNIK *et al.* [21]:

***Erro Relativo Médio* (ERM):** Apresenta o erro relativo médio em porcentagem, ou seja: (abs(pitchEstimado − pitchReferência)/pitchReferência).

***Gross Error High* (GEH):** Apresenta a porcentagem dos segmentos de voz cuja estimação de *pitch* esteja acima do valor referência em 20% deste, ou seja, (pitchEstimado > (1,2).pitchReferência).

***Gross Error Low* (GEL):** Apresenta a porcentagem dos segmentos de voz cuja estimação de *pitch* esteja abaixo do valor referência em 20% deste, ou seja, (pitchEstimado < (0,8).pitchReferência).

***Voiced Error* (VE):** Apresenta a porcentagem dos segmentos de voz erroneamente classificados como *não-vozeados*, ou seja, (pitchEstimado = 0,0 e pitchReferência > 0,0).

***Unvoiced Error* (UVE):** Apresenta a porcentagem dos segmentos *não-vozeados* erroneamente classificados como *vozeados*, ou seja, (pitchEstimado > 0,0 e pitchReferência = 0,0).

Gerou-se uma base de dados para a realização da avaliação dos algoritmos tendo como foco as partituras mostradas na Figura 4.16. As partituras foram criadas pensando nas seguintes condições de avaliação dos algoritmos:

Faixa de *pitch*: Utilizar como entrada para os algoritmos um sinal que abranja todo o intervalo de notas desejadas. Esse intervalo desejado restringe-se a C2-B5, ou de 65,4064 a 987,767 Hz, respectivamente;

Pausas: Utilizar um sinal que possua pausas de durações variáveis como entrada para os algoritmos. Essas pausas devem ser reconhecidas pelos algoritmos e estes devem indicá-las por 0,00 Hz;

Vibrato: Utilizar como entrada para os algoritmos um sinal que possua vibrato. Sendo este um efeito que provoca no sinal "ondulações" — vistas no domínio da frequência — é importante que os algoritmos também consigam lidar com tal característica no *pitch tracking*;

"Saltos" de *pitch*: Utilizar como entrada para os algoritmos um sinal que possua "saltos" grandes de *pitch*. Neste contexto, o "salto" refere-se à diferença em Hz entre notas consecutivas. Uma diferença de 1 oitava (ou $\frac{\text{nota}_i(Hz)}{\text{nota}_{i-1}(Hz)} = 2$) é considerada uma diferença grande e, consequentemente, um "salto" grande de *pitch* entre notas consecutivas.

Para cada partitura foram gerados sinais em formato `WAV` utilizando-se um *software* de síntese de voz.

O *software* de síntese de voz adotado foi o **Vocaloid**, desenvolvido pela Universidade de Pompeu Fabra e a empresa Yamaha, que usa tecnologia de síntese de voz a partir de amostras gravadas por atores e cantores. É simples a composição dos sinais de áudio através dele, pois o mesmo permite que o usuário crie a melodia de forma visual, bem como importar um arquivo `MIDI` para atuar como a melodia. Sua utilização fez-se útil, uma vez que os sinais nele gerados possuem uma referência de comparação: a partitura ou o arquivo `MIDI`.

Os sinais de áudio produzidos no *software* **Vocaloid** possuem as seguintes características configuradas:

- A frequência de amostragem foi de 48.000 amostras por segundo;
- Os sinais de solfejo foram gerados utilizando apenas a sílaba: "lá";
- Todas as notas possuem vibrato de duração equivalente a 66% da duração total da nota, localizado na parte final de cada nota;
- Foram escolhidas 3 vozes para compor a base de dados: Hatsune Miku (língua japonesa, sexo feminino), Kaai Yuki (língua japonesa, sexo feminino) e Hiyama Kiyoteru (língua japonesa, sexo masculino). As 3 vozes apresentaram boa semelhança com a pronúncia brasileira ao solfejar a sílaba "lá", mesmo sendo todas de língua japonesa. Entretanto, a pronúncia do solfejo algumas vezes assemelha-se com "rá", o que não apresenta qualquer problema para os algoritmos de estimação de f_0.

As seções **Sinais MIDI importados no sintetizador de voz Vocaloid** e **Sinais gerados pelo sintetizador de voz Vocaloid em formato de onda** mostram as Figuras dos sinais `MIDI` de partituras definidas na Figura 4.16 em forma de *piano roll* no *software* **Vocaloid** e em forma de onda, respectivamente.

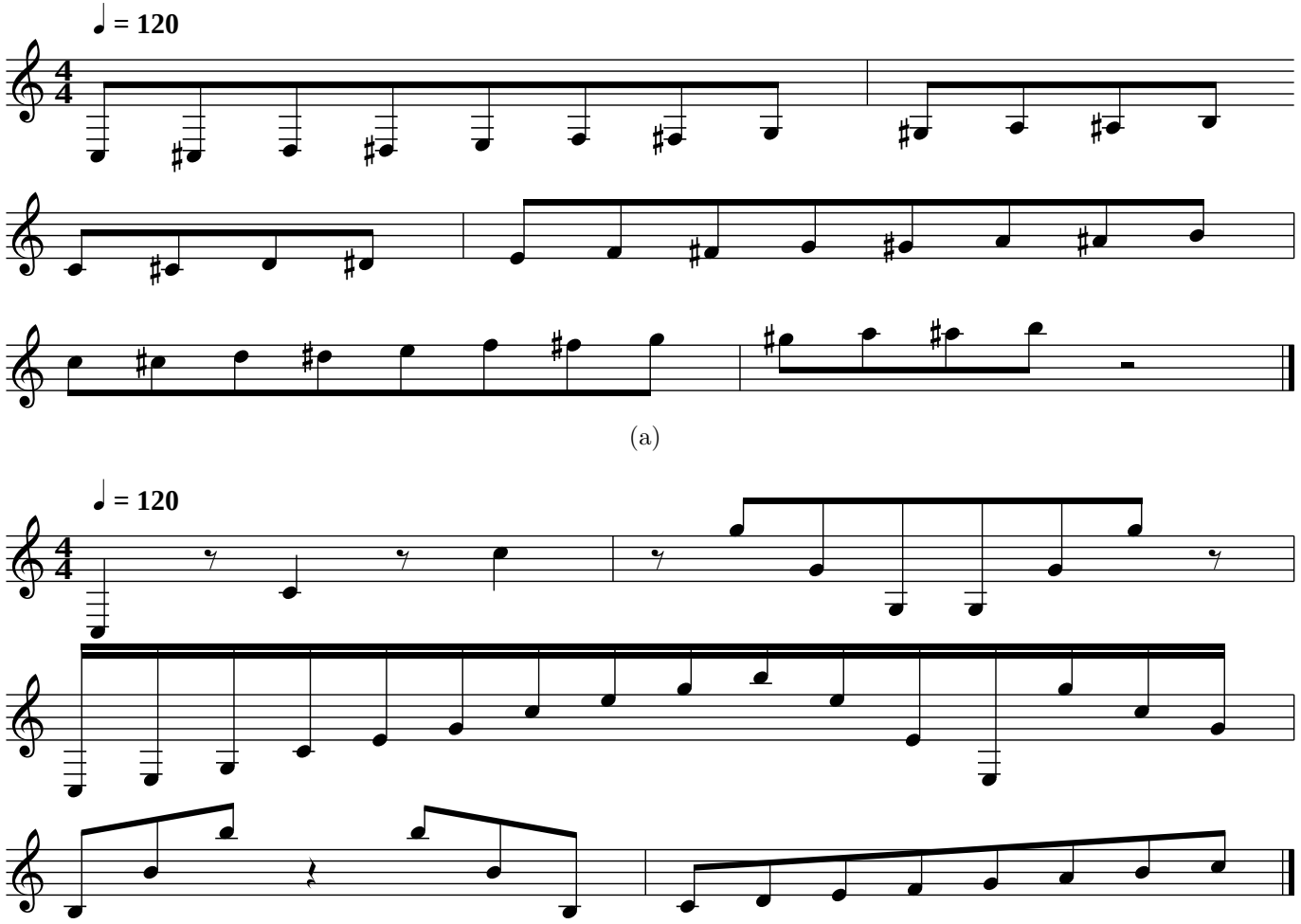


Figura 4.16: Partituras criadas para: (a) sinal 1, compreendendo o intervalo C2-B4 e (b) sinal 2, possui "saltos" de *pitch* de 1 oitava no máximo e notas e pausas de durações diversas.

Sinais `MIDI` importados no sintetizador de voz Vocaloid.

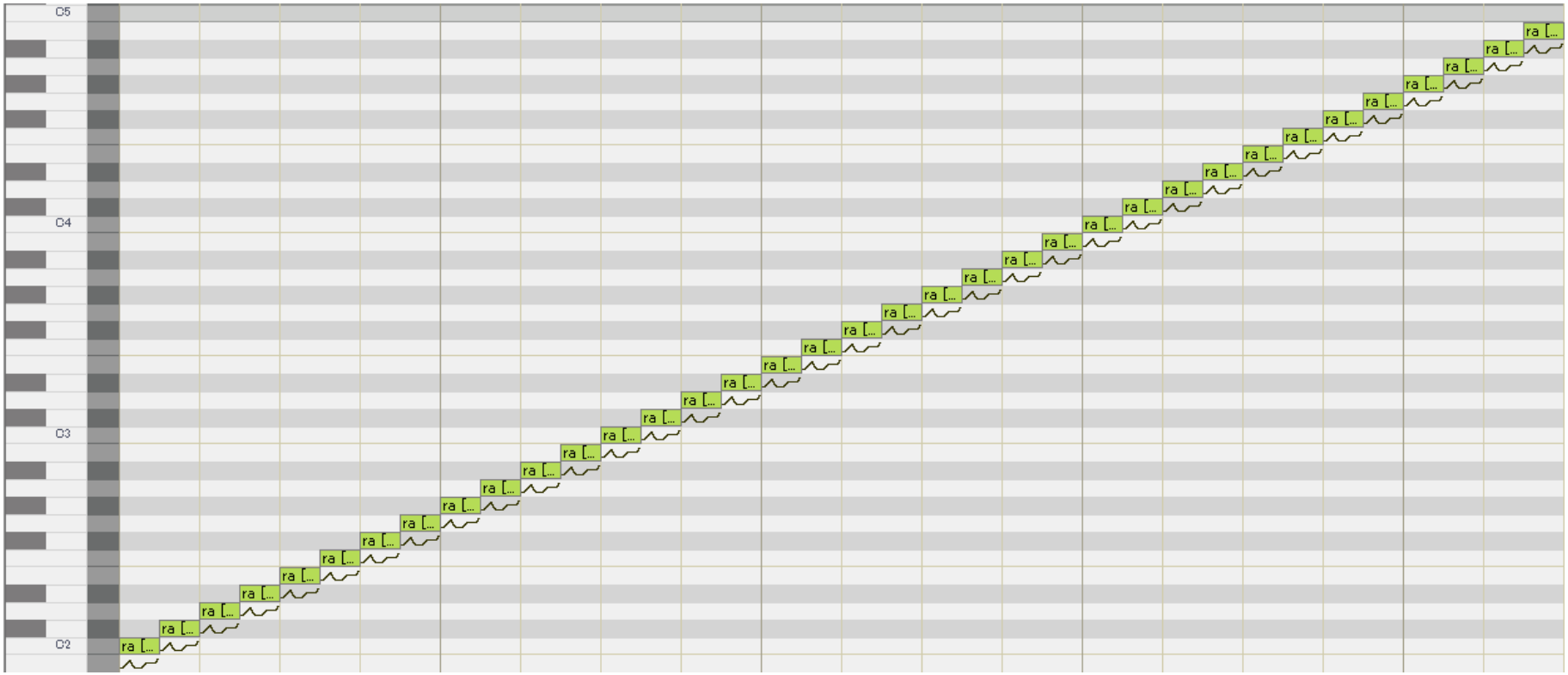


Figura 4.17: Sinal 1, referente à Figura 4.16(a).

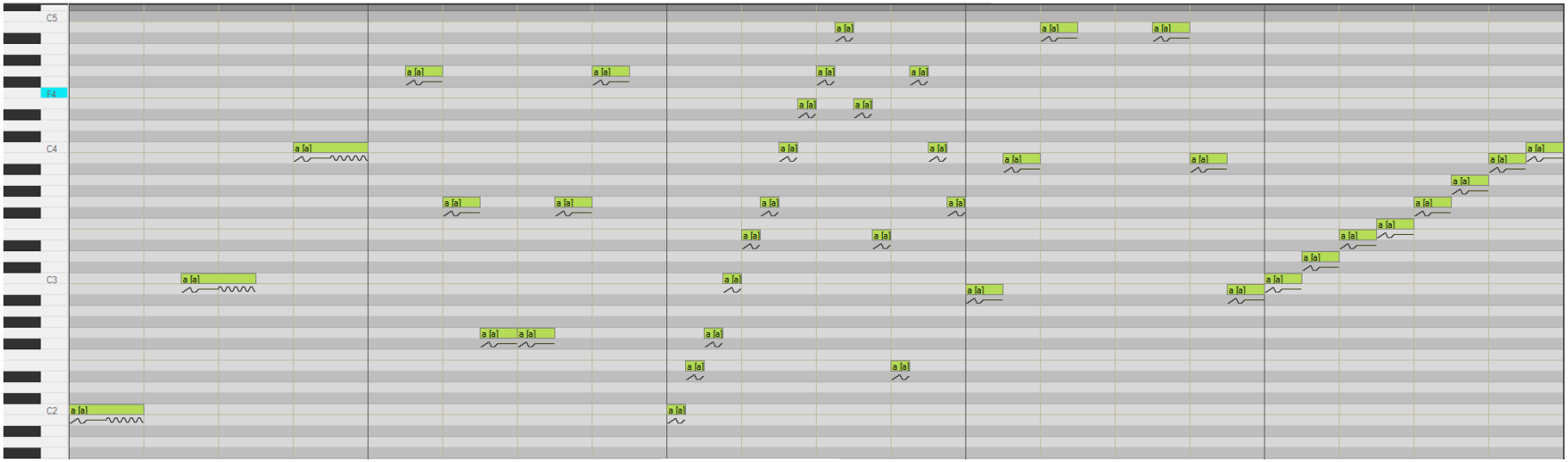
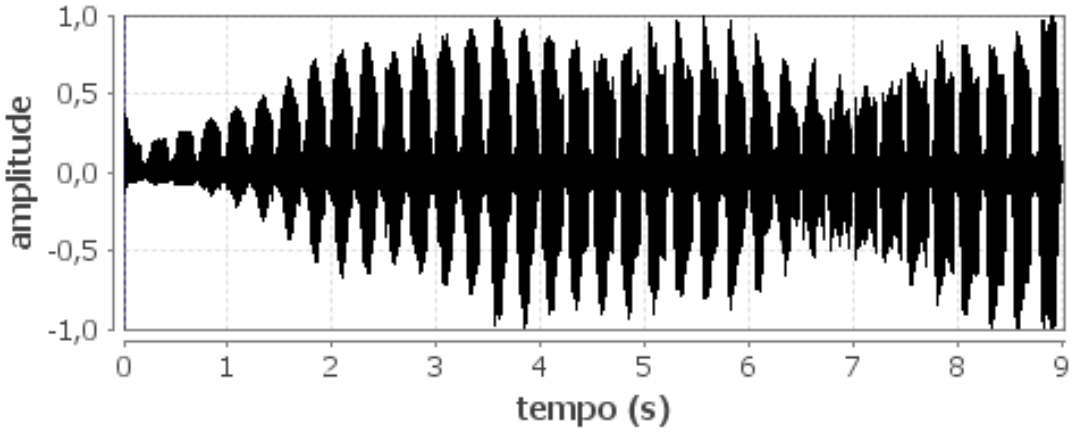
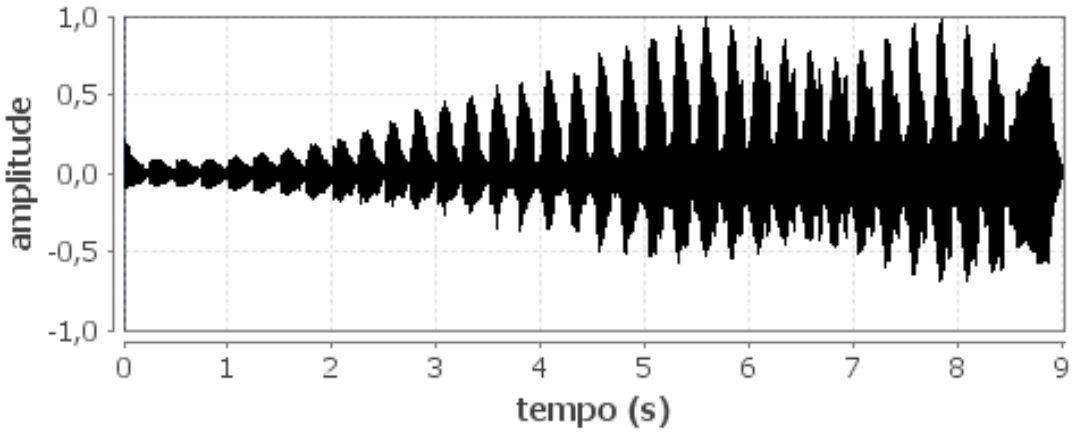


Figura 4.18: Sinal 2, referente à Figura 4.16(b).

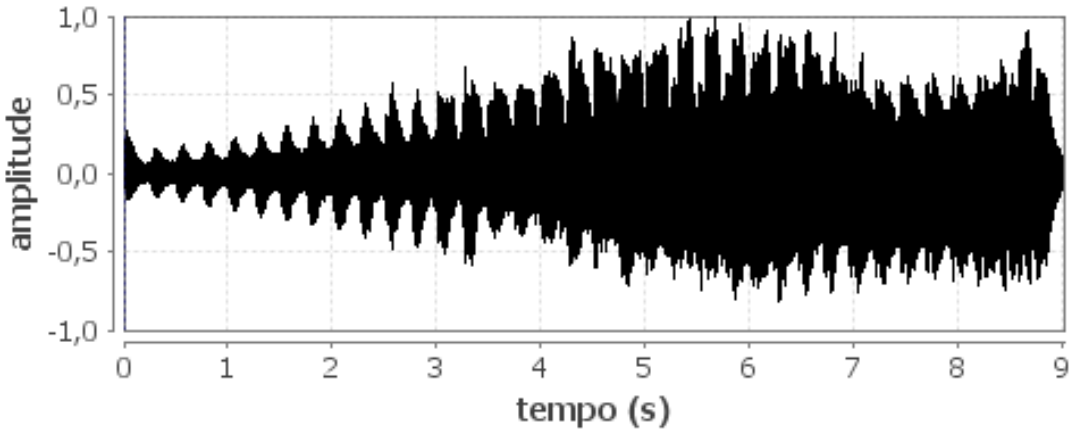
Sinais gerados pelo sintetizador de voz Vocaloid em formato de onda.



(a) Voz de Hiyama Kiyoteru

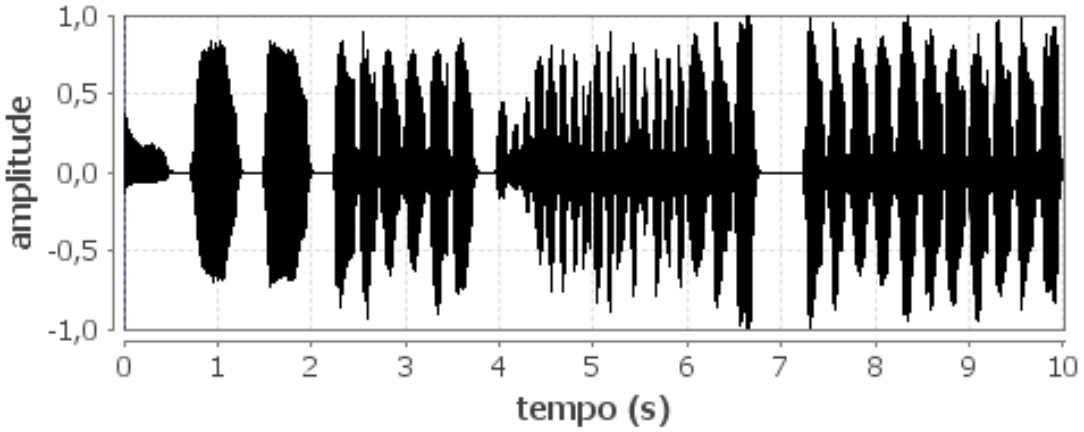


(b) Voz de Hatsune Miku

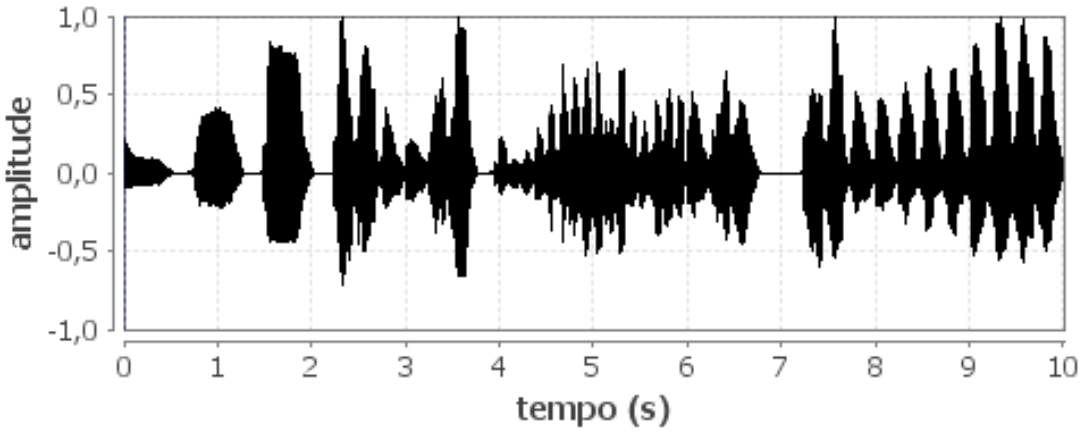


(c) Voz de Kaai Yuki

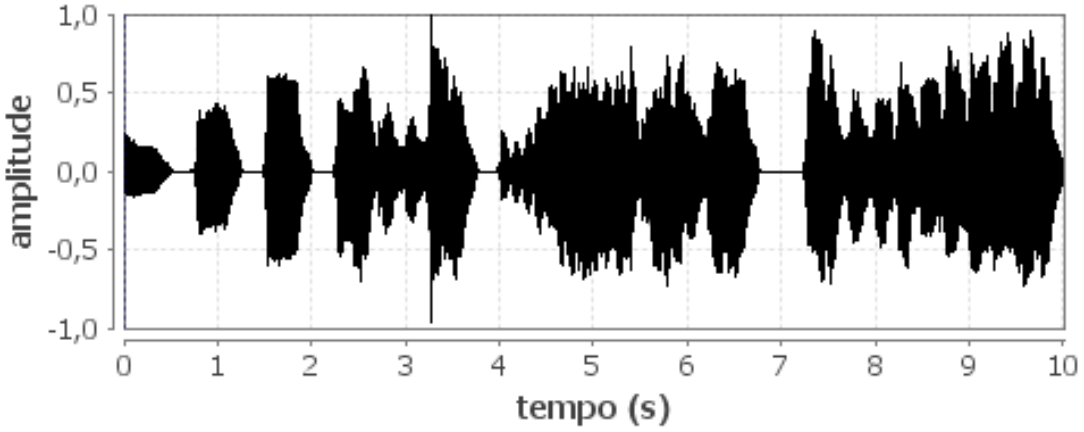
Figura 4.19: Sinal 1, referente à Figura 4.17.



(a) Voz de Hiyama Kiyoteru



(b) Voz de Hatsune Miku



(c) Voz de Kaai Yuki

Figura 4.20: Sinal 2, referente à Figura 4.18.

Aqui, cabe uma pequena ressalva acerca dos testes realizados. Seria possível gerar arquivos de áudio de referência cuja variação de *pitch* seguisse quase estritamente aquela determinada pelos arquivos `MIDI`, e sobre eles realizar a avaliação de desempenho dos estimadores de f_0, cuja meta seria, então, erro nulo. Ao sintetizarmos sinais de voz para realizarmos os testes, buscamos nos aproximar mais das condições em que os algoritmos sob análise deverão operar. Mantivemos, contudo, as medidas de erro calculadas contra os valores determinados pelo `MIDI`. Naturalmente, os erros encontrados englobarão tanto a inexatidão dos estimadores em si mesmos quanto das próprias emissões vocais. Não se pode esperar, portanto, erro nulo; contudo, menores erros ainda indicam melhor desempenho para a aplicação-destino.

4.5.1 Resultados

Apresentam-se os resultados provenientes da avaliação dos algoritmos de estimação de f_0 a partir dos sinais mostrados na seção 4.5, gerados pelo *software* sintetizador de voz **Vocaloid** utilizando 3 vozes de idioma japonês, tendo como referência as partituras da Figura 4.16.

Os resultados estão divididos em duas seções, a primeira em tabelas e a segunda em gráficos, ambas fornecendo dados tanto numéricos quanto visuais do desempenho dos algoritmos. As considerações levantadas a partir destes resultados são comentadas a seguir:

- **Sinal 1, voz de Hiyama Kiyoteru**: Segundo a Tabela 4.1, os algoritmos CEPS e YIN apresentam comportamentos próximos entre si. Ao observar a Figura 4.21, nota-se que o algoritmo ACF, embora erre na estimação logo após o segundo de número 6, aproxima-se dos algoritmos mencionados. Já o algoritmo HPS, apresentou inúmeros erros de estimação, em especial na região de notas mais agudas, localizada a partir do segundo de número 7. Todos os algoritmos não estimaram corretamente o *pitch* referente às 3 notas mais graves do sinal.

- **Sinal 1, voz de Hatsune Miku**: Embora o algoritmo CEPS apresente os valores mais baixos para as classes **GEH** e **UVE**, segundo a Tabela 4.2, o algoritmo YIN mostrou maior fidelidade ao sinal referência, atingindo a menor porcentagem de erro relativo médio (**ERM**). Conforme pode ser visto na Figura 4.22, o algoritmo ACF não consegue estimar o *pitch* para as 5 notas mais graves do sinal e erra na estimação da penúltima nota mais aguda. Já o algoritmo HPS, não estima corretamente o *pitch* das primeiras 7 notas do sinal, assim como algumas das notas mais agudas, situadas a partir do segundo de número 7.

- **Sinal 1, voz de Kaai Yuki**: O algoritmo YIN obteve melhor desempenho

que os demais algoritmos, ao atingir a menor porcentagem de erros para as classes **ERM**, **GEH** e **UVE**, e nas demais classes desempenho igual aos outros algoritmos (vide Tabela 4.3). Na Figura 4.23 verifica-se o quão próximo a estimação deste algoritmo foi fiel ao sinal-referência.

- **Sinal 2, voz de Hiyama Kiyoteru**: Os algoritmos CEPS e YIN apresentaram comportamentos próximos entre si e os menores erros alcançados, como se vê tanto na Tabela 4.4 quanto na Figura 4.24. O algoritmo HPS não conseguiu estimar corretamente o *pitch* das notas mais agudas do sinal.

- **Sinal 2, voz de Hatsune Miku**: O algoritmo YIN obteve menos da metade dos erros da classe **ERM** quando comparado ao algoritmo CEPS e também os menores erros para as classes **GEH** e **GEL**, segundo a Tabela 4.5. O algoritmo HPS não estimou corretamente a altura das notas mais agudas do sinal. Já o algoritmo ACF mostrou-se de comportamento mediano quando comparado aos outros estimadores. Todos os algoritmos não estimaram o *pitch* de duas das notas mais graves do sinal, localizadas a partir do segundo de número 4.

- **Sinal 2, voz de Kaai Yuki**: Segundo a Tabela 4.6, o Algoritmo YIN obteve dois dos melhores resultados nas categorias **ERM** e **GEH**, e de acordo com a Figura 4.26 o melhor desempenho em nível de fidelidade ao sinal referência. O algoritmo HPS não estimou corretamente todas as notas de altura superior a 500 Hz, e o ACF aproximou-se do desempenho do estimador CEPS.

- **Sinais 1 e 2**: Observando as categorias avaliadas, em especial a **ERM**, percebe-se que os valores apresentados referentes ao sinal 2 são de maior amplitude que os do sinal 1. Isso remete concluir que o sinal 2 é mais complexo que o sinal 1, uma vez que os sinais-referência são advindos dos arquivos `MIDI`, o que condiz com a estrutura dos sinais criados para o propósito da avaliação neste capítulo.

Tabelas de avaliação de desempenho de algoritmos de estimação de f_0:

Tabela 4.1: Tabela comparativa: Sinal 1, voz de Kiyoteru.

KIYOTERU (Sinal 1)					
Algoritmo	**ERM(%)**	**GEH(%)**	**GEL(%)**	**VE(%)**	**UVE(%)**
ACF	4,1965	10,1868	0,1698	0,1698	7,9796
CEPS	1,9030	8,1494	0,1698	0,1698	7,9796
HPS	27,0459	27,6740	9,3379	0,3396	7,9796
YIN	1,5795	7,9796	0,1698	0,1698	7,9796

Tabela 4.2: Tabela comparativa: Sinal 1, voz de Miku.

MIKU (Sinal 1)					
Algoritmo	**ERM(%)**	**GEH(%)**	**GEL(%)**	**VE(%)**	**UVE(%)**
ACF	6,0116	15,4499	0,1698	0,1698	13,0730
CEPS	2,8935	11,7148	0,1698	0,1698	10,3565
HPS	19,3179	30,3905	0,1698	0,1698	14,4312
YIN	1,3221	13,0730	0,1698	0,1698	13,0730

Tabela 4.3: Tabela comparativa: Sinal 1, voz de Yuki.

YUKI (Sinal 1)					
Algoritmo	**ERM(%)**	**GEH(%)**	**GEL(%)**	**VE(%)**	**UVE(%)**
ACF	5,8053	5,7725	0,3396	0,3396	3,9049
CEPS	2,6111	2,7165	0,3396	0,3396	2,0374
HPS	42,1857	46,5195	0,3396	0,3396	7,1307
YIN	1,6652	0,0000	0,3396	0,3396	0,0000

Tabela 4.4: Tabela comparativa: Sinal 2, voz de Kiyoteru.

KIYOTERU (Sinal 2)					
Algoritmo	**ERM(%)**	**GEH(%)**	**GEL(%)**	**VE(%)**	**UVE(%)**
ACF	16,0131	18,4733	1,6794	0,9160	8,0916
CEPS	5,6180	12,3664	1,2214	0,9160	8,2443
HPS	20,2589	26,5649	5,6489	1,0687	8,5496
YIN	5,7999	11,9084	3,3588	0,7634	8,2443

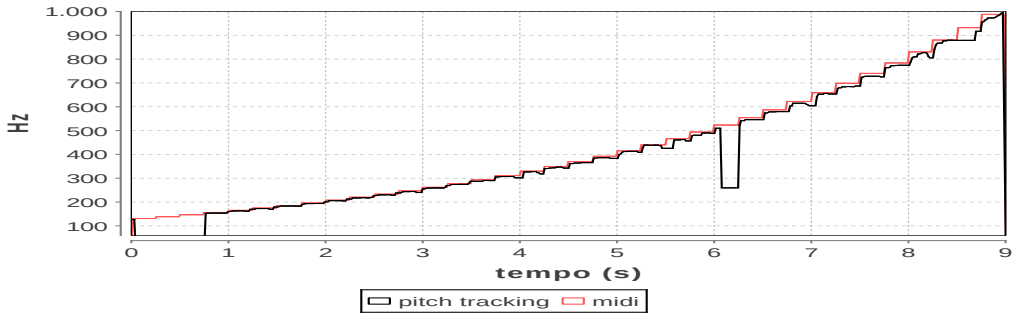
Tabela 4.5: Tabela comparativa: Sinal 2, voz de Miku.

MIKU (Sinal 2)					
Algoritmo	**ERM(%)**	**GEH(%)**	**GEL(%)**	**VE(%)**	**UVE(%)**
ACF	15,5473	18,9313	1,3740	0,6107	7,9389
CEPS	9,2470	12,5191	1,8321	0,9160	7,0229
HPS	30,3338	30,0763	7,0229	0,9160	7,3282
YIN	4,5129	10,5344	1,6794	0,9160	7,4809

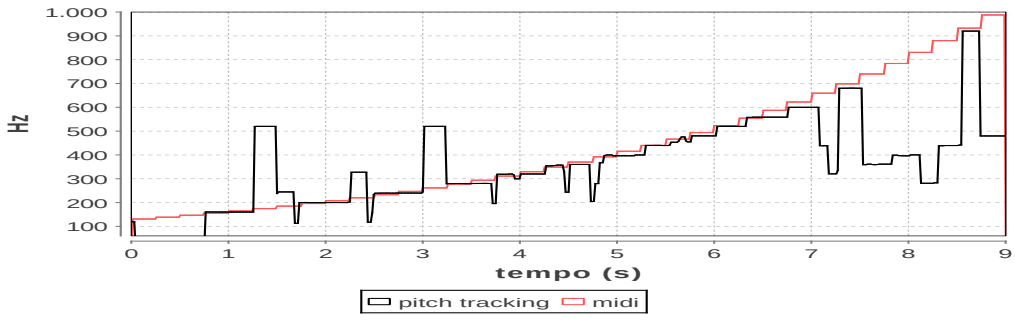
Tabela 4.6: Tabela comparativa: Sinal 2, voz de Yuki.

YUKI (Sinal 2)					
Algoritmo	**ERM(%)**	**GEH(%)**	**GEL(%)**	**VE(%)**	**UVE(%)**
ACF	10,0546	12,5191	1,2214	0,3053	2,7481
CEPS	6,7358	8,7023	1,3740	0,4580	2,1374
HPS	38,6046	42,5954	0,6107	0,4580	8,5496
YIN	5,1884	6,4122	1,5267	0,4580	2,1374

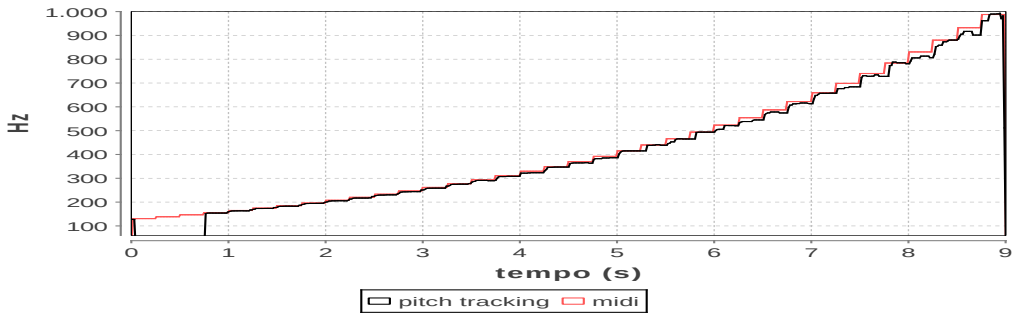
Gráficos de avaliação de desempenho de algoritmos de estimação de f_0:



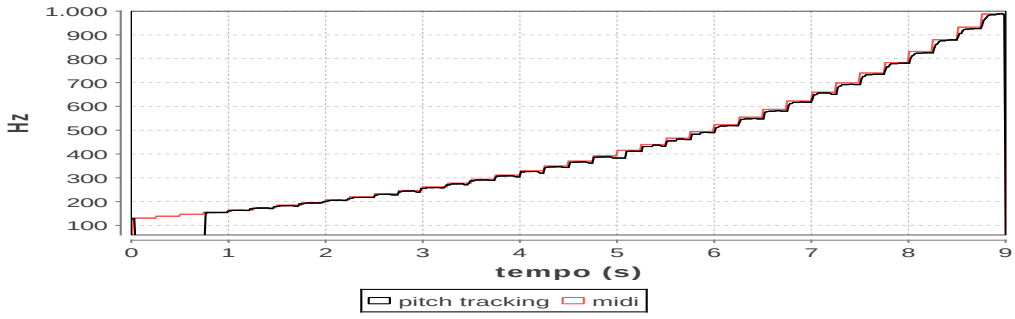
(a)



(b)

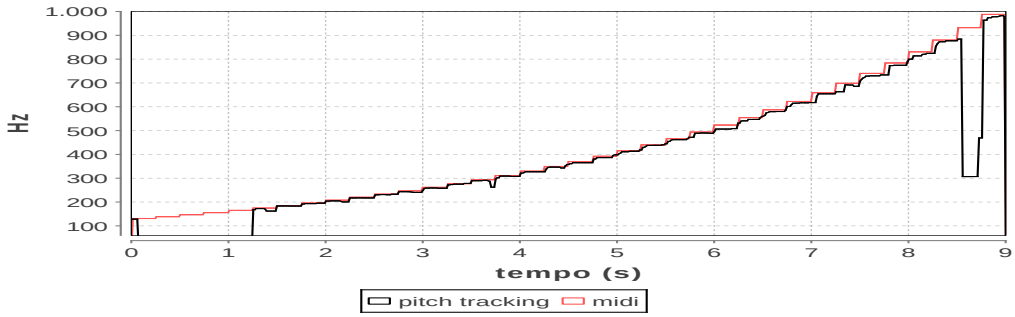


(c)

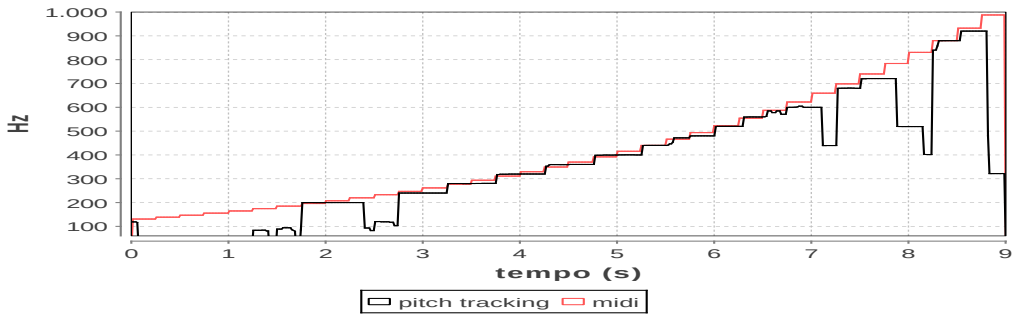


(d)

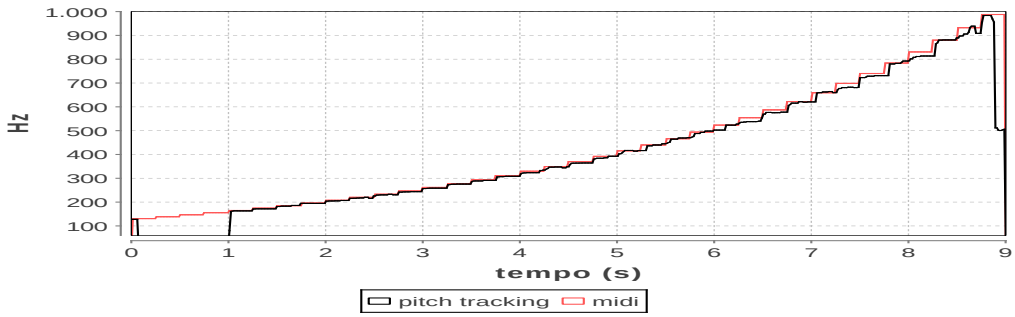
Figura 4.21: Voz Hiyama Kiyoteru sinal 1, *pitch tracking* de saída dos algoritmos e referência `MIDI`: a) ACF, b) HPS, c) CEPS e d) YIN.



