ESCOLA POLITÉCNICA
DE PERNAMBUCO

d.s.c. UPE
Departamento de Sistemas e Computação

Ferramenta de Apoio ao Ensino de Instrumento Musical

Trabalho de Conclusão de Curso

Engenharia da Computação

Fábio Delicato Feijó de Melo
Orientador: Prof. Carmelo Jose Albanez Bastos Filho

UNIVERSIDADE
DE PERNAMBUCO

Fábio Delicato Feijó de Melo

Ferramenta de Apoio ao Ensino de Instrumento Musical

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Novembro de 2009.

Aos meus pais.

Agradecimentos

Agradeço primeiramente à minha família por ter sempre me fornecido todo o suporte necessário. Isso é uma das coisas mais importantes na minha vida.

Aos meus colegas de turma que puderam tornar a convivência mais leve e agradável durante os anos da graduação.

Ao meu professor e orientador Carmelo, por essa e outras orientações, contribuindo assim diretamente para minha formação.

Agradeço também a todas as pessoas que contribuíram, direta ou indiretamente, para essa realização.

Resumo

O ensino da música para iniciantes é uma atividade que sempre oferece um grau de dificuldade, porque o aluno é exposto a uma grande quantidade de informações novas, tanto para o aprendizado do instrumento, quanto para o aprendizado de teoria musical. Este trabalho descreve o desenvolvimento de uma ferramenta computacional para o apoio ao ensino de instrumento musical para iniciantes. Essa ferramenta utiliza-se de técnicas de processamento de sinais para comparar uma partitura previamente importada pelo sistema com a execução real feita pelo aluno, apontando as diferenças encontradas. A ferramenta importa as partituras a serem executadas através de arquivos no formato MusicXML, e verifica a execução do aluno obtendo os dados da interface de som do computador, ao qual o instrumento a ser executado pelo aluno deve estar ligado. A ferramenta também dispõe de um visualizador do espectro de freqüência do sinal recebido da interface de som do computador. É apresentado também neste trabalho um estudo de caso, em que a principal funcionalidade da ferramenta é testada, através da análise da execução de duas versões diferentes da música Asa Branca, de Luiz Gonzaga e Humberto Teixeira. Em uma das versões a música é executada corretamente, e em outra existem alguns erros inseridos para realização dos testes. Esses erros devem ser identificados pela ferramenta, que deve indicar o tipo de cada erro, e também a sua localização na partitura.

Abstract

The teaching of music for beginners always offers some difficulty, because the student is exposed to a lot of new information, mainly regarding the instrument and the music theory. This paper describes the development of a computational tool to support the teaching of musical instrument for beginners. This tool uses signal processing techniques to compare a score previously imported by the system with an actual student‘s performance, pointing out the differences found. The tool imports the scores to be performed from a MusicXML file, and checks the student’s performance by obtaining the data from the computer’s sound interface, to which the student’s instrument must be connected. The tool has also a frequency spectrum viewer of the data received from the computer’s sound interface. This paper also presents a case study, where the main feature provided by the tool has been tested by analyzing the performance of two different versions of the music Asa Branca, by Luiz Gonzaga and Humberto Teixeira. In one version the music is performed correctly, and in another there are some errors inserted as test cases. These errors must be identified by the tool, which should indicate each error’s type and its location in the score.

Sumário

Índice de Figuras

Tabela de Símbolos e Siglas

A/D – Analógico Digital

AF – Audio-freqüência

API - *Application Programming Interface* (*Interface* de Programação de Aplicativos)

D/A – Digital/Analógico

DFT – *Discrete Fourier Transform* (Transformada Discreta de Fourier)

DSP – *Digital Signal Processing* (Processamento Digital de Sinais)

FFT – *Fast Fourier Transform* (Transformada Rápida de Fourier)

RF – Rádio-Freqüência

SNR – Signal-to-Noise Ratio (Relação Sinal-Ruído)

XML – *Extensible Markup Language*

Capítulo 1

Introdução

O ensino da música para iniciantes é uma atividade que sempre oferece um certo grau de dificuldade, pois o aprendizado trata-se, inicialmente, de conhecer uma linguagem completamente nova, que é a de notação musical em partituras [1][2]. Essa linguagem utiliza-se de uma série de símbolos para representar todos os detalhes de uma composição musical, e um grande período de tempo é necessário até que o aluno torne-se fluente em sua leitura.

Além disso, o aluno iniciante também começa com os primeiros passos do trabalho de sua percepção para os ritmos (intervalos de tempo entre cada uma das notas em uma seqüência), sons monofônicos (onde no máximo uma nota pode estar soando num certo momento) e polifônicos (pode existir mais de uma nota soando ao mesmo tempo), além de desenvolver sua percepção harmônica (onde se estuda a função de cada nota dentro de uma seqüência polifônica). Tudo isso, obviamente, em paralelo com o aprendizado de um instrumento musical, e com isso a necessidade de habituar-se com uma série de coisas novas, como uma postura diferente [3][4][5], forma de segurar o instrumento, posicionamento das mãos, etc.

Com todos esses fatores é natural que um iniciante na música sinta, a princípio, uma certa dificuldade de ambientar-se, pois é exposto a uma grande quantidade de informações com as quais não está habituado.

1.1 Motivação

Geralmente, ao iniciar seus estudos em um instrumento musical, o aluno tem pouco ou nenhum conhecimento teórico-musical. Isso é mais uma barreira para a clareza do entendimento das primeiras peças com que entra em contato, o que o leva a interpretar muitas vezes de forma errônea as primeiras partituras de peças do seu instrumento. Por isso, uma ferramenta de apoio ao ensino pode ser de grande valia para esse momento do aprendizado.

A informática já é utilizada de diversas formas como ferramenta de apoio ao ensino, inclusive da música, onde se pode encontrar muitas ferramentas que auxiliam o aprendizado do estudante, como afinadores de instrumentos [6], dicionários de acordes [7], cursos completos multimídia [8], etc.

Utilizando-se de métodos de processamento sinais digitais (DSP – *Digital Signal Processing*) [9][10], pode-se, através do monitoramento do sinal obtido da interface de áudio de um computador ligado a um instrumento musical, construir uma ferramenta que possa auxiliar o aluno iniciante no entendimento das primeiras peças por ele executadas.

1.2 Objetivos

O presente trabalho tem como objetivo a construção de uma ferramenta de apoio ao ensino de instrumento musical voltada para os iniciantes no estudo de um instrumento, com pouco ou nenhum conhecimento teórico-musical.

A ferramenta deve ter como entrada, primeiramente, uma partitura da peça que o aluno deseje executar, peça essa que será inserida no sistema através de um arquivo *MusicXML* [11]. Após a entrada da partitura, o aluno poderá executar a peça com o seu instrumento ligado ao computador. A ferramenta fará então uma comparação da partitura inserida no sistema com as notas que forem realmente tocadas pelo aluno, verificando as diferenças (causadas por erros na execução) e indicando as suas localizações e tipos. Desta forma, o aluno poderá ver que partes da peça foram executadas a contento, e em que partes cometeu enganos, podendo mais facilmente fazer as correções necessárias.

Para redução da complexidade no desenvolvimento da ferramenta, o projeto contemplará unicamente sons monofônicos, não sendo permitidas portanto peças a mais de uma voz. Com o mesmo objetivo de redução da complexidade, será usado somente um instrumento para testar a ferramenta. O instrumento usado para os testes será o teclado (com som de piano), por ser um instrumento muito comum, de fácil ligação com a interface de som do computador e por oferecer uma onda sonora

relativamente limpa (e portanto de mais fácil reconhecimento), quando comparada à onda obtida de alguns outros instrumentos, como o violão, por exemplo.

1.3 Estrutura do Documento

O Capítulo 2 (Fundamentação Teórica) apresenta uma descrição das principais ferramentas estudadas e utilizadas durante o desenvolvimento do projeto. Todas as ferramentas descritas nesse capítulo foram utilizadas como base para tornar possível a implementação do sistema final.

No Capítulo 3 (Métodos Utilizados), pode-se encontrar a abordagem que foi adotada para a implementação das principais funcionalidades da ferramenta desenvolvida, desde a importação dos arquivos *MusicXML* pelo sistema, até a comparação das Notas tocadas com as notas obtidas do arquivo importado e a apresentação dos resultados, passando por todo o tratamento recebido pelo sinal de som dentro da aplicação. Esse capítulo oferece uma visão mais específica dos detalhes de implementação.

O Capítulo 4 (A Ferramenta Desenvolvida) apresenta uma visão geral da ferramenta, mostrando suas telas e como utilizá-las. Neste capítulo também é apresentado um estudo de caso, onde pode-se encontrar os resultados obtidos para a execução de duas versões da música Asa Branca, de Luiz Gonzaga e Humberto Teixeira, sendo uma das versões uma execução correta da música e a outra uma execução onde foram inseridos alguns erros, afim de testar as funcionalidades da ferramenta.

O Capítulo 5 (Conclusões e Trabalhos Futuros) conclui o trabalho, indicando os possíveis trabalhos que podem ser desenvolvidos para o aperfeiçoamento da ferramenta, assim como as dificuldades encontradas no decorrer do desenvolvimento.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentadas as principais ferramentas utilizadas no desenvolvimento de todo o projeto. Nele, são encontrados os referenciais teóricos necessários para a implementação do sistema final.

Como a ferramenta desenvolvida envolve processamento de som, uma parte deste capítulo trata desse assunto e mais alguns assuntos envolvidos de alguma forma, como a FFT (*Fast Fourier Transform* – Transformada Rápida de Fourier) [9][12][13][14][15][16] e as freqüências das notas musicais. Outro assunto tratado neste capítulo é a notação musical em partituras e sua representação em *MusicXML* [11].

2.1 O Som e Sua Representação Digital

O Som na natureza se dá por aumentos e reduções periódicas da densidade do ar (ou outro meio elástico) [17], dentro da faixa de Áudio-Freqüência (AF) [18], que se estende desde 20 Hz até 20 KHz. Esses aumentos e reduções de densidade se propagam pelo meio, causando uma perturbação em volta do seu ponto inicial (fonte sonora). As perturbações com freqüências acima de 20 KHz são chamadas de Rádio-Freqüência (RF), e não são percebidos pela nossa audição.

Essas variações na densidade do meio causam compressões e rarefações por todo o meio em volta, fazendo com que a pressão sofra uma variação no decorrer do tempo, formando assim uma onda sonora. Estas variações na pressão podem então ser percebidas por algum instrumento próprio para isso, como por exemplo o ouvido humano, desde que estejam em uma freqüência adequada ao instrumento usado para captá-las.

Se a variação de pressão for medida no decorrer do tempo, pode-se fazer um gráfico da onda sonora, dado pela variação da pressão no ambiente (eixo Y) em

função do tempo (eixo X). A Figura 1 mostra um exemplo de um gráfico de onda sonora.