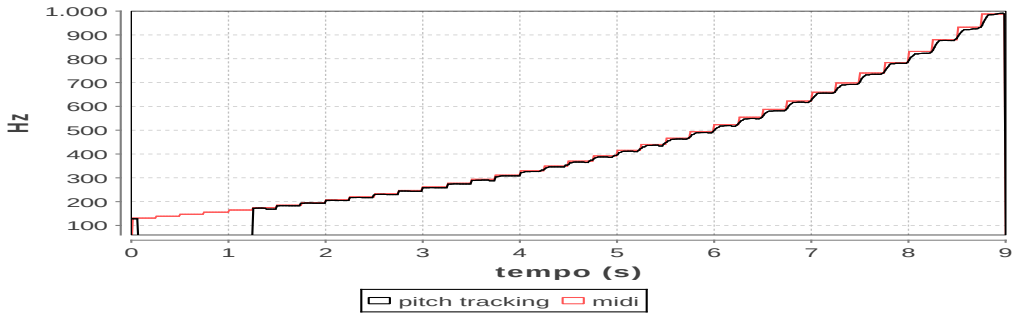
(a)



(b)

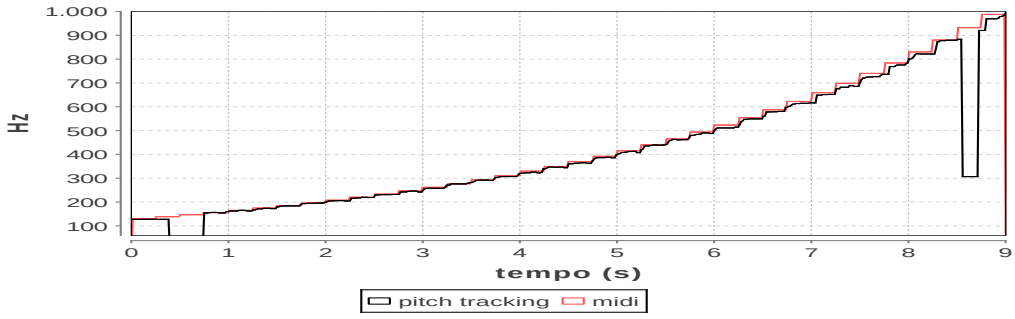


(c)

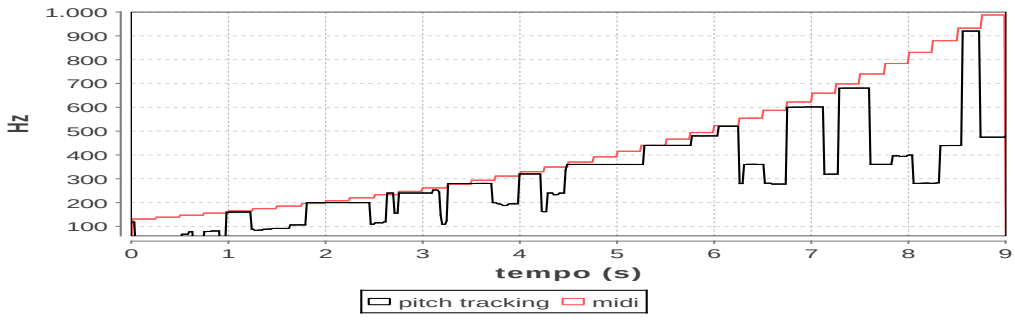


(d)

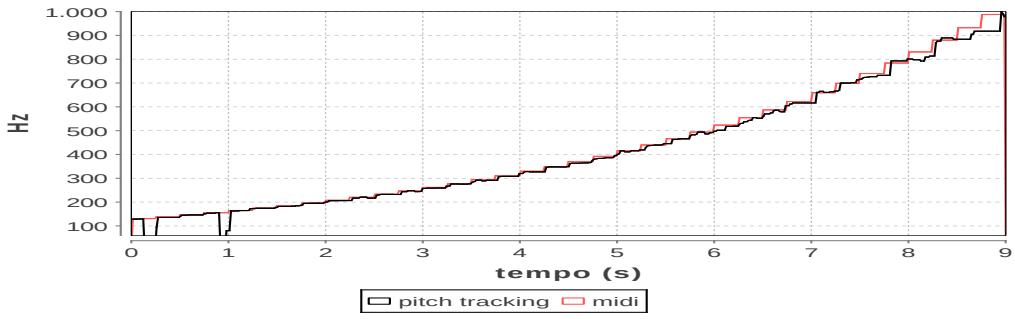
Figura 4.22: Voz Hatsune Miku sinal 1, *pitch tracking* de saída dos algoritmos e referência `MIDI`: a) ACF, b) HPS, c) CEPS e d) YIN.



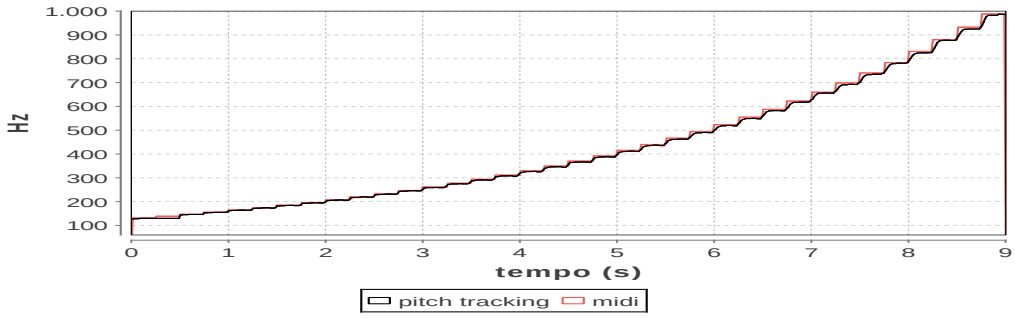
(a)



(b)

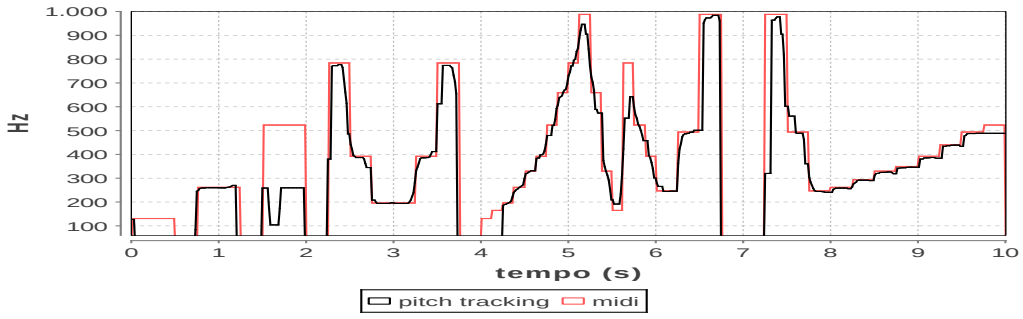


(c)

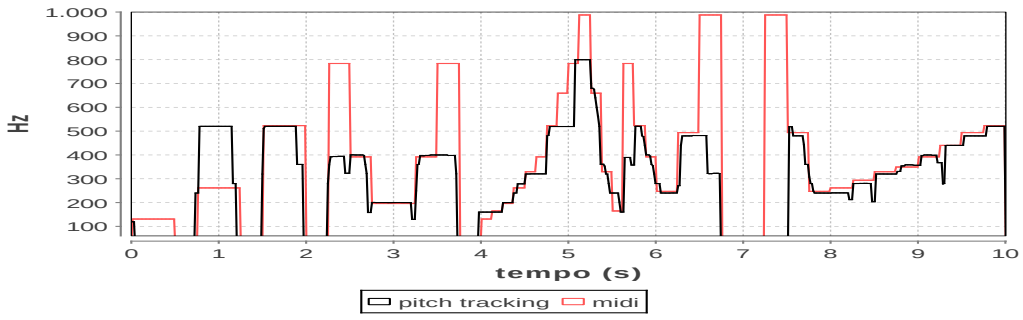


(d)

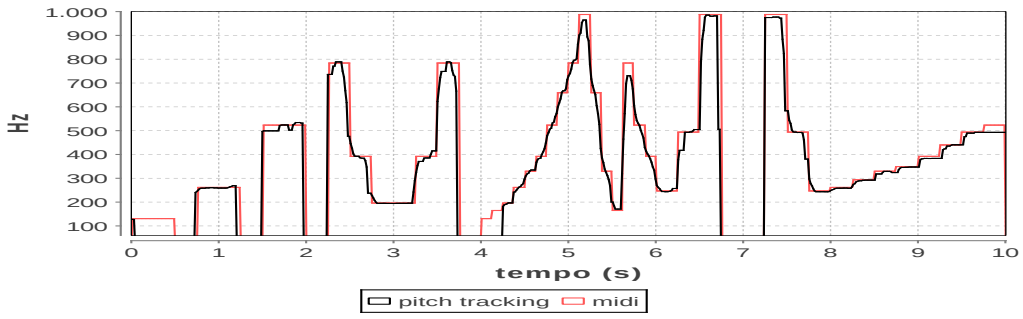
Figura 4.23: Voz Kaai Yuki sinal 1, *pitch tracking* de saída dos algoritmos e referência `MIDI`: a) ACF, b) HPS, c) CEPS e d) YIN.



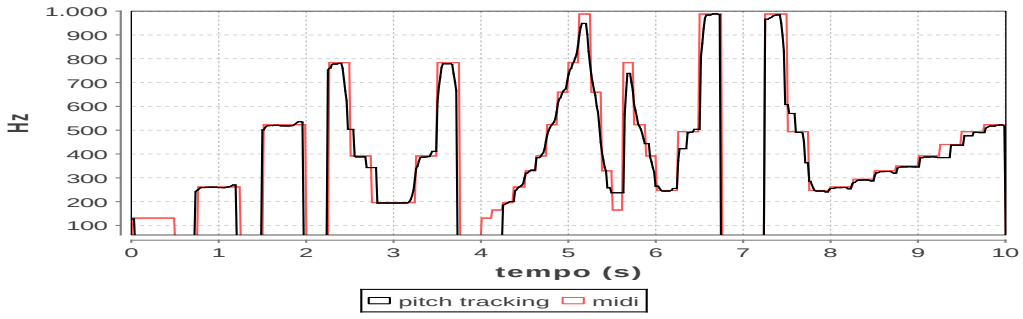
(a)



(b)

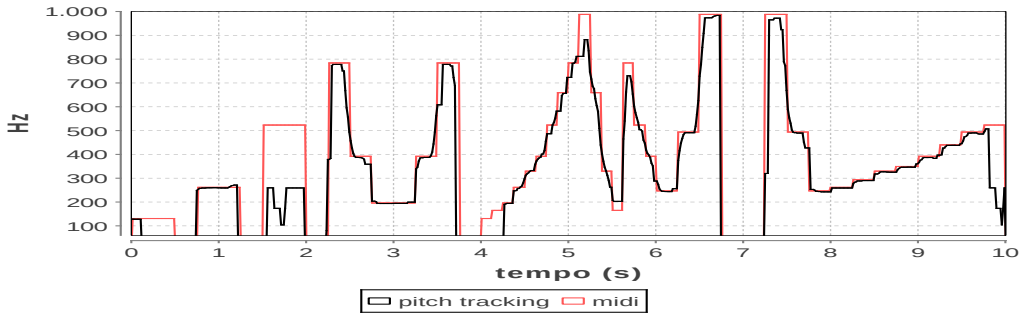


(c)

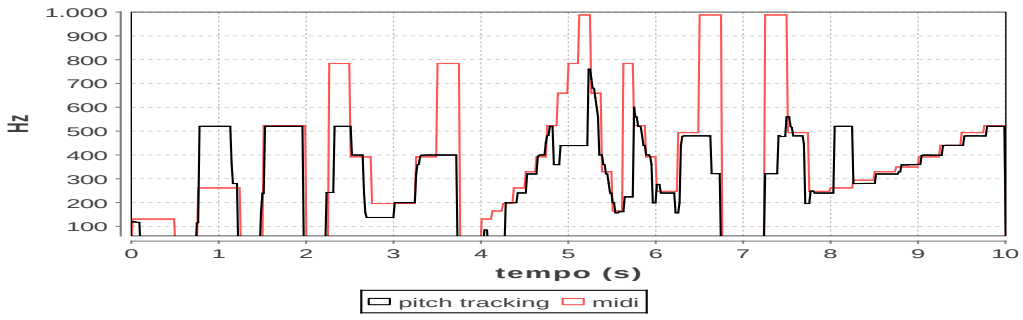


(d)

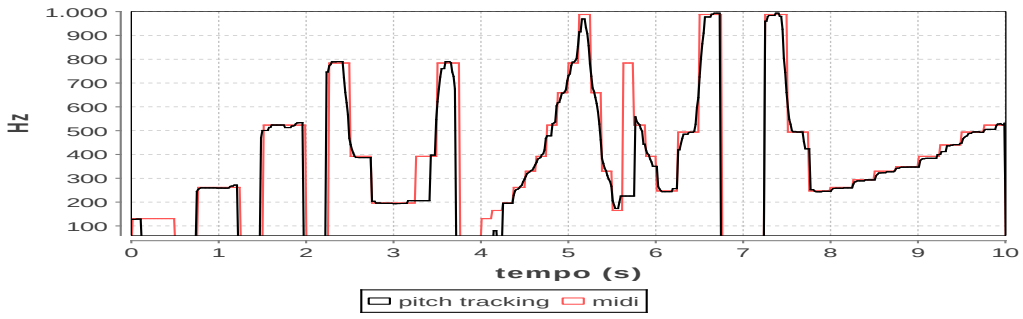
Figura 4.24: Voz Hiyama Kiyoteru sinal 2, *pitch tracking* de saída dos algoritmos e referência `MIDI`: a) ACF, b) HPS, c) CEPS e d) YIN.



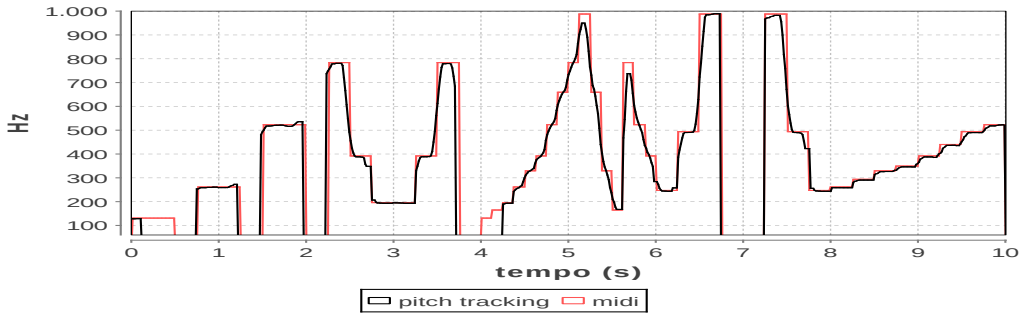
(a)



(b)

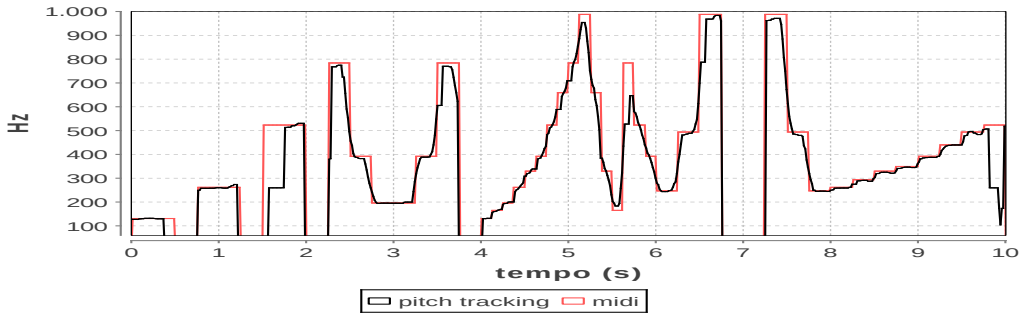


(c)

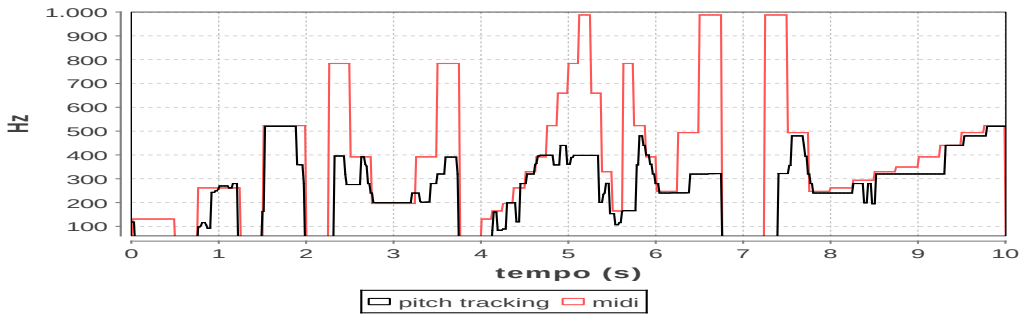


(d)

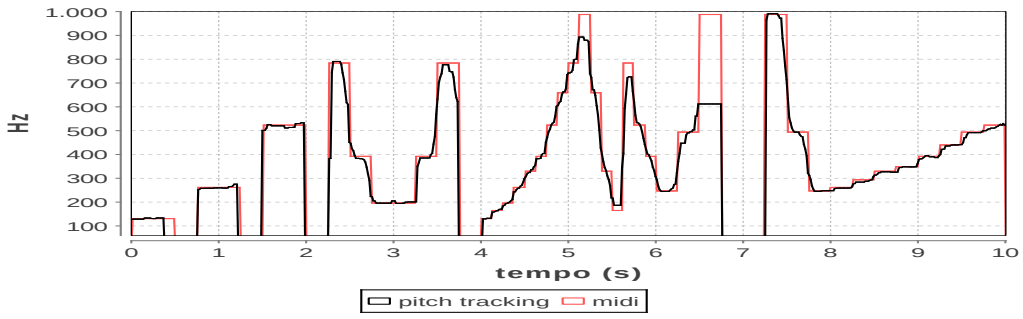
Figura 4.25: Voz Hatsune Miku sinal 2, *pitch tracking* de saída dos algoritmos e referência `MIDI`: a) ACF, b) HPS, c) CEPS e d) YIN.



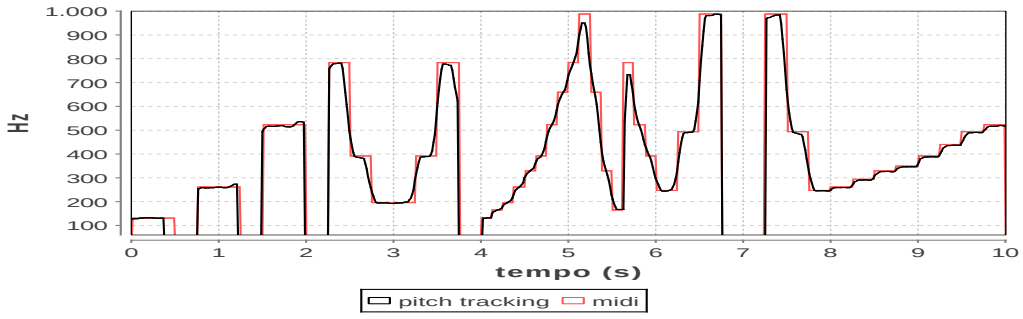
(a)



(b)



(c)



(d)

Figura 4.26: Voz Hatsune Kaai Yuki 2, *pitch tracking* de saída dos algoritmos e referência `MIDI`: a) ACF, b) HPS, c) CEPS e d) YIN.

4.6 Conclusão

Neste capítulo, foram estudados, implementados e avaliados algoritmos de estimação de frequência fundamental. Etapas de pré e pós-processamento foram empregadas e alguns passos presentes nessas etapas criados, de modo a melhorar o sinal de saída (*pitch tracking*) de todos os algoritmos apresentados.

Quanto à avaliação dos algoritmos, é importante salientar que em trabalhos dessa natureza, normalmente são utilizadas bases de dados construídas com foco em sinais laringográficos, existindo um sinal de *pitch tracking* de referência, devidamente analisado, sendo este um sinal confiável para comparação com o *pitch tracking* gerado pelos estimadores. No entanto, foi abordada uma forma alternativa a esta, com sinais gerados pelo sintetizador de voz **Vocaloid** que satisfizessem certas características desejadas, como o uso de "saltos" de 1 oitava nas notas, vibratos, pausas e utilização de toda a faixa desejada de medição de f_0, fazendo uso de notas em regiões "graves" e "agudas" da escala temperada. Apesar do reduzido número de sinais da base de dados, a avaliação dos algoritmos foi abrangente, tendo a base sido construída pensando em sinais com certas características que permitissem avaliar problemas considerados importantes em algoritmos estimadores de *pitch*, a *priori*.

Busca-se utilizar um algoritmo de estimação de f_0 que apresente a menor quantidade de erros, levando em consideração principalmente as classes de erro relativo médio (**ERM**) e grosseiro (**GEH** e **GEL**) tendo como referência sinais advindos de arquivos `MIDI`.

Segundo a avaliação realizada, o algoritmo YIN apresentou melhor desempenho que os demais nessas categorias. Um ponto a ressaltar está no fato de este algoritmo ter obtido o menor erro relativo médio em praticamente todos os sinais analisados. Deste modo, o algoritmo **YIN** é o algoritmo estimador de frequência fundamental escolhido para compor a aplicação final a que se propõe esta dissertação, devido ao seu desempenho superior em relação aos demais estimadores abordados.

Capítulo 5

Algoritmos de detecção de eventos

5.1 Introdução

Em um sistema de transcrição musical frequentemente é útil localizar a ocorrência de eventos especiais em um sinal de áudio (monofônico ou polifônico), tais como o início de notas (*onsets*) e o fim de notas (*offsets*). Entretanto, em geral, é particularmente importante a determinação do instante de tempo em que cada nota musical é iniciada.

Conhecida também por Detecção de *Onsets* ou *Onset Detection*, esta etapa encontra aplicação em:

- Transcrição automática de música [22, 23];
- Extração automática de tempo (andamento) e batida (*beat*, em inglês) [24];
- Reconhecimento de acorde em tempo real em execuções ao vivo [25];
- Sincronização de tablatura com letras de músicas cifradas [26];
- Classificação de gênero musical [27, 28].

A Figura 5.1 mostra o diagrama de blocos comumente usado em detectores de *onsets*.

Figura 5.1: Diagrama de blocos de um detector de *onsets*.

O bloco *Pré-processamento* é responsável por destacar as partes mais importantes do sinal para os passos seguintes do detector de *onsets* [29].

O bloco *Redução* tem como objetivo detectar mudanças nas propriedades do sinal.

Por último, o bloco *Pós-processamento* realiza a busca por picos (indicados pelos pontos que possuem valores máximos no sinal sob processamento), indicando assim o início das notas. Para isso, muitos algoritmos utilizam um limiar, fixo ou variável, de modo a localizar tais picos.

Os parâmetros para o processamento de voz utilizados nos algoritmos de detecção de *onset* foram os mesmos usados no capítulo anterior.

Em [29], os autores ROSÃO e RIBEIRO realizam uma revisão recente dos principais algoritmos de detecção de *onsets*, organizados em categorias: com funções de redução no domínio do tempo, com funções de redução no domínio da frequência, baseados em *pitch* e com funções de redução probabilística.

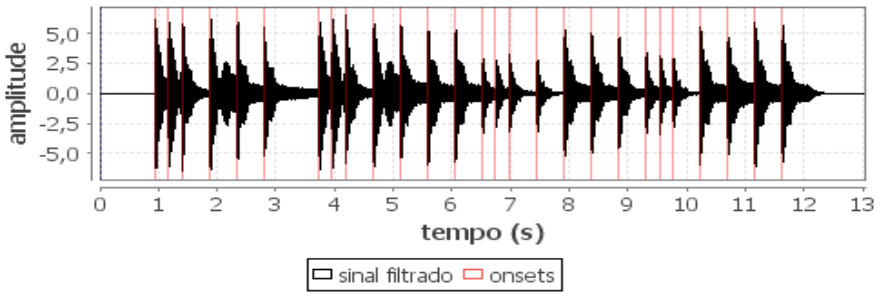
5.2 Pré-processamento

A etapa de pré-processamento tem como objetivo melhorar o desempenho do algoritmo de detecção de *onsets*. O pré-processamento empregado obedece a seguinte ordem:

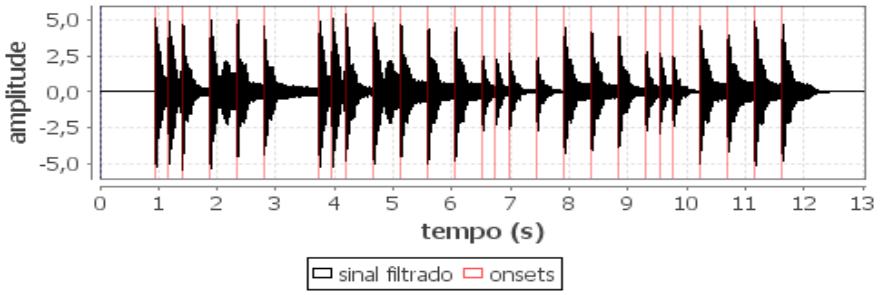
1. **Reamostragem**: O sinal de áudio em formato `WAV` é reamostrado a 8.000 Hz, com o objetivo de diminuir o volume de dados a ser processado. Sinais de mesma duração e com taxa de amostragem diferentes entre si passam a obter o mesmo tempo de processamento pelo detector de *onsets* devido a essa reamostragem para taxa fixa;

2. **Filtragem em sub-bandas**: Conceitos psicoacústicos são normalmente utilizados em detectores de *onsets*. Sendo a filtragem em sub-bandas uma estrutura que emula aproximadamente a percepção do som pelo ouvido humano, foi utilizada uma filtragem em bandas de oitava, como propõe KLAPURI em [22]. Através de filtros Butterworth de ordem 4, o sinal é dividido em 4 bandas de frequências limitadas de: 250 a 500 Hz, 500 a 1000 Hz, 1000 a 2000 Hz e 2000 a 4000 Hz.

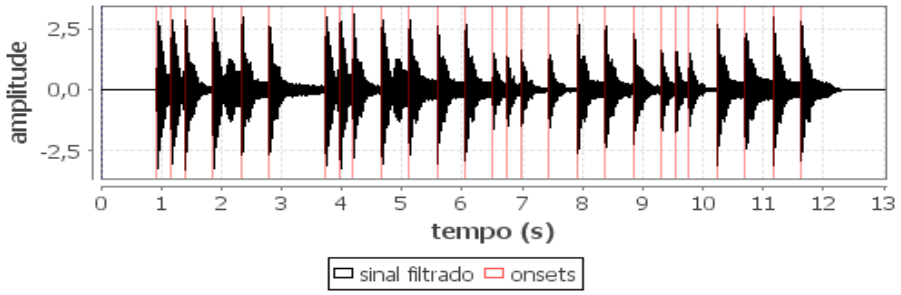
Nas Figuras 5.2 e 5.3 são mostrados os sinais de entrada de piano e solfejo (Figuras 4.2(a) e 4.2(b), respectivamente) após o pré-processamento empregado referentes à música "Parabéns a você", acompanhados de marcações que indicam o início de notas obtidas da referência `MIDI`. É nessa forma que serão passados ao módulo seguinte.



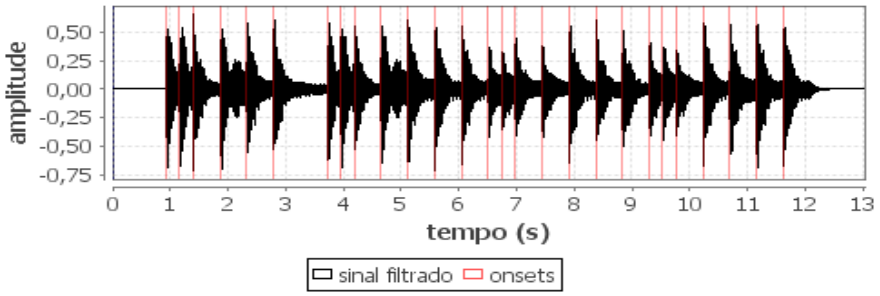
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

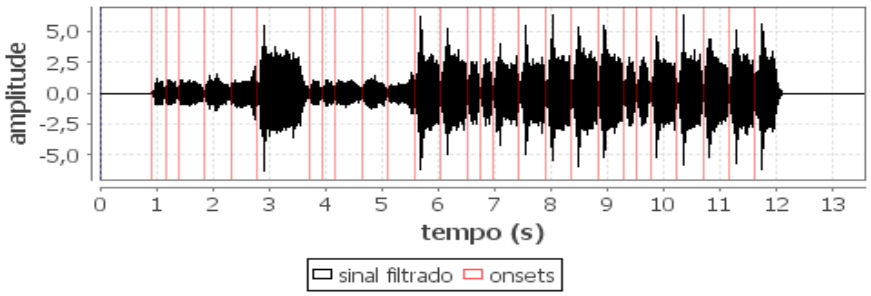


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

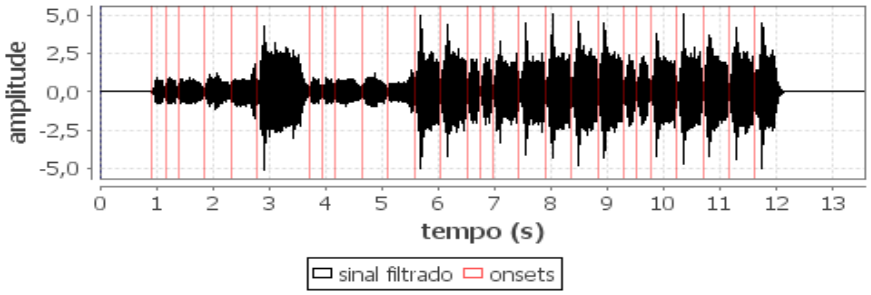


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

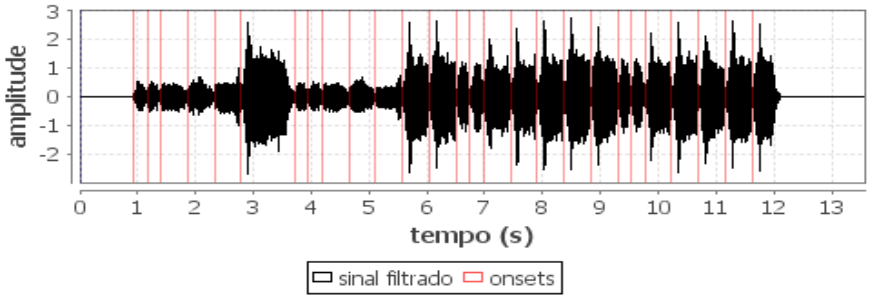
Figura 5.2: Gráfico gerado após o emprego do pré-processamento no sinal de piano da Figura 4.2(a). Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



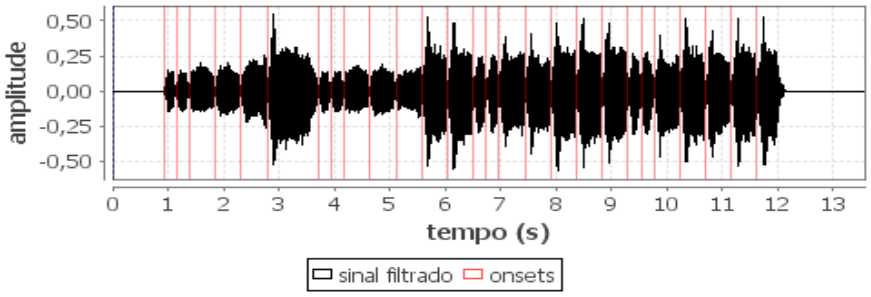
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz



(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz



(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

Figura 5.3: Gráfico gerado após o emprego do pré-processamento no sinal de solfejo da Figura 4.2(b). Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.

5.3 Detectores de *onsets*

Nesta Seção, descrevem-se alguns dos detectores mais comuns da literatura.

5.3.1 *Phase Deviation* - PD

Baseia-se em mudanças espectrais levando em consideração o desvio de fase de cada ponto do espectro de um sinal. Segundo BELLO *et al.* [30], muito da estrutura temporal de um sinal está contido no seu espectro de fase.

Considere a STFT (*Short-Time Fourier Transform*) de um sinal $x(t)$ como mostra a Equação (5.1):

$$X(n,k) = \sum_{m=0}^{N-1} x(nh+m)w(m)e^{-j\frac{2\pi}{N}mk}, \tag{5.1}$$

onde a função descreve o k-ésimo coeficiente espectral do n-ésimo *frame*, com uma janela $w(m)$ de N pontos e um deslocamento temporal entre janelas h (também conhecido por *hop-size*). Em representação polar, temos:

$$X(n,k) = |X(n,k)|e^{j\psi(n,k)},$$

com amplitude $|X(n,k)|$ e fase $\psi(n,k)$ mapeadas no intervalo $]-\pi, +\pi]$.

Uma vez obtida a representação espectral de todos os *frames*, realiza-se o desdobramento de fases pertencentes à mesma posição (coeficiente espectral) nos *frames*, com o intuito de obter a menor diferença/desvio de fase nessa análise — sem esse desdobramento, fases como 2π rad e 0 rad, por exemplo, estariam distantes de 2π rad ao invés de 0 rad, uma vez que representam a mesma fase.

A frequência instantânea da k-ésima componente espectral é calculada pela primeira diferença [31]:

$$\psi^{'}(n,k) = \psi(n,k) - \psi(n-1,k). \tag{5.2}$$

Um indicador de possível *onset* é fornecido pela segunda diferença de fase, isto é, pela variação da frequência instantânea:

$$\psi^{''}(n,k) = \psi^{'}(n,k) - \psi^{'}(n-1,k), \tag{5.3}$$

com ambos os valores de $\psi^{'}(n,k)$ e $\psi^{''}(n,k)$ mapeados em $]-\pi, +\pi]$.

Combinando as Equações (5.2) e (5.3), representa-se a segunda diferença de fase

diretamente por:

$$\psi''(n,k) = \psi(n,k) - 2\psi(n-1,k) + \psi(n-2,k). \tag{5.4}$$

A média dos desvios de frequência instantânea $\psi''(n,k)$ em módulo descreve o detector PD, como mostra a Equação (5.5):

$$\mathrm{PD}(n) = \frac{1}{N}\sum_{k=0}^{N-1} |\psi''(n,k)|. \tag{5.5}$$

Nas Figuras B.1 e B.2 do Apêndice B é possível verificar o comportamento do detector PD para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.3.2 *Weighted Phase Deviation* - WPD

Proposto em 2006 por DIXON [31], o detector *Weighted Phase Deviation* emprega na função de redução *Phase Deviation* um peso $X(n,k)$ com o objetivo de tornar a função redução menos suscetível a ruído introduzido por componentes de energia insignificante [30].

A Equação (5.6) descreve o detector WPD para o n-ésimo *frame* como a média dos desvios de frequência instantânea ponderados pelos coeficientes espectrais do *frame*, em módulo:

$$\mathrm{WPD}(n) = \frac{1}{N}\sum_{k=0}^{N-1} |X(n,k)\psi''(n,k)|. \tag{5.6}$$

Nas Figuras B.3 e B.4 do Apêndice B é possível verificar o comportamento do detector WPD para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

Outra forma de combinar amplitude e fase em uma função de redução — não abordada neste trabalho — é normalizar o somatório presente na função WPD pela soma das amplitudes dos coeficientes espectrais do n-ésimo *frame*. Esta função descreve o detector conhecido por *Normalised Weighted Phase Deviation*:

$$\mathrm{NWPD}(n) = \frac{\sum_{k=0}^{N-1} |X(n,k)\psi''(n,k)|}{\sum_{k=0}^{N-1} |X(n,k)|}. \tag{5.7}$$

5.3.3 *Complex Domain* - CD

Baseia-se em mudanças espectrais do sinal levando em consideração a amplitude e a fase dos *bins* $X(n-1,k)$ e $X(n-2,k)$.

Este detector calcula a posição esperada, no domínio complexo, de cada coeficiente espectral do *frame*, representada por $X_T(n,k)$ ou alvo (*target*, em inglês) de $X(n,k)$. Para isso, adota-se como constante a amplitude dos coeficientes do *frame* anterior $n-1$ e calcula-se a fase $\psi_T(n,k)$ deste coeficiente de acordo com a Equação (5.10):

$$\psi_T(n,k) = \psi(n-1,k) + \psi^{'}(n-1,k) \tag{5.8}$$
$$= \psi(n-1,k) + [\psi(n-1,k) - \psi(n-2,k)] \tag{5.9}$$
$$= 2\psi(n-1,k) - \psi(n-2,k). \tag{5.10}$$

O alvo de $X(n,k)$, em notação polar, é indicado na Equação (5.12):

$$X_T(n,k) = |X(n-1,k)|e^{j\psi_T(n,k)} \tag{5.11}$$
$$= |X(n-1,k)|e^{j[2\psi(n-1,k)-\psi(n-2,k)]} \tag{5.12}$$

A Figura 5.4 mostra o diagrama no domínio complexo dos fasores envolvidos na construção deste algoritmo. O erro de predição de fase é indicado por $\psi^{''}(n,k)$ no diagrama.

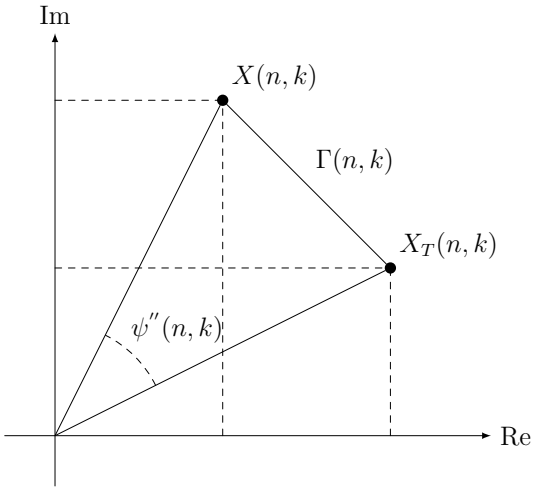


Figura 5.4: Diagrama dos fasores utilizados no algoritmo *Complex Domain*.

A função de redução *Complex Domain* é definida como a soma das distâncias $\Gamma(n,k)$ entre $X(n,k)$ e $X_T(n,k)$ ao longo dos *bins*, conforme indicam a Equações (5.13) e (5.14):

$$\Gamma(n,k) = |X(n,k) - X_T(n,k)|, \tag{5.13}$$

$$CD(n) = \sum_{k=0}^{N-1} \Gamma(n,k). \tag{5.14}$$

Nas Figuras B.5 e B.6 do Apêndice B é possível verificar o comportamento do detector CD para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.3.4 *Complex Domain Simplified* - CDS

Baseado no detector *Complex Domain*, este algoritmo faz uma simplificação no processo, que consiste em:

- Projetar $X_T(n,k)$ no eixo real;
- Rotacionar $X(n,k)$ de modo que sua fase seja igual ao desvio de fase $\psi''(n,k)$.

Assim, com a parte imaginária de $X_T(n,k)$ igual a zero, a Equação (5.12) torna-se:

$$X_T(n,k) = |X(n-1,k)|,$$

e por sua vez a distância $\Gamma(n,k)$ entre os fasores é simplificada de acordo com a Equação (5.17):

$$\Gamma(n,k) = |X(n,k) - X_T(n,k)| \tag{5.15}$$

$$= \{[\mathrm{Re}\{X(n,k)\} - \mathrm{Re}\{X_T(n,k)\}]^2 + [\mathrm{Im}\{X(n,k)\}]^2\}^{\frac{1}{2}} \tag{5.16}$$

$$\begin{aligned} &= \{\mathrm{Re}\{X(n,k)\}^2 + \mathrm{Re}\{X_T(n,k)\}^2 \\ &\quad - 2.\mathrm{Re}\{R_T(n,k)\}.\mathrm{Re}\{R(n,k)\}.\cos(\psi''(n,k))\}. \end{aligned} \tag{5.17}$$

A Figura 5.5 mostra o diagrama no domínio complexo da nova disposição dos fasores envolvidos.

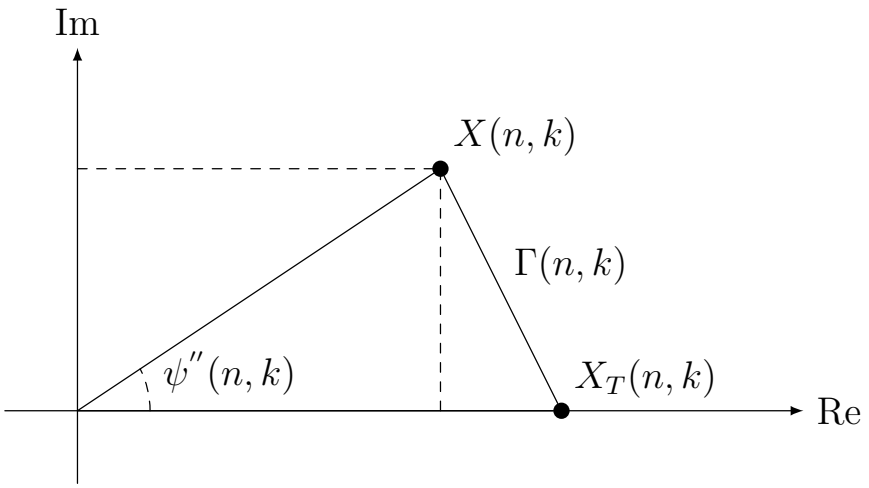


Figura 5.5: Diagrama dos fasores utilizados no algoritmo *Complex Domain Simplified*.

A função de redução *Complex Domain Simplified* é definida como a soma das distâncias $\Gamma(n,k)$ entre $X(n,k)$ e $X_T(n,k)$ ao longo dos *bins*, conforme indica a Equação (5.18):

$$\mathrm{CDS}(n) = \sum_{k=0}^{N-1} \Gamma(n,k). \tag{5.18}$$

Nas Figuras B.7 e B.8 do Apêndice B é possível verificar o comportamento do detector CDS para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.3.5 *Rectified Complex Domain* - RCD

Baseado no detector *Complex Domain*, o algoritmo *Rectified Complex Domain* enfatiza a ocorrência de *onsets* realizando uma retificação de meia onda no sinal.

Esse algoritmo foi construído devido a o método *Complex Domain* não fazer distinção entre "subidas" e "descidas" de amplitude no sinal, o que faz com que *onsets* e *offsets* não sejam diferenciados pelo algoritmo CD.