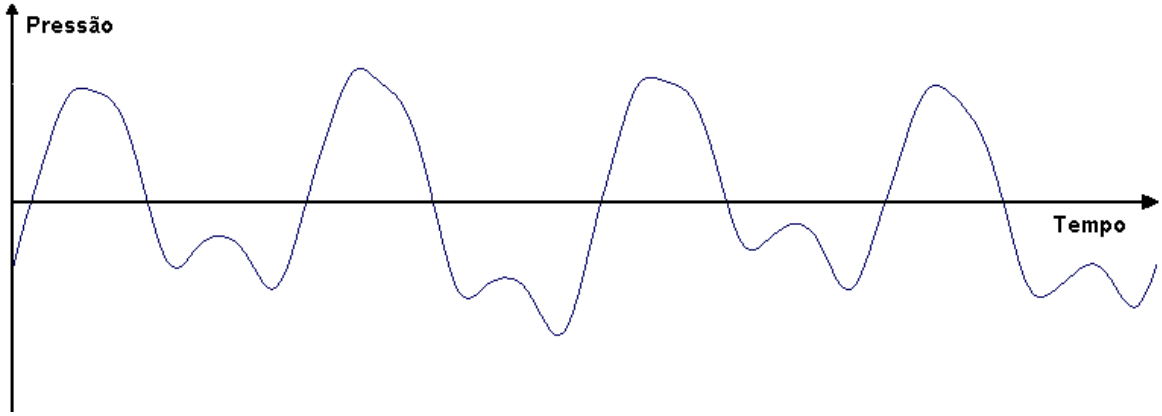

Figura 1. Exemplo de evolução temporal de uma onda sonora (Pressão do meio em função do Tempo).

Após a passagem por um conversor A/D (Analógico/Digital), a forma de onda analógica é separada em amostras individuais em intervalos de tempo regulares, onde um conjunto dessas amostras, que é a representação digital do sinal, representa uma aproximação do sinal original. Assim, quanto maior a quantidade de amostras para um período de tempo, tem-se um sinal mais fiel ao sinal original analógico. Para representar cada uma dessas amostras, é utilizada uma certa quantidade de bits. Quanto maior a quantidade de bits utilizada para representar uma amostra, tem-se um sinal digital mais próximo do sinal analógico original. Essas amostras são utilizadas para a reconstrução do sinal analógico, e este processo é executado por um conversor D/A (Digital/Analógico).

Desta forma, uma representação digital do som é armazenada definindo-se a quantidade de amostras por segundo (Taxa de Amostragem – *Sample Rate*) e pela quantidade de bits utilizada para representar cada amostra (Quantização - *Quantization*). A Figura 2 mostra o gráfico de um som em sua representação digital com duas taxas de amostragem diferentes. Pode-se notar pela figura que a representação com uma taxa de amostragem maior (32000 Hz, na parte de cima da figura) representa bem melhor a onda do que a representação com a menor taxa de amostragem (11025 Hz, na parte de baixo da figura). Essas formas de onda foram obtidas diretamente de uma ferramenta de edição de áudio, e representam sons reais gravados.

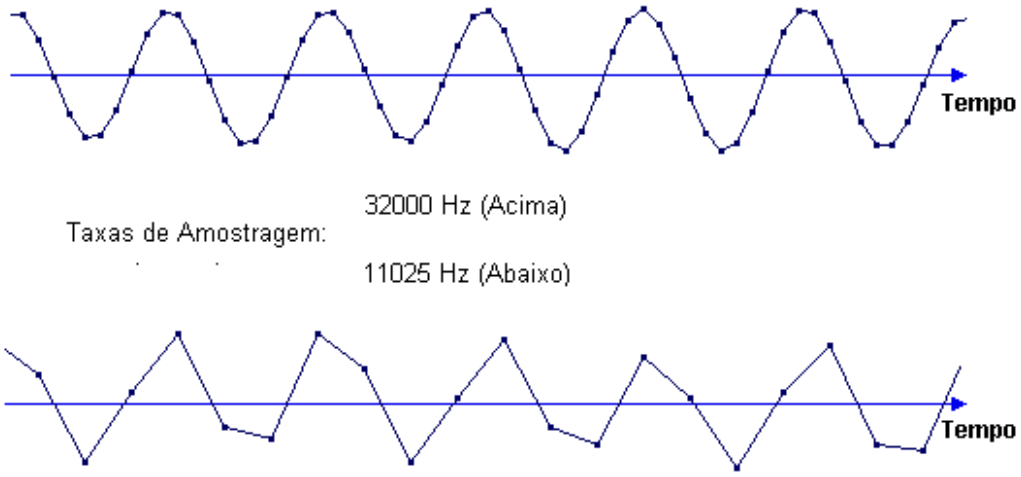

Figura 2. Exemplos de ondas sonoras com diferentes taxas de amostragem.

Um cd de áudio, por exemplo, tem uma taxa de amostragem de 44100 Hz e taxa de quantização de 16 bits, o que lhe confere uma alta fidelidade aos sons da gravação original.

2.2 Processamento de Som e a FFT

Pode-se entender por processamento (digital) de som, alguma manipulação em um sinal digital de áudio, visando um objetivo específico. Muitas dessas manipulações podem ser feitas diretamente sobre o sinal amostrado, como por exemplo, uma amplificação do volume. Pode-se identificar também os instantes em que a fonte sonora foi ativada, como por exemplo, um ataque de uma nota em um instrumento (momento em que o instrumentista toca uma nota). Além disso, pode-se identificar o instrumento gerador do som, visto que diferentes instrumentos geram, para uma mesma nota, formas de onda diferentes. Isto ocorre porque a potência dos harmônicos (componentes de freqüência existentes no sinal quando uma nota está soando, que são ativados por ressonância) variam de acordo com o instrumento, causando uma diferença no formato da onda.

Na figura 3, pode-se observar o resultado de uma atenuação do volume de um pequeno trecho (de 4 segundos) de uma peça. O sinal mostrado na parte inferior da figura é o mesmo da parte superior, mas com o volume reduzido a 25% do

original. Pela observação do sinal, pode-se também identificar em quem momentos as notas estão sendo atacadas (tocadas) pelo instrumentista.

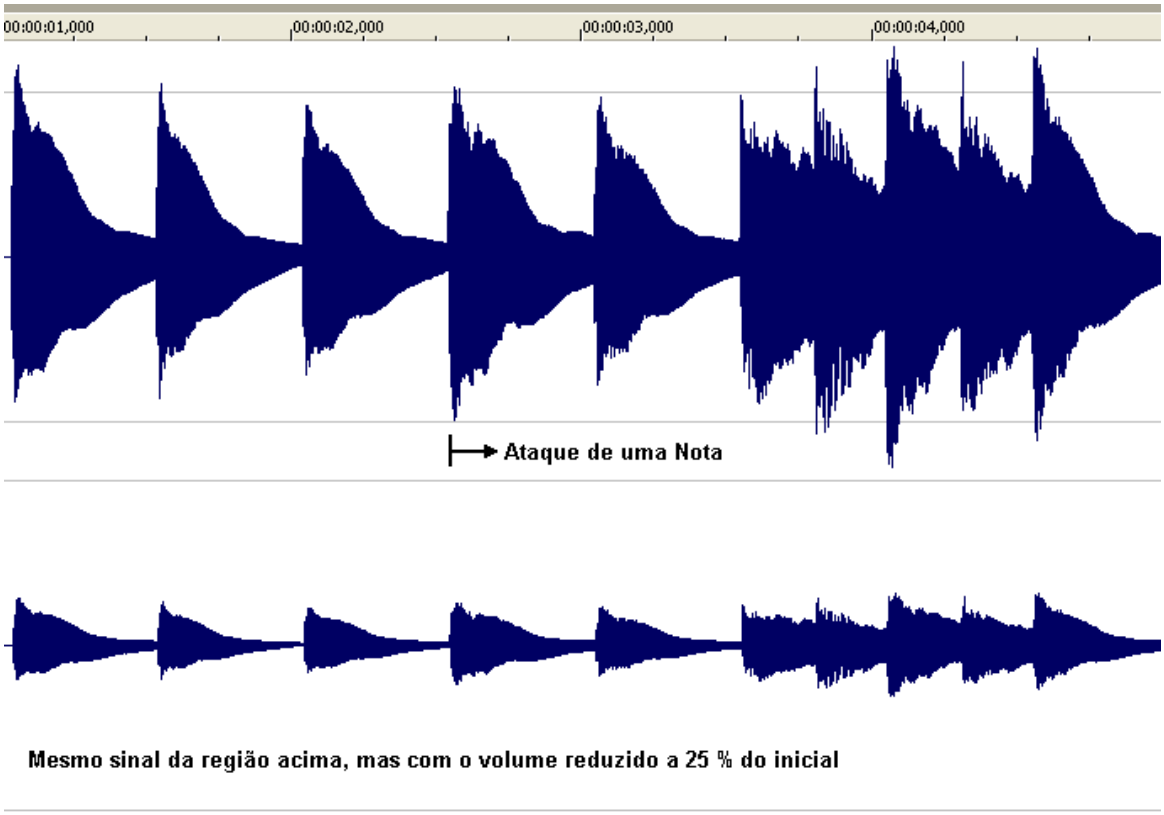

Figura 3. Alteração no volume de um sinal digital de som.

Já na Figura 4, pode-se observar os sinais obtidos a partir da mesma nota sendo tocada em diferentes instrumentos, sendo a parte superior da figura um sinal obtido de um violoncelo e a parte inferior um sinal obtido de um piano.

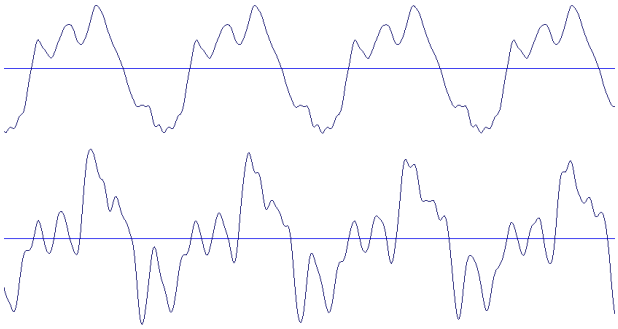

Figura 4. Sinais de uma mesma nota tocada em instrumentos diferentes: um violoncelo (acima) e um piano (abaixo).

Todos os exemplos anteriores de processamento de som levaram em consideração somente o sinal digital em sua representação no domínio do tempo, visto que a forma de onda é formada por amostras das variações de pressão no decorrer do tempo. Para todos esses casos, esse sinal oferece dados suficientes para realizar as operações necessárias.

Entretanto, o sinal no domínio do tempo não é adequado para muitas aplicações em processamento de som. Principalmente quando as manipulações no sinal são na verdade realizadas nas freqüências que o compõem. Esse tipo de processamento é extremamente comum, sendo a base para filtros, equalizadores, afinadores de instrumentos, softwares com recursos de afinação de trechos específicos de uma música, remoção de alguns tipo de ruídos em faixas de áudio, etc. Todas essas aplicações manipulam de alguma forma as componentes de freqüência do som, e para possibilitar sua existência, existe uma ferramenta que transforma o sinal com representação no domínio do tempo para uma representação desse sinal no domínio da freqüência, facilitando a análise das freqüências que compõem o sinal. Essa ferramenta é a FFT (*Fast Fourier Transform*).