A Equação (5.19) descreve o algoritmo RCD como a mesma soma das distâncias entre $X(n,k)$ e $X_T(n,k)$ ao longo dos *bins*, porém, apenas quando $|X(n,k)| \geq |X(n-1,k)|$:

$$\mathrm{RCD}(n) = \sum_{k=0}^{N-1} \Gamma(n,k), \tag{5.19}$$

onde $\Gamma(n,k)$ corresponde a:

$$\Gamma(n,k) = \begin{cases} |X(n,k) - X_T(n,k)|, & \text{se } |X(n,k)| \geq |X(n-1,k)|, \\ 0, & \text{em caso contrário.} \end{cases}$$

Nas Figuras B.9 e B.10 do Apêndice B é possível verificar o comportamento do detector RCD para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.3.6 *Spectral Flux* - SF

Baseia-se em mudanças espectrais do sinal levando em consideração a magnitude de cada ponto do espectro. Para isso, calcula-se a diferença entre amostras sucessivas a partir da função de distância das normas L1 [29], segundo a Equação (5.20):

$$\mathrm{SF}(n) = \sum_{k=0}^{N-1} H(|X(n,k)| - |X(n-1,k)|). \tag{5.20}$$

Sabendo que $H(x) = \frac{x+|x|}{2}$ retorna zero para valores negativos de x e o próprio valor de x para os demais casos, apenas as frequências que apresentam aumento de energia — enfatizando *onsets* — são consideradas. A largura da janela em amostras é indicada por N, e $X(n,k)$ representa o conjunto de k pontos na frequência do n-ésimo *frame*.

Em geral, são relatados ótimos resultados no uso deste método na procura de *onsets* em sinais de percussão sem *pitch* (NPP, *non-pitched percussive*) [30].

Outro detector de *onsets* semelhante ao *Spectral Flux* é o *Spectral Difference*. Este utiliza a função distância de norma L2. No entanto, testes práticos indicam melhores resultados quando a função de distância de norma L1 é utilizada em vez da norma L2 [31]. Sendo assim, o detector *Spectral Difference* não foi abordado neste trabalho.

Nas Figuras B.11 e B.12 do Apêndice B é possível verificar o comportamento do detector SF para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.3.7 *High-Frequency Content* - HFC

Proposto por MASRI [32] em 1996, o algoritmo *High- Frequency Content* identifica *onsets* através da caracterização da quantidade de conteúdo de alta frequência no sinal.

O algoritmo utiliza as funções $\mathrm{HF}(n)$ e $\mathrm{E}(n)$, que indicam, respectivamente, o conteúdo de alta frequência e a energia do sinal, descritas nas Equações (5.21) e

(5.22):

$$\mathrm{HF}(n) = \sum_{k=2}^{N} k.|X(n,k)|^2, \tag{5.21}$$

$$\mathrm{E}(n) = \sum_{k=2}^{N} |X(n,k)|^2, \tag{5.22}$$

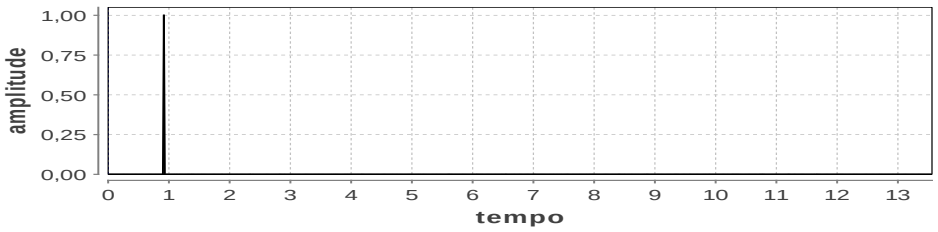
onde $X(n,k)$ corresponde ao k-ésimo coeficiente espectral do n-ésimo *frame* e N à metade do total de coeficientes espectrais.

De modo a evitar *bias* indesejado e componentes de muito baixa frequência, os dois primeiros *bins* são descartados [32].

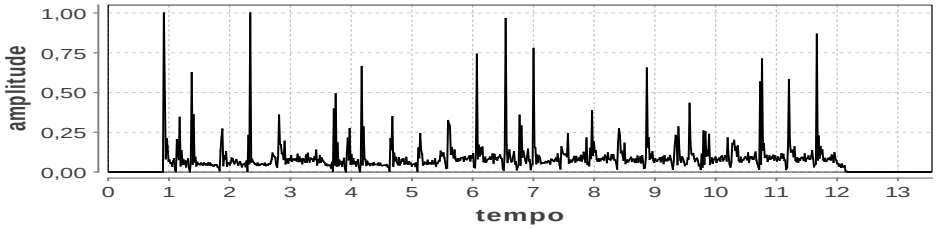
A função de redução HFC utiliza dados dos *frames* atual e imediatamente anterior na sua configuração, n e $n-1$, respectivamente:

$$\mathrm{HFC}(n) = \frac{\mathrm{HF}(n)^2}{\mathrm{HF}(n-1).\mathrm{E}(n)}. \tag{5.23}$$

A Figura 5.6(a) mostra a saída do detector para o sinal de solfejo mostrado na Figura 4.2(b). Observando este sinal de entrada, verifica-se nos trechos de silêncio valores localizados em zero, o que provoca, por exemplo, um pico de altíssima intensidade situado próximo ao instante de segundo número 1. Vale recordar que o sinal de entrada foi gerado pelo *software* **Vocaloid**, que adota o valor de zero absoluto nos momentos de pausas, comportamento que dificilmente é encontrado em gravações reais, que costumam ter algum ruído de fundo. Entretanto, para minimizar os efeitos desse artefato, foi incluído um passo extra no detector, que consiste em encontrar a 2ª maior amostra em amplitude e usá-la como valor máximo permitido para o conjunto de amostras do sinal resultante. Assim, todas as amostras que tiverem amplitude superior a este valor terão suas amplitudes corrigidas para este limiar. A Figura 5.6(b) mostra o sinal após este ajuste.



(a)



(b)

Figura 5.6: Gráfico gerado pelo estimador HFC tendo como entrada um sinal de solfejo (Figura 4.2(b)) filtrado de 2.000 a 4.000 Hz, sendo (a) sinal original normalizado pelo valor máximo e (b) sinal com o ajuste proposto.

São relatados ótimos resultados deste detector de *onsets* em sinais percussivos na literatura.

Nas Figuras B.13 e B.14 do Apêndice B é possível verificar o comportamento do detector HFC para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.3.8 Derivada da Envoltória - DE

O detector DE utiliza a derivada da envoltória do sinal para a obtenção das localizações de início de notas.

A Equação (5.24) define como a envoltória do sinal é encontrada [33]:

$$\mathrm{Env}(n) = \mathrm{AVG}_{k=0}^{L-1}\{|x(n,k)|\}, \tag{5.24}$$

onde L corresponde à largura em amostras dos *frames*, $x(n,k)$ à k-ésima amostra temporal do n-ésimo *frame* do sinal e AVG à média dos valores de $|x(n,k)|$.

A derivada da envoltória é calculada pela Equação (5.25):

$$\mathrm{DE(n)} = \mathrm{Env}(n) - \mathrm{Env}(n-1), \tag{5.25}$$

para o n-ésimo *frame*.

Nas Figuras B.15 e B.16 do Apêndice B é possível verificar o comportamento

do detector DE para os sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.3.9 Derivada Relativa da Envoltória - DRE

Proposto por KLAPURI [34] em 1999, este algoritmo concentra-se em obter a derivada relativa da envoltória do sinal filtrado em bandas e na combinação destas por meio de uma heurística de modo a ressaltar o início das notas.

Assim como SCHEIRER [35], diversos pesquisadores buscam aprimorar os sistemas de detecção de *onsets* tendo como base o órgão auditivo humano. O algoritmo de KLAPURI é um deles.

O diagrama da Figura 5.7 mostra as etapas do detector DRE. A primeira etapa deste detector foi retirada do algoritmo neste trabalho devido ao pré-processamento abordado no início desta seção já conter um banco de filtros para a filtragem em subbandas. Para a etapa de **Extração de Envoltória** do sinal foi utilizada a Equação (5.24). A etapa **Derivada Relativa** tem como finalidade enfatizar os prováveis *onsets* presentes no sinal.

Sabe-se que o uso da derivada relativa na tarefa de detectar *onsets* em sinais resulta em estimação mais fiel ao verdadeiro início de notas, quando comparado com o emprego da derivada simples no processo. Seu comportamento é regido pela Equação (5.26):

$$\mathrm{DRE(n)} = \frac{\mathrm{Env}(n) - \mathrm{Env}(n-1)}{\mathrm{Env}(n)}, \tag{5.26}$$

onde n representa o n-ésimo *frame*.

A etapa **Limiarização** utiliza um limiar fixo de modo a considerar apenas as amostras mais relevantes do sinal. O limiar adotado corresponde ao valor de 30% da maior amplitude considerando todas as amostras do sinal de saída da etapa **Derivada Relativa**. As amplitudes abaixo do limiar são zeradas.

Por último, a etapa **Combinação de bandas** combina todos os sinais limiarizados (cada um referente a uma banda específica) em uma soma linear de modo a gerar um sinal com as indicações de localização temporal de início de notas. Uma vez que no diagrama da Figura 5.1 o bloco **Pós-processamento** já realiza essa combinação e a detecção dos picos no sinal, essa etapa do detector de KLAPURI não foi implementada.

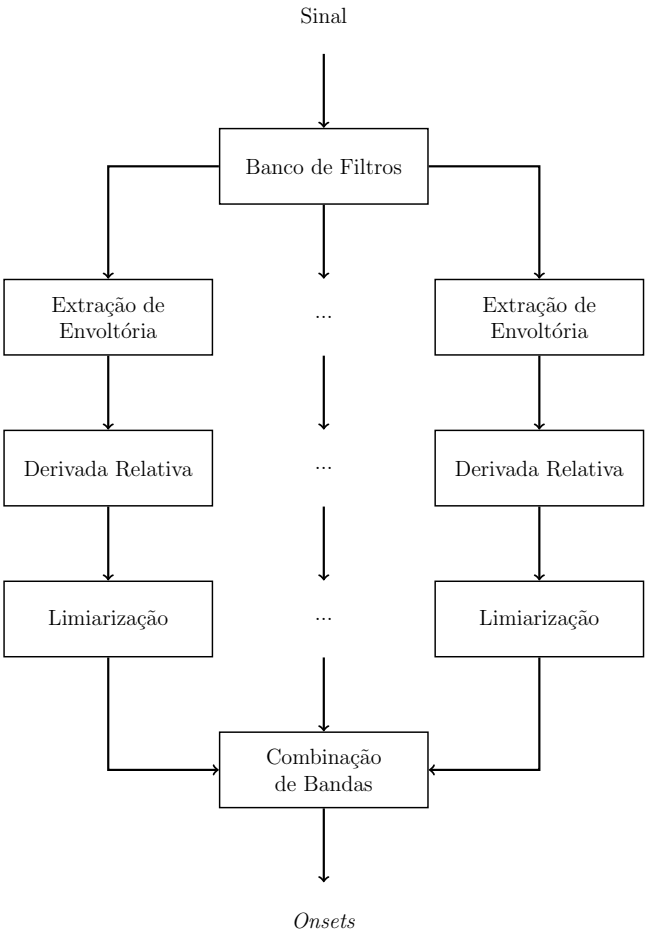


Figura 5.7: Diagrama do detector de KLAPURI.

Nas Figuras B.17 e B.18 do Apêndice B é possível verificar o comportamento do detector por meio dos sinais de entrada de piano e solfejo referentes à música "Parabéns a você", mostrados nas Figuras 4.2(a) e 4.2(b), respectivamente.

5.4 Pós-processamento

A etapa de pós-processamento tem como objetivo processar os sinais de cada banda, combinando-os para gerar um sinal que apresente as indicações de início de notas existentes no sinal de entrada fornecido pelo usuário.

O pós-processamento empregado divide-se em duas partes, sendo a primeira constituída de um conjunto de passos realizados no sinal de cada sub-banda, e a segunda em um conjunto de passos realizados no sinal resultante da soma dos sinais das sub-bandas.

A seguir são descritas detalhadamente as etapas de cada parte do pós-processamento:

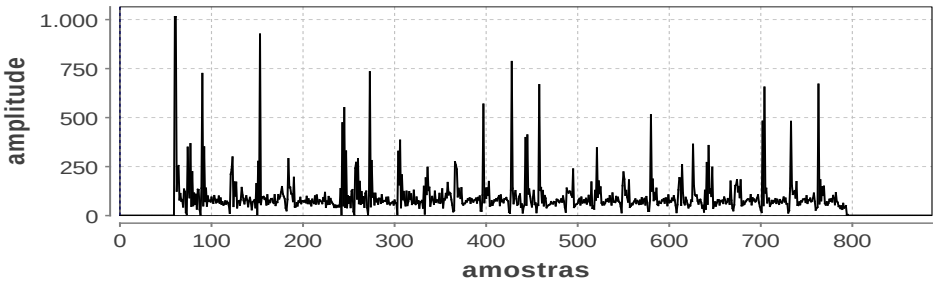
Parte 1: Passos:

- Normaliza-se o sinal da sub-banda pelo valor máximo de amplitude das amostras do sinal, tornando assim cada sinal igualmente relevante no processo.
- Deriva-se o sinal para que nos pontos onde existam *onsets* sejam evidenciados tais eventos.
- Aplica-se ao sinal um limiar definido pela média móvel de 3 amostras subtraída de um *offset* fixo (escolhido como 0,1) que garanta que as amplitudes referentes a *onsets* fiquem acima do limiar; amostras abaixo do limiar são zeradas.

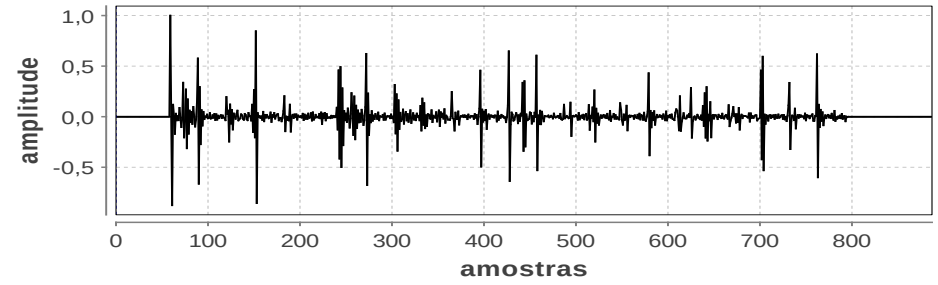
Parte 2: Passos:

- A partir de um janelamento de largura 150 ms no sinal resultante da soma dos sinais das sub-bandas, encontra-se a amostra de maior amplitude por janela e utiliza-se desta amplitude como referência. As amostras da janela que possuam amplitudes abaixo desta referência passam a ter amplitude zerada.
- Normaliza-se o sinal pelo valor máximo de amplitude das amostras.
- Por fim, eliminam-se do sinal amostras abaixo de um limiar fixo (escolhido como 0,1), indicativas de *onsets* menos proeminentes, possivelmente espúrios.

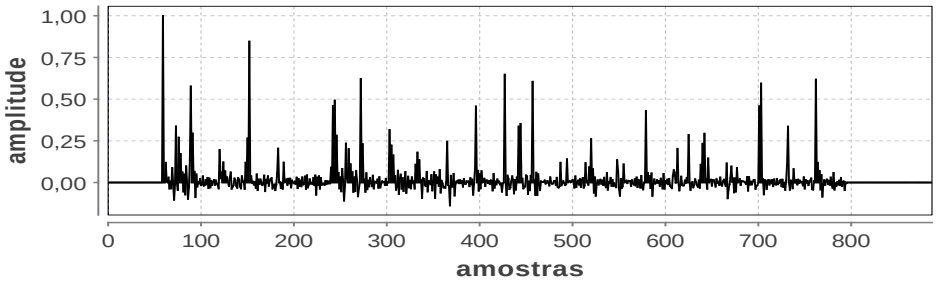
São mostrados nas Figuras 5.8 e 5.9 os sinais após cada passo das partes do pós-processamento para um sinal de exemplo.



(a) Sub-banda 1: Sinal de saída do estimador HFC filtrado de 250 a 500 Hz.

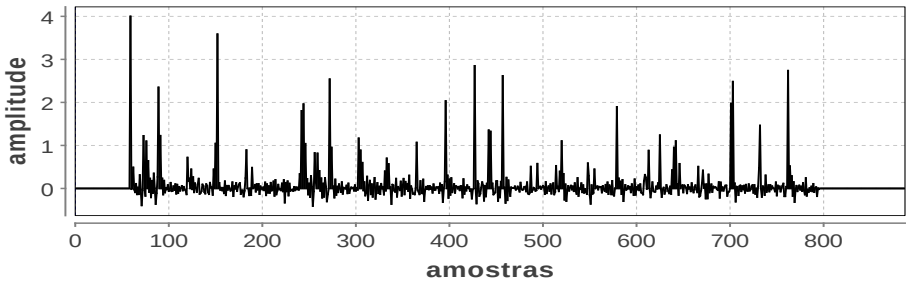


(b) Derivada do sinal normalizado pela amplitude máxima.

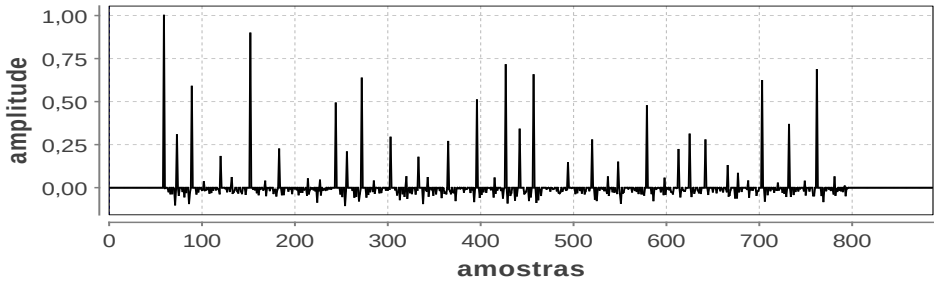


(c) Limiarização pela média móvel decrescida pelo nível 0,1.

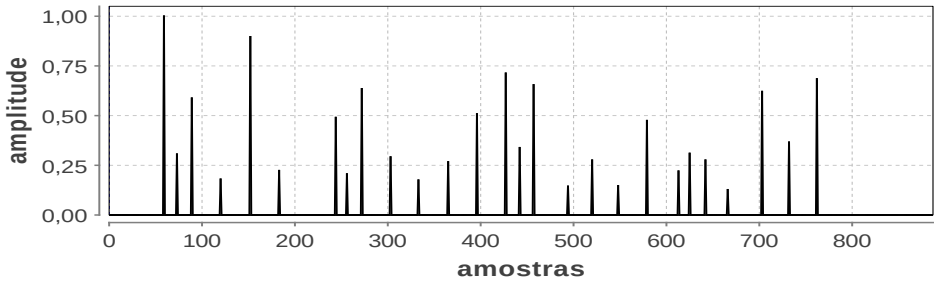
Figura 5.8: Gráficos referentes aos passos da parte 1 do pós-processamento, para um sinal de redução obtido pelo solfejo (Figura 4.2(b)) usado como entrada no estimador HFC.



(a) Sinal resultante da soma dos sinais das sub-bandas.



(b) Sinal resultante após o janelamento com normalização pela amplitude máxima.



(c) Sinal resultante após a limiarização pelo nível fixo em 0,1.

Figura 5.9: Gráficos referentes aos passos da parte 2 do pós-processamento, para um sinal de redução obtido pelo solfejo (Figura 4.2(b)) usado como entrada no estimador HFC.

5.5 Avaliação

De modo a permitir a escolha do algoritmo utilizado como detector de *onsets* na aplicação desenvolvida nesta dissertação, avaliou-se o desempenho dos mesmos com base nas classes apresentadas por KLAPURI [34]:

***Total de eventos* (TE):** Apresenta a quantidade total de eventos conhecidos no sinal referência `MIDI`.

***Eventos não detectados* (END):** Apresenta a quantidade de eventos existentes do sinal referência, mas não existentes no sinal de voz, considerando uma tolerância de 150 ms.

***Eventos Adicionais* (EA):** Apresenta a quantidade de eventos existentes no sinal de voz, mas não no sinal referência, para uma tolerância de 150 ms.

***Precisão %*:** Apresenta a precisão em porcentagem de acordo com a expressão abaixo:

$$\text{Precisão } \% = \frac{(\text{TE} - \text{END} - \text{EA})}{\text{TE}}.100\ \% \tag{5.27}$$

Utilizou-se a mesma base de dados do capítulo anterior. No entanto, por se tratar de um sinal impossível de ser solfejado (considerando tanto a emissão em curtíssimo intervalo de tempo de um conjunto de notas quanto em variação de amplitude de frequências que chegam a 1 oitava em diversos momentos), o sinal referente à partitura mostrada na Figura 4.16(b) foi ignorado nesta avaliação.

Para a avaliação dos algoritmos de detecção de *onsets* não ser realizada tendo como referência apenas um sinal, foi adicionado ao experimento um sinal da base de dados `MIDI`. Este sinal, refere-se à música "Parabéns a você", cuja partitura é mostrada na Figura 4.1. São mostradas nas Figuras 5.10 e 5.11 o sinal `MIDI` importado na tela do *software* **Vocaloid** e em formato de onda para as mesmas vozes utilizadas no Capítulo 4, respectivamente.

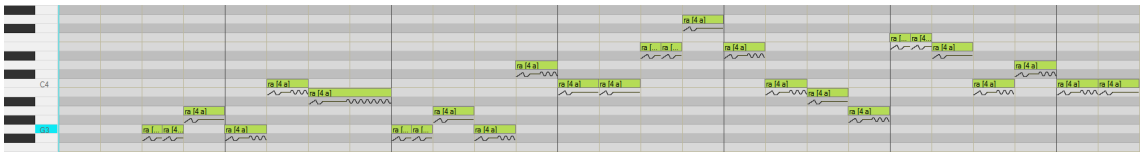
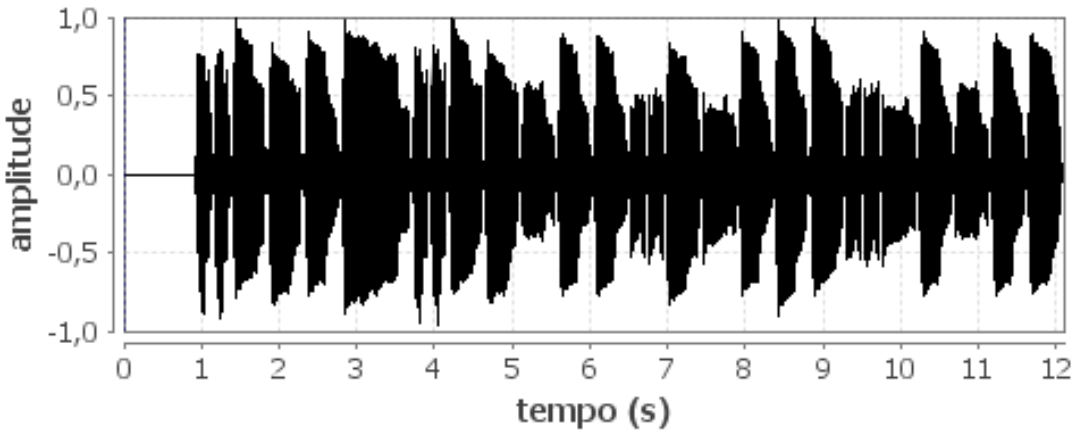
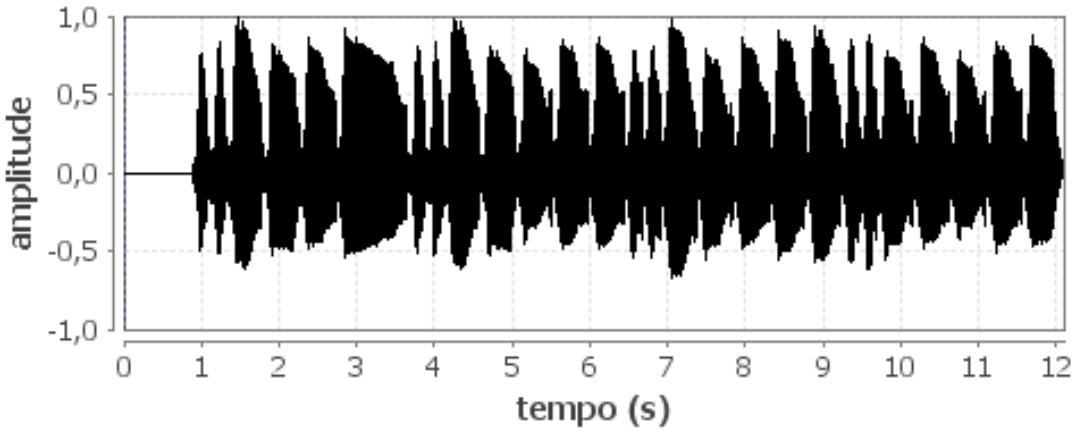


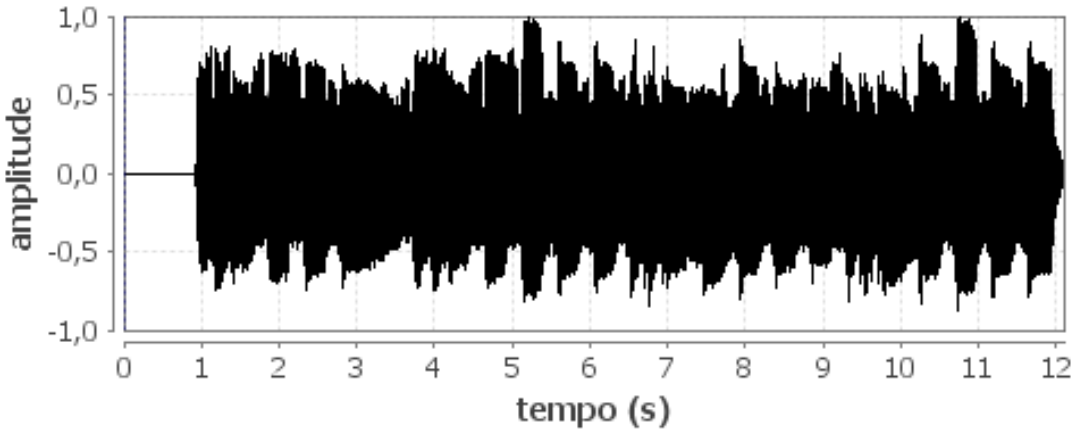
Figura 5.10: Sinal 2, referente à música "Parabéns a você", cuja partitura é mostrada na Figura 4.1.



(a) Voz de Hiyama Kiyoteru



(b) Voz de Hatsune Miku



(c) Voz de Kaai Yuki

Figura 5.11: Sinal 2, referente à Figura 4.18.

5.5.1 Resultados

Apresentam-se os resultados provenientes da avaliação dos algoritmos de detecção de *onsets* a partir dos sinais gerados pelo *software* sintetizador de voz **Vocaloid** utilizando 3 vozes de idioma japonês, tendo como referência as partituras das Figuras 4.16(a) e 4.1, nomeados de sinal 1 e sinal 2.

Os resultados estão divididos em duas seções, a primeira em tabelas e a segunda em gráficos, ambas fornecendo dados tanto numéricos quanto visuais do desempenho dos algoritmos. As considerações levantadas a partir destes resultados são comentadas a seguir:

- **Sinal 1, voz de Hiyama Kiyoteru**: Segundo a Tabela 5.1, somente três algoritmos conseguiram nenhuma adição de evento não existente no sinal referência; foram eles **DE**, **DRE** e **HFC**. Observando a Figura 5.1, nota-se que os detectores **DRE**, **HFC** e **PD** foram os únicos a detectar os eventos referentes às notas mais graves do sinal. O algoritmo de melhor desempenho foi o **DRE**, alcançando 97,22 % de precisão.

- **Sinal 1, voz de Hatsune Miku**: Com apenas 3 eventos não detectados e nenhum evento adicional, o detector **DRE** alcançou novamente o melhor desempenho, com 91.67 % de precisão. Os detectores **CD**, **CDS**, **RCD** e **SF** obtiveram o pior desempenho, 72,22 % de precisão, como mostram a Tabela 5.2 e a Figura 5.13.

- **Sinal 1, voz de Kaai Yuki**: O algoritmo **RCD** apresentou o desempenho mais baixo dentre os demais, não detectando 9 eventos, enquanto que o **DRE** não detectou apenas 2 eventos, atingindo precisão de 94,44 %, conforme a Tabela 5.3. Somente os algoritmos **DRE**, **HFC** e **PD** conseguiram detectar virtualmente todos os eventos, como mostra a Figura 5.14.

- **Sinal 2, voz de Hiyama Kiyoteru**: Sendo o único a alcançar 100,00 % de precisão, como mostram a Tabela 5.4 e a Figura 5.4, o algoritmo **DRE** mais uma vez teve desempenho melhor que os demais. Já o algoritmo **PD** obteve o pior desempenho, com 18,52 % de precisão. O segundo melhor desempenho foi do detector **DR**, com 88,89 % de precisão.

- **Sinal 2, voz de Hatsune Miku**: Segundo a Tabela 5.5, o detector **DRE** atingiu a melhor precisão, com 96,29 %. Já o algoritmo **CDS** o pior desempenho, com 29,62 % de precisão, embora tenha detectado todos os eventos existentes no sinal referência.

- **Sinal 2, voz de Kaai Yuki**: Mais uma vez os algoritmos de Derivada de Envoltória, **DRE** e **DE** obtiveram os melhores desempenhos, com 100,00 %

e 92,59 % de precisão, respectivamente. Já o algoritmo **CDS** apresentou 16 eventos adicionais e o algoritmo **HFC** 16 eventos não detectados, alcançando cada um a precisão de 40,74 %.

- **Sinais 1 e 2**: Sendo ambos os sinais de estruturas completamente diferentes, os sinais produziram resultados condizentes em questão de desempenho dos algoritmos.

Tabelas de avaliação de desempenho de algoritmos de detecção de *onsets*:

Tabela 5.1: Tabela comparativa: Sinal 1, voz de Kiyoteru.

Kiyoteru (Sinal 1)				
Algoritmo	**TE**	**END**	**EA**	**Precisão (%)**
Complex Domain	36,0000	8,0000	1,0000	75,0000
Complex Domain Simplified	36,0000	4,0000	1,0000	86,1111
Derivada da Envoltória	36,0000	6,0000	0,0000	83,3333
Derivada Relativa da Envoltória	36,0000	1,0000	0,0000	97,2222
High Frequency Content	36,0000	2,0000	0,0000	94,4444
Phase Deviation	36,0000	1,0000	1,0000	94,4444
Rectified Complex Domain	36,0000	3,0000	0,0000	91,6667
Spectral Flux	36,0000	7,0000	0,0000	80,5556
Weighted Phase Deviation	36,0000	4,0000	0,0000	88,8889

Tabela 5.2: Tabela comparativa: Sinal 1, voz de Miku.

Miku (Sinal 1)				
Algoritmo	**TE**	**END**	**EA**	**Precisão (%)**
Complex Domain	36,0000	9,0000	1,0000	72,2222
Complex Domain Simplified	36,0000	9,0000	1,0000	72,2222
Derivada da Envoltória	36,0000	11,0000	0,0000	69,4444
Derivada Relativa da Envoltória	36,0000	3,0000	0,0000	91,6667
High Frequency Content	36,0000	8,0000	0,0000	77,7778
Phase Deviation	36,0000	3,0000	1,0000	88,8889
Rectified Complex Domain	36,0000	10,0000	0,0000	72,2222
Spectral Flux	36,0000	10,0000	0,0000	72,2222
Weighted Phase Deviation	36,0000	9,0000	0,0000	75,0000

Tabela 5.3: Tabela comparativa: Sinal 1, voz de Yuki.

Yuki (Sinal 1)				
Algoritmo	**TE**	**END**	**EA**	**Precisão (%)**
Complex Domain	36,0000	6,0000	1,0000	80,5556
Complex Domain Simplified	36,0000	7,0000	1,0000	77,7778
Derivada da Envoltória	36,0000	9,0000	0,0000	75,0000
Derivada Relativa da Envoltória	36,0000	2,0000	0,0000	94,4444
High Frequency Content	36,0000	3,0000	0,0000	91,6667
Phase Deviation	36,0000	3,0000	1,0000	88,8889
Rectified Complex Domain	36,0000	9,0000	0,0000	75,0000
Spectral Flux	36,0000	8,0000	0,0000	77,7778
Weighted Phase Deviation	36,0000	7,0000	0,0000	80,5556

Tabela 5.4: Tabel comparativa: Sinal 3, voz de Kiyoteru.

Kiyoteru (Sinal 2)				
Algoritmo	**TE**	**END**	**EA**	**Precisão (%)**
Complex Domain	27,0000	2,0000	12,0000	48,1481
Complex Domain Simplified	27,0000	1,0000	13,0000	48,1481
Derivada da Envoltória	27,0000	1,0000	6,0000	74,0741
Derivada Relativa da Envoltória	27,0000	0,0000	0,0000	100,0000
High Frequency Content	27,0000	3,0000	0,0000	88,8889
Phase Deviation	27,0000	3,0000	19,0000	18,5185
Rectified Complex Domain	27,0000	0,0000	16,0000	40,7407
Spectral Flux	27,0000	0,0000	14,0000	48,1481
Weighted Phase Deviation	27,0000	1,0000	16,0000	37,0370

Tabela 5.5: Tabela comparativa: Sinal 3, voz de Miku.

Miku (Sinal 2)				
Algoritmo	**TE**	**END**	**EA**	**Precisão (%)**
Complex Domain	27,0000	0,0000	14,0000	48,1481
Complex Domain Simplified	27,0000	0,0000	19,0000	29,6296
Derivada da Envoltória	27,0000	0,0000	5,0000	81,4815
Derivada Relativa da Envoltória	27,0000	0,0000	1,0000	96,2963
High Frequency Content	27,0000	6,0000	0,0000	77,7778
Phase Deviation	27,0000	3,0000	9,0000	55,5556
Rectified Complex Domain	27,0000	2,0000	14,0000	40,7407
Spectral Flux	27,0000	1,0000	9,0000	62,9630
Weighted Phase Deviation	27,0000	0,0000	12,0000	55,5556

Tabela 5.6: Tabela comparativa: Sinal 3, voz de Yuki.

Yuki (Sinal 2)				
Algoritmo	**TE**	**END**	**EA**	**Precisão (%)**
Complex Domain	27,0000	0,0000	10,0000	62,9630
Complex Domain Simplified	27,0000	0,0000	16,0000	40,7407
Derivada da Envoltória	27,0000	0,0000	2,0000	92,5926
Derivada Relativa da Envoltória	27,0000	0,0000	0,0000	100,0000
High Frequency Content	27,0000	16,0000	0,0000	40,7407
Phase Deviation	27,0000	2,0000	11,0000	51,8519
Rectified Complex Domain	27,0000	0,0000	15,0000	44,4444
Spectral Flux	27,0000	0,0000	12,0000	55,5556
Weighted Phase Deviation	27,0000	1,0000	13,0000	48,1481

Gráficos de avaliação de desempenho de algoritmos de detecção de *onsets*:

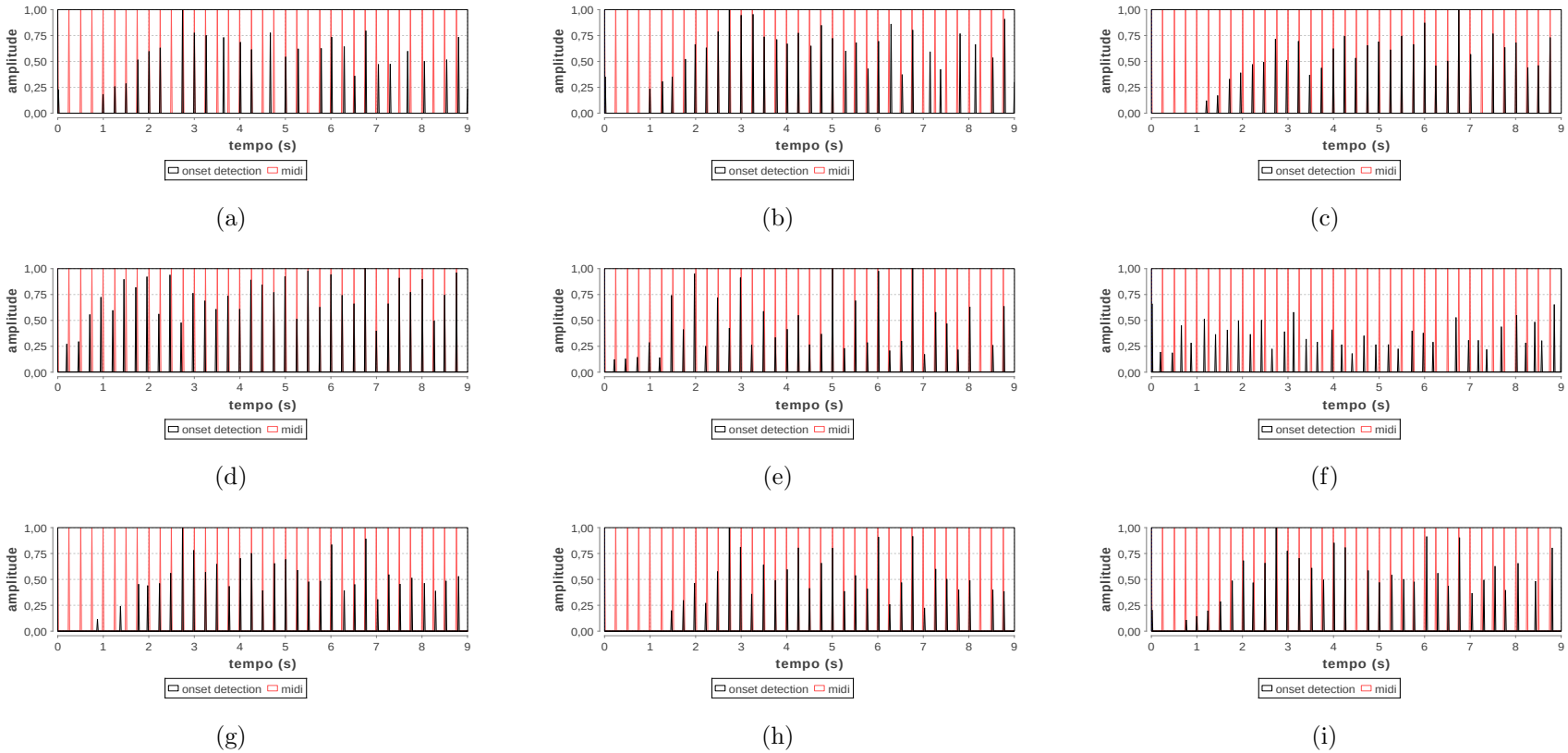


Figura 5.12: Voz Hiyama Kiyoteru sinal 1, sinal de saída dos algoritmos e referência `MIDI` (em vermelho): a) CD, b) CDS, c) DE, d) DRE, e) HFC, f) PD, g) RCD, h) SF e i) WPD.

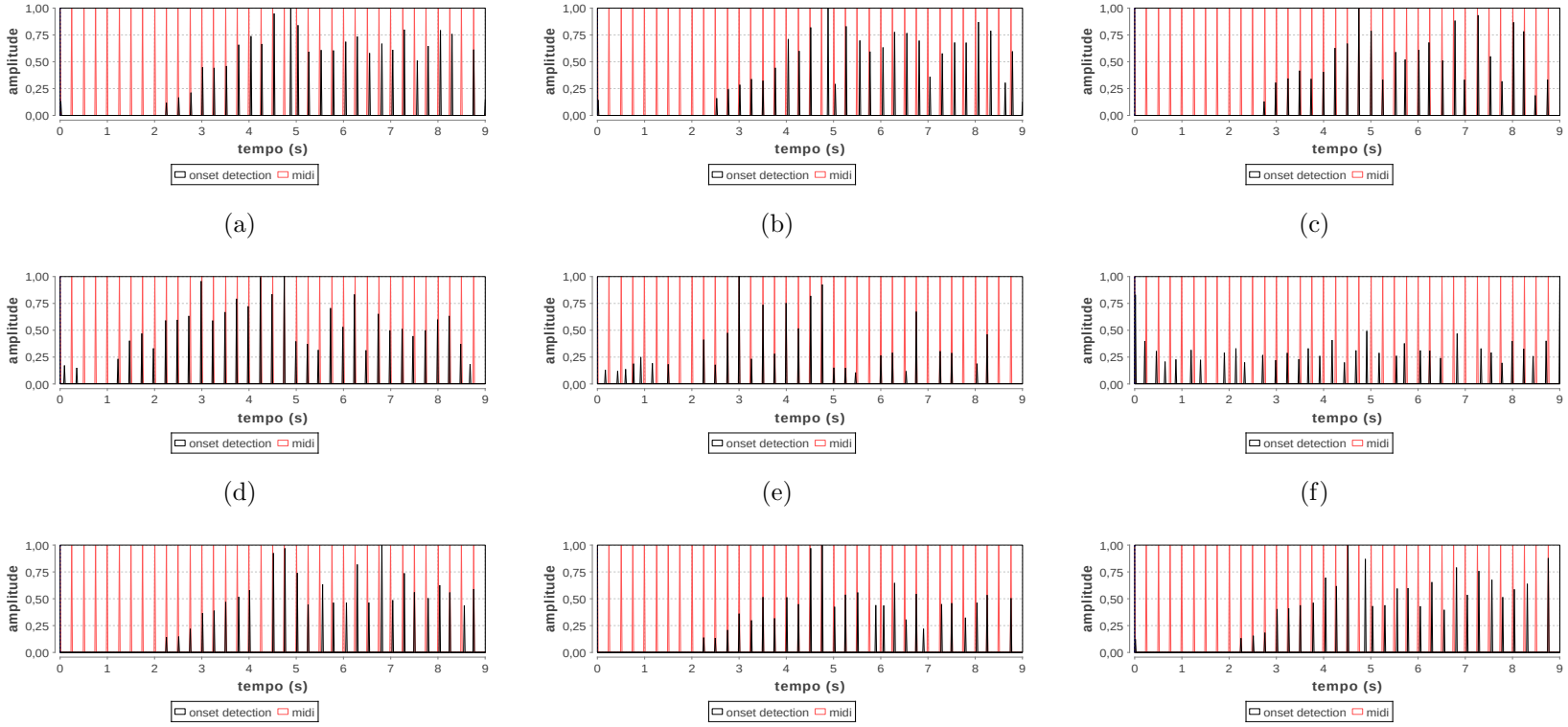


Figura 5.13: Voz Hatsune Miku sinal 1, sinal de saída dos algoritmos e referência `MIDI` (em vermelho): a) CD, b) CDS, c) DE, d) DRE, e) HFC, f) PD, g) RCD, h) SF e i) WPD.