A FFT é uma otimização do algoritmo para cálculo da DFT (*Discrete Fourier Transform* – Transformada Discreta de Fourier) [9][12][15][16], que é a Transformada de Fourier [12][15] aplicada a uma seqüência de valores complexos discretos. A Transformada de Fourier é uma ferramenta muito utilizada na área de processamento de sinais, por possibilitar a transformação de uma função no domínio do tempo para uma representação da função no domínio da freqüência. Evidentemente, para os dispositivos digitais é utilizada sua versão discreta, a DFT.

Dada uma função contínua de uma variável $f(t)$, a sua transformada de Fourier $F(\omega)$ é definida por:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \,, \qquad (2.1)$$

e sua versão discreta, a DFT, é definida por:

$$X(m) = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi mn}{N}}, \qquad (2.2)$$

onde *x*[*n*] é uma seqüência discreta no domínio do tempo, e *X*[*m*] é a DFT de *x*[*n*]. Uma outra possível representação da DFT é dada por:

$$X(m) = \sum_{n=0}^{N-1} x[n] \left[\cos\left(\frac{2\pi mn}{N}\right) - jsen\left(\frac{2\pi mn}{N}\right) \right], \qquad (2.3)$$

que utiliza-se da relação de Euler [15], que é dada por $e^{j\theta} = \cos\theta + jsen\theta$, para apresentar a DFT de uma forma diferente.

Considerando um sinal com taxa de amostragem *S* (amostras/segundo), a DFT, sendo calculada para um sinal com *N* amostras, resulta no conteúdo espectral do sinal, onde pode-se verificar os componentes de freqüência que formam o sinal original. A freqüência fundamental do sinal é dada por $\frac{S}{N}$ [15], e sua magnitude será dada por $X(1)$. $X(0)$ é a componente contínua do sinal. $X(2)$ será a magnitude da freqüência duas vezes maior que a fundamental (segundo harmônico), $X(3)$ será a magnitude da freqüência três vezes maior que a fundamental (terceiro harmônico), e generalizando, $X(n)$ será a magnitude da freqüência *n* vezes maior que a fundamental (*n*-ésimo harmônico). Isso gera uma visão geral dos componentes de freqüência do sinal original.

A execução de uma implementação da DFT tem complexidade $O(n^2)$ [13]. Portanto, ao aumentar os números de amostras da seqüência de entrada, o desempenho do algoritmo degrada-se rapidamente, o que causa limitações às suas implementações. Assim, mesmo o DFT sendo um algoritmo de implementação simples, esta ineficiência torna-se um fator limitante para sua maior utilização em aplicações práticas.

Em 1965, Cooley e Tukey apresentaram um artigo [14] mostrando um algoritmo para o cálculo da DFT com complexidade $O(n\log(n))$ [13]. Esse algoritmo, por ser mais eficiente do que o algoritmo original, ficou conhecido como

transformada rápida de Fourier (*Fast Fourier Transform* – FFT). Esse algoritmo utiliza a abordagem dividir para conquistar, quando divide recursivamente a DFT em duas partes, calculando-as separadamente, e depois somando-as. Essa divisão é feita para os números de ordem par e ímpar da seqüência, representando a DFT da seguinte forma:

$$X(m) = \sum_{n=0}^{(N/2)-1} x[2n] e^{-\frac{j2\pi m 2n}{N}} + e^{-\frac{j2\pi m}{N}} \sum_{n=0}^{(N/2)-1} x[2n+1] e^{-\frac{j2\pi m(2n+1)}{N}} . \quad (2.4)$$

Simplificando a notação pode-se definir $W_N = e^{-\frac{j2\pi}{N}}$, ficando assim a equação anterior como:

$$X(m) = \sum_{n=0}^{(N/2)-1} x[2n] W_N^{2nm} + W_N^m \sum_{n=0}^{(N/2)-1} x[2n+1] W_N^{2nm}, \quad (2.5)$$

mas como $W_N^2 = e^{-\frac{j2\pi 2}{N}} = e^{-\frac{j2\pi}{\frac{N}{2}}} = W_{\frac{N}{2}}$.

Então, tem-se:

$$X(m) = \sum_{n=0}^{(N/2)-1} x[2n] W_{\frac{N}{2}}^{nm} + W_N^m \sum_{n=0}^{(N/2)-1} x[2n+1] W_{\frac{N}{2}}^{nm} . \quad (2.6)$$

Considerando que:

$$Y(m) = \sum_{n=0}^{(N/2)-1} x[2n] W_{\frac{N}{2}}^{nm} \text{ e } Z(m) = \sum_{n=0}^{(N/2)-1} x[2n+1] W_{\frac{N}{2}}^{nm}, \quad (2.7)$$

nota-se que $Y(m)$ é simplesmente a DFT da seqüência formada pelos itens de índice par da seqüência original, e $Z(m)$ é a DFT da seqüência formada pelos itens de índice ímpar. Então, tem-se:

$$X(m) = Y(m) + W_N^m Z(m)\,. \tag{2.8}$$

Além disso, utilizando-se do fato de que $W_N^{\frac{N}{2}+m} = -W_N^m$ e que $W_{\frac{N}{2}}^{\frac{N}{2}} = 1$, pode-se demonstrar que:

$$X(m + N/2) = Y(m) - W_N^m Z(m)\,. \tag{2.9}$$

A FFT tem uma implementação naturalmente recursiva, dado que cada uma das parcelas da soma é um subcaso dessa soma, e portanto pode ser dividida recursivamente em outras duas parcelas. Esse algoritmo é mais facilmente implementado para calcular a FFT de seqüências com números de itens sendo potência de 2, embora a fórmula original sirva para calcular a FFT de seqüências com qualquer número par de itens.