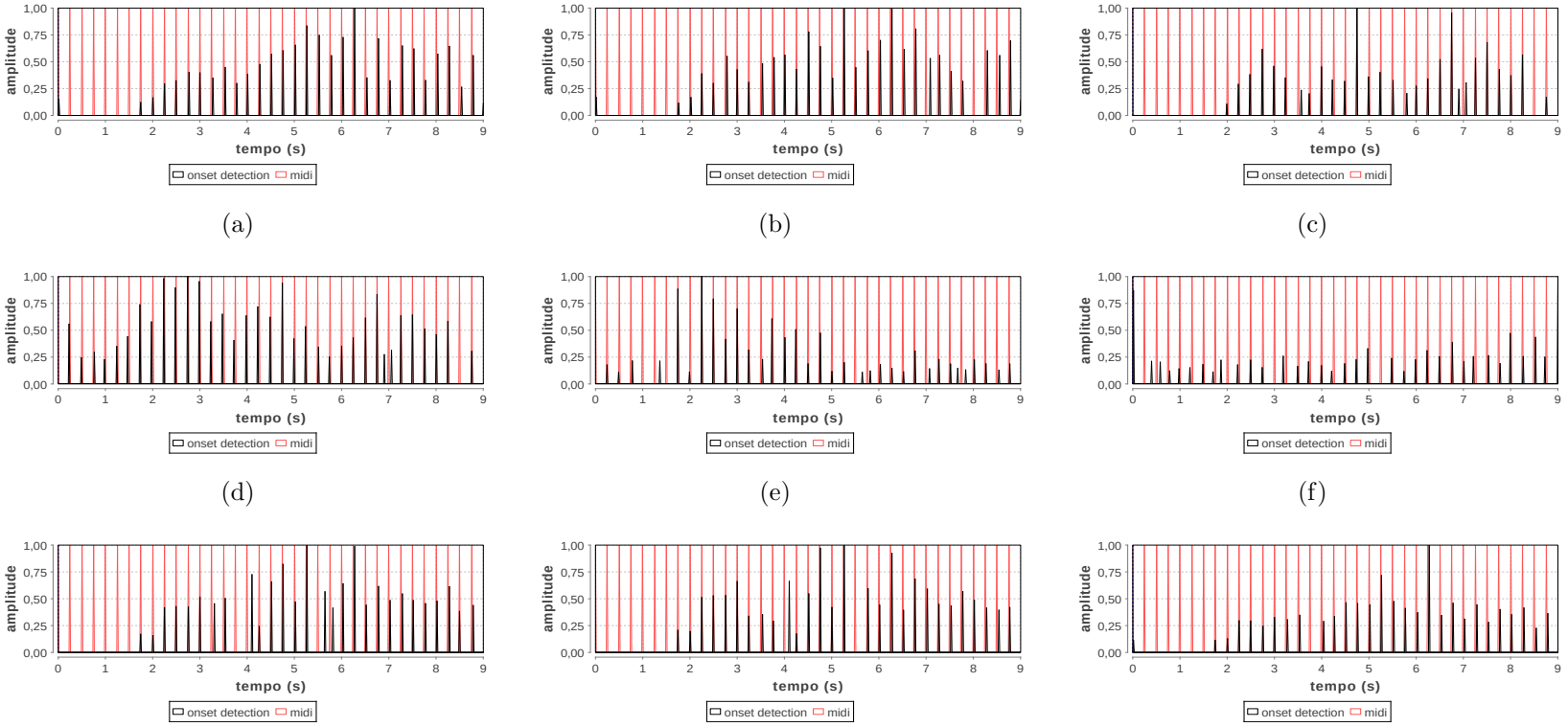


Figura 5.14: Voz Kaai Yuki sinal 1,sinal de saída dos algoritmos e referência `MIDI` (em vermelho): a) CD, b) CDS, c) DE, d) DRE, e) HFC, f) PD, g) RCD, h) SF e i) WPD.

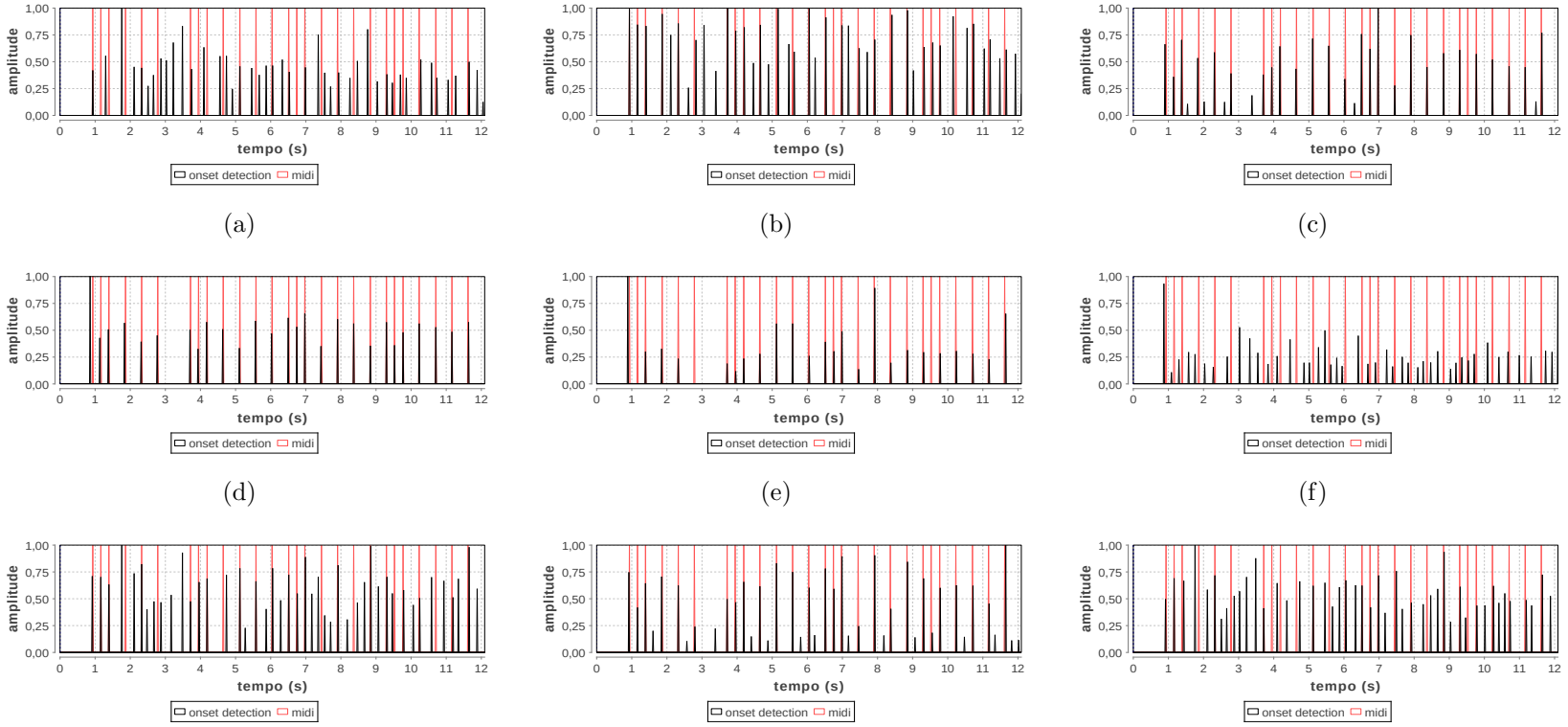


Figura 5.15: Voz Hiyama Kiyoteru sinal 2, sinal de saída dos algoritmos e referência `MIDI` (em vermelho): a) CD, b) CDS, c) DE, d) DRE, e) HFC, f) PD, g) RCD, h) SF e i) WPD.

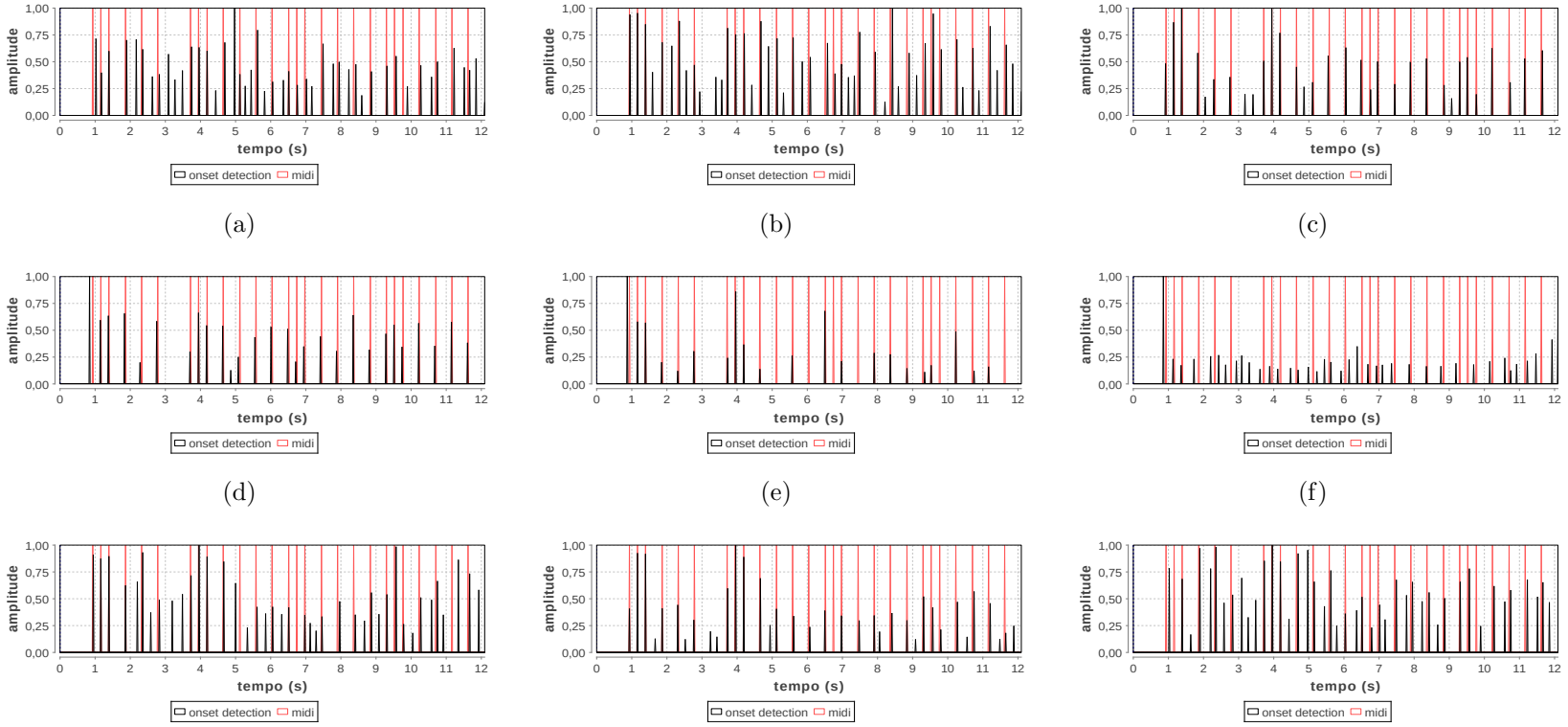


Figura 5.16: Voz Hatsune Miku sinal 2, sinal de saída dos algoritmos e referência `MIDI` (em vermelho): a) CD, b) CDS, c) DE, d) DRE, e) HFC, f) PD, g) RCD, h) SF e i) WPD.

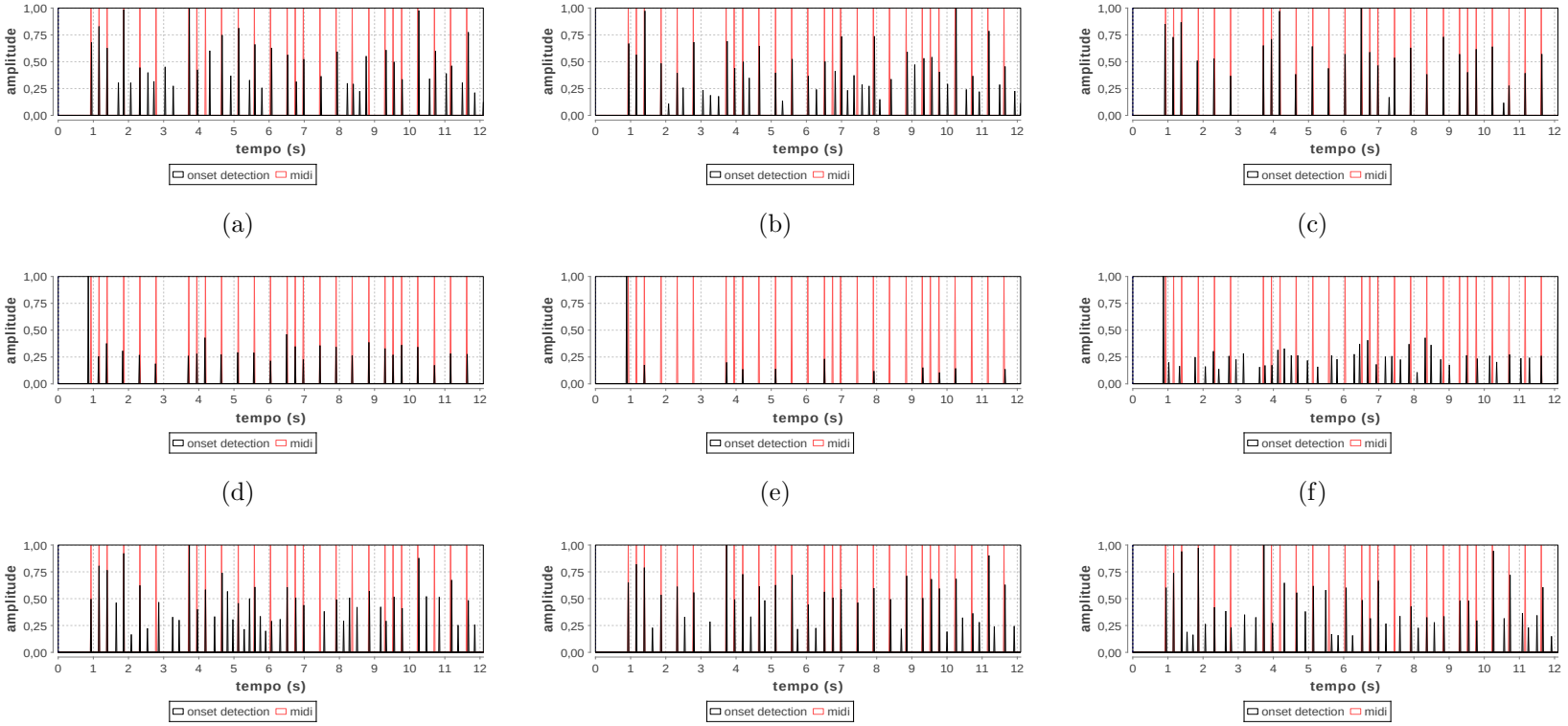


Figura 5.17: Voz Kaai Yuki sinal 2, sinal de saída dos algoritmos e referência `MIDI` (em vermelho): a) CD, b) CDS, c) DE, d) DRE, e) HFC, f) PD, g) RCD, h) SF e i) WPD.

5.6 Conclusão

Neste capítulo, foram estudados, implementados e avaliados algoritmos de detecção de início de notas, também conhecidos por detectores de *onsets*. Etapas de pré e pós-processamento foram empregadas e alguns passos presentes nessas etapas criados, de modo a melhorar o sinal de saída de todos os algoritmos apresentados.

Os algoritmos foram avaliados tendo como referência os sinais 1 da avaliação dos algoritmos do capítulo anterior, sendo este um sinal com a escala cromática, e o sinal referente à música "Parabéns a você", com suas partituras mostradas nas Figuras 4.16(a) e 4.1, respectivamente, para três vozes sintetizadas pelo *software* **Vocaloid**: Hatsune Miku, Kaai Yuki e Hiyama Kiyoteru.

Buscou-se utilizar um algoritmo de estimação de detecção de *onsets* que apresentasse o melhor desempenho possível, levando em consideração principalmente as classes de precisão (**Precisão %**) e de eventos não detectados (**END**) tendo como referência sinais advindos dos arquivos `MIDI` baseados nas partituras anteriormente mencionadas.

Segundo a avaliação realizada, o algoritmo **DRE** apresentou melhor desempenho que os demais em todos os sinais. Um ponto a ressaltar está no fato de que este algoritmo obteve na avaliação dos sinais precisão mínima de 91,6667 % e máxima de 100,00 %. Deste modo, o algoritmo da Derivada Relativa da Envoltória - **DRE** é o algoritmo detector de início de notas escolhido para compor a aplicação final a que se propõe esta dissertação, devido ao seu desempenho superior em relação aos demais estimadores abordados.

Capítulo 6

Representação melódica

A etapa denominada Representação Melódica ou *Melody Representation* é responsável por gerar uma representação simbólica de melodia, tenha sido ela solfejada ou obtida através de arquivo em formato `MIDI`. Para isso, utiliza de informações de início de notas e de frequência fundamental como ponto de partida, convertendo as informações de alto nível de melodia em um formato reduzido.

O formato definido consiste na tripla [*Onset*, *Offset*, n_{MIDI}], representando assim início e fim (em segundos) e a altura como um número inteiro (em notação `MIDI`) das notas de uma melodia. Este formato não deve ser interpretado como sendo, *a priori*, o modelo utilizado pelos algoritmos de comparação melódica no processo de busca, mas sim como um formato intermediário, o qual servirá de base para a extração dos dados no modelo em que cada algoritmo de busca trabalha.

No diagrama mostrado na Figura (1.1), é possível observar que tanto o sinal de solfejo fornecido pelo usuário quanto os sinais advindos dos arquivos em formato `MIDI` da base **Melodias MIDI** necessitam estar na mesma representação melódica (de alto nível), para que seja possível posteriormente buscar as músicas mais próximas em semelhança ao áudio-consulta do usuário através de algoritmos de comparação de similaridades. As etapas nomeadas de **Processamento MIDI** e **Processamento WAV** realizam a conversão dos sinais de formato `MIDI` e `WAV` para a representação melódica, respectivamente.

O processamento realizado utiliza dos sinais de *onsets* e *pitch tracking* como entrada para a etapa de **Combinação em notas musicais**, e em seguida a etapa **Quantização de alturas de notas musicais**, como mostra o diagrama da Figura 6.1.

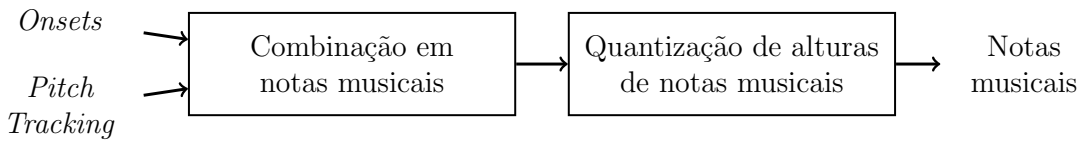


Figura 6.1: Diagrama de blocos do processamento de conversão do sinal em formato `WAV` para a representação melódica.

Combinação em notas musicais: Este bloco é responsável por definir a localização de fim das notas musicais, produzindo como saída o conjunto de notas com início e fim (em segundos) e o conjunto de alturas delimitadas por este intervalo de cada nota musical.

A localização de término de uma nota musical será aqui estabelecida, a princípio, como a localização de início da nota sucessora. Para a última nota, o término é configurado como a duração em segundos do sinal de *pitch tracking*. Após essa atribuição inicial, é realizada uma correção para que a marcação seja mais fiel ao real fim da nota: o *offset* é corrigido para a localização temporal da última amostra do intervalo [*onset*, *offset*] cuja amplitude seja superior a zero.

Quantização de alturas de notas musicais: Este bloco tem como objetivo definir uma única altura para cada nota musical. Devido ao valor médio das alturas do conjunto fornecido na etapa anterior ser sensível a erros de oitava e a outros desvios de altura (como vibrato, por exemplo), como expõe PAUWS [33], utilizou-se a mediana em seu lugar.

A Equação (6.1) realiza a conversão do valor de *pitch* de Hz para a notação `MIDI`, sem arredondamento, considerando a nota Lá de 440 Hz como referência para a escala temperada e sua indicação 69 em `MIDI`:

$$n_{\mathrm{MIDI}}(f) = 69 + 12.\log_2\left(\frac{f}{440}\right), \tag{6.1}$$

onde f representa a mediana do conjunto de alturas de uma nota e n_{MIDI} a altura da nota musical em valor inteiro limitado de 0 a 127.

Alguns trabalhos realizam a correção do desajuste entre a afinação da transcrição e a escala temperada através de métodos de normalização de *pitch*, como em [36], [37], [38] e [39]. No entanto, optou-se pela conversão direta em `MIDI`, sem arredondamento, neste trabalho.

A Tabela 6.1 mostra a representação melódica para a música "Parabéns a você" de formato `MIDI` cujo *pitch tracking* encontra-se na Figura (4.3).

Tabela 6.1: Representação melódica da música: “Parabéns a você” para o sinal de formato `WAV` apresentado na Figura 4.3.

Parabéns a você		
Onset	***Offset***	***Pitch***
0,9302	1,1628	67,0000
1,1628	1,3953	67,0000
1,3953	1,8605	69,0000
1,8605	2,3256	67,0000
2,3256	2,7907	72,0000
2,7907	3,7209	71,0000
3,7209	3,9535	67,0000
3,9535	4,1860	67,0000
4,1860	4,6512	69,0000
4,6512	5,1163	67,0000
5,1163	5,5814	74,0000
5,5814	6,0465	72,0000
6,0465	6,5116	72,0000
6,5116	6,7442	76,0000
6,7442	6,9767	76,0000
6,9767	7,4419	79,0000
7,4419	7,9070	76,0000
7,9070	8,3721	72,0000
8,3721	8,8372	71,0000
8,8372	9,3023	69,0000
9,3023	9,5349	77,0000
9,5349	9,7674	77,0000
9,7674	10,2326	76,0000
10,2326	10,6977	72,0000
10,6977	11,1628	74,0000
11,1628	11,6279	72,0000
11,6279	12,0930	72,0000

Capítulo 7

Algoritmos de comparação de melodia

7.1 Introdução

Após a realização da etapa de transcrição dos sinais, tanto das músicas da base de dados quanto do sinal de solfejo informado pelo usuário, a etapa de comparação de similaridades de melodias (também conhecida como *Melody Matching*) é então acionada no sistema de *Query by Humming*, conforme pode ser observado no diagrama da Figura 1.1. Esta etapa tem como objetivo medir o grau de similaridade entre as melodias da base de dados e a melodia do áudio-consulta do usuário, de modo a permitir que o sistema liste, dentre aquelas, as músicas mais assemelhadas a esta última.

O diagrama da Figura 7.1 mostra a estrutura mais comumente usada nos algoritmos de comparação de melodia. A abordagem tem como requisito a conversão das melodias em notas discretas. Outra, no entanto, é também encontrada na literatura: a abordagem derivada da envoltória de *pitch* contínuo [40], [39]. Os trabalhos que utilizam esta última abordagem reportam os resultados obtidos como promissores, apesar de que o processamento apresenta-se muito mais lento comparado com o requerido pela primeira. No presente trabalho, foram abordados e implementados apenas os da primeira abordagem, baseada num formato de alto nível, pelo seu menor tempo de processamento e por ser a mais frequentemente adotada nos trabalhos da literatura recente.

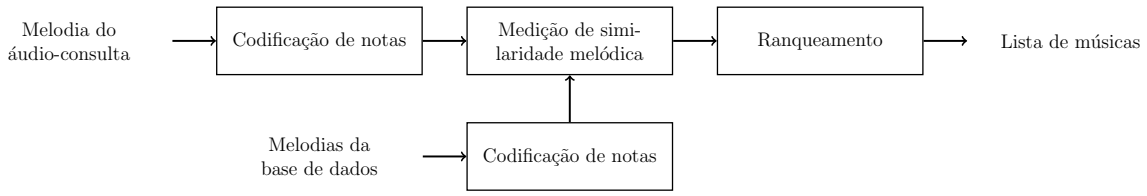


Figura 7.1: Diagrama de blocos da etapa da etapa de *Melody Matching*.

A melodia do usuário representada em notas discretas é codificada (obedecendo as particularidades do algoritmo de comparação utilizado), assim como cada melodia da base de dados de melodias, com o intuito de convertê-las para uma representação mais apropriada ao método de comparação do algoritmo. A medição de similaridade produz, então, um valor escalar, conhecido por distância ou custo, que servirá de parâmetro na determinação da lista de músicas mais semelhantes, gerada no processo de ranqueamento.

A melodia fornecida pelo usuário será indicada por $X = [x_1, x_2, \ldots, x_m]$, contendo esta a sequência de notas-alvo. Já as melodias da base de dados serão indicadas por $Y = [y_1, y_2, \ldots, y_n]$, onde cada sequência de notas é "confrontada" com X no intuito de medir o grau de similaridade entre elas.

7.2 Codificação de notas

Em [41], os autores DANNENBERG e HU abordam como o algoritmo desenvolvido por eles pode realizar a pesquisa nas distintas codificações de sequência melódica. Adotando o conjunto $S = [N_1, N_2, \ldots, N_n]$, onde N_i representa a i-ésima nota do total de n na sequência de notas da melodia, as alturas podem ser expressadas nas formas:

1. ***Pitch* Absoluto**: Utiliza a representação `MIDI` de altura de notas.

$$P_{\text{abs}}(N_i) \in \{1, 2, \ldots, 127\}. \tag{7.1}$$

2. ***Pitch* Relativo**: Utiliza as diferenças de *pitch* absoluto das alturas de notas consecutivas da sequência S.

$$P_{\text{rel}}(N_i) = \begin{cases} P_{\text{abs}}(N_i) - P_{\text{abs}}(N_{i-1}), & 2 \leq i \leq n \\ 0, & i = 1. \end{cases} \tag{7.2}$$

3. **Código de Parson (*Parson's Code*)**: Usado em [42] e [43] este algoritmo converte a sequência de notas em uma sequência de transições de alturas re-

lativas como segue:

$$P_{\text{Parson}}(N_i) = \begin{cases} \text{'R'}, & \text{se } N_i = N_{i-1} \\ \text{'U'}, & \text{se } N_i > N_{i-1} \\ \text{'D'}, & \text{em caso contrário.} \end{cases} \tag{7.3}$$

onde $2 \leq i \leq n$. No entanto, devido ao fato de a quantização da altura das notas adotada conforme a Equação (6.1) resultar em valores não-inteiros, a definição acima foi ajustada para a que segue:

$$P_{\text{Parson}}(N_i) = \begin{cases} \text{'R'}, & \text{se } |N_i - N_{i-1}| < S \\ \text{'U'}, & \text{se } N_i - N_{i-1} \geq S \\ \text{'D'}, & \text{em caso contrário,} \end{cases} \tag{7.4}$$

onde S representa 1 semitom.

De modo similar, quanto ao tempo, as notas podem ser expressas como:

1. ***Inter-onset-interval* (IOI)**: Representa o intervalo de *onsets* de notas consecutivas. Em [41] também foi adotado que o *offset* de cada nota equivale ao *onset* da nota posterior na sequência melódica.

$$\begin{aligned} T_{\text{IOI}}(N_i) &= t_{\text{onset}}(N_{i+1}) - t_{\text{onset}}(N_i) \\ &= t_{\text{offset}}(N_i) - t_{\text{onset}}(N_i), \end{aligned} \tag{7.5}$$

onde $1 \leq i < n$, $t_{\text{onset}}(N_i)$ indica a localização temporal de início da nota N_i e $t_{\text{offset}}(N_i)$ a localização temporal de fim desta nota.

2. **IOI *Ratio* (IOIR)**: Representa a proporção dos intervalos entre *onsets* de notas consecutivas.

$$T_{\text{IOI}}(N_i) = \begin{cases} \frac{T_{\text{IOI}}(N_i)}{T_{\text{IOI}}(N_{i-1})}, & 2 \leq i \leq N \\ 1, & i = 1. \end{cases} \tag{7.6}$$

3. **Log IOI *Ratio* (LogIOIR)**: Representa a proporção dos intervalos entre *onsets* de notas consecutivas em logaritmo.

$$T_{\text{LogIOI}}(N_i) = \log(T_{\text{IOIR}}(N_i)), \tag{7.7}$$

onde $1 \leq i \leq n$.

7.3 Medidores de similaridade

7.3.1 Distância de *Levenshtein*

A distância *Levenshtein* ou distância de edição, normalmente empregada em busca de cadeia de caracteres, consiste na determinação do menor número de alterações necessárias para transformar uma sequência em outra.

Partindo das sequências $X = [x_1, x_2, \ldots, x_m]$ e $Y = [y_1, y_2, \ldots, y_m]$, utiliza-se programação dinâmica para a determinação de sua distância. Uma matriz de distância $D_{(\mathrm{M+1})\mathrm{x}(\mathrm{N+1})}$ é, então, construída recursivamente. Em condições iniciais a matriz possui os valores $d(i, 0) = i$ para $0 \leq i \leq m$ e $d(0, j) = j$ para e $0 \leq j \leq n$. Os valores $d(i, j)$ são obtidos como segue [42]:

$$d(i,j) = \min \begin{cases} d(i, j-1) + w(0, y_j), & \text{(inserção)} \\ d(i-1, j) + w(x_i, 0), & \text{(deleção)} \\ d(i-1, j-1) + w(x_i, y_j) & \text{(correspondência/mudança)} \end{cases} \tag{7.8}$$

$$d(0,0) = 0 \tag{7.9}$$

$$d(i,0) = d(i-1,0) + w(a_i, 0) \qquad \text{para } i \geq 1 \tag{7.10}$$

$$d(0,j) = d(0,j-1) + w(0, b_j) \qquad \text{para } j \geq 1, \tag{7.11}$$

onde $1 \leq i \leq m$, $1 \leq j \leq n$, $w(0, y_j)$ é o peso associado com a inserção de y_j, $w(x_i, 0)$ o peso associado com a deleção de x_i e $w(x_i, y_j)$ o peso associado com a substituição do elemento x_i pelo elemento y_j, sendo que $w(x_i, y_j) = 0$ quando os elementos forem correspondentes ($x_i = y_j$) e $w(x_i, y_j) > 0$ quando for necessário realizar a mudança de um elemento por outro ($x_i \neq y_j$).

A distância de *Levenshtein* utiliza os pesos associados à inserção, deleção e mudança iguais a 1 e 0 para o peso associado à correspondência. Outras abordagens, no entanto, podem ser exploradas ao empregar pesos diferentes. Em [42] foram usados $w(0, y_j) = 1$, $w(x_i, 0) = 1$, $w(x_i, y_j) = 1$ quando $x_i \neq y_j$ e $w(x_i, y_j) = 0$ quando $x_i = y_j$. Já em [8], os autores utilizaram um peso de valor -1 associado com a correspondência quando altura e duração de notas são idênticas, e um peso de valor 0 associado à nota que coincide em altura mas não coincide em duração.

O elemento $d(M + 1, N + 1)$ (no algoritmo de *Levenshtein* clássico) contém o número de alterações (inserção, deleção e mudança) necessárias para transformar a sequência X em Y ou vice-versa. Este valor é inversamente proporcional ao grau de similaridade entre as sequências X e Y.

7.3.2 *Dynamic Time Warping* - DTW

Muito empregado em reconhecimento de fala desde o fim da década de 70 [44], [45], [46] e nas áreas de vídeo, áudio e gráficos, o algoritmo DTW é um algoritmo de medição de similaridade e alinhamento entre duas sequências que devem variar em tempo e/ou velocidade.

Baseado no algoritmo de *Linear Matching*, o qual realiza a comparação de duas sequências de igual comprimento, onde a i-ésima amostra de uma sequência é comparada com a i-ésima da outra sequência, conforme NIELS [47], o algoritmo DTW define a distância entre as sequências de modo mais complexo, por possuir algumas condições ou limitações que devem ser obedecidas. Considerando a sequência $P = [p_1, p_2, \ldots, p_k] = [d(i_1, j_1), d(i_2, j_2), \ldots, d(i_K, j_K)]$ de comprimento K como o caminho ótimo entre duas sequências X e Y de comprimentos m e n, respectivamente, as condições seguem:

- **Condição de continuidade:** Segundo NIELS [47], essa condição é o coração do algoritmo DTW, e consiste em limitar os pontos de P de modo que $i_l - i_{l-1} \leq 1$ e $j_l - j_{l-1} \leq 1$, onde $2 \leq l \leq K$.

- **Condição de monotonicidade:** Os pontos de P devem ser monotonicamente ordenados com respeito ao tempo, ou seja, $i_{l-1} \leq i_l$ e $j_{l-1} \leq j_l$, onde $2 \leq l \leq K$.

- **Condição de contorno (*Boundary condition*)**: Essa condição restringe o espaço de pesquisa, de modo que os pontos $p_1 = d(0, 0)$ e $p_k = d(m+1, n+1)$, forçando a comparação entre as sequências a acontecer desde a primeira até a última amostra de cada sequência.

O cálculo da distância de edição entre duas sequências $X = [x_1, x_2, \ldots, x_M]$ de comprimento $M \in \mathbb{N}$, e $Y = [y_1, y_2, \ldots, y_M]$ de comprimento $N \in \mathbb{N}$ é realizado utilizando a abordagem da programação dinâmica. Para isso, constrói-se uma matriz D_{MxN} com as distâncias[1] entre os pontos x_i e y_j, onde $1 \leq i \leq M$ e $1 \leq j \leq N$. De modo a simplificar a definição de $d(i, j)$, DANNENBERG e HU [41] utilizaram como pesos associados com a inserção e a deleção, respectivamente: $w(x_i, 0) = k.C_{\text{ins}}$ e $w(0, y_i) = k.C_{\text{del}}$, onde k representa o peso relativo à importância de altura e tempo de notas. O peso de substituição é indicado por $w(x_i, y_j) = |P(x_i) - P(y_j)| + k.|T(x_i) - T(y_j)|$, onde $P()$ pode ser P_{abs} ou P_{rel} e $T()$ pode ser T_{IOI} ou T_{LogIOIR}. No caso de uso de IOIR, o peso deverá ser $w(a_i, b_j) = |P(a_i) - P(b_j)| + k.\max\left(\frac{T_{\text{IOIR}}(a_i)}{T_{\text{IOIR}}(b_j)}, \frac{T_{\text{IOIR}}(b_j)}{T_{\text{IOIR}}(a_i)}\right)$.

[1] Muitas medidas de distâncias podem ser adotadas; as mais comuns são a diferença de magnitude $|x_i - y_j|$ e o quadrado da diferença $(x_i - y_j)^2$.

A distância entre as sequências X e Y pode ser expressa por:

$$D(X,Y) = \frac{\sum_{l=1}^{K} d(i_l, j_l).w(l)}{\sum_{l=1}^{K} w(l)}, \tag{7.12}$$

onde K representa a quantidade de pontos do caminho ótimo e $w(l)$ é o peso, podendo ser de inclusão, deleção, correspondência ou mudança. No entanto, devido à complexidade na otimização de $\sum_{l=1}^{K} w(l)$, foi adotada neste trabalho a seguinte simplificação:

$$D(X,Y) = \frac{\sum_{l=1}^{K} d(i_l, j_l).w(l)}{K}. \tag{7.13}$$

7.4 Conclusão

Neste capítulo, foram estudados e implementados algoritmos de comparação de melodias, também conhecidos por *Melody Matching*. Etapas de codificação de alturas e durações de notas foram descritas e dois métodos de comparação foram abordados.

Buscou-se utilizar codificações de alturas em representação numérica e alfanumérica, de modo a ser possível o uso em algoritmos de diferentes abordagens. Já quanto aos métodos de comparação de melodia, os dois algoritmos aprentados para este fim são comumente utilizados na literatura, uma vez que possuem características de serem invariantes ao tempo de andamento musical e quando combinados com uma codificação apropriada, à transposição de altura de notas.

Apesar de ser seguida nesta seção a abordagem de extração de sequência de notas e a partir delas a realização da busca pelas músicas de maior semelhança com a melodia informada pelo usuário, uma segunda abordagem, que realiza a busca considerando apenas o *pitch tracking* (apesar de o processamento ser muito lento quando comparado com o processamento da outra abordagem, por considerar muitas amostras) pode ser empregada, conforme PHIWMA e SANGUANSAT [39] para melhorar o desempenho de algoritmos da abordagem aqui seguida. Este pode ser um aspecto a explorar futuramente.

Capítulo 8

Avaliação

8.1 Introdução

A saída do sistema de reconhecimento de melodias desenvolvido nesta dissertação é uma lista ordenada das músicas consideradas mais parecidas com o áudio-referência fornecido pelo usuário (diz-se, então, que cada uma ocupa uma posição no ranque). De modo a permitir o levantamento de alguns aspectos do desempenho do sistema, buscou-se responder as seguintes questões:

Questão 1: Qual algoritmo de comparação de melodias teve melhor desempenho quanto ao reconhecimento das músicas da base de dados?

Questão 2: Qual tipo de gravação de solfejo[1] teve melhor reconhecimento pelo sistema para as dez músicas com maior número de gravações (para garantir uma estatística aceitável)?

Questão 3: Levando em consideração os tipos de gravação de solfejo e os algoritmos de comparação de melodias, qual a probabilidade de reconhecimento pelo sistema nas dez primeiras posições do ranque para a base de dados de solfejos?

Questão 4: Para as dez músicas com maior número de gravações de solfejos (para garantir uma estatística aceitável), qual a ocorrência de reconhecimentos por posição do ranque, considerando separadamente algoritmos de comparação de melodias e tipos de gravação de solfejo?

Questão 5: Quais as músicas da base de dados com maior percentual de reconhecimento nas dez primeiras posições do ranque?

Os algoritmos de comparação melódica abordados são definidos a seguir:

[1] Tipo 1 - Com acompanhamento `MIDI` de piano; Tipo 2 - Com acompanhamento `WAV` de gravação comercial; e Tipo 3 - Sem acompanhamento algum.

Algoritmo 1: Utiliza o **Código de Parson** para codificação de notas e **Distância de *Levenshtein* Clássico**, ou seja, com pesos associados à inclusão, deleção e mudança iguais a 1, e peso de valor 0 (zero) para pontos de correspondência.

Algoritmo 2: Utiliza a codificação de ***Pitch* Absoluto** obtido pela conversão de frequência de Hz para notação `MIDI` sem arredondamento apresentada na Equação 6.1 e algoritmo ***Dynamic Time Warping***, com pesos associados à inclusão, deleção, mudança e correspondência iguais a 0 (zero).

Algoritmo 3: Utiliza a codificação de ***Pitch* Relativo** e algoritmo ***Dynamic Time Warping***, com pesos associados à inclusão, deleção, mudança e correspondência, iguais a 0 (zero).

Adotou-se como principal medida a contagem de vezes que as músicas foram encontradas pelo sistema em cada uma das posições possíveis do ranque, levando em consideração todas as gravações desta pelos usuários, sob determinado aspecto a ser avaliado.

Num caso, também foi utilizada outra medida, conhecida por MRR (***Mean Reciprocal Ranking***), definida a seguir:

Mean Reciprocal Ranking - Medida estatística usada na avaliação de qualquer processo que produza uma lista de possíveis respostas a uma referência, ordenadas por probabilidade de acerto. A equação a seguir formaliza matematicamente sua definição [48]:

$$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{rank}_i}, \tag{8.1}$$

onde rank_i representa a posição no ranque e N à quantidade total de medições. Quanto mais próximo de 1 for o valor em **MRR**, melhor é o desempenho do que se está avaliando.

Como exemplo, imagine que três usuários solfejam uma mesma música e o sistema a classifica nas posições de números 1, 2 e 3 do ranque. O valor em MRR, levando em consideração essas três medições, será de $\frac{1}{3}\left(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}\right) \approx 0{,}61$.

8.2 Resultados

Apresentam-se a seguir os resultados provenientes da avaliação do desempenho do sistema para cada uma das questões:

8.2.1 *Questão 1*: Qual algoritmo de comparação de melodias teve melhor desempenho quanto ao reconhecimento das músicas da base de dados?

A Figura 8.1 confronta os três algoritmos de comparação de melodias através de um gráfico de quantidades acumuladas de vezes que as músicas foram encontradas por posição do ranque. Essa figura mostra de forma geral que o Algoritmo 3 teve o melhor desempenho, uma vez que atinge valores maiores que os dos outros algoritmos desde as posições iniciais.

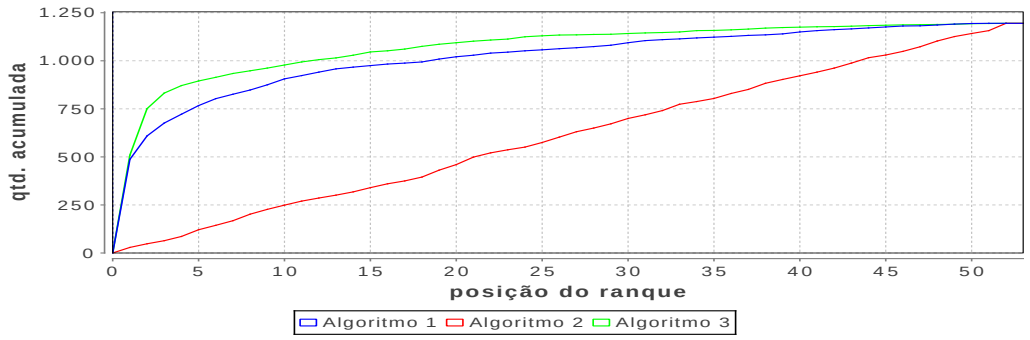


Figura 8.1: Comparativo entre os algoritmos através da quantidade acumulada de vezes que o sistema encontrou as músicas por posição do ranque.

Uma outra forma de avaliar essa questão é através da análise da medida MRR levando em consideração os algoritmos e o tipo de gravação, como mostra a Tabela 8.1. Nela, é possível ver que o Algoritmo 3 possui os valores mais altos de MRR para todos os tipos de gravação. A última linha da tabela mostra os valores MRR para todos os tipos de gravação de forma sintética. O Algoritmo 1 tem comportamento próximo ao do Algoritmo 3. Isso se deve ao fato de que estes algoritmos são invariantes à transposição de altura de notas, pois utilizam **Código de Parson** e ***Pitch* Relativo**, respectivamente, como codificação de altura de notas. A análise é feita horizontalmente, ou seja, fixando-se o tipo de gravação vemos o comportamento dos algoritmos, pois temos a mesma quantidade de gravações sendo avaliada.

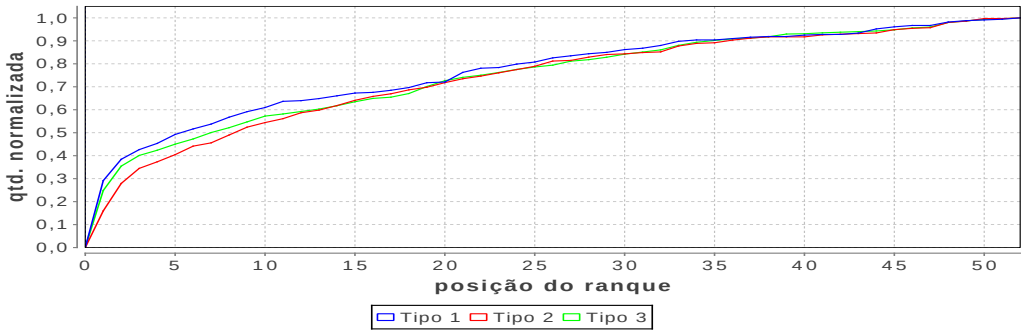
Tabela 8.1: Tabela comparativa dos valores medididos em MRR para os algoritmos de comparação de melodias por tipo de gravação.

Tipo	**MRR Algoritmo 1**	**MRR Algoritmo 2**	**MRR Algoritmo 3**
1	0,5590	0,0859	0,6041
2	0,4828	0,0900	0,5473
3	0,5217	0,0929	0,5854
	0,5209	0,0906	0,5806

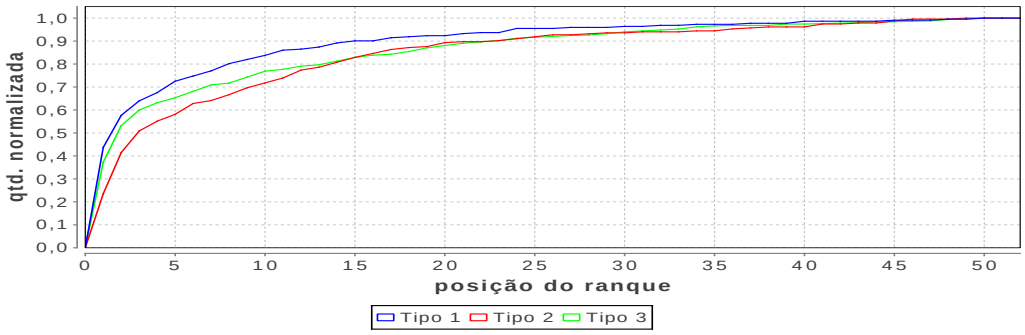
8.2.2 *Questão 2*: Qual tipo de gravação de solfejo teve maior reconhecimento pelo sistema para as dez músicas com maior número de gravações?

Essa questão foi abordada através do reconhecimento das músicas contando as ocorrências de encontro das dez músicas com maior número de gravações (vide Tabela A.1) em todas as posições do ranque. Uma vez obtidas, essas contagens foram normalizadas pelo total de gravações em cada tipo e, por sua vez, acumuladas partindo da posição inicial do ranque. Como as quantidades de gravações em cada tipo de gravação não são iguais, esse procedimento foi necessário para possibilitar a comparação entre eles.

A Figura 8.2(a) foi gerada levando em consideração todos os algoritmos de comparação de melodias abordados nesta dissertação, enquanto que para a Figura 8.2(b) apenas os Algoritmos 1 e 3 foram considerados, por serem os únicos que produziram resultados aceitáveis. Nota-se que em ambas as figuras o tipo de gravação 1 apresenta melhores medições, seguido dos tipos 2 e 3. Explica-se este comportamento devido ao fato de as gravações realizadas através do acompanhamento de `MIDI` de piano (tipo 1) seguirem a melodia (tanto em andamento quanto em tonalidade) de referência para a comparação de melodias. Isso favoreceu o seu melhor desempenho, ficando isso mais evidente no primeiro gráfico da primeira até a posição do ranque de número onze, aproximadamente. Já no segundo gráfico, isso fica evidente da primeira até a posição de número quinze. Percebeu-se que os solfejos que foram gravados sem acompanhamento (tipo 3) apresentaram em grande parte andamento mais lento que a versão `MIDI` das músicas. Isso favoreceu o reconhecimento deste tipo pelo sistema, em relação aos solfejos gravados com o acompanhamento `WAV` das versões comerciais das músicas (tipo 2).



(a) Levando em consideração os Algoritmos 1, 2 e 3.



(b) Levando em consideração, apenas, o Algoritmo 1 e Algoritmo 3.

Figura 8.2: Comparativo entre os algoritmos através da quantidade acumulada de vezes que o sistema encontrou as músicas por posição do ranque.

8.2.3 *Questão 3*: Levando em consideração os tipos de gravação de solfejo e os algoritmos de comparação de melodias, qual a probabilidade de reconhecimento pelo sistema nas dez primeiras posições do ranque para a base de dados de solfejos?

A Tabela 8.2 mostra a porcentagem de acerto do sistema para a base de dados de solfejos considerando os tipos de gravação e algoritmos de comparação de melodias. A porcentagem foi calculada a partir da contagem de classificação no ranque de todas as gravações de solfejos para cada tipo de gravação e algoritmo de comparação de melodias utilizado pelo total de gravações do tipo e algoritmo em questão. Nota-se que o Algoritmo 3 mostrou-se sistematicamente um pouco melhor que o Algoritmo 1, e o Algoritmo 2 foi incapaz de fornecer resultados aceitáveis. Em particular, considerando os arquivos do tipo 1, o Algoritmo 3 chegou a 46% de acerto na primeira posição do ranque contra 45,85 % do Algoritmo 1 e 1,81 % do Algoritmo 2.

Tabela 8.2: Probabilidade de acerto em porcentagem para tipo de gravação e algoritmo de comparação de melodias considerando as dez primeiras posições do ranque.

Tipo	**Algoritmo**	**1ª**	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**	**10ª**
1	1	45,85	9,03	3,97	3,61	4,33	2,17	1,08	2,89	3,25	2,17
	2	1,81	2,17	1,08	2,53	2,53	3,25	1,08	1,44	2,17	1,44
	3	46,57	16,97	6,14	2,89	2,53	2,89	2,17	1,44	1,44	0,72
2	1	36,11	10,42	5,90	4,86	4,17	3,82	1,39	2,43	1,74	1,74
	2	2,08	2,08	2,08	1,04	2,78	1,04	1,39	3,47	2,78	1,74
	3	37,85	21,53	8,68	3,47	2,08	1,39	1,39	1,39	1,04	2,43
3	1	40,63	10,63	6,19	3,49	3,33	3,02	2,54	1,11	2,06	3,17
	2	2,86	1,11	1,11	1,90	3,17	1,75	2,70	3,17	1,75	2,06
	3	43,02	21,11	6,19	3,33	1,75	1,11	1,59	0,95	1,11	1,11

8.2.4 *Questão 4*: Para as dez músicas com maior número de gravações de solfejos, qual a ocorrência de acertos por posições do ranque, considerando separadamente algoritmos de comparação de melodias e tipos de gravação de solfejo?

Os gráficos do Apêndice D mostram as quantidades de ocorrências normalizadas pelo total de gravações por posição no ranque de cada uma das dez músicas com maior número de gravações, vistas por algoritmo de comparação de melodias e por tipo de gravação (neste caso, sem a influência do Algoritmo 2). Segue a respectiva discussão:

- **Hino Nacional:** Vê-se clara predominância do Algoritmo 3 na Figura D.1(a). Na Figura D.1(b), o tipo 1 forneceu os maiores valores nas posições iniciais do ranque.

- **Parabéns a você:** Novamente, na Figura D.2(a) os melhores resultados foram os do Algoritmo 3. Na Figura D.2(b), os tipos 1 e 3 tiveram comportamente próximo, sendo este porém mais consistente.

- **Anna Júlia:** Na Figura D.3(a), o Algoritmo 3 alcançou os melhores resultados. No gráfico D.3(b), novamente o tipo 3 resultou mais consistente que o tipo 1, embora tenham levado a desempenhos próximos. De fato, esta música não

foi reconhecida preferencialmente na primeira posição, isto é, foi uma música "difícil" para o sistema.

- **Ciranda cirandinha:** Para a primeira posição do ranque, o Algoritmo 3 alcançou o melhor resultado na Figura D.4(a), enquanto que os tipos 1 e 3 se equilibraram na Figura D.4(b).

- **Asa branca:** Na Figura D.5(a), os Algoritmos 1 e 3 se equilibraram, com alguma avantagem para este último. Na Figura D.5(b), nota-se que os resultados localizados nas três primeiras colocações do ranque são dominantes, tendo o tipo de gravação 3 o melhor desempenho na primeira posição. Pode-se dizer que foi uma música "fácil" para o sistema.

- **Garota de Ipanema:** Pela Figura D.6(a), observa-se que na primeira posição do ranque venceu o Algoritmo 1. Já para o gráfico da Figura D.6(b) os tipos 1 e 3 obtiveram desempenhos próximos.

- **Você não soube me amar:** Pela Figura D.7(a), o Algoritmo 1 foi o único a concentrar os resultados na primeira posição. Os tipos 1 e 3 novamente se equilibraram na Figura D.7(b).

- **Eu sei que vou te amar:** Nota-se na Figura D.8(a) alguma vantagem do Algoritmo 3 sobre o Algoritmo 1. Já na Figura D.8(b), todos os tipos de gravação resultaram equivalentes em comportamento, embora o Tipo 1 ter atingido melhor medição para a primeira posição.

- **Que país é este?:** Pela Figura D.9(a), observa-se que a melhor concentração nas primeiras posições do ranque foi obtida pelo Algoritmo 1. Já na Figura D.9(b), o melhor resultado ocorreu para as gravações do tipo 1.

- **País tropical:** Novamente, na Figura D.10(a) predomina o Algoritmo 1, com alta concentração nas 2 primeiras posições. Na Figura D.4(b), tipos 1 e 3 se equilibram.

8.2.5 *Questão 5*: Quais as músicas da base de dados com maior percentual de acerto quanto ao seu reconhecimento pelo sistema nas dez primeiras posições do ranque?

Na Tabela A.3 é possível encontrar a lista de todas as músicas, ordenadas pelo percentual de acerto em cada uma das dez primeiras posições do ranque. A Tabela A.4 mostra os valores excluindo as medições do Algoritmo 2, devido a este não ser

invariante à altura de notas e por isso ter apresentado resultados sistematicamente ruins. Segundo a primeira tabela, a música com maior percentual de acerto é a música "Ilariê" alcançando 71,17 % de precisão do sistema na primeira posição do ranque. Já na segunda tabela, a música com maior percentual de acerto é a "Inútil", com 87,50 % para a primeira posição do ranque. É fácil entender por que ambas foram as mais fáceis: ambas têm grande quantidade de notas repetidas, o que lhes confere um padrão muito particular.

8.3 Conclusão

Neste capítulo, foi avaliado o desempenho da aplicação desenvolvida nesta dissertação sobre a base de dados de músicas brasileiras. As medidas utilizadas para isso foram desde a contagem de ocorrências em cada posição do ranque, ora completo (cinquenta e duas posições), ora restrito às dez primeiras posições — para prover uma estatística melhor — até o uso de valores normalizados e em porcentagem, além da **Mean Reciprocal Ranking**, utilizada em vários trabalhos na literatura desta linha de pesquisa.

Avaliou-se o desempenho dos algoritmos de comparação de melodias quanto ao reconhecimento das músicas presentes no conjunto de dados, assim como qual tipo de gravação de solfejo obteve melhores resultados para o conjunto das dez músicas com maior número de gravações; investigou-se a probabilidade de acerto do sistema nas dez primeiras posições do ranque; obteve-se também a ocorrência de acertos por posições do ranque para as dez músicas com maior número de gravações de solfejos, observando separadamente os algoritmos de comparação de melodias e os tipos de gravação adotados neste trabalho; e por último, levantou-se a lista das músicas com maior percentual de acerto pelo sistema observando as dez primeiras posições do ranque.

Em linhas gerais, os resultados encontrados foram os seguintes.

- O Algoritmo 2, por se basear em altura absoluta, não conseguiu fornecer resultados aceitáveis.

- Os Algoritmos 1 e 3 tiveram desempenho em média similar, entretanto houve tanto músicas em que seus desempenhos se equilibraram quanto músicas em que um deles se destacou fortemente em relação ao outro. Esse comportamento merece ser investigado.

- O grau de dificuldade oferecido pelos tipos de gravação foi esperado no que se refere às gravações acompanhando `MIDI` (mais fáceis, já que muito mais próximas do próprio padrão utilizado para comparação). Entretanto, foi uma

surpresa verificar que as gravações *a capella* foram mais fáceis que as acompanhadas de gravação comercial. Isso tem duas possíveis explicações: as gravações *a capella* tenderem a ser mais lentas; e os intérpretes nas gravações comerciais fugirem interpretativamente das melodias originais, o que o usuário instintivamente tende a copiar.

- Houve músicas que foram difíceis para o sistema; seriam muito próximas a outras? Já as duas ocorrências mais evidentes de músicas fáceis continham muitas notas repetidas, gerando um padrão pouco comum.

Capítulo 9

Conclusão

Neste trabalho, abordou-se o tema de pesquisa de músicas através de solfejos, tendo sido descrito e desenvolvido um sistema de *Query by Humming*, por sua vez testado sobre uma base de dados de músicas brasileiras.

Como principais contribuições do trabalho, podemos citar as seguintes:

- Construiu-se uma base de dados de solfejos composta exclusivamente de músicas brasileiras, até então inexistente na literatura. A escolha das músicas se deu pela filtragem e extensão de uma listagem publicada pela Revista Rolling Stone [10]. A base contém 52 músicas de referência em formato `MIDI` contendo o trecho mais popular (início ou refrão) de cada música. Os solfejos totalizam 1.015 gravações feitas em ambiente preparado com equipamento profissional por 43 usuários, de três formas: acompanhando o `MIDI` de referência, ou acompanhando uma gravação comercial ou *a capella*.

- Desenvolveu-se um sistema completo de *Query by Humming*. Dentre suas características, destacam-se:

  - ***Software* livre**: Atende aos quatro tipos de liberdade aos usuários: liberdade para executar o programa para qualquer propósito; liberdade de estudar o *software*; liberdade de redistribuir cópias do programa, tendo como finalidade ajudar ao próximo; e por fim, liberdade de modificar o programa e distribui-lo para que a comunidade se beneficie.

  - **Extensibilidade**: Cada etapa do sistema é dividida em módulos, podendo cada um deles ser estendido a partir da inclusão de novos métodos e algoritmos;

  - **Interface amigável**: O sistema apresenta telas e componentes com textos/rótulos/dicas autoexplicativos, além de apresentar gráficos manuseáveis (com possibilidade de aplicação de *zoom* e exportação em for-

matos de imagens) das etapas do sistema: *pitch tracking*, detecção de *onsets* e representação de melodia.

- **Configurável**: A tela de configuração do sistema permite a edição do valor de cada constante usada por algoritmo, assim como qual algoritmo será usado em cada etapa do sistema.
- **Base de dados de músicas de fácil criação**: Pelo sistema, é possível adicionar músicas na base de dados a partir da importação de músicas em formato `MIDI`, assim como excluir músicas. Isso permite que o sistema seja utilizado tanto para conjuntos personalizados de músicas como em outras bases de dados, compostas neste formato e encontradas em outros trabalhos na literatura.

  Uma possível funcionalidade a ser acrescentada é a possibilidade de o usuário inserir na base de dados músicas solfejadas, a partir da representação melódica adotada atualmente no sistema.

Algumas informações técnicas acerca do sistema e de passos para a sua utilização são encontrados no Apêndice C.

- Documentação detalhada das etapas envolvendo a construção do sistema. Foram abordados métodos/algoritmos em cada etapa e estes foram avaliados (confrontados entre si) de modo a permitir a escolha do mais apropriado em cada tarefa para a aplicação final, buscando o estado da arte em cada tema.

- O sistema foi avaliado sobre a base de dados, como um guia para a continuação das pesquisas.

Outras contribuições são dadas em todas as etapas do sistema, que vão desde alguns métodos adicionais empregados em pré ou pós-processamentos até a avaliação dos algoritmos e do sistema como um todo.

O refinamento dos métodos empregados (por exemplo, a utilização de um conjunto de pesos mais diversificado nos métodos de comparação de melodia) e a inclusão de mais algoritmos (por exemplo, algoritmos de estimação de frequência fundamental e de comparação de melodias) nas etapas do sistema podem ser explorados futuramente, assim como a utilização de repositórios *online* de músicas `MIDI` ou até mesmo a migração do aplicativo *desktop* para a plataforma `WEB` ou *mobile*, com o intuito de torná-lo mais acessível aos usuários.

Referências Bibliográficas

[1] MIDOMI. Disponível em: <http://www.midomi.com>. Último acesso em Dezembro de 2013.

[2] MUSIPEDIA. "The Open Music Encyclopedia". Disponível em: <http://www.musipedia.org>. Último acesso em Dezembro de 2011.

[3] DANNENBERG, R. B., BIRMINGHAM, W. P., PARDO, B., et al. "A comparative evaluation of search techniques for query-by-humming using the MUSART testbed", *Journal of the American Society of Information and Science Technology*, v. 58, n. 3, pp. 687–701, March 2007. ISSN: 1532-2882. doi: 10.1002/asi.v58:5. Disponível em: <http://dl.acm.org/citation.cfm?id=1231003.1231010>.

[4] DANNENBERG, R. B., BIRMINGHAM, W. P., PARDO, B. "Query by humming with the VocalSearch system", *Communications of the ACM - Music information retrieval*, v. 49, n. 8, pp. 49–52, Agosto 2006.

[5] PITCHPERFECTOR. Disponível em: <http://www.onpitchsinging.com/>. Último acesso em Dezembro de 2013.

[6] THORPE, W., CALLAGHAN, J., WILSON, P. "Sing & See". Disponível em: <http://www.singandsee.com>. Último acesso em Dezembro de 2013.

[7] FURUI, S. *Digital Speech Processing, Synthesis, and Recognition.* 2 ed. New York, USA, Marcel Dekker, Inc., 2000.

[8] LÓPEZ, E., ROCAMORA, M., SOSA, G. "Búsqueda de Música por Tarareo". Julho 2004. Projeto Final, Universidad de la República Oriental del Uruguay, Montevideo, Uruguay.

[9] ZHU, Y., SHASHA, D. "Warping indexes with envelope transforms for query by humming". In: *Proceedings of the SIGMOD 2003 (International Conference on Management of Data)*, San Diego, USA, Junho 2003. ACM.

[10] ROLLING STONE. "As 100 maiores músicas brasileiras". Outubro 2009. n. 37.

[11] PARK, T. H. *Introduction to Digital Signal Processing - Computer Musically Speaking*. Singapore, World Scientific, 2009. ISBN: 978-981-279-027-9. Disponível em: <http://www.worldscibooks.com/compsci/6705.html;http://www.bibsonomy.org/bibtex/2581bbba5ead3ab429fdc81c63a846347/dblp>.

[12] HSU, C.-L., WANG, D., JANG, J.-S. R. "A trend estimation algorithm for singing pitch detection in musical recordings". In: *Proceedings of the ICASSP'11 (International Conference on Acoustics, Speech, and Signal Processing)*, pp. 393–396, Prague, Czech Republic, Maio 2011. IEEE.

[13] LUENGO, I., SARATXAGA, I., NAVAS, E., et al. "Evaluation of pitch detection algorithms under real conditions". In: *Proceedings of the IACSSP'07 (International Conference on Acoustics, Speech, and Signal Processing)*, v. 4, Honolulu, USA, Abril 2007. IEEE. doi: 10.1109/ICASSP.2007.367255.

[14] KLAPURI, A., DAVY, M. *Signal Processing Methods for Music Transcription*. New York, USA, Springer-Verlag, 2006. ISBN: 0387306676.

[15] TADJIKOV, M., AHMADI, A. *Pitch Estimation Using a Full/Multi-Band Approaches*. Relatório técnico, University of California, Los Angeles, USA, Junho 2010. Disponível em: <http://tadjikov.com/sites/tadjikov.com/files/Final.Report.pdf>.

[16] SCHROEDER, M. R. "Period histogram and product spectrum: new methods for fundamental frequency measurement", *Journal of the Acoustical Society of America*, v. 43, n. 4, pp. 829–834, Abril 1968.

[17] CUADRA, P. D. L., MASTER, A. "Efficient pitch detection techniques for interactive music". In: *Proceedings of the International Computer Music Conference*, La Habana, Cuba, Setembro 2001.

[18] NOLL, A. M. "Cepstrum pitch determination." *Journal of the Acoustical Society of America*, v. 41, n. 2, pp. 293–309, Fevereiro 1967.

[19] RABINER, L. R., SCHAFER, R. W. "Introduction to digital speech processing", *Foundations and Trends in Signal Processing*, v. 1, n. 1/2, pp. 1–194, 2007. doi: http://dx.doi.org/10.1561/2000000001.

[20] DE CHEVEIGNÉ, A., KAWAHARA, H. "YIN, a fundamental frequency estimator for speech and music." *Journal of the Acoustical Society of America*, v. 111, n. 4, pp. 1917–1930, Abril 2002. Disponível em: <http://link.aip.org/link/JASMAN/v111/i4/p1917/s1&Agg=doi>.

[21] KOTNIK, B., HÖGE, H., KACIC, Z. "Evaluation of pitch detection algorithms in adverse conditions". In: *Proceedings of the 3rd International Conference on Speech Prosody*, Dresden, Germany, Maio 2006.

[22] KLAPURI, A. P. "Automatic Transcription of Music". In: *Proceedings of the Stockholm Music Acoustics Conference*, Stockholm, Sweden, Agosto 2003.

[23] BENETOS, E., DIXON, S. "Polyphonic music transcription using note onset and offset detection". In: *Proceedings of the ICASSP'11 (International Conference on Acoustics, Speech, and Signal Processing)*, pp. 37–40, Prague, Czech Republic, Maio 2011. doi: 10.1109/ICASSP.2011.5946322.

[24] DIXON, S. "Automatic extraction of tempo and beat from expressive performances", *Journal of New Music Research*, v. 30, pp. 39–58, 2001.

[25] STARK, A. M., PLUMBLEY, M. D. "Real-time chord recognition for live performance". In: *Proceedings of the International Computer Music Conference*, Montreal, Canada, Agosto 2009. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.152.6977&rep=rep1&type=pdf>.

[26] MACRAE, R., DIXON, S. "A guitar tablature score follower". In: *Proceedings of the ICME 2010 (International Conference on Multimedia and Expo)*, pp. 725–726, Singapore, Julho 2010. IEEE. doi: 10.1109/ICME.2010.5582963.

[27] LI, T. "Musical genre classification of audio signals", *IEEE Transactions on Speech and Audio Processing*, v. 10, n. 5, pp. 293–302, Julho 2002.

[28] SOLTAU, H., SCHULTZ, T., WESTPHAL, M., et al. "Recognition of music types". In: *Proceedings of the ICASSP'98 (International Conference on Acoustics, Speech, and Signal Processing)*, v. 2, pp. 1137–1140, Seattle, USA, Maio 1998.

[29] ROSÃO, C., RIBEIRO, R. "Trends in onset detection". In: *Proceedings of the OSDOC'11 (Workshop on Open Source and Design of Communication)*, pp. 75–81, New York, USA, Julho 2011. ISBN: 978-1-4503-0873-1. doi: http://doi.acm.org/10.1145/2016716.2016736. Disponível em: <http://doi.acm.org/10.1145/2016716.2016736>.

[30] BELLO, J. P., DAUDET, L., ABDALLAH, S., et al. "A tutorial on onset detection in music signals", *IEEE Transactions On Speech And*

Audio Processing, v. 13, n. 5, pp. 1035–1047, Setembro 2005. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1495485>.

[31] DIXON, S. "Onset detection revisited". In: *Proceedings of the DAFx-06 (International Conference on Digital Audio Effects)*, pp. 133–137, Montreal, Canada, Setembro 2006. http://www.dafx.ca/proceedings/papers/p_133.pdf.

[32] MASRI, P. *Computer modeling of sound for transformation and synthesis of musical signal*. Tese de Doutorado, University of Bristol, 1996.

[33] PAUWS, S. "CubyHum: A Fully Operational Query by Humming System". In: *Proceedings of the ISMIR 2002 (International Society for Music Information Retrieval Conference)*, pp. 187–196, Porto, Portugal, Outubro 2002.

[34] KLAPURI, A. "Sound onset detection by applying psychoacoustic knowledge". In: *Proceedings of the ICASSP'99 (International Conference on Acoustics, Speech, and Signal Processing)*, v. 6, pp. 3089–3092, Phoenix, USA, Março 1999. ISBN: 0-7803-5041-3. doi: 10.1109/ICASSP.1999.757494. Disponível em: <http://dx.doi.org/10.1109/ICASSP.1999.757494>.

[35] SCHEIRER, E. D. "Tempo and beat analysis of acoustic musical signals", *Journal of the Acoustical Society of America*, v. 103, n. 1, pp. 588–601, Janeiro 1998. doi: 10.1121/1.421129. Disponível em: <http://dx.doi.org/10.1121/1.421129>.

[36] MCNAB, R. J., SMITH, L. A., WITTEN, I. H., et al. "Towards the digital music library: tune retrieval from acoustic input". In: *Proceedings of the first ACM International Conference on Digital Libraries*, pp. 11–18, Bethesda, USA, Março 1996.

[37] POLLASTRI, E. "An audio front end for query-by-humming systems". In: *Proceedings of the ISMIR 2001 (International Symposium on Music Information Retrieval)*, pp. 65–72, Bloomington, USA, Outubro 2001. Disponível em: <http://dblp.uni-trier.de/db/conf/ismir/ismir2001.html#Pollastri01>.

[38] VIITANIEMI, T., KLAPURI, A., ERONEN, A. "A probabilistic model for the transcription of single-voice melodies". In: *Proceedings of the Finnish Signal Processing Symposium*, pp. 59–63, Tampere, Finland, Maio 2003.

[39] PHIWMA, N., SANGUANSAT, P. "An improved melody contour feature extraction for query by humming", *International Journal of Computer Theory and Engineering*, v. 2, n. 4, pp. 1793–8201, Agosto 2010. Disponível em: <http://www.ijcte.org/papers/196-H059.pdf>.

[40] MAZZONI, D., DANNENBERG, R. B. "Melody matching directly from audio". In: *Proceedings of the ISMIR 2001 (International Symposium on Music Information Retrieval)*, pp. 17–18, Bloomington, USA, Outubro 2001.

[41] DANNENBERG, R. B., HU, N. "Understanding search performance in query-by-humming systems". In: *Proceedings of the 5th International Conference on Music Information Retrieval*, Barcelona, Spain, Outubro 2004. http://ismir2004.ismir.net/proceedings/p043-page-232-paper236.pdf.

[42] TRIPATHY, A. K., CHHATRE, N., SURENDRANATH, N., et al. "Query by humming", *International Journal of Recent Trends in Engineering*, v. 2, n. 5, pp. 373–379, Novembro 2009.

[43] PRECHELT, L., TYPKE, R. "An interface for melody input", *ACM Transactions on Computer-Human Interaction*, v. 8, n. 2, pp. 133–149, Junho 2001. ISSN: 1073-0516. doi: 10.1145/376929.376978. Disponível em: <http://doi.acm.org/10.1145/376929.376978>.

[44] RABINER, L. R., ROSENBERG, A. E., LEVINSON, S. E. "Considerations in dynamic time warping algorithms for discrete word recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 26, n. 6, pp. 575–582, Dezembro 1978. Disponível em: <http://link.aip.org/link/?JAS/63/S79/1>.

[45] SAKOE, H., CHIBA, S. "Dynamic programming algorithm optimization for spoken word recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, , n. 1, pp. 43–49, Fevereiro 1978.

[46] BERNDT, D. J., CLIFFORD, J. "Using dynamic time warping to find patterns in time series". In: *Proceedings of the KDD-94 (Workshop on Knowledge Discovery in Databases)*, pp. 359–370, Seattle, USA, Abril 1994.

[47] NIELS, R. *Dynamic time warping: an intuitive way of handwriting recognition?* Tese de Mestrado, Radboud University Nijmegen, Nijmegen, The Nederlands, Novembro/Dezembro 2004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.1553>.

[48] VOORHEES, E. "The TREC-8 question answering track report". In: *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, pp. 77–82. NIST, 1999.

Apêndice A

Lista de Músicas da Base de Dados

Tabela A.1: Lista de músicas da base de dados com a quantidade de gravações por tipo (1 - com acompanhamento de `MIDI` piano, 2 - com acompanhamento de gravação comercial `WAV` e 3 - sem acompanhamento) ordenada decrescentemente pelo total de gravações.

Código	**Título**	**Tipo 1**	**Tipo 2**	**Tipo 3**	**Total**
104	Hino nacional	13	15	30	58
101	Parabéns a você	9	17	31	57
100	Anna Júlia	9	15	26	50
004	Asa branca	14	9	24	47
102	Ciranda cirandinha	12	10	25	47
027	Garota de ipanema	11	12	23	46
081	Que país é este?	6	14	24	44
097	Você não soube me amar	10	11	23	44
024	Eu sei que vou te amar	12	9	23	44
025	País tropical	15	5	22	42
103	Escravos de jó	10	9	22	41
019	Quero que vá tudo pro inferno	9	9	21	39
109	Mamãe eu quero	11	7	20	38
107	Ilariê	9	8	20	37
015	Trem das onze	7	10	18	35
067	A banda	7	8	19	34
020	Preta pretinha	12	4	17	33
072	Gita	7	8	17	32
108	Fim de Ano	7	7	16	30

Continua na próxima página

Tabela A.1 – Continuação da página anterior.

Código	Título	Tipo 1	Tipo 2	Tipo 3	Total
002	Águas de março	6	8	15	29
003	Carinhoso	8	5	15	28
010	Alegria, alegria	4	8	13	25
008	Detalhes	6	4	12	22
062	Luar do sertão	4	6	11	21
028	Pra não dizer que não falei das flores	5	4	11	20
006	Chega de saudade	4	5	9	18
083	Ideologia	2	6	9	17
087	Meu mundo e nada mais	5	3	8	16
012	Aquarela do Brasil	3	5	8	16
091	Felicidade	1	6	8	15
063	Alagados	4	3	8	15
085	O barquinho	4	3	7	14
068	Comida	3	3	7	13
074	Sentado à beira do caminho	3	2	7	12
057	Conversa de botequim	2	4	6	12
105	Hino da bandeira	4	2	6	12
031	Travessia	2	3	6	11
089	A flor e o espinho	2	2	6	10
096	Disritmia	2	2	5	9
047	Me chama	3	1	4	8
023	Inútil	1	3	4	8
111	Coelhinho de olhos vermelhos	1	2	4	7
038	Eu quero é botar meu bloco na rua	0	3	4	7
046	Ponteio	2	1	3	6
064	As curvas da estrada de Santos	2	1	3	6
070	Ronda	0	2	2	4
112	Chegou a hora da foqueira	0	2	2	4
093	Casa no campo	2	0	2	4
098	A noite de meu bem	1	1	2	4
075	Foi um rio que passou em minha vida	0	1	1	2
041	Manhã de carnaval	1	0	1	2
084	Rosa	0	0	0	0

Tabela A.2: Lista de músicas da base de dados com a quantidade de gravações saturadas por tipo (1 - com acompanhamento `MIDI` de piano, 2 - com acompanhamento de gravação comercial `WAV` e 3 - sem acompanhamento) ordenada decrescentemente pelo total de gravações.

Código	**Título**	**Tipo 1**	**Tipo 2**	**Tipo 3**	**Total**
081	Que país é este?	1	7	6	14
101	Parabéns a você	0	8	5	13
100	Anna Júlia	1	6	3	10
103	Escravos de jó	2	4	3	9
097	Você não soube me amar	1	4	3	8
107	Ilariê	4	1	3	8
019	Quero que vá tudo pro inferno	2	2	3	7
067	A banda	2	1	4	7
109	Mamãe eu quero	1	4	2	7
072	Gita	1	3	2	6
104	Hino nacional	1	3	2	6
028	Pra não dizer que não falei das flores	0	4	2	6
004	Asa branca	2	2	1	5
020	Preta pretinha	1	2	2	5
102	Ciranda cirandinha	0	2	3	5
074	Sentado à beira do caminho	1	1	2	4
002	Águas de março	3	0	1	4
062	Luar do sertão	0	3	1	4
025	País tropical	1	1	2	4
108	Fim de Ano	1	1	2	4
089	A flor e o espinho	0	1	2	3
003	Carinhoso	0	1	2	3
008	Detalhes	1	0	2	3
010	Alegria, alegria	0	2	1	3
015	Trem das onze	0	2	1	3
024	Eu sei que vou te amar	1	0	2	3
031	Travessia	0	1	1	2
111	Coelhinho de olhos vermelhos	1	0	1	2
083	Ideologia	0	1	1	2

Continua na próxima página

Tabela A.2 – Continuação da página anterior.

Código	Título	Tipo 1	Tipo 2	Tipo 3	Total
091	Felicidade	0	1	1	2
096	Disritmia	0	1	1	2
063	Alagados	1	0	1	2
027	Garota de ipanema	1	0	1	2
075	Foi um rio que passou em minha vida	0	1	0	1
038	Eu quero é botar meu bloco na rua	0	0	1	1
087	Meu mundo e nada mais	0	1	0	1
046	Ponteio	0	1	0	1
047	Me chama	1	0	0	1
006	Chega de saudade	1	0	0	1
023	Inútil	0	1	0	1
068	Comida	0	0	1	1
070	Ronda	0	0	0	0
112	Chegou a hora da foqueira	0	0	0	0
084	Rosa	0	0	0	0
041	Manhã de carnaval	0	0	0	0
085	O barquinho	0	0	0	0
093	Casa no campo	0	0	0	0
098	A noite de meu bem	0	0	0	0
012	Aquarela do Brasil	0	0	0	0
057	Conversa de botequim	0	0	0	0
064	As curvas da estrada de Santos	0	0	0	0
105	Hino da bandeira	0	0	0	0

Tabela A.3: Lista de músicas ordenadas pelo percentual de acerto em cada uma das dez primeiras posições do ranque.

Código	**Título**	**1ª**	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**	**10ª**
107	Ilariê	71,17	12,61	1,80	3,60	0,90	0,90	0,90	0,90	0,90	0,00
023	Inútil	66,67	4,17	0,00	0,00	0,00	0,00	0,00	0,00	0,00	4,17
002	Águas de março	52,87	8,05	1,15	3,45	3,45	0,00	4,60	1,15	1,15	2,30
057	Conversa de botequim	52,78	5,56	0,00	0,00	0,00	2,78	0,00	0,00	0,00	0,00
075	Foi um rio que passou em minha vida	50,00	0,00	0,00	16,67	0,00	0,00	0,00	0,00	0,00	0,00
041	Manhã de carnaval	50,00	0,00	0,00	0,00	16,67	0,00	0,00	0,00	0,00	0,00
067	A banda	49,02	10,78	3,92	0,00	0,00	0,00	0,00	0,00	0,98	0,00
020	Preta pretinha	47,47	5,05	2,02	1,01	2,02	0,00	1,01	0,00	2,02	1,01
074	Sentado à beira do caminho	47,22	8,33	2,78	0,00	2,78	5,56	0,00	8,33	2,78	2,78
028	Pra não dizer que não falei das flores	45,00	10,00	3,33	1,67	0,00	0,00	0,00	0,00	1,67	0,00
108	Fim de Ano	43,33	4,44	5,56	1,11	4,44	2,22	2,22	1,11	1,11	0,00
109	Mamãe eu quero	42,98	15,79	5,26	7,02	4,39	3,51	0,88	1,75	1,75	3,51
111	Coelhinho de olhos vermelhos	42,86	14,29	4,76	4,76	0,00	0,00	0,00	0,00	0,00	0,00
004	Asa branca	42,55	13,48	2,84	0,00	0,71	2,84	1,42	2,84	2,84	1,42
112	Chegou a hora da foqueira	41,67	16,67	0,00	0,00	0,00	8,33	0,00	0,00	0,00	0,00
027	Garota de ipanema	41,30	13,04	5,07	2,17	0,72	1,45	0,72	0,72	0,00	0,00
010	Alegria, alegria	40,00	5,33	1,33	2,67	1,33	0,00	0,00	0,00	0,00	2,67

Continua na próxima página

Tabela A.3 – Continuação da página anterior.

Código	Título	1ª	2ª	3ª	4ª	5ª	6ª	7ª	8ª	9ª	10ª
064	As curvas da estrada de Santos	38,89	5,56	0,00	0,00	5,56	0,00	0,00	0,00	5,56	0,00
072	Gita	38,54	16,67	4,17	5,21	6,25	6,25	3,13	4,17	0,00	2,08
085	O barquinho	35,71	11,90	0,00	7,14	4,76	2,38	0,00	2,38	0,00	2,38
093	Casa no campo	33,33	33,33	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
046	Ponteio	33,33	5,56	5,56	5,56	11,11	0,00	0,00	5,56	0,00	0,00
070	Ronda	33,33	0,00	8,33	0,00	0,00	0,00	8,33	0,00	0,00	0,00
102	Ciranda cirandinha	32,62	9,22	2,84	3,55	6,38	4,26	3,55	8,51	2,84	4,26
103	Escravos de jó	31,71	5,69	5,69	3,25	0,00	0,81	3,25	0,81	0,81	0,81
024	Eu sei que vou te amar	30,30	18,18	9,09	1,52	1,52	2,27	3,79	4,55	3,03	2,27
087	Meu mundo e nada mais	29,17	12,50	12,50	0,00	2,08	2,08	0,00	2,08	2,08	0,00
097	Você não soube me amar	28,03	5,30	1,52	0,76	0,76	1,52	0,00	0,76	1,52	0,76
091	Felicidade	26,67	8,89	4,44	0,00	0,00	0,00	2,22	4,44	4,44	8,89
019	Quero que vá tudo pro inferno	26,50	20,51	5,98	3,42	3,42	3,42	0,85	0,85	0,00	0,85
083	Ideologia	23,53	9,80	5,88	0,00	3,92	1,96	1,96	0,00	7,84	3,92
025	País tropical	19,05	11,11	2,38	2,38	1,59	3,17	0,79	1,59	1,59	1,59
101	Parabéns a você	18,71	11,70	5,26	3,51	5,85	4,09	1,75	1,75	3,51	3,51
047	Me chama	16,67	16,67	8,33	0,00	0,00	0,00	4,17	0,00	0,00	0,00
089	A flor e o espinho	16,67	13,33	10,00	0,00	6,67	0,00	0,00	0,00	0,00	0,00
098	A noite de meu bem	16,67	0,00	16,67	8,33	0,00	8,33	0,00	0,00	0,00	0,00
015	Trem das onze	16,19	16,19	3,81	4,76	2,86	0,95	0,95	0,95	0,95	0,00

Continua na próxima página

Tabela A.3 – Continuação da página anterior.