Um algoritmo recursivo (em pseudo-código) para cálculo da FFT é apresentado a seguir. Esse algoritmo calcula a FFT de um vetor (x):

```
1.  FFT_Recursiva(x)                          # x é um vetor de números complexos
2.  N = comprimento(x)                        # N é uma potência de 2
3.  Se N == 1 então, retorne x                # caso base
4.  wn = e^(-j2π/N)
5.  w =1
6.  x_pares = (x[0], x[2], ..., x[N-2])       # itens de x com índices pares
7.  x_impares = (x[1], x[3], ..., x[N-1])     # itens de x com índices ímpares
8.  y = FFT_Recursiva(x_pares)                # chamada recursiva
9.  z = FFT_Recursiva(x_impares)              # chamada recursiva
10. Para n = 0 até N/2 – 1 faça               # laço principal
11.         t = wz[n]
12.         fft[n] = y[n] + t
13.         fft[n+N/2] = y[n] – t
14.         w = w * wn
15. retorne fft
```

A operação realizada no laço ***Para*** (linhas 10 a 15) é chamada de operação de borboleta (*Butterfly*) [13][16], e é mostrada esquematicamente na Figura 5.

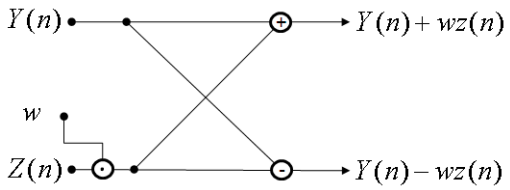

Figura 5. Esquema da Operação de Borboleta.

A recursividade, apesar de ser uma ferramenta poderosa, quando implementada em máquinas reais, apresenta alto custo computacional. Isto ocorre principalmente em termos de memória, dado que para cada chamada recursiva, todo o contexto de execução antes da chamada deve ser guardado, para utilização depois do retorno. Como é comum o cálculo de FFT com seqüências (vetores) de muitos itens, é interessante que seja implementada uma versão iterativa desse algoritmo.

Na Figura 6 são exibidos, em uma estrutura de árvore, os vetores de entrada para as chamadas recursivas da execução do algoritmo FFT para um vetor de 8 entradas. Se for possível organizar o vetor de entrada inicial na ordem em que os itens aparecem nas folhas, será possível implementar uma versão iterativa da função, simulando o comportamento das chamadas recursivas.

Assim, pode-se iniciar fazendo operações de borboleta utilizando os itens do vetor organizado aos pares, e substituir os pares por suas respectivas DFTs. Em seguida, tem-se *N/2* pares de DFTs, e pode-se portanto calcular as DFTs dos *N/4* vetores que os geraram, através também de operações de borboleta. Isto pode ser realizado sucessivamente, até que se tenha 2 vetores de *N/2* DFTs, os quais serão utilizados para calcular os valores da FFT do vetor original.

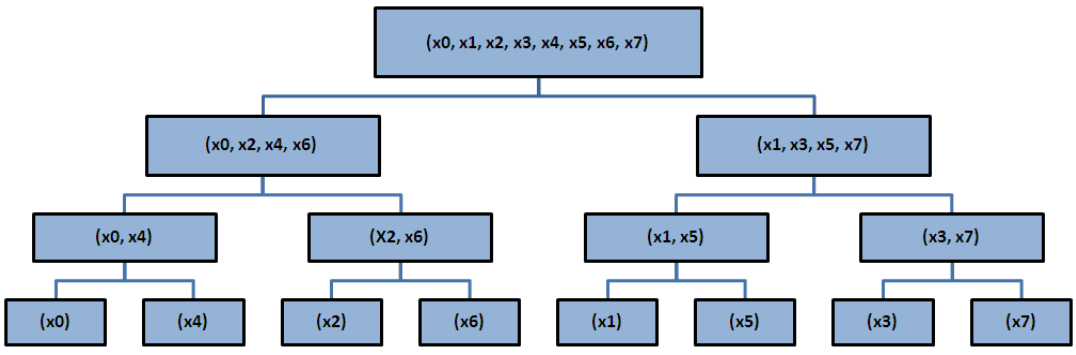

Figura 6. Chamadas recursivas do algoritmo FFT para um vetor de 8 entradas.

Pode-se organizar o vetor de entrada da forma necessária (conforme as folhas da árvore da Figura 6) realizando uma operação chamada permutação com inversão de bits [13][16] nos índices do vetor. Essa operação é simplesmente uma inversão dos bits do número binário representativo do índice do item. Dessa forma, o valor do item de índice 3 (011 em binário), vai para o índice 6 (110 em binário) do vetor, e o valor do item de índice 4 (100 em binário) vai para o índice 1 (001 em binário), e assim por diante. Essa operação é facilmente implementada nas linguagens de programação com operações sobre bits.

Finalmente, é apresentado abaixo o pseudo-código de um algoritmo iterativo que calcula a FFT do vetor de entrada:

```
1.  FFT_Iterativa(x)                          # x é um vetor de números complexos
2.  N = comprimento(x)                        # N é uma potência de 2
3.  Para k = 0 até N – 1 faça                 # permutação com inversão de bits
4.          fft[bit_reversal(k)] = x[k]
5.  Para s = 1 até log2 N faça                # laço principal
6.          m = 2^s
7.          wm = e^(-j2π/N)
8.          Para k = 0 até N – 1 passo m faça
9.                  w = 1
10.                 Para n = 0 até m/2 – 1 faça
11.                         t = w*fft[k + n + m/2]
12.                         u = fft[k + n]
13.                         fft[k + n] = u + t
14.                         fft[k + n + m/2] = u – t
15.                         w = w * wm
16. retorne fft
```

Apesar de a DFT (e a FFT) terem como entrada seqüências de números complexos, elas servem perfeitamente para atuar sobre seqüências de valores reais, bastando somente zerar a parte imaginária de todos os valores da seqüência, e calcular normalmente a DFT (ou FFT).

2.3 Notação Musical (Partituras)

Apesar de existirem muitas outras formas de notação musical, a forma mais utilizada atualmente é conhecida como notação em partituras, e utiliza-se de uma série de símbolos (figuras) que são escritos sobre uma pauta de 5 linhas chamada de

pentagrama. A partitura permite ao instrumentista executar com detalhes a música (ou peça musical) idealizada pelo compositor ou arranjador.

Nesta notação, cada nota é representada por uma figura desenhada sobre o pentagrama (em cima de uma linha ou entre duas linhas), onde a forma da nota define a duração da mesma (por quanto tempo a nota deve ficar soando), e o lugar no pentagrama onde a figura foi desenhada define a altura da nota tocada (altura do som, ou sua freqüência).

A Figura 7 apresenta uma relação das diferentes figuras utilizadas para determinar os tempos das notas (acima) e suas respectivas pausas (abaixo). Quando se caminha para o lado direito da figura as notas têm a metade da duração da figura anterior. Assim a segunda figura, que tem o nome de mínima, tem a metade da duração da primeira, que tem o nome de semibreve. As pausas têm a mesma duração de suas figuras correspondentes, mas determina um tempo de silêncio. Os números abaixo das figuras representam a fração a que cada uma corresponde em relação à semibreve.

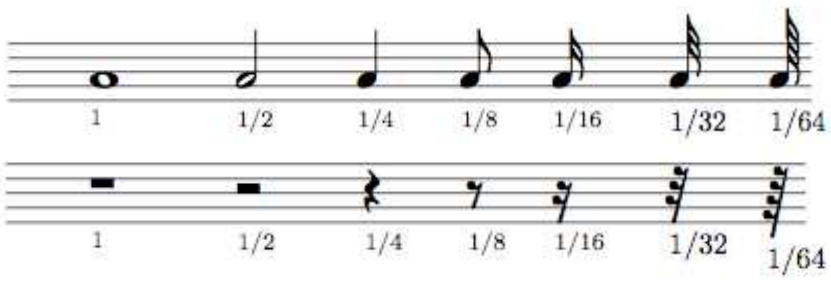

Figura 7. Figuras musicais (acima) e suas pausas (abaixo).

Para se determinar a altura de uma nota em uma partitura, duas coisas devem ser consideradas: a clave, que indica as alturas que correspondem a cada posição (linha ou espaço entre linhas) no pentagrama, e onde a figura foi desenhada no mesmo. A Figura 8 mostra algumas notas nas claves de Sol e de Fá, com suas respectivas alturas. Pode-se ver pela figura que, dependendo da clave utilizada, as notas são diferentes para uma mesma posição no pentagrama.

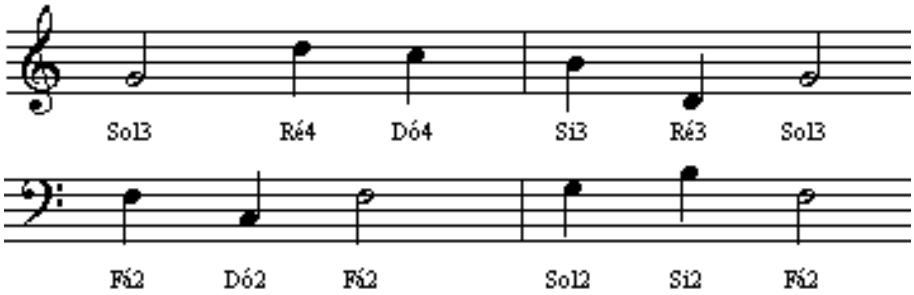

Figura 8. Exemplos de notas nas claves de Sol (acima) e de Fá (abaixo).

Existem alterações que as notas podem sofrer de um semitom (menor intervalo entre duas notas adotado na música ocidental [1]) acima ou abaixo, e essas alterações podem ser indicadas na partitura imediatamente antes da nota. As principais dessas alterações são o Sustenido, cujo símbolo é ♯, e o Bemol, que tem como símbolo ♭. Existe também uma figura que anula uma alteração indicada anteriormente. Essa figura tem o nome de Bequadro, e tem por símbolo ♮.

Normalmente as peças são divididas em compassos, e estes são divididos em tempos. O compasso define a estrutura rítmica da música, e sua fórmula é indicada logo no início da partitura. A fórmula de compasso é indicada por dois números, um numerador (acima) e um denominador (abaixo), onde o numerador indica quantos tempos terá cada compasso, e o denominador indica qual a figura que representará um tempo no compasso.

Na Figura 9 são exibidas algumas fórmulas de compasso diferentes. Na primeira, por exemplo (em cima à esquerda), a fórmula indica que cada compasso terá 4 tempos, e que a figura que corresponderá a um tempo é a semínima (1/4). Dessa forma, cada compasso será preenchido por 4 semínimas, ou qualquer combinação de figuras cuja soma das durações dê igual à duração de 4 semínimas (2 mínimas, por exemplo). Entre os compassos da peça existe uma barra, chamada barra de compasso, que é usada para indicar a separação dos compassos da peça.

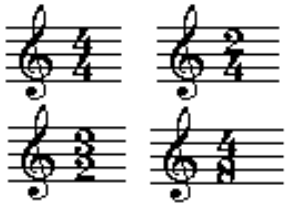

Figura 9. Exemplos de fórmulas de compasso.