Código	Título	1ª	2ª	3ª	4ª	5ª	6ª	7ª	8ª	9ª	10ª
008	Detalhes	13,64	7,58	1,52	6,06	4,55	4,55	3,03	3,03	1,52	3,03
104	Hino nacional	12,07	5,75	5,17	2,30	2,30	0,57	0,57	0,57	2,30	1,15
003	Carinhoso	11,90	17,86	3,57	7,14	3,57	3,57	0,00	2,38	1,19	0,00
081	Que país é este?	11,36	9,85	8,33	1,52	1,52	1,52	3,03	0,76	2,27	3,03
063	Alagados	11,11	2,22	15,56	8,89	4,44	4,44	11,11	2,22	4,44	2,22
038	Eu quero é botar meu bloco na rua	9,52	19,05	0,00	4,76	0,00	9,52	0,00	0,00	4,76	9,52
031	Travessia	9,09	9,09	3,03	0,00	6,06	0,00	3,03	0,00	0,00	3,03
012	Aquarela do Brasil	6,25	12,50	4,17	2,08	0,00	0,00	2,08	2,08	0,00	0,00
006	Chega de saudade	5,56	11,11	0,00	5,56	5,56	3,70	0,00	1,85	0,00	0,00
100	Anna Júlia	5,33	10,00	7,33	6,67	8,00	4,67	7,33	4,67	6,67	4,00
096	Disritmia	3,70	11,11	14,81	3,70	0,00	0,00	0,00	0,00	0,00	11,11
062	Luar do sertão	1,59	7,94	7,94	6,35	9,52	1,59	4,76	7,94	0,00	4,76
105	Hino da bandeira	0,00	11,11	5,56	5,56	0,00	0,00	0,00	0,00	0,00	5,56
068	Comida	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	5,13	2,56
084	Rosa	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Tabela A.4: Lista de músicas ordenadas pelo percentual de acerto em cada uma das dez primeiras posições do ranque excluindo os resultados do Algoritmo 2.

Código	**Título**	**1ª**	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**	**10ª**
023	Inútil	87,50	6,25	0,00	0,00	0,00	0,00	0,00	0,00	0,00	6,25
002	Águas de março	79,31	8,62	0,00	1,72	1,72	0,00	0,00	0,00	0,00	1,72
057	Conversa de botequim	79,17	8,33	0,00	0,00	0,00	4,17	0,00	0,00	0,00	0,00
075	Foi um rio que passou em minha vida	75,00	0,00	0,00	25,00	0,00	0,00	0,00	0,00	0,00	0,00
041	Manhã de carnaval	75,00	0,00	0,00	0,00	25,00	0,00	0,00	0,00	0,00	0,00
067	A banda	73,53	16,18	5,88	0,00	0,00	0,00	0,00	0,00	1,47	0,00
107	Ilariê	72,97	9,46	1,35	4,05	0,00	1,35	1,35	0,00	1,35	0,00
020	Preta pretinha	71,21	7,58	3,03	1,52	3,03	0,00	1,52	0,00	3,03	1,52
074	Sentado à beira do caminho	70,83	12,50	4,17	0,00	0,00	4,17	0,00	0,00	0,00	0,00
028	Pra não dizer que não falei das flores	67,50	15,00	5,00	2,50	0,00	0,00	0,00	0,00	2,50	0,00
108	Fim de Ano	65,00	6,67	8,33	1,67	6,67	3,33	3,33	1,67	1,67	0,00
111	Coelhinho de olhos vermelhos	64,29	21,43	7,14	7,14	0,00	0,00	0,00	0,00	0,00	0,00
004	Asa branca	63,83	20,21	4,26	0,00	1,06	4,26	1,06	1,06	1,06	1,06
109	Mamãe eu quero	63,16	17,11	3,95	5,26	0,00	0,00	0,00	1,32	0,00	2,63
112	Chegou a hora da foqueira	62,50	12,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
027	Garota de ipanema	61,96	19,57	7,61	3,26	1,09	2,17	1,09	1,09	0,00	0,00
010	Alegria, alegria	60,00	8,00	2,00	4,00	2,00	0,00	0,00	0,00	0,00	4,00

Continua na próxima página

Tabela A.4 – Continuação da página anterior.

Código	Título	1ª	2ª	3ª	4ª	5ª	6ª	7ª	8ª	9ª	10ª
064	As curvas da estrada de Santos	58,33	8,33	0,00	0,00	8,33	0,00	0,00	0,00	8,33	0,00
072	Gita	57,81	25,00	4,69	1,56	0,00	0,00	0,00	3,13	0,00	0,00
085	O barquinho	53,57	17,86	0,00	10,71	7,14	3,57	0,00	3,57	0,00	3,57
093	Casa no campo	50,00	50,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
046	Ponteio	50,00	8,33	8,33	8,33	16,67	0,00	0,00	8,33	0,00	0,00
070	Ronda	50,00	0,00	12,50	0,00	0,00	0,00	12,50	0,00	0,00	0,00
102	Ciranda cirandinha	48,94	13,83	4,26	3,19	2,13	2,13	2,13	1,06	0,00	0,00
103	Escravos de jó	47,56	8,54	8,54	4,88	0,00	1,22	4,88	1,22	1,22	1,22
024	Eu sei que vou te amar	45,45	27,27	13,64	2,27	2,27	3,41	1,14	1,14	0,00	1,14
087	Meu mundo e nada mais	43,75	18,75	18,75	0,00	3,13	3,13	0,00	3,13	3,13	0,00
097	Você não soube me amar	42,05	7,95	2,27	1,14	1,14	2,27	0,00	1,14	2,27	1,14
091	Felicidade	40,00	13,33	6,67	0,00	0,00	0,00	3,33	3,33	3,33	6,67
019	Quero que vá tudo pro inferno	39,74	30,77	8,97	5,13	5,13	5,13	1,28	1,28	0,00	1,28
083	Ideologia	35,29	14,71	8,82	0,00	5,88	0,00	2,94	0,00	0,00	2,94
025	País tropical	28,57	16,67	3,57	3,57	2,38	4,76	1,19	2,38	2,38	2,38
101	Parabéns a você	28,07	17,54	7,89	5,26	8,77	6,14	2,63	2,63	5,26	5,26
047	Me chama	25,00	25,00	12,50	0,00	0,00	0,00	6,25	0,00	0,00	0,00
089	A flor e o espinho	25,00	20,00	15,00	0,00	10,00	0,00	0,00	0,00	0,00	0,00
098	A noite de meu bem	25,00	0,00	25,00	12,50	0,00	12,50	0,00	0,00	0,00	0,00
015	Trem das onze	24,29	24,29	5,71	7,14	4,29	1,43	1,43	1,43	1,43	0,00

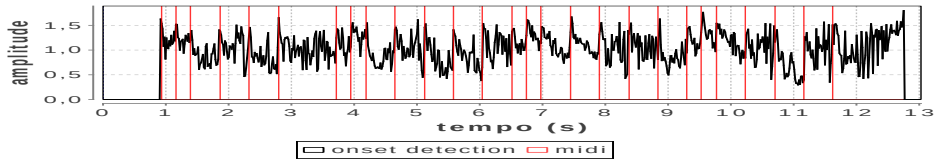
Continua na próxima página

Tabela A.4 – Continuação da página anterior.

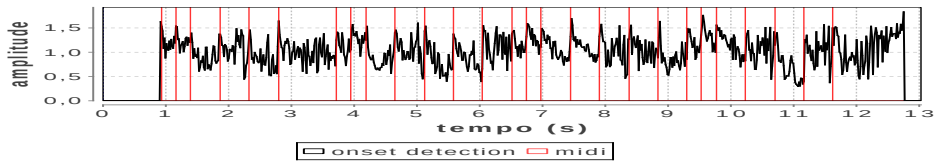
Código	Título	1ª	2ª	3ª	4ª	5ª	6ª	7ª	8ª	9ª	10ª
008	Detalhes	20,45	6,82	0,00	0,00	6,82	4,55	2,27	0,00	2,27	0,00
104	Hino nacional	18,10	8,62	7,76	3,45	3,45	0,86	0,00	0,86	3,45	1,72
003	Carinhoso	17,86	26,79	5,36	10,71	5,36	5,36	0,00	3,57	1,79	0,00
081	Que país é este?	17,05	14,77	12,50	2,27	2,27	2,27	4,55	1,14	3,41	4,55
063	Alagados	16,67	3,33	10,00	6,67	3,33	3,33	10,00	3,33	3,33	3,33
038	Eu quero é botar meu bloco na rua	14,29	28,57	0,00	7,14	0,00	14,29	0,00	0,00	7,14	14,29
031	Travessia	13,64	13,64	4,55	0,00	9,09	0,00	4,55	0,00	0,00	4,55
012	Aquarela do Brasil	9,38	18,75	6,25	3,13	0,00	0,00	3,13	3,13	0,00	0,00
006	Chega de saudade	8,33	16,67	0,00	8,33	8,33	5,56	0,00	2,78	0,00	0,00
100	Anna Júlia	7,00	15,00	9,00	10,00	4,00	3,00	9,00	5,00	6,00	5,00
096	Disritmia	5,56	16,67	22,22	5,56	0,00	0,00	0,00	0,00	0,00	16,67
062	Luar do sertão	2,38	7,14	4,76	2,38	4,76	2,38	2,38	7,14	0,00	4,76
105	Hino da bandeira	0,00	16,67	8,33	8,33	0,00	0,00	0,00	0,00	0,00	8,33
068	Comida	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	7,69	3,85
084	Rosa	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Apêndice B

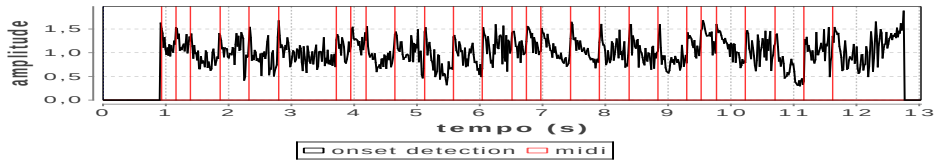
Gráficos dos detectores de *onsets*



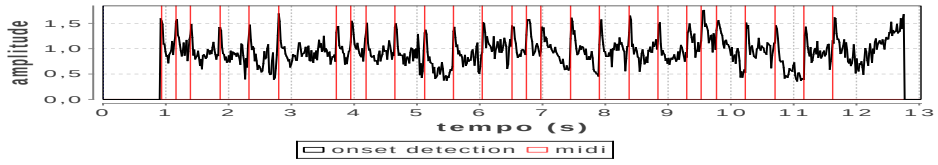
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

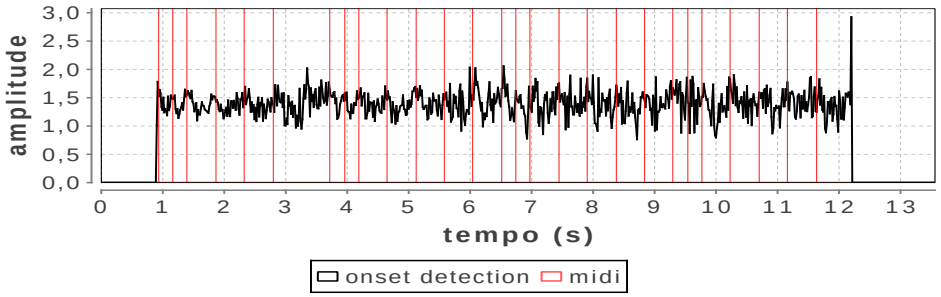


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

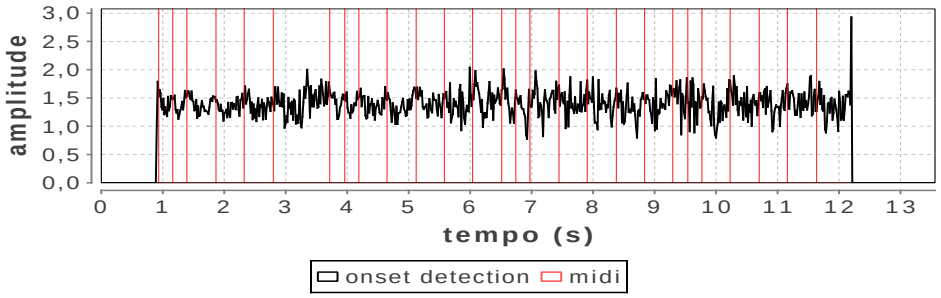


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

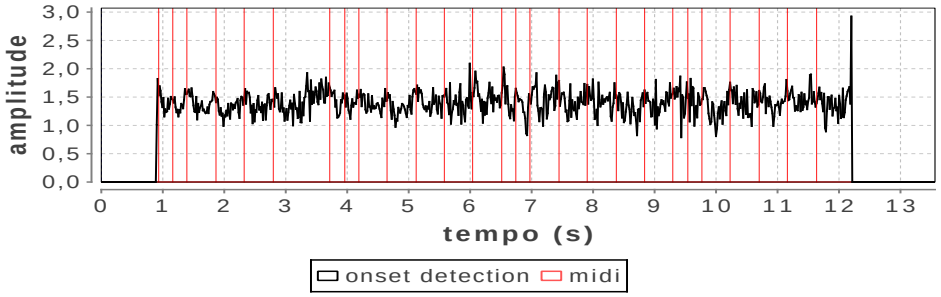
Figura B.1: Gráfico gerado pelo estimador PD tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



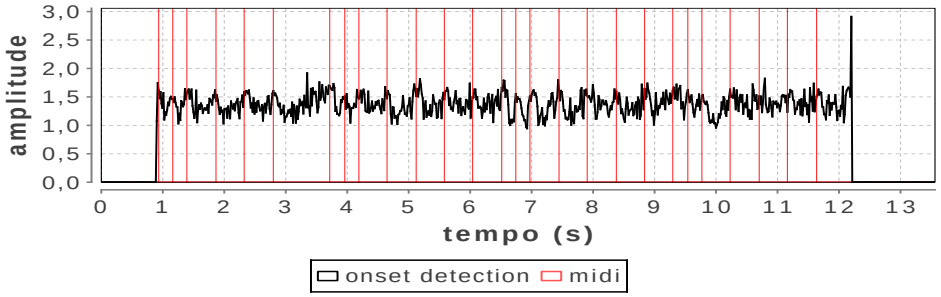
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

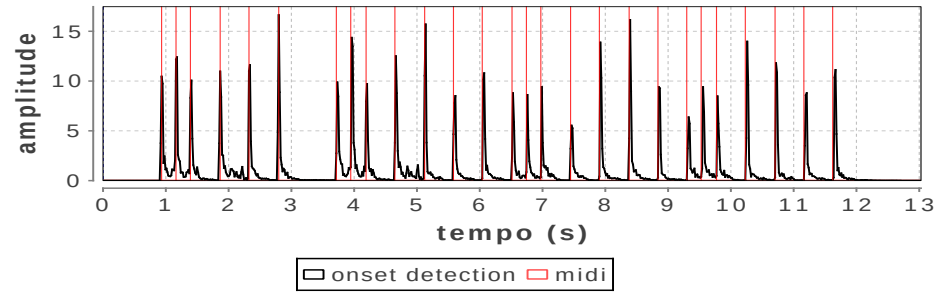


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

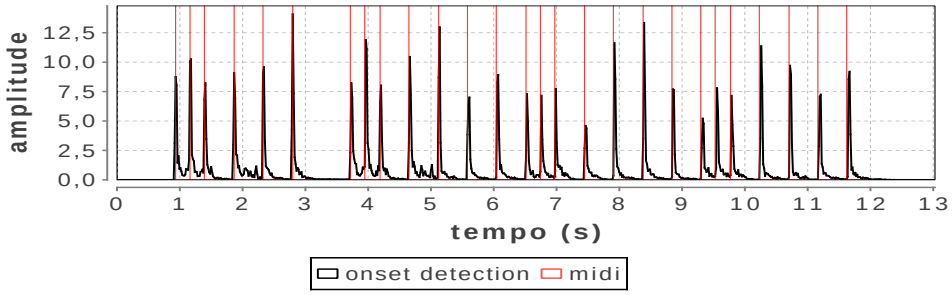


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

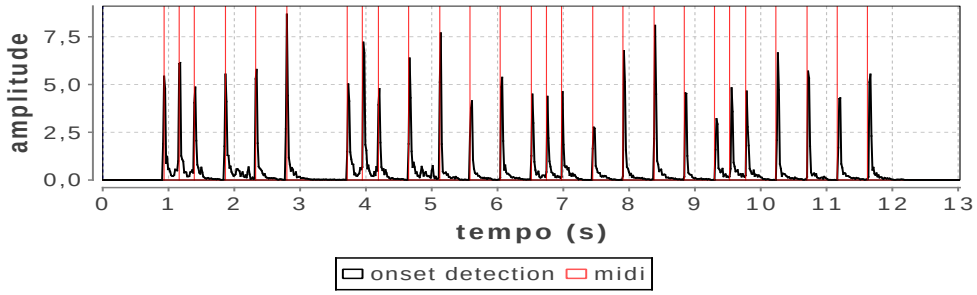
Figura B.2: Gráfico gerado pelo estimador PD tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



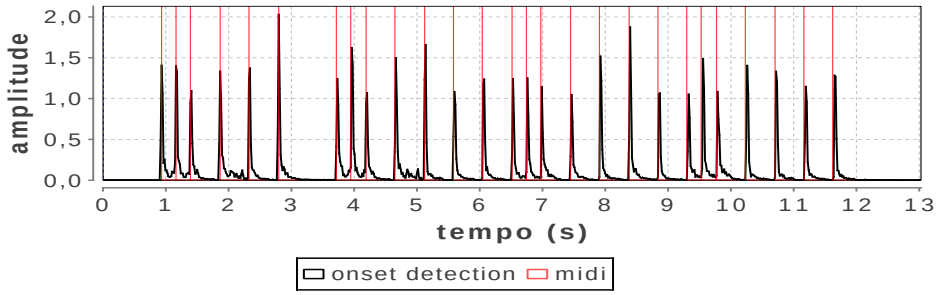
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

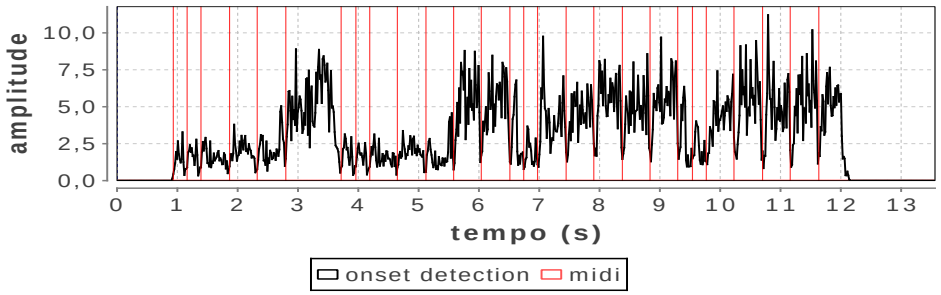


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

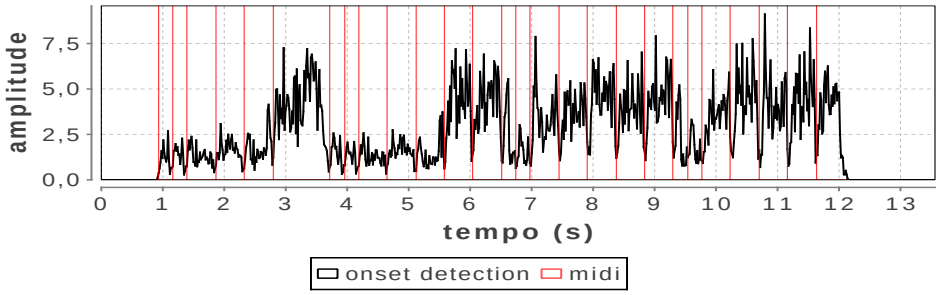


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

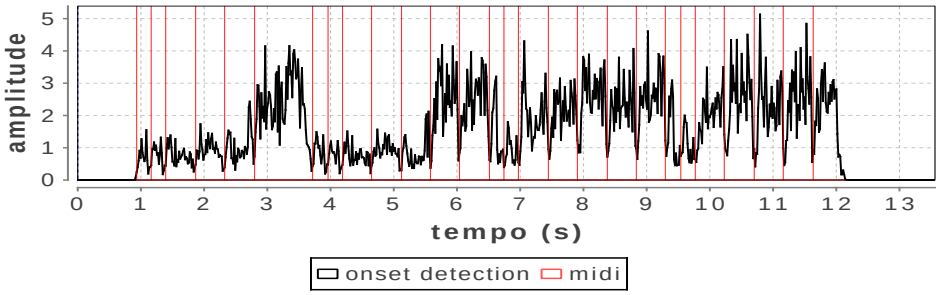
Figura B.3: Gráfico gerado pelo estimador WPD tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



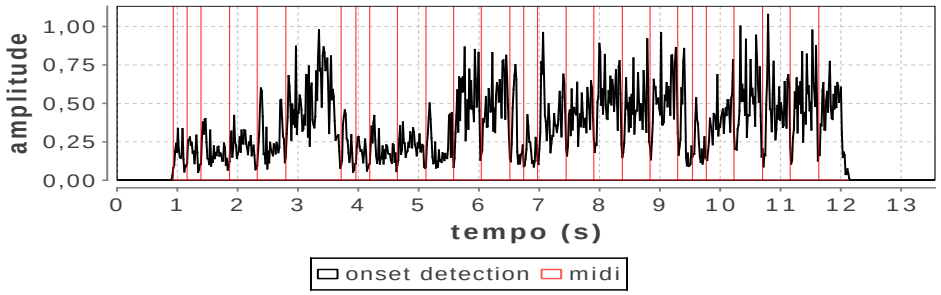
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

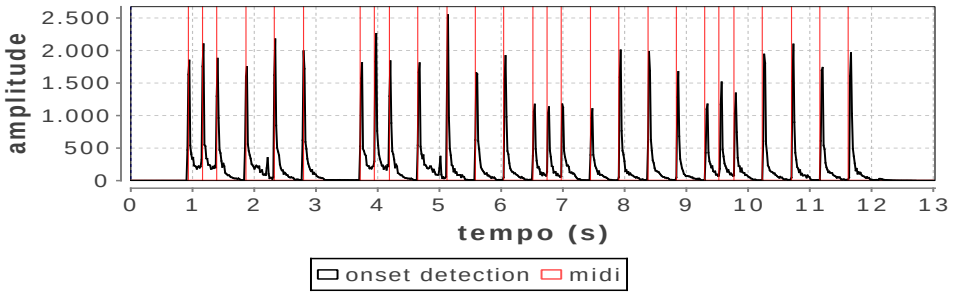


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

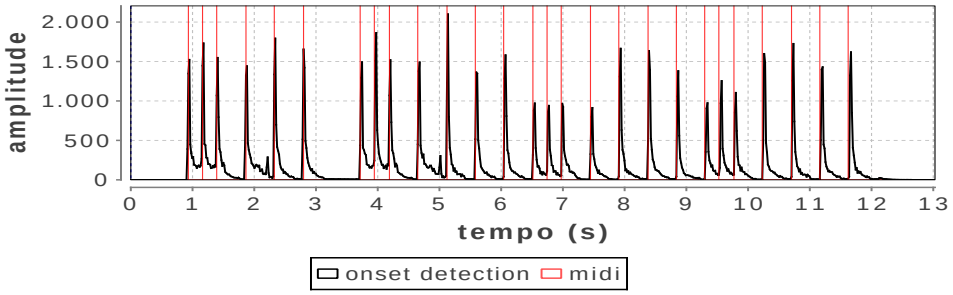


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

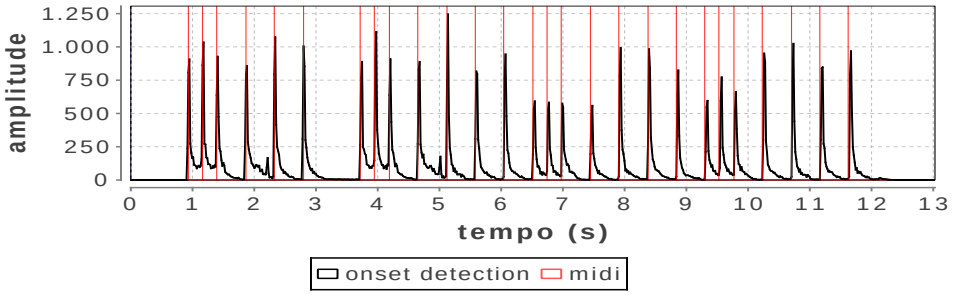
Figura B.4: Gráfico gerado pelo estimador WPD tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



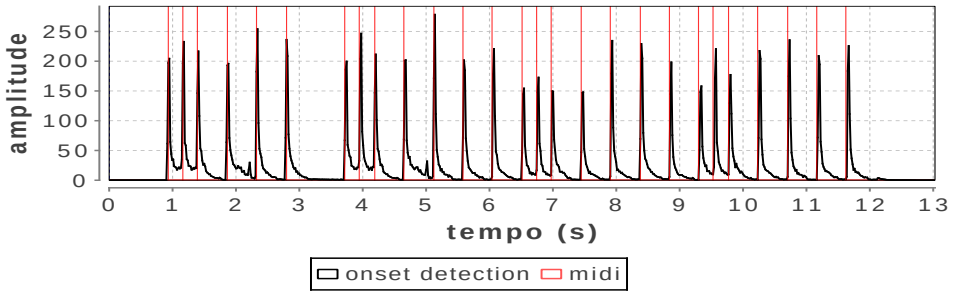
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

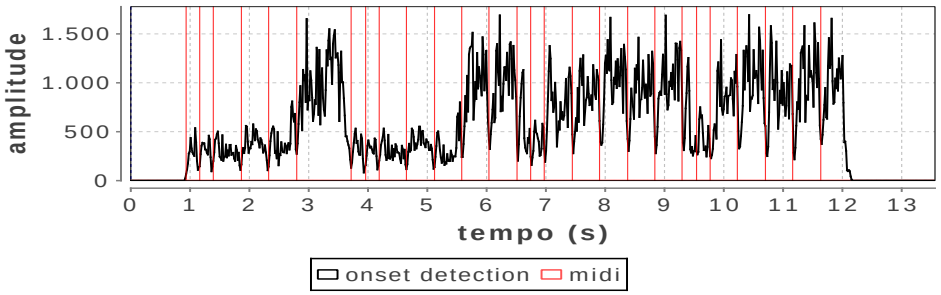


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

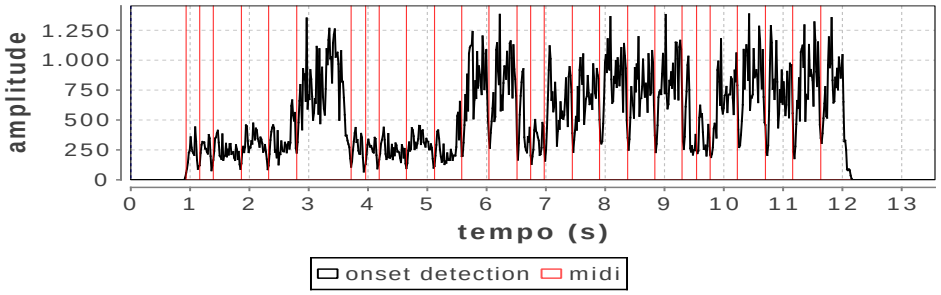


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

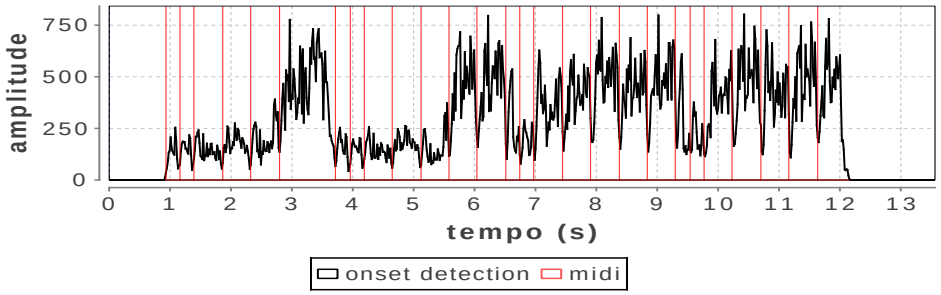
Figura B.5: Gráfico gerado pelo estimador CD tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



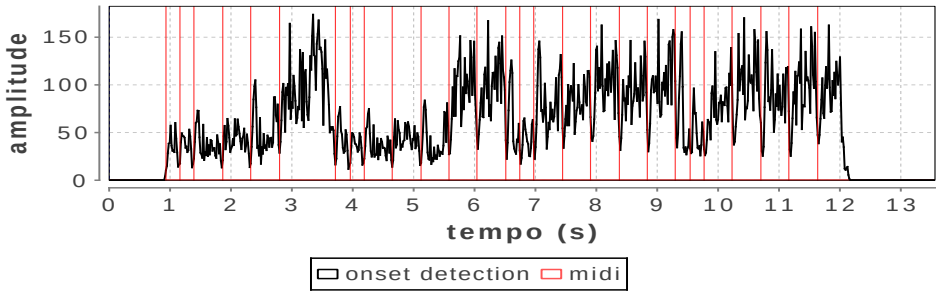
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

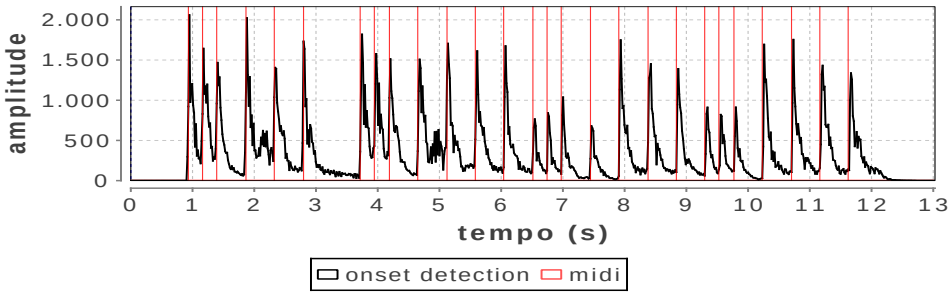


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

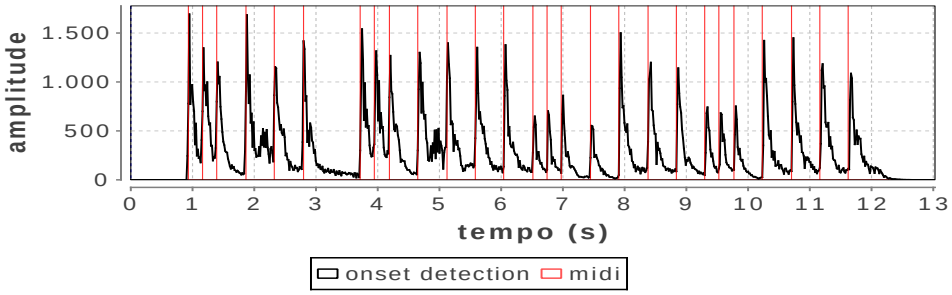


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

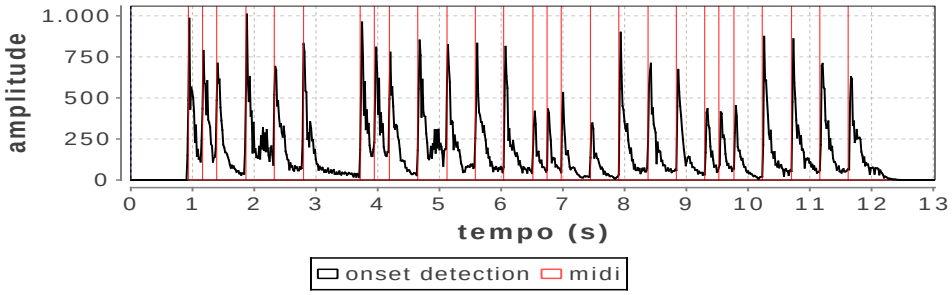
Figura B.6: Gráfico gerado pelo estimador CD tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



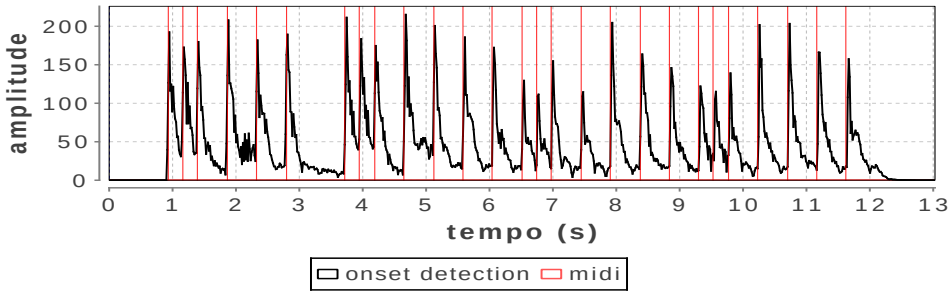
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

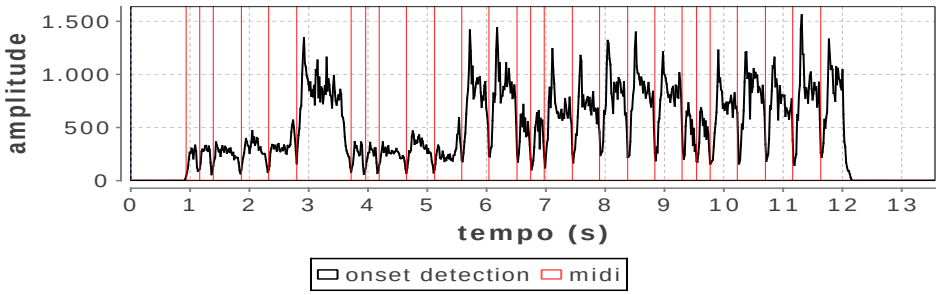


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

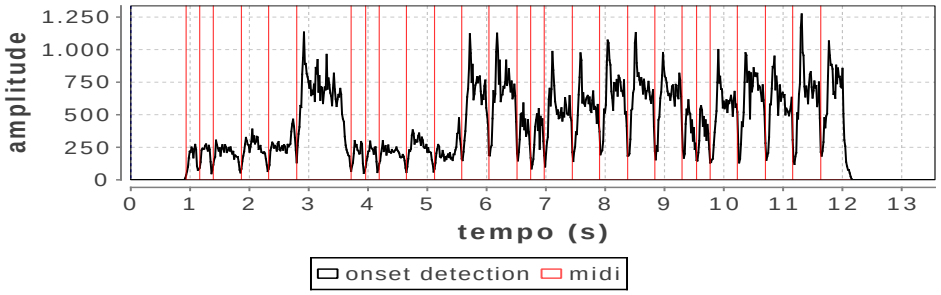


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

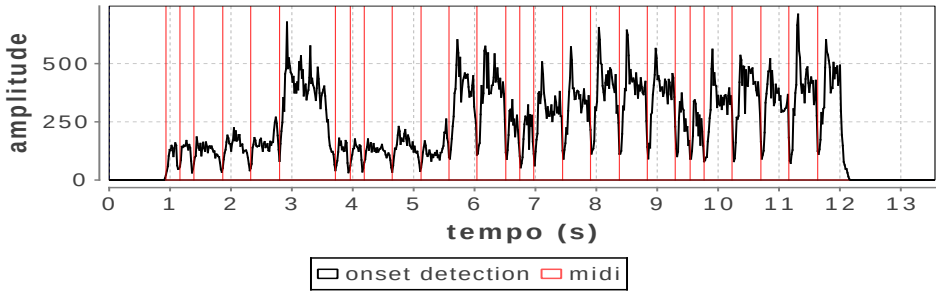
Figura B.7: Gráfico gerado pelo estimador CDS tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



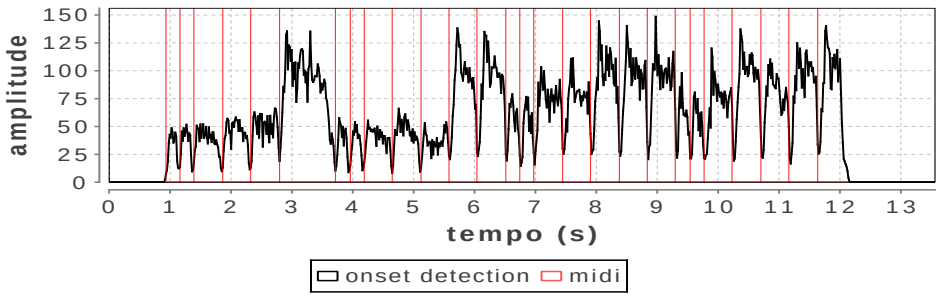
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

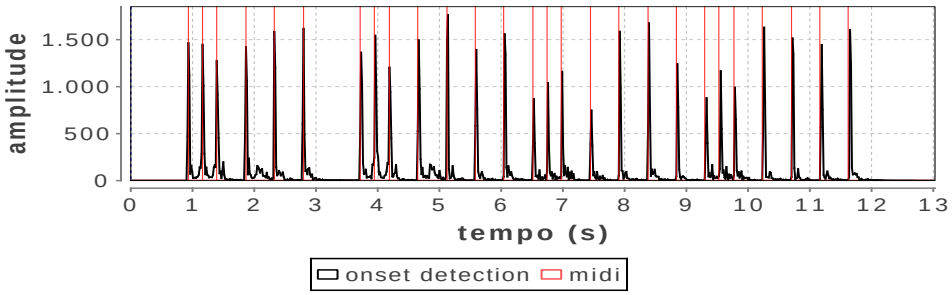


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

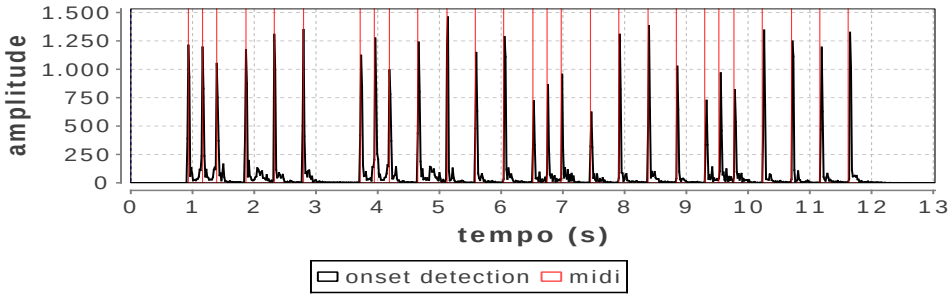


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

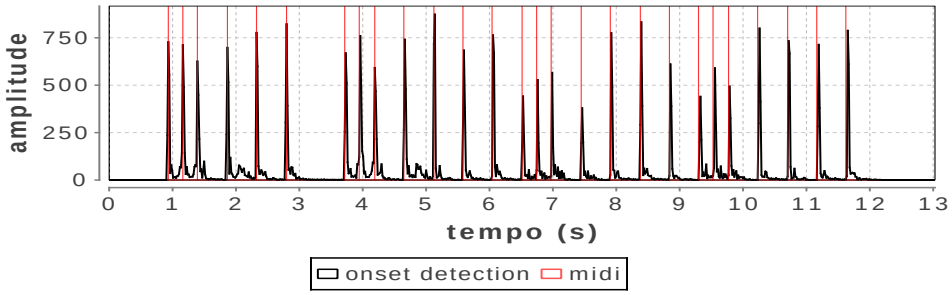
Figura B.8: Gráfico gerado pelo estimador CDS tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



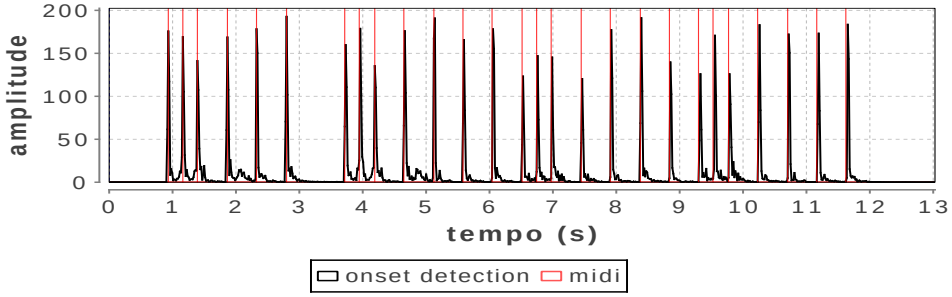
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

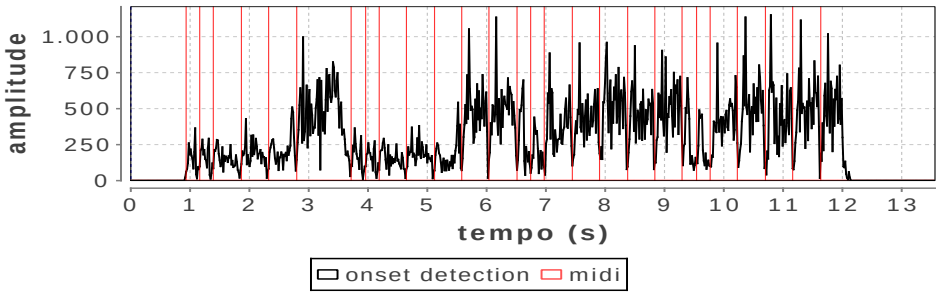


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

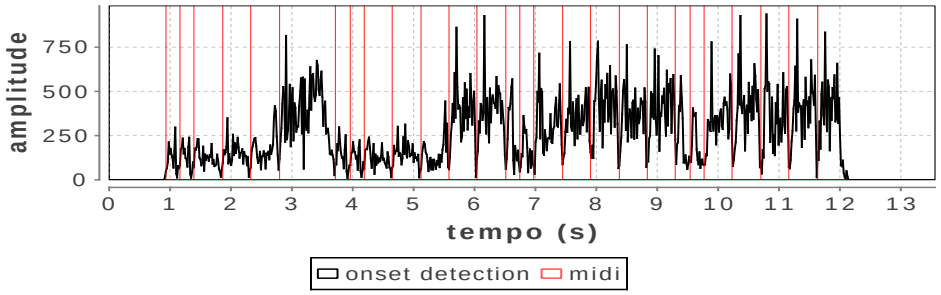


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

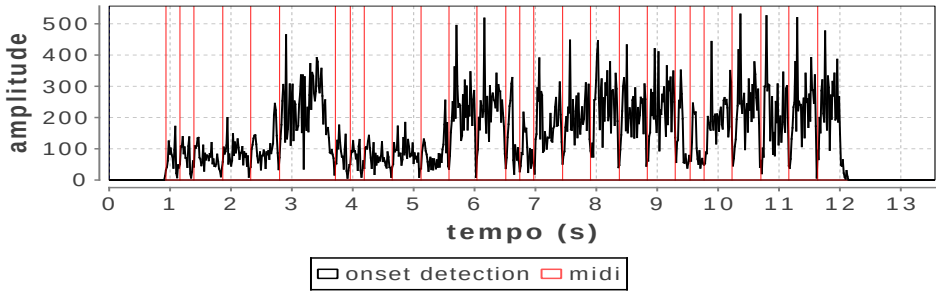
Figura B.9: Gráfico gerado pelo estimador RCD tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



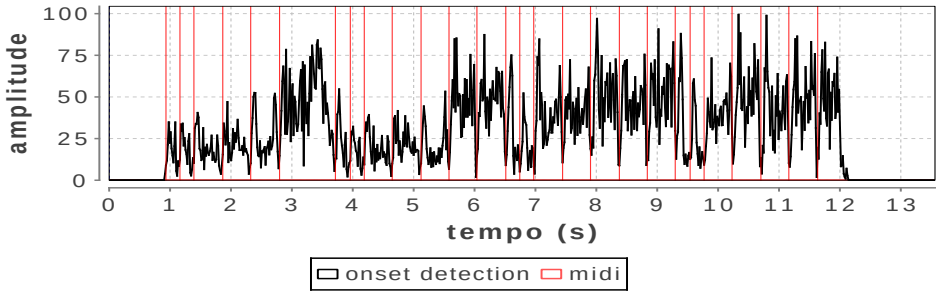
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

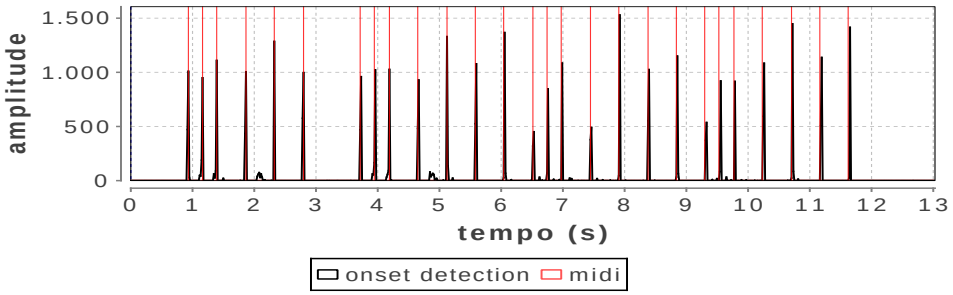


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

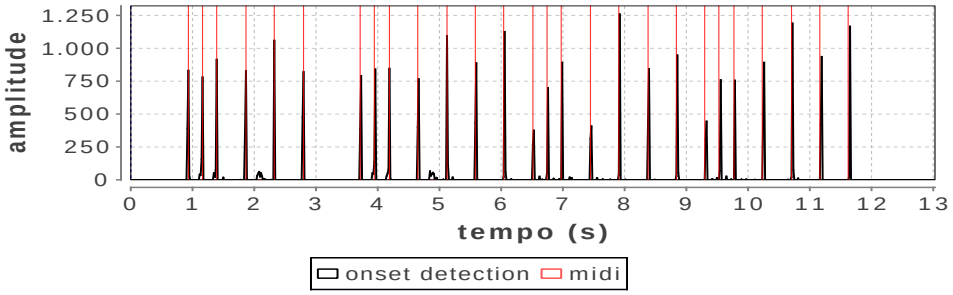


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

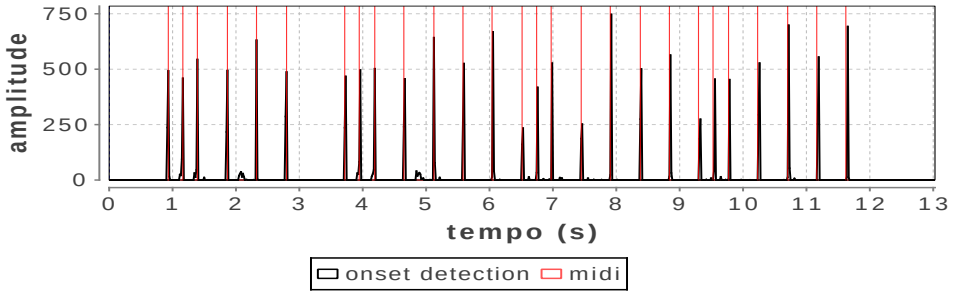
Figura B.10: Gráfico gerado pelo estimador RCD tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



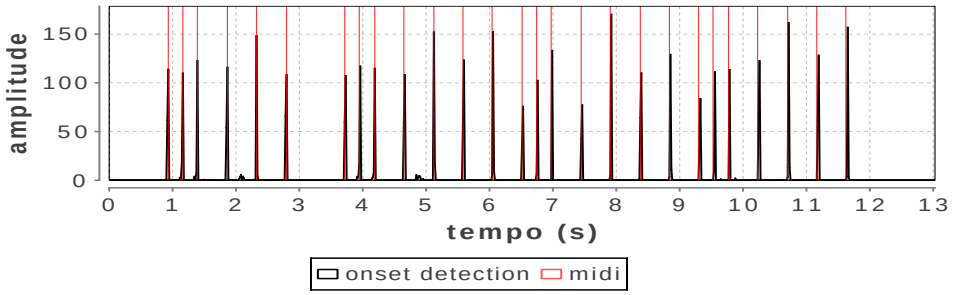
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

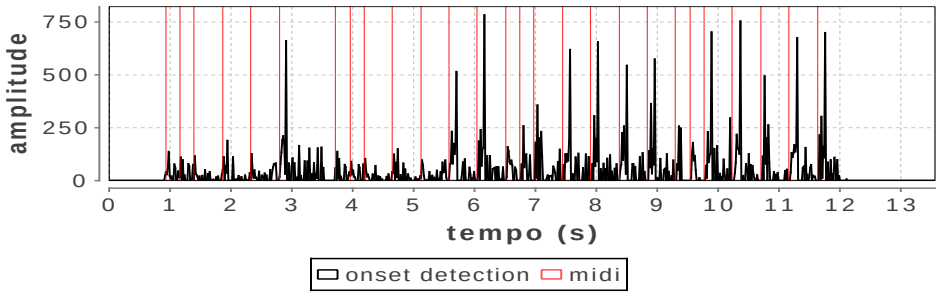


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

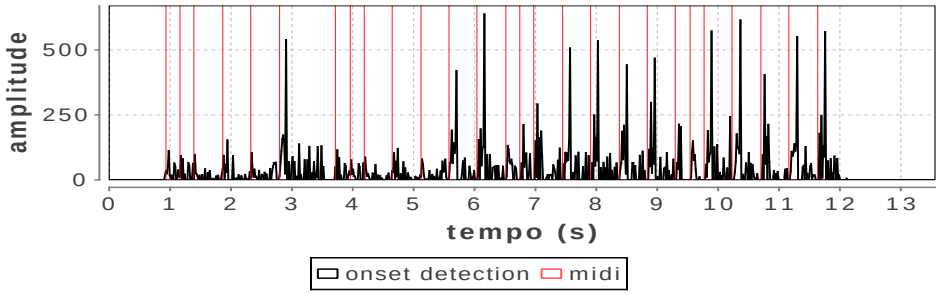


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

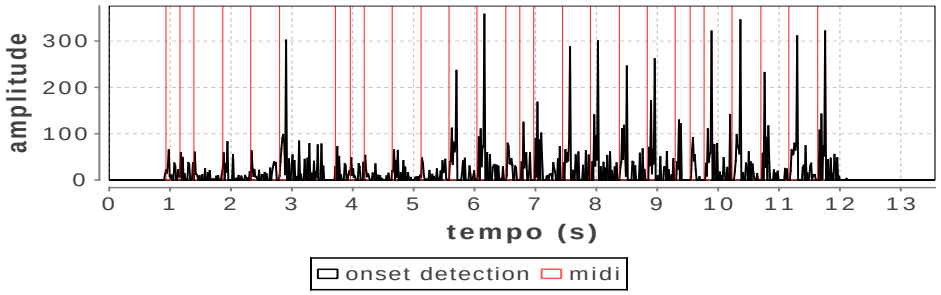
Figura B.11: Gráfico gerado pelo estimador SF tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



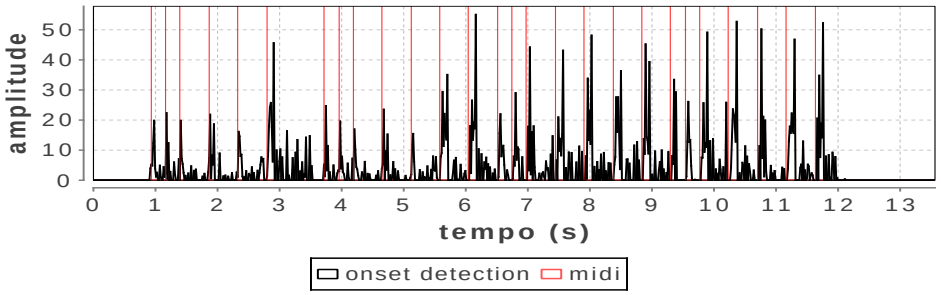
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

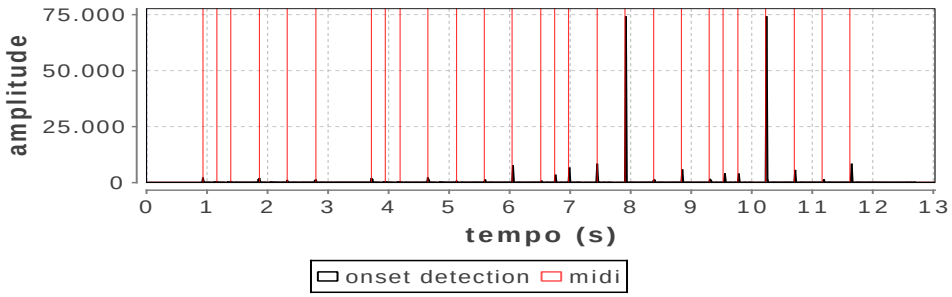


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

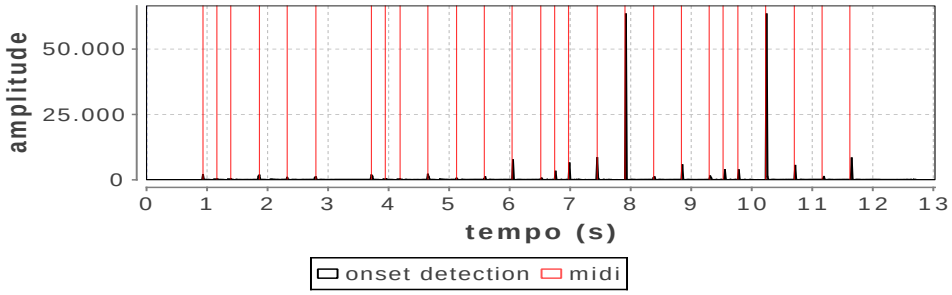


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

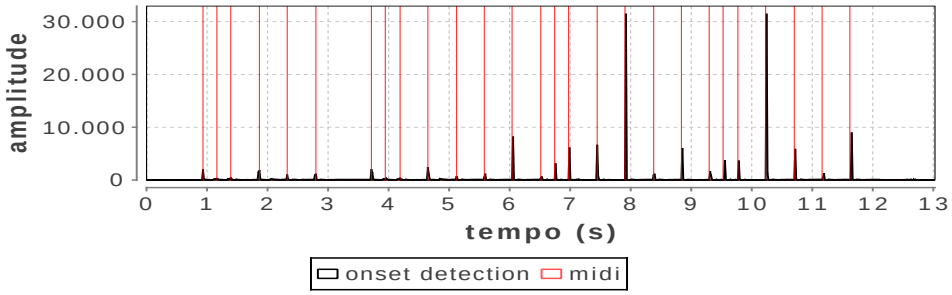
Figura B.12: Gráfico gerado pelo estimador SF tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



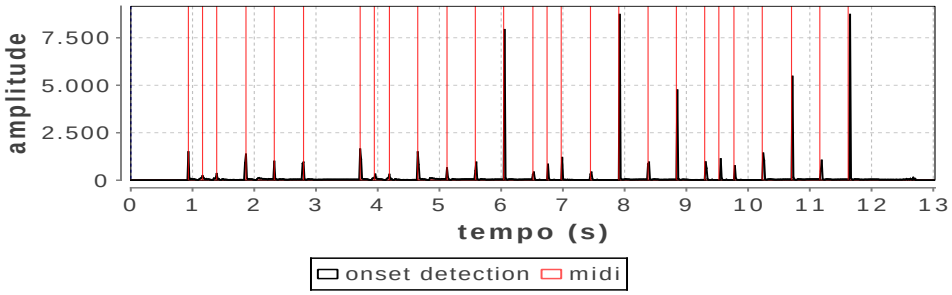
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

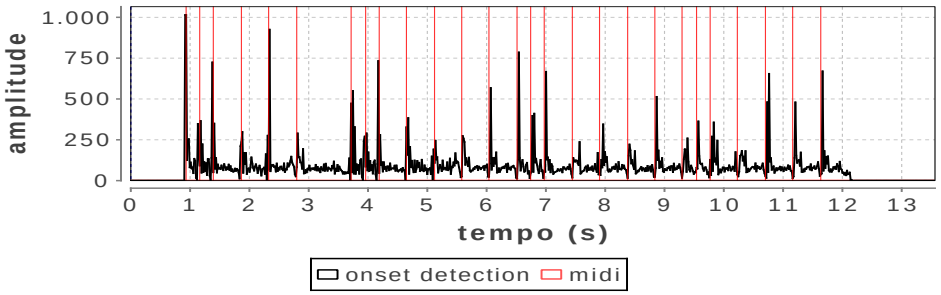


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

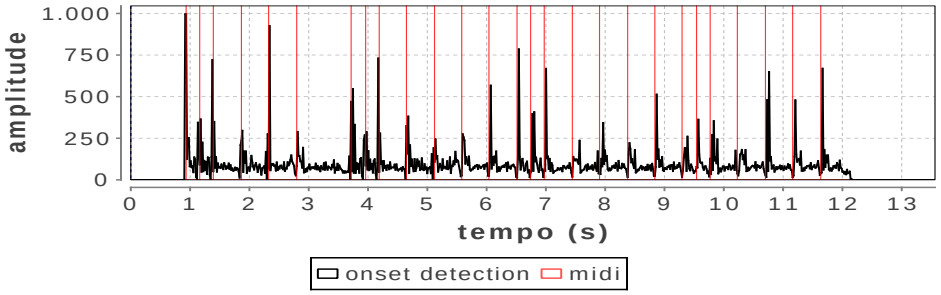


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

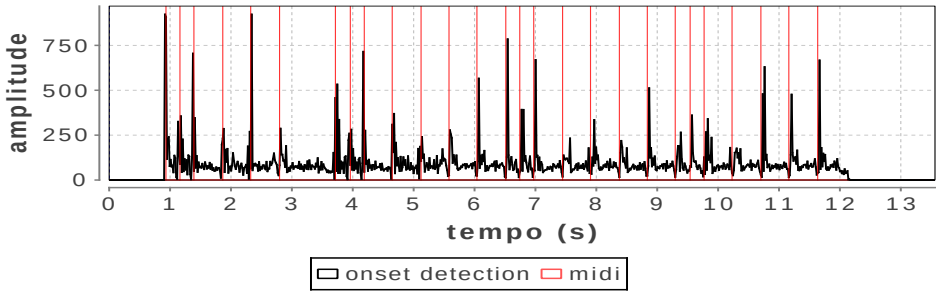
Figura B.13: Gráfico gerado pelo estimador HFC tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



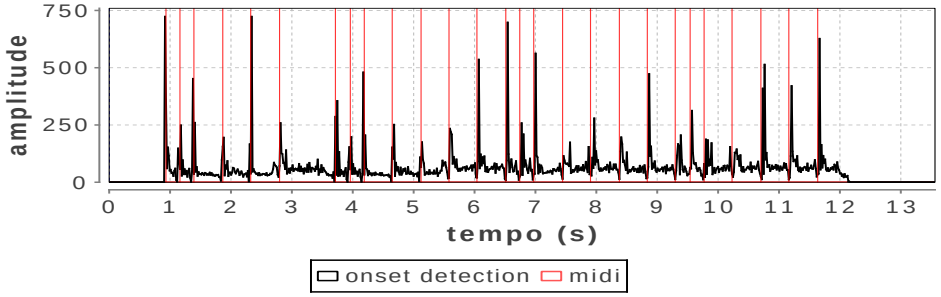
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

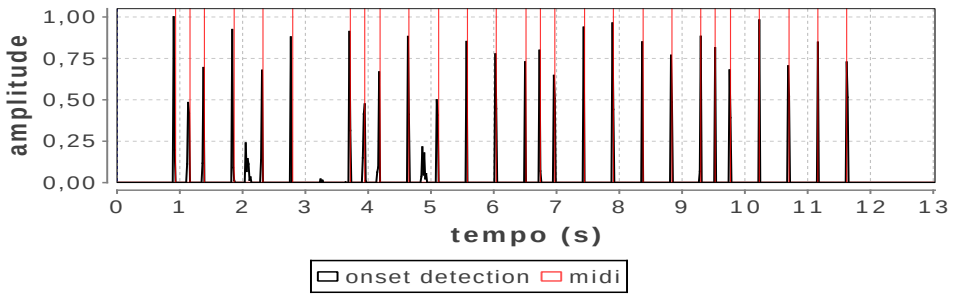


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

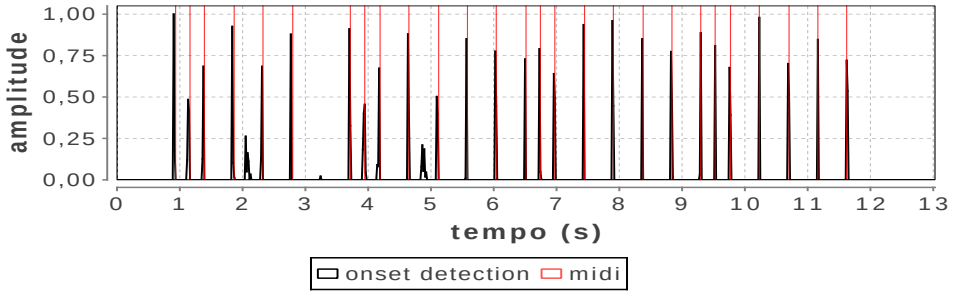


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

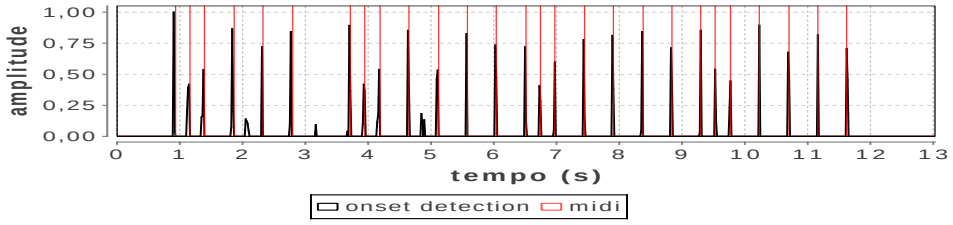
Figura B.14: Gráfico gerado pelo estimador HFC tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



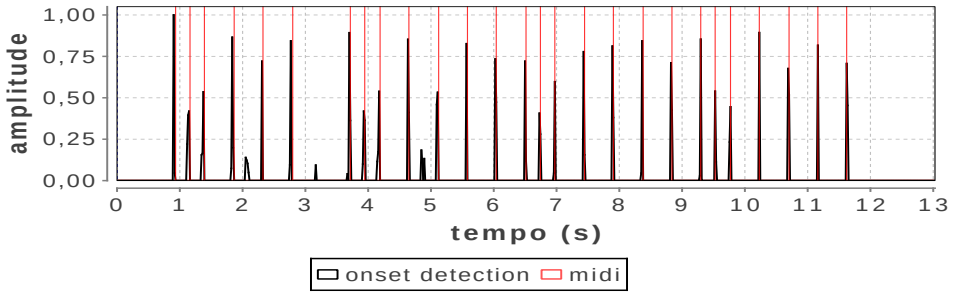
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

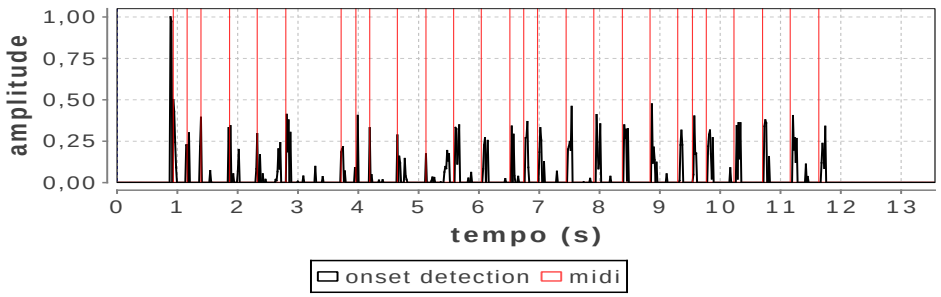


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

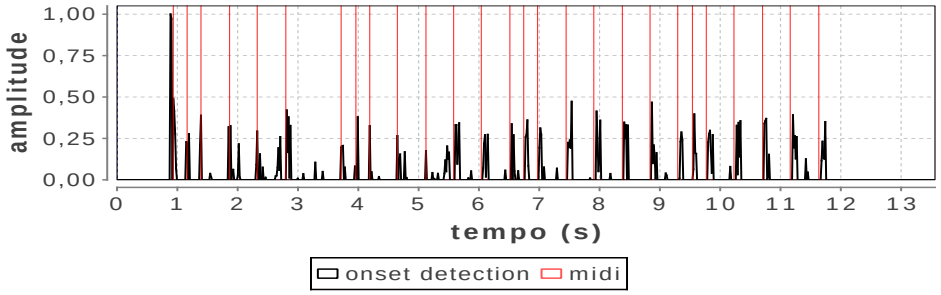


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

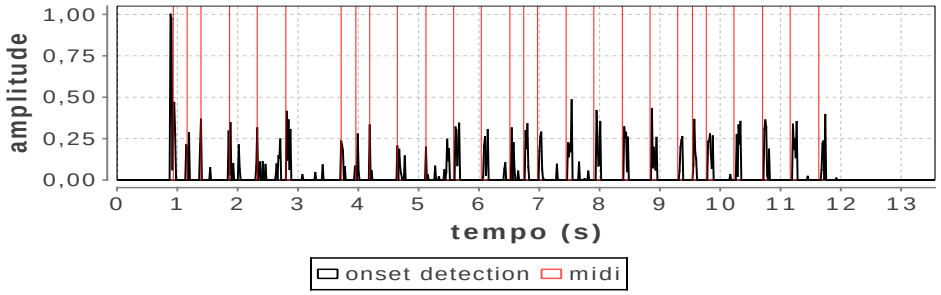
Figura B.15: Gráfico gerado pelo estimador DE tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



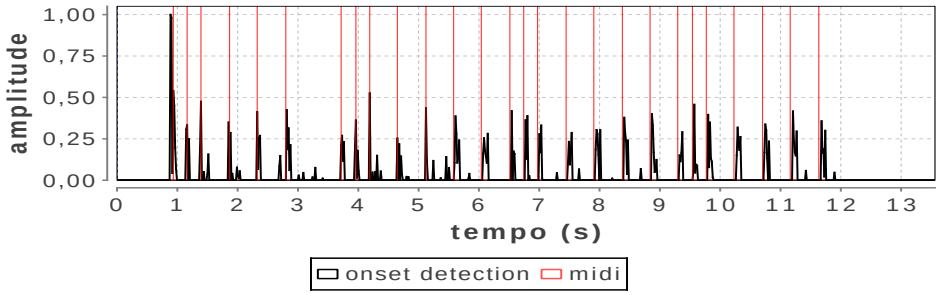
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

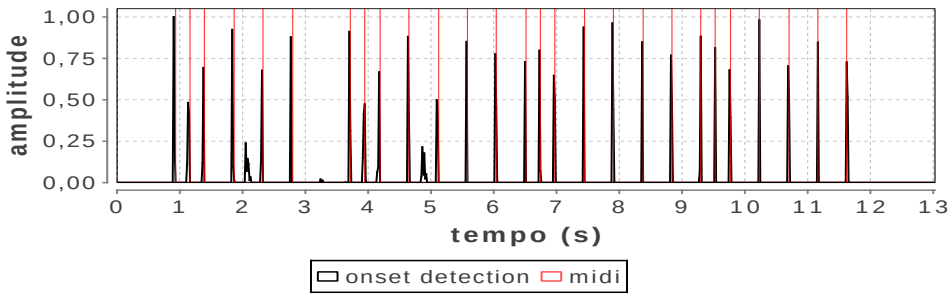


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

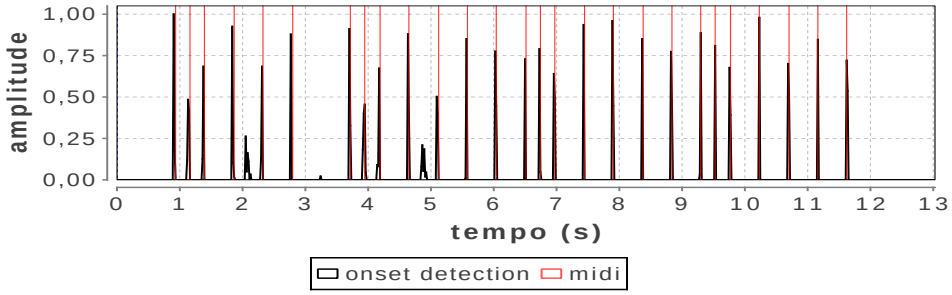


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

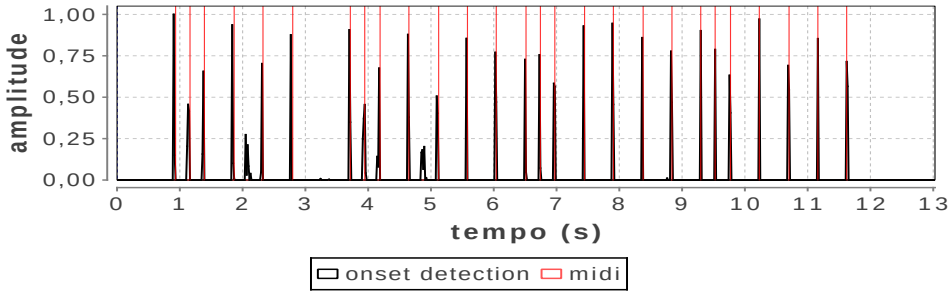
Figura B.16: Gráfico gerado pelo estimador DE tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



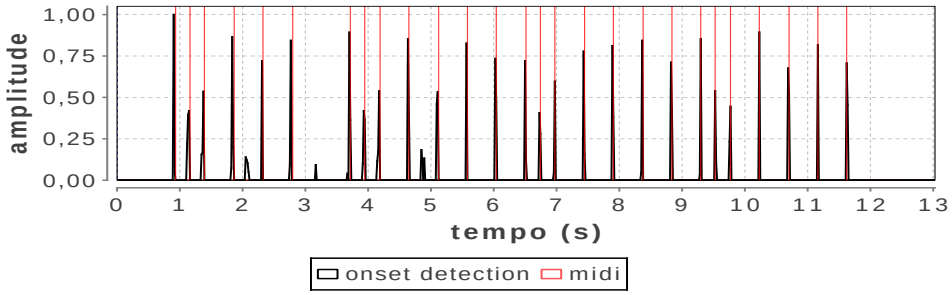
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz

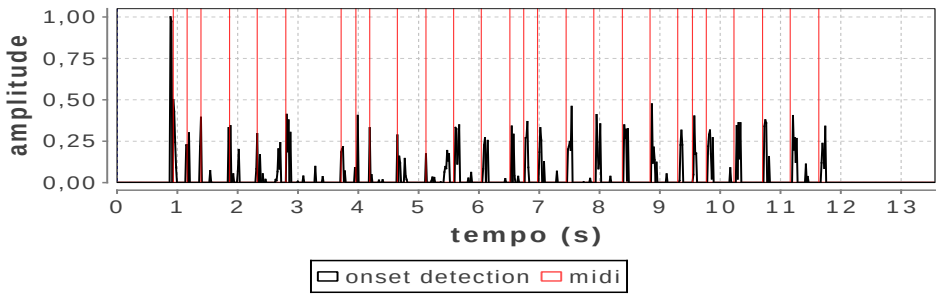


(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz

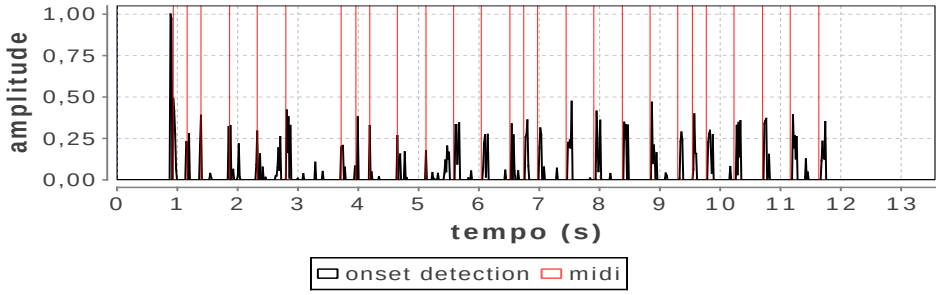


(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

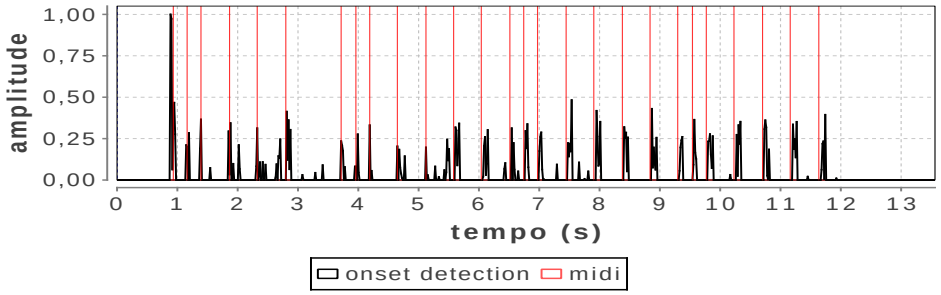
Figura B.17: Gráfico gerado pelo estimador DRE tendo como entrada um sinal de piano (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.



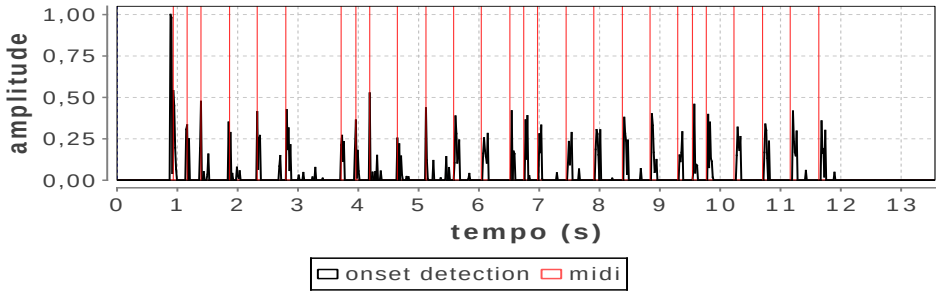
(a) Sub-banda 1: Sinal de entrada limitado de 250 a 500 Hz



(b) Sub-banda 2: Sinal de entrada limitado de 500 a 1000 Hz



(c) Sub-banda 3: Sinal de entrada limitado de 1000 a 2000 Hz



(d) Sub-banda 4: Sinal de entrada limitado de 2000 a 4000 Hz

Figura B.18: Gráfico gerado pelo estimador DRE tendo como entrada um sinal de solfejo (Figura 4.2(a)) filtrado em sub-bandas. Os sinais são mostrados com marcações de *onset* obtidas do sinal de referência `MIDI` da Figura 4.3.

Apêndice C

Aplicativo: jQueryByHumming

Desenvolvido como aplicativo fim deste trabalho, o sistema de pesquisa de músicas através de solfejos denominado **jQueryByHumming** possui as seguintes características técnicas:

- Criado em linguagem JAVA, com JDK (*Java Development Kit*) 1.7;
- Utiliza *Spring*: *Framework* baseado nos padrões de projeto inversão de controle (IoC) e injeção de dependência;
- Utiliza *Maven*: Ferramenta de automação de compilação;
- Utiliza arquitetura MVC (*Model*, *View* e *Control*) sem a utilização de biblioteca ou *framework* de terceiros.
- Utiliza *Elastic Search*: Baseado no Apache Lucene e desenvolvido em JAVA, é um servidor de buscas distribuído, REST, de código-fonte aberto.

O aplicativo está hospedado no serviço de *Web Hosting* de compartilhamento de projetos GitHub no endereço `https://github.com/AlexCaranha/jQueryByHumming`. Os passos seguintes descrevem como proceder para a primeira execução do aplicativo:

1. Instalar o controle de versão distribuído Git. Em `http://git-scm.com/book/pt-br/Primeiros-passos-Instalando-Git` é descrito um passo-a-passo de como instalar este controle de versão nos sistemas operacionais *Windows*, *Linux* e *Mac*;
2. Instalar a ferramente de automação de compilação *Maven*. Em `http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html` é descrito um passo-a-passo de como instalar esta ferramenta.

3. Instalar o servidor de buscas *Elastic Search*. Em `http://www.elasticsearch.org/guide/reference/setup/installation/` é descrito como a instalação deve ser realizada.

4. Obter o código-fonte do aplicativo. Isso é feito no terminal/console através do comando: "git clone https://github.com/AlexCaranha/jQueryByHumming";

5. Gerar versão `jar` com dependências. Estando no diretório clonado pelo comando anterior e ainda no terminal/console, digite: "mvn install". O *Maven* encarregar-se-á de baixar a versão mais atual das dependências, inclusive das dependências das dependências se necessário. Uma conexão com a internet é indispensável. Após a conclusão será criada no diretório atual uma pasta de nome *target*;

6. Iniciar o processo do servidor de buscas. No endereço informado no passo 3, é explicado como iniciar o processo.

7. Instalação da base de dados no servidor de buscas. Estando na pasta *jQueryByHumming/target*, digite no terminal/console o comando: "java -jar jQueryByHumming-1.0-jar-with-dependencies.jar db:reset". Estando todos os passos reproduzidos corretamente você verá alguns registros informando dados de *id*, *type* e *index*. Em caso contrário procure certificar-se que os passos foram realizados corretamente.

8. Execução do aplicativo **jQueryByHumming**. Estando na pasta *jQueryByHumming/target*, digite no terminal/console o comando: "java -jar jQueryByHumming-1.0-jar-with-dependencies.jar".

A seguir são listadas as bibliotecas de terceiros utilizada no aplicativo, com as suas respectivas licenças:

- **musicg**: Biblioteca de análise de áudio. Licença: *Apache License 2.0*;
- **android-midi-lib**: Biblioteca de manipulação de dados `MIDI`. Licença: *Apache License 2.0*;
- **jfftpack**: Biblioteca de processsamento de FFT baseada, sendo a versão java da fftpack. Licença: Domínio Público;
- **guava-libraries**: Bibliotecas de classes utilitárias da empresa Google. Licença: *Apache License 2.0*;
- **jama**: Pacote de manipulação de matrizes. Licença: Domínio Público;

- **jFreeChart**: Biblioteca de construção de gráficos de qualidade profissional. Licença: *LGPL*;
- **Xstream**: Biblioteca de serialização de objetos em XML e vice-versa. Licença: *BSD*;
- **Spring Framework**: *Framework* baseado nos padrões de projeto inversão de controle (IoC) e injeção de dependência. Licença: *Apache License*.
- **Maven**: Ferramenta de automação de compilação. Licença: *Apache License 2.0*.

A seguir são mostradas um conjunto de telas extraídas do aplicativo **jQueryByHumming** para efeito de ilustração:

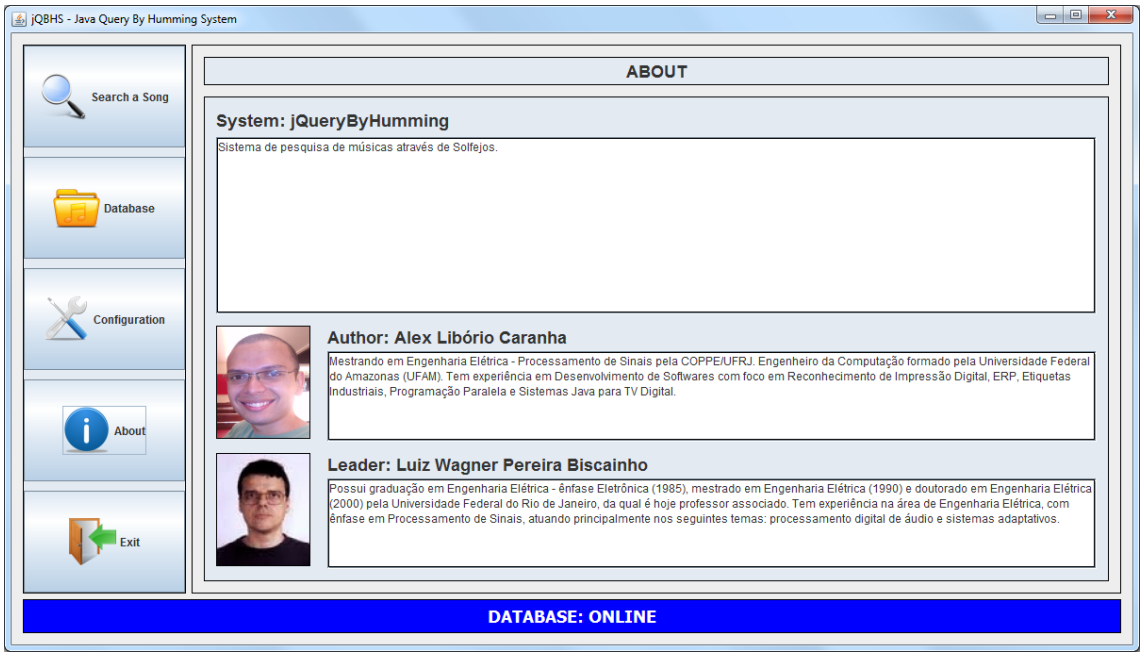


Figura C.1: Tela de *About* do aplicativo.

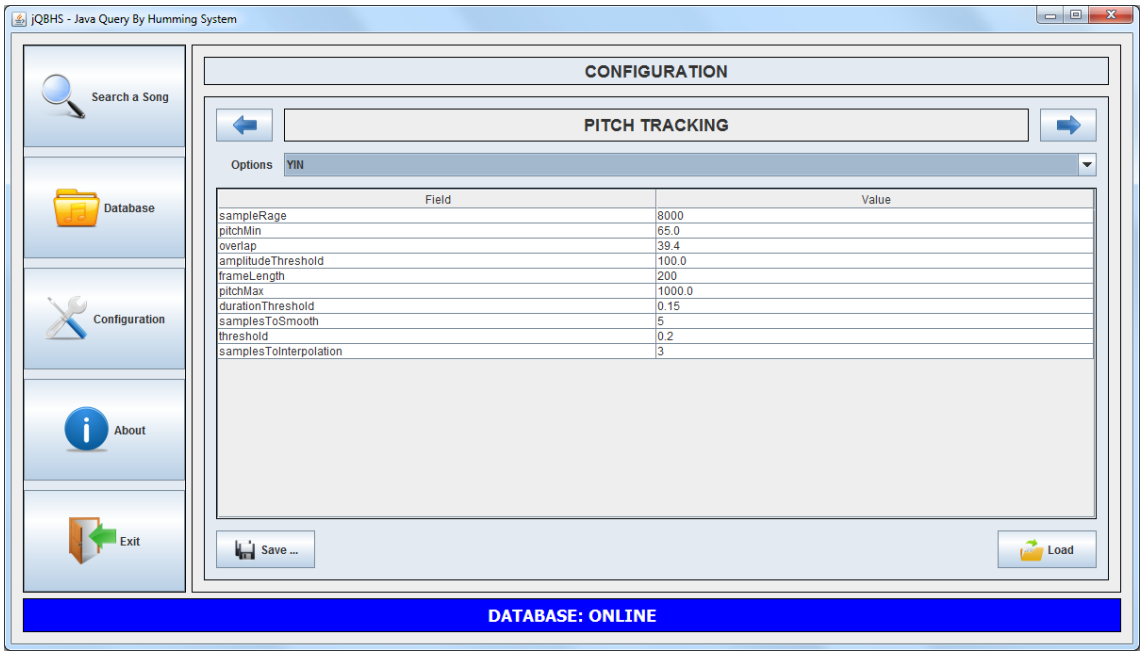


Figura C.2: Tela de Configuração do aplicativo.

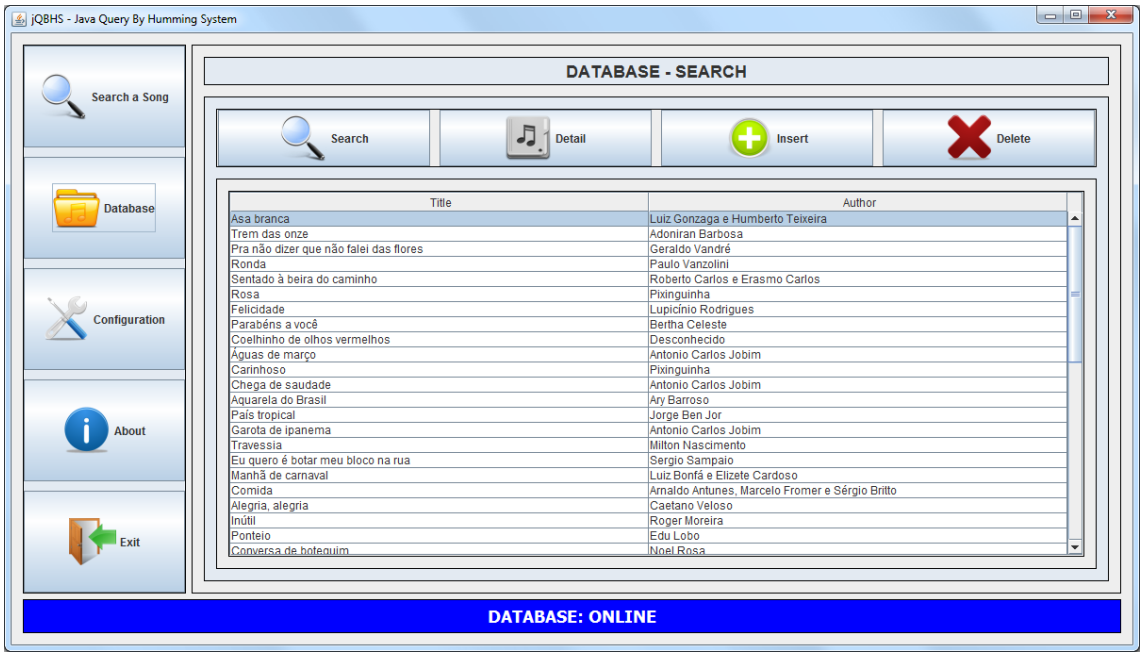


Figura C.3: Tela de listagem de músicas da base de dados do aplicativo.

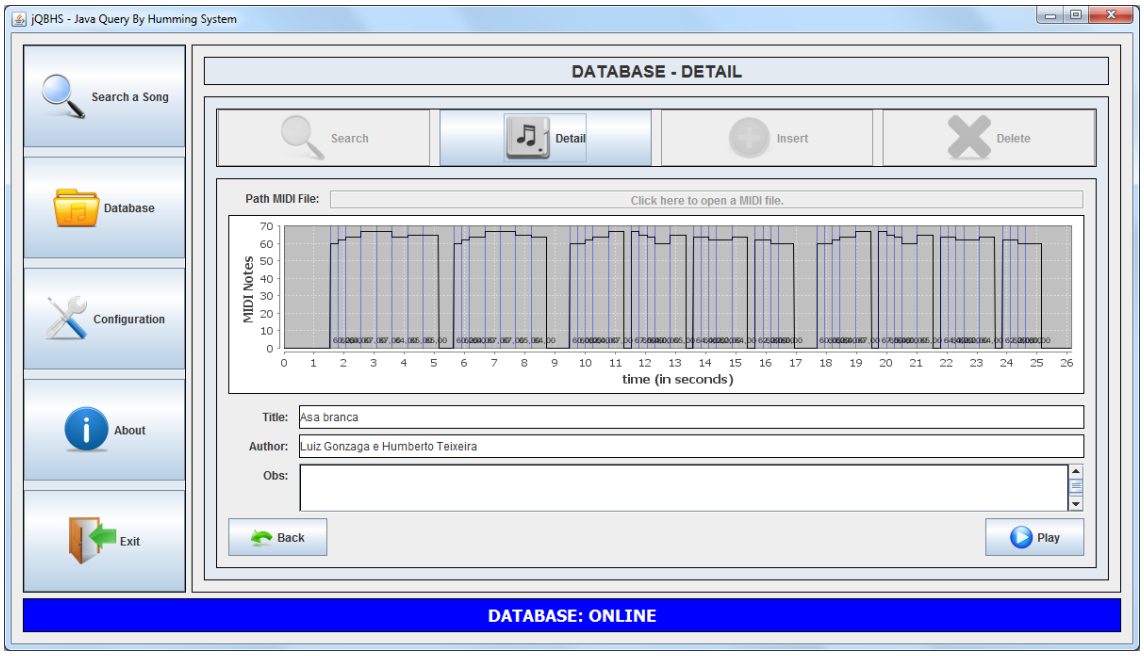


Figura C.4: Tela de detalhe de música da base de dados do aplicativo.

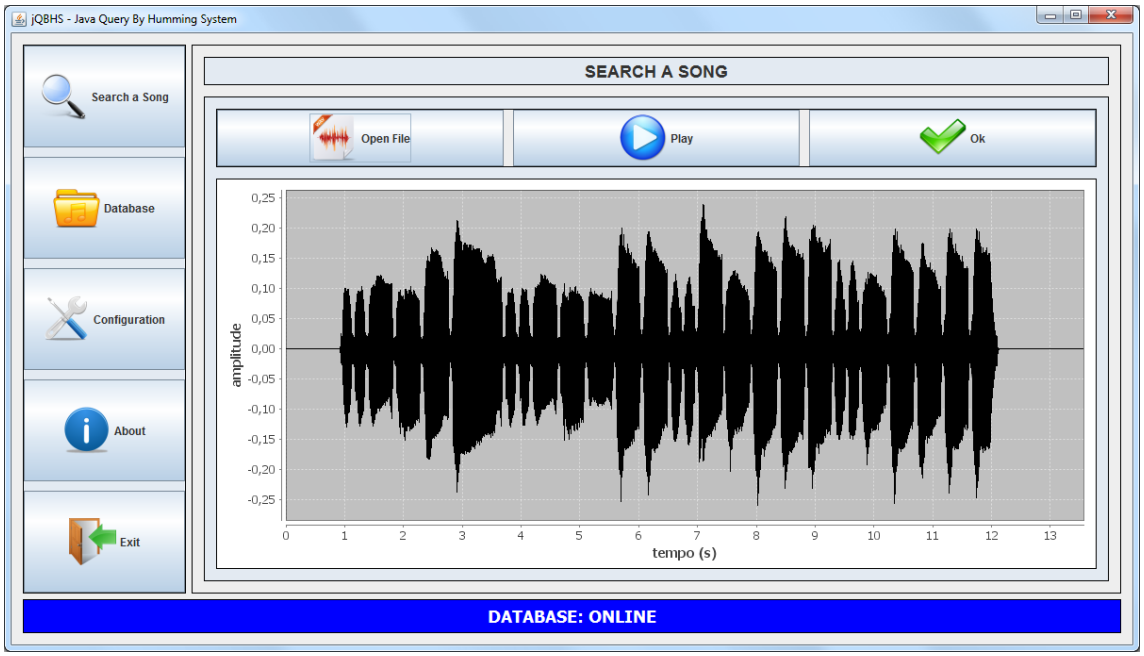


Figura C.5: Tela de pesquisa de música do aplicativo.

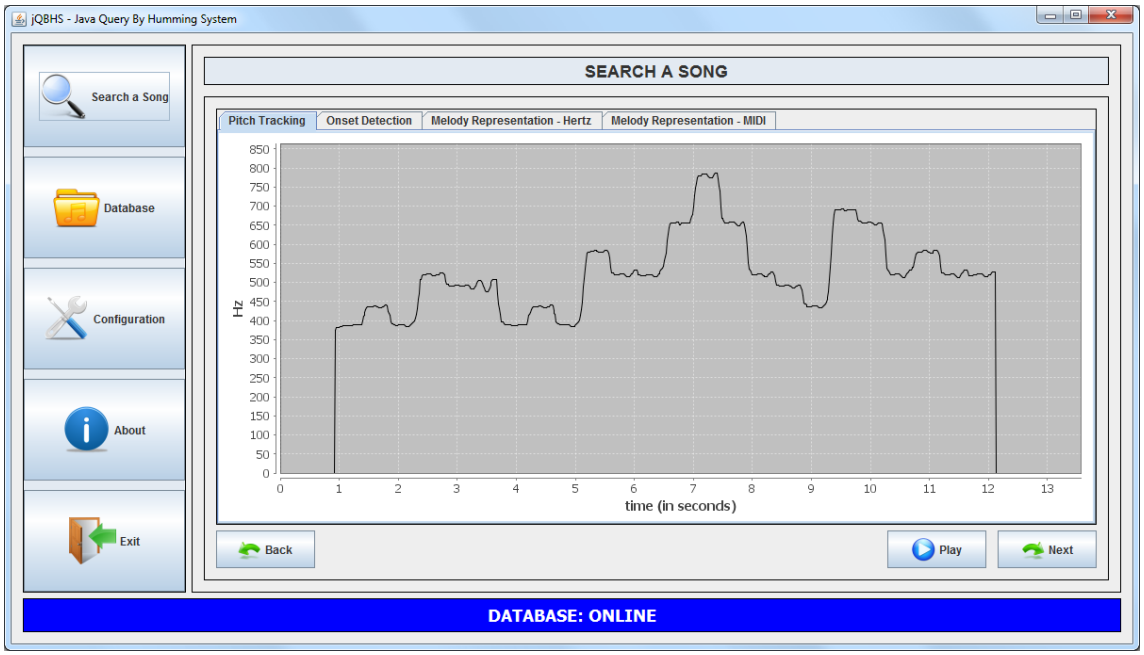


Figura C.6: Tela de pesquisa de música - *Pitch Tracking* do aplicativo.

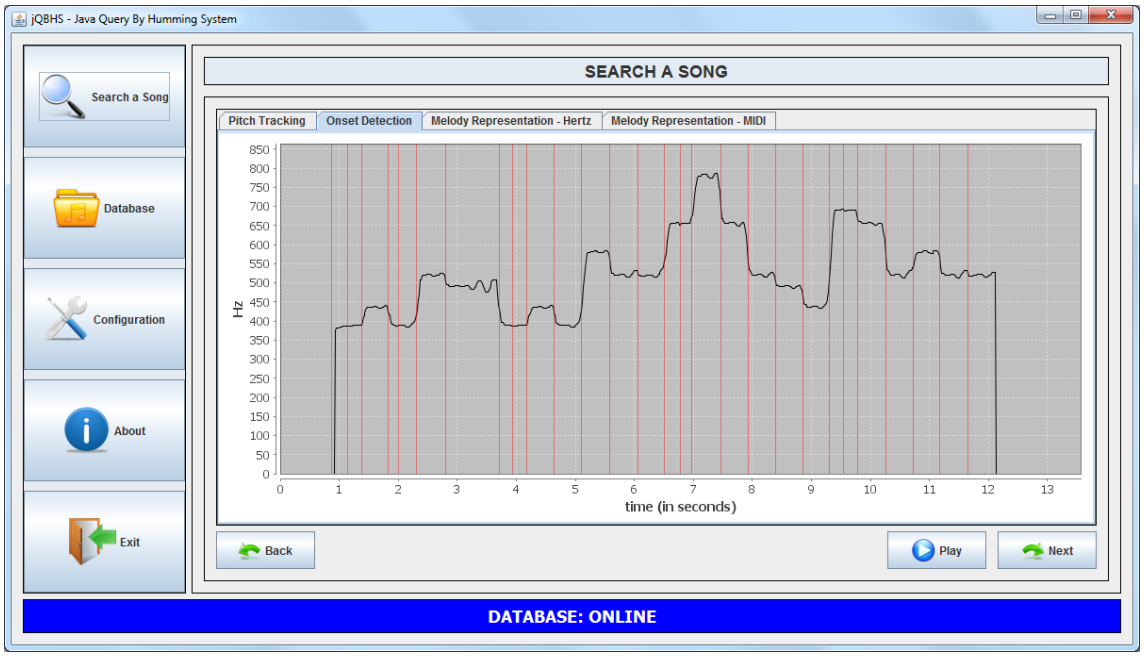


Figura C.7: Tela de pesquisa de música - *Onset Detection* do aplicativo.

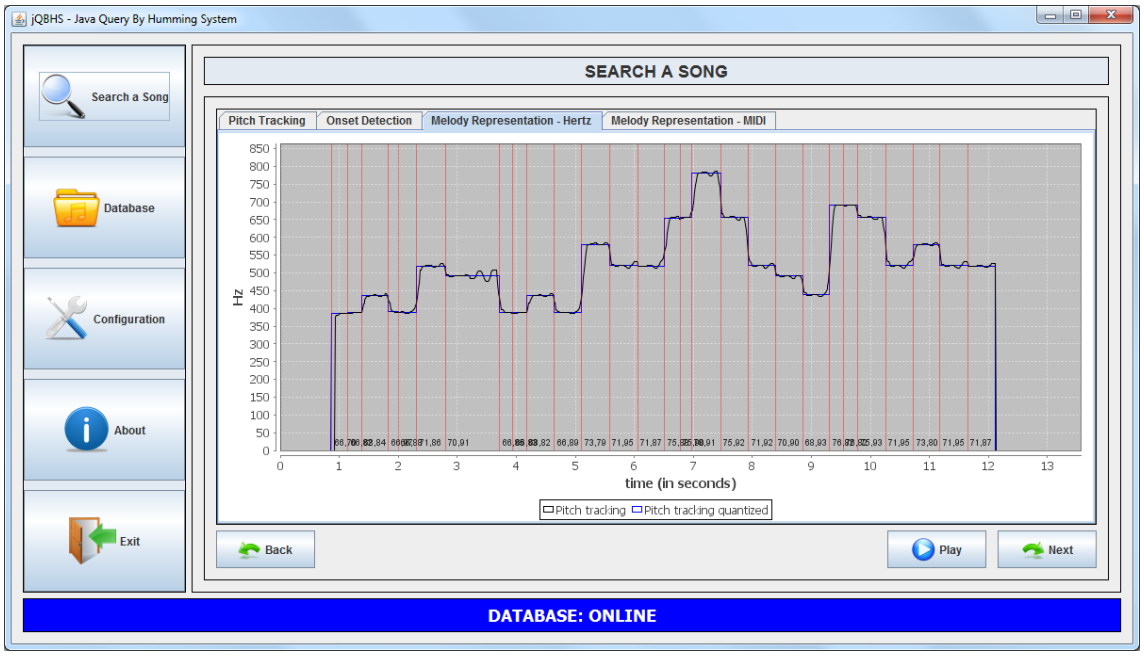


Figura C.8: Tela de pesquisa de música - *Melody Representation* do aplicativo.

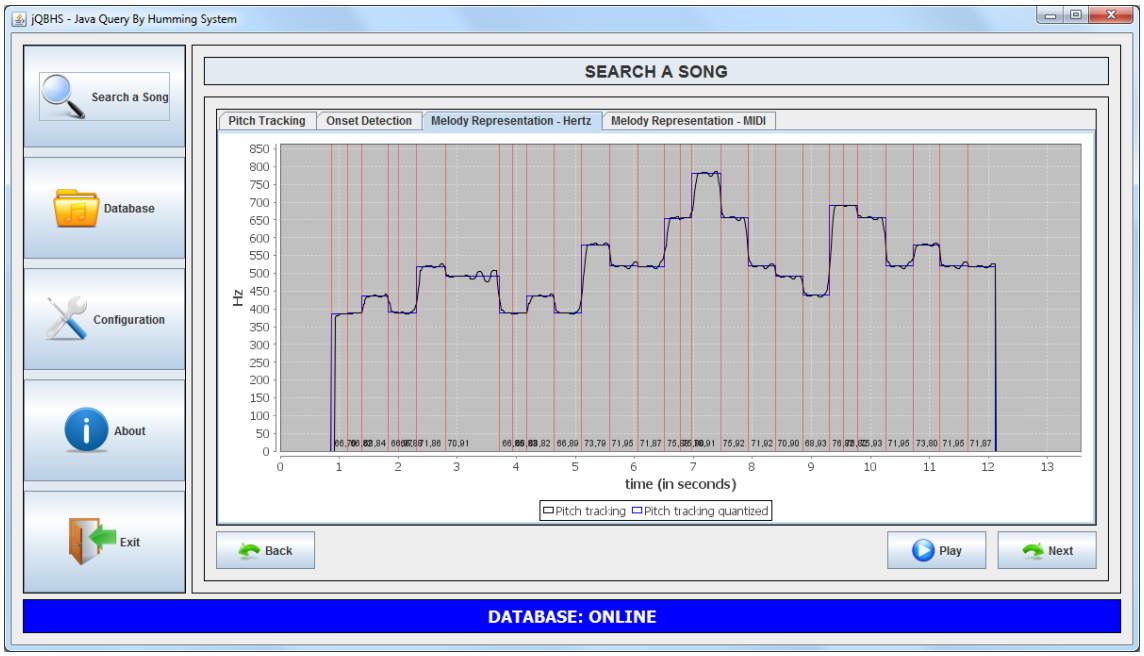


Figura C.9: Tela de pesquisa de música - *Melody Representation* em `MIDI` do aplicativo.

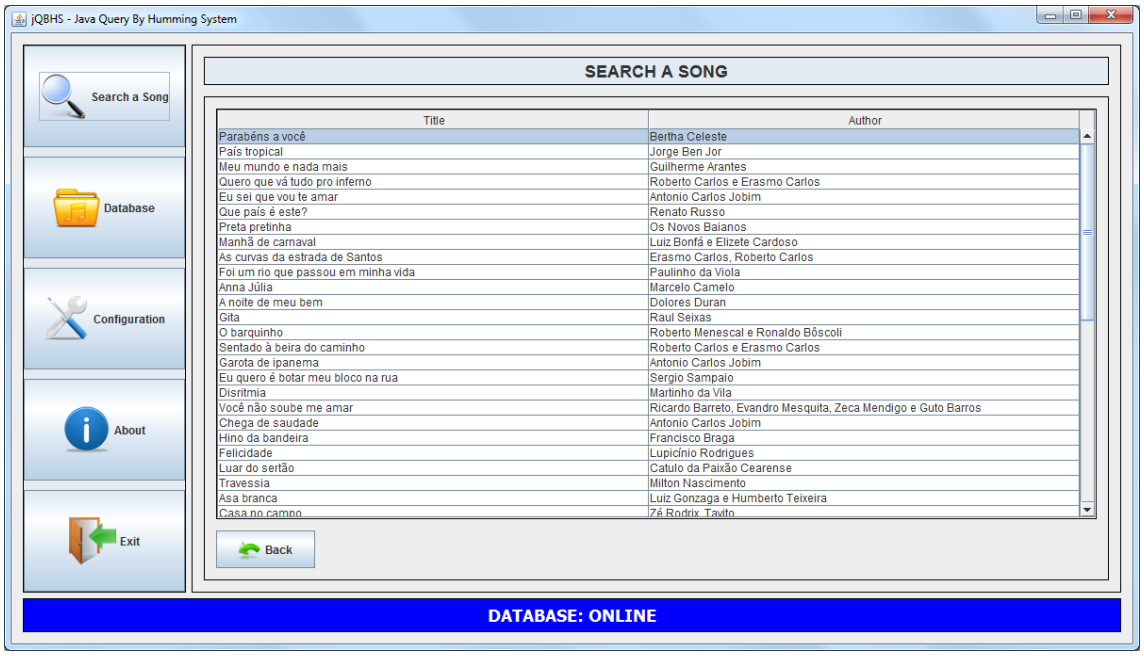
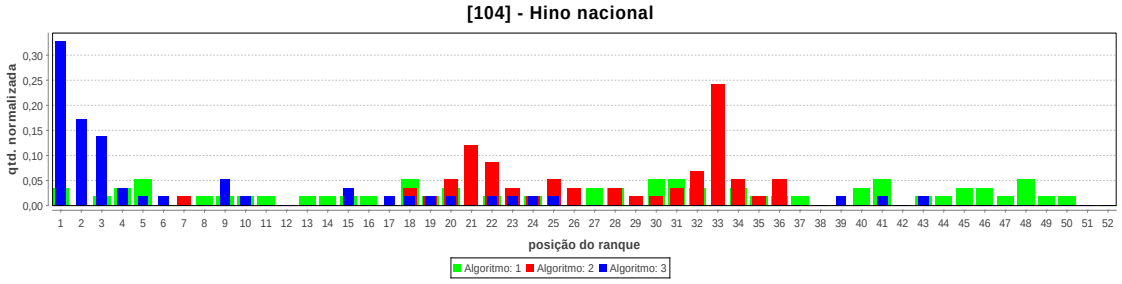


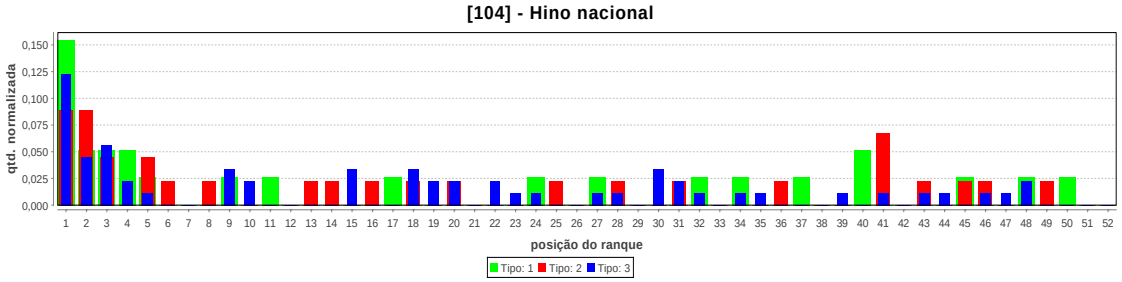
Figura C.10: Tela de pesquisa de música - lista de músicas retornadas pelo aplicativo.

Apêndice D

Gráficos de análise das dez músicas de maior número de gravações de solfejos

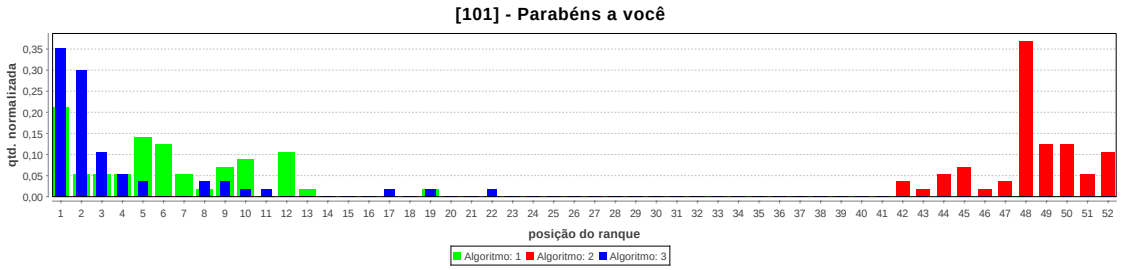


(a) Por Algoritmos.

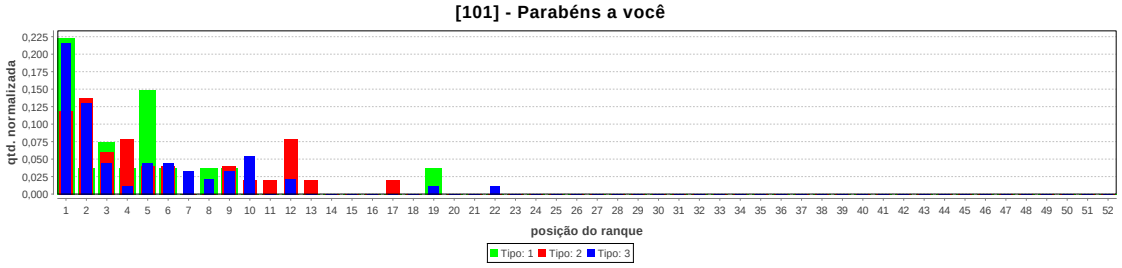


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.1: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música “Hino Nacional” para os algoritmos de comparação de melodias (a) e tipos de gravação (b).

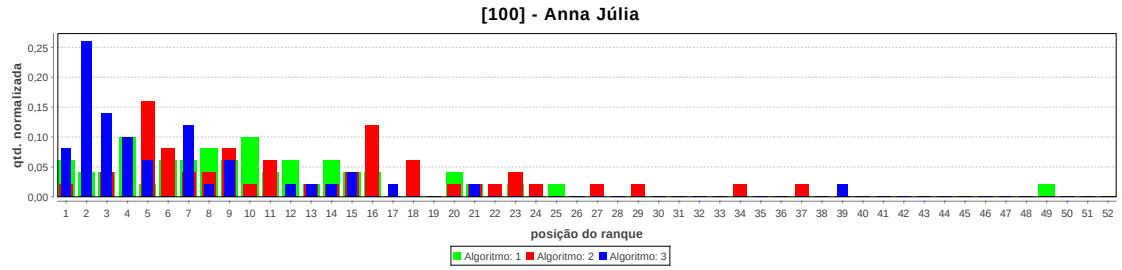


(a) Por Algoritmos.

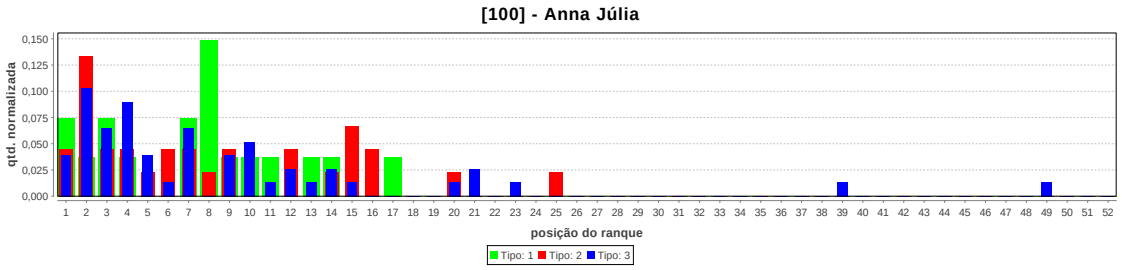


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.2: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Parabéns a você" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).

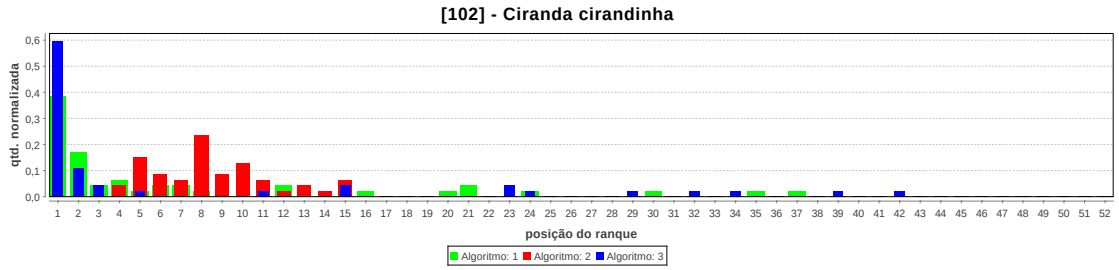


(a) Por Algoritmos.

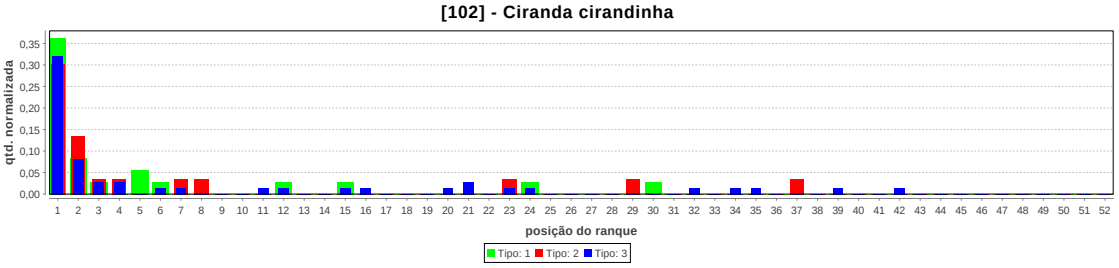


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.3: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Anna Júlia" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).

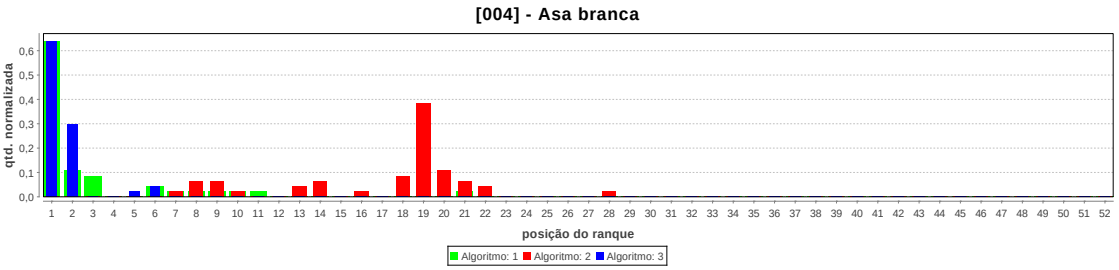


(a) Por Algoritmos.

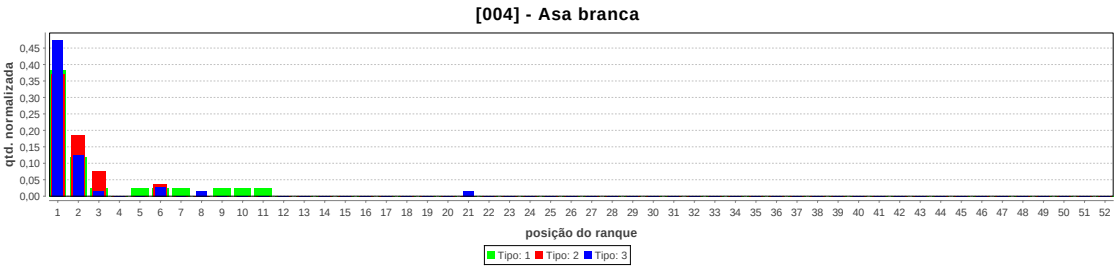


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.4: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Ciranda cirandinha" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).

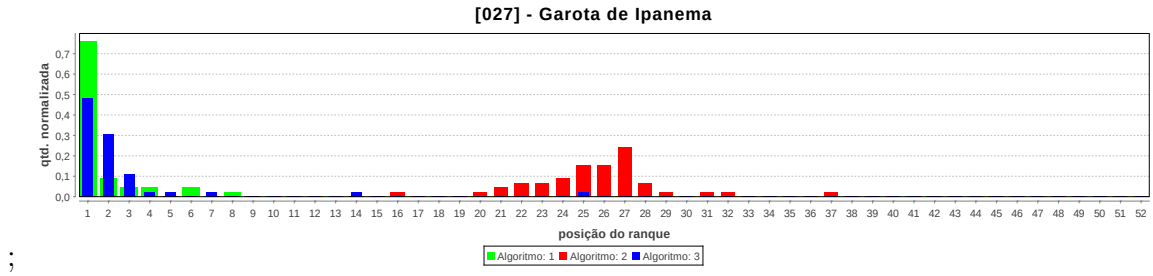


(a) Por Algoritmos.



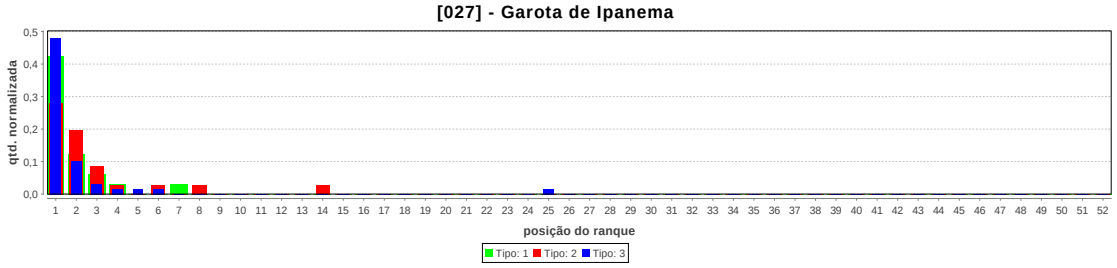
(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.5: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Asa branca" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).



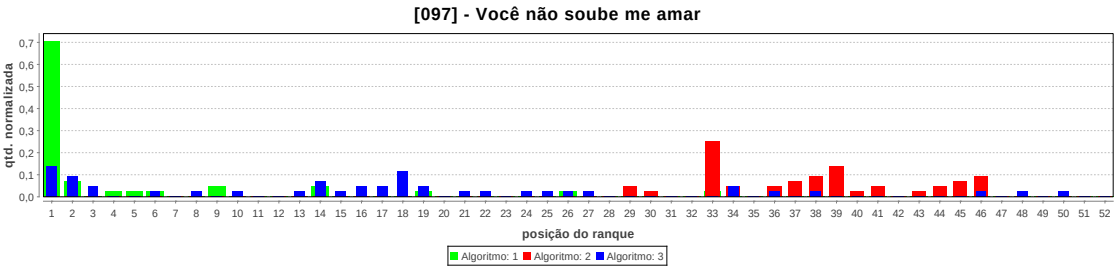
;

(a) Por Algoritmos.

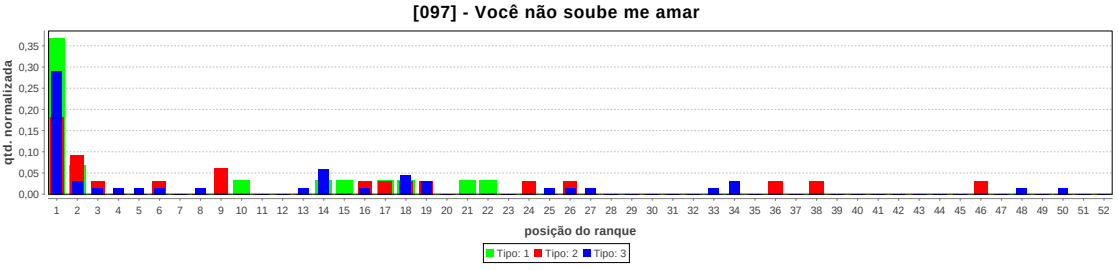


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.6: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Garota de Ipanema" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).

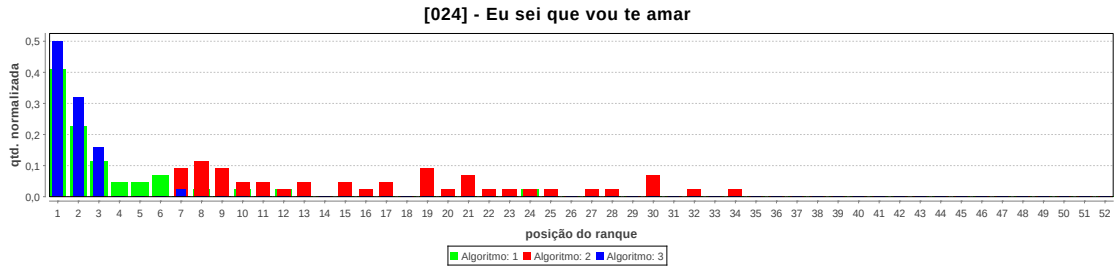


(a) Por Algoritmos.

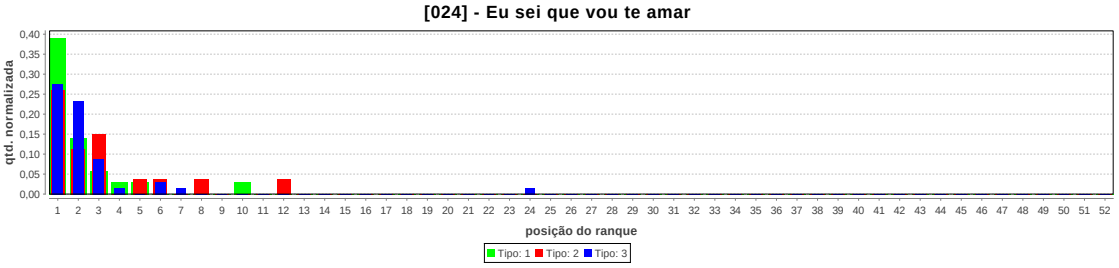


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.7: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Você não soube me amar" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).

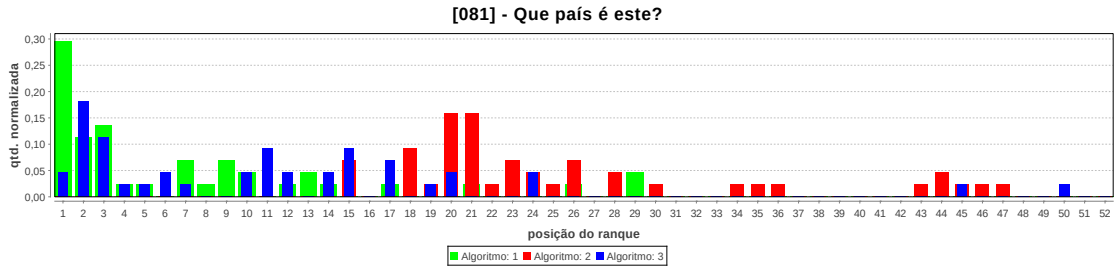


(a) Por Algoritmos.

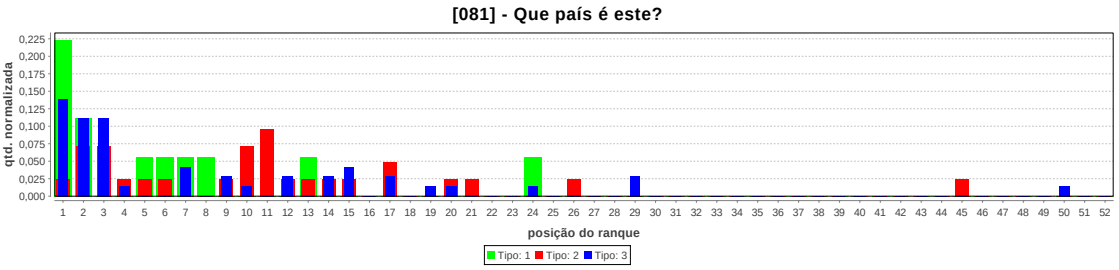


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.8: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Eu sei que vou te amar" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).

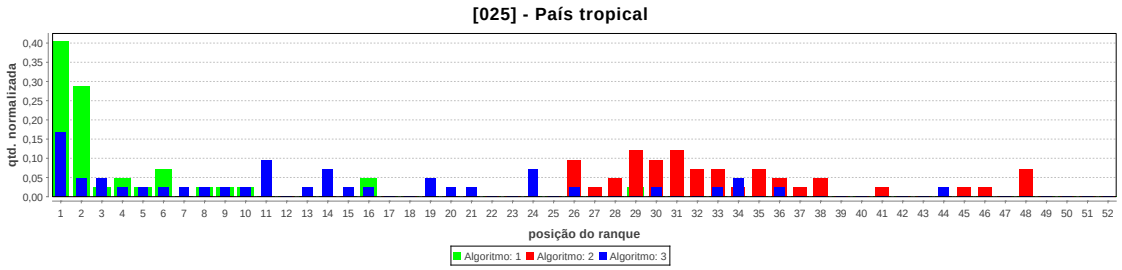


(a) Por Algoritmos.

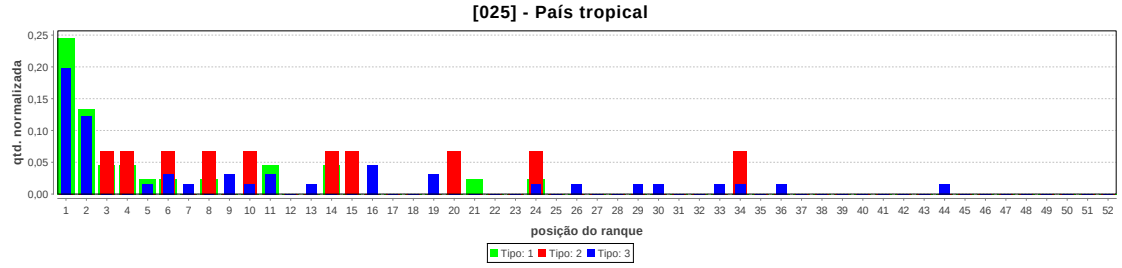


(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.9: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "Que país é este?" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).



(a) Por Algoritmos.



(b) Por Tipos, excluindo as medições referentes ao Algoritmo 2.

Figura D.10: Gráfico gerado a partir da medição de ocorrências normalizadas por posição no ranque da música "País tropical" para os algoritmos de comparação de melodias (a) e tipos de gravação (b).