Essa notação em partituras é muita rica, e conta com uma série de recursos além dos já descritos, mas esses recursos não serão apresentados nesse documento, pois foge ao escopo do mesmo.

2.4 Freqüências das Notas Musicais

Como se pode ver em um gráfico, uma nota pode ser vista como uma forma de onda, que é formada enquanto a nota está soando. Nessa forma de onda existem alguns componentes de freqüência, que podem ser isolados e verificados através do cálculo da FFT sobre o sinal. Nesses componentes de freqüência existe uma que se destaca, que é chamada de freqüência fundamental da nota (componente de menor freqüência). O que diferencia uma nota de outra é justamente o valor da sua componente de freqüência fundamental. O Lá3, por exemplo, tem a freqüência da fundamental de 440 Hz. Assim, quando um instrumentista toca um Lá3 em seu instrumento, algumas componentes de freqüência podem estar soando ao mesmo tempo naquele som, mas a nota é identificada pela freqüência fundamental.

Na escala musical temperada a diferença de freqüência mínima entre duas notas é o semitom, e essa escala é formada dividindo-se uniformemente todos os intervalos dentro de uma oitava (diferença entre duas notas de mesmo nome seguidas, Dó3 e Dó4, por exemplo). Como as notas são divididas em intervalos regulares, as freqüências das notas crescem (ou diminuem) de acordo com uma progressão geométrica, cuja razão (q) ainda será apresentada.

Para uma mesma nota, a cada vez que se sobe uma oitava, a sua freqüência dobra. Então, como existem 12 notas no total em uma escala (Dó, Dó#, Ré, Ré#, Mi,

Fá, Fá#, Sol, Sol#, Lá, Lá#, Si), o valor da razão *q* é tal que $q^{12} = 2$. Assim, $q = 2^{\frac{1}{12}} \cong 1{,}0594631$.

De posse do valor dessa razão, basta se saber a freqüência de uma nota musical qualquer, e pode-se descobrir a freqüência de qualquer outra nota. Por exemplo, a nota Si3 tem a freqüência de $440 * 1{,}0594631^2 \cong 493{,}88 Hz$, já que essa nota é separada do Lá3 por 2 semitons, e a freqüência do Lá3 é de 440 Hz.

2.5 MusicXML

Atualmente existe uma tendência que está sendo seguida por desenvolvedores de aplicativos dos mais diversos tipos, como editores de texto, planilhas e muitos outros, que é a de usar formatos de arquivos baseados em XML (*Extensible Markup Language*) [19] para os arquivos utilizados em suas aplicações. Geralmente esses tipos de arquivos são definidos em padrões abertos (disponíveis livremente para outros desenvolvedores).

O formato MusicXML, desenvolvido pela empresa Recordare [20], representa justamente mais um esforço nesse sentido, e nesse caso, o de definir um formato aberto de representação em XML para partituras, que poderá ser utilizado em editores. Muitos desses programas (editores de partituras) já oferecem suporte para arquivos do tipo MusicXML. O *layout* do arquivo MusicXML está disponível na Internet, e pode ser usado livremente por qualquer desenvolvedor em suas aplicações.

Na Figura 10 é exibida (circulada em vermelho) uma partitura contendo apenas uma nota, que é um Dó 3 (Dó Central), na figura de uma semibreve, na clave de Sol, em um compasso 4/4, e sua representação em formato MusicXML (texto à esquerda). Por essa figura pode-se ver algumas das características desse formato.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <!DOCTYPE score-partwise PUBLIC
     "-//Recordare//DTD MusicXML 2.0 Partwise//EN"
     "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="2.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

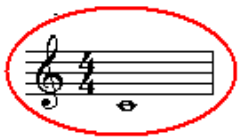

Figura 10. Dó central em uma clave de Sol, em compasso 4/4, e sua representação em MusicXML.

O formato MusicXML geralmente tem como elemento principal (ou raiz) o score-partwise, e todos os elementos representativos da partitura se encontram hierarquicamente abaixo desse elemento. Pela figura pode-se ver, por exemplo, como se faz uma definição de compasso, que é com a *tag* measure, e todos os elementos que caracterizam o compasso se encontram hierarquicamente abaixo dessa *tag*, como por exemplo os elementos beats e beat-type, que definem a quantidade de tempos do compasso e a figura que representa um tempo, respectivamente (fórmula de compasso). Vê-se também que a clave é definida através da *tag* clef. A nota, definida pela *tag* note, também apresenta os elementos

que descrevem a nota por completo, como a nota em si (*tag* pitch), e o tipo da figura da nota (*tag* type), que no caso da figura é uma semibreve (*whole note*, em inglês).

Capítulo 3

Métodos Utilizados

Neste capítulo são apresentadas as formas de abordagem utilizadas para implementar as principais funcionalidades da ferramenta desenvolvida. Nele, pode-se ter uma visão mais detalhada de como foi desenvolvido o sistema em sua estrutura interna.

Os assuntos tratados vão desde a obtenção do sinal da interface de som do computador, passando por todo o tratamento feito no sinal recebido, até a apresentação dos resultados ao usuário. Isto além de mostrar como algumas entidades necessárias ao processamento realizado estão representadas no sistema.

Primeiramente, é importante ressaltar que a linguagem escolhida para o desenvolvimento do sistema foi a linguagem Java [21], por oferecer uma série de facilidades, como uma API (*Application Programming Interface – Interface* de Programação de Aplicativos) nativa com suporte à captura de áudio da interface do computador, facilidade de implementação da interface gráfica do sistema, API's disponíveis com suporte à importação de arquivos MusicXML, etc.

3.1 Tratamento do Sinal de Som

Nessa seção é descrito todo o tratamento que o sinal de som recebe internamente na ferramenta.

3.1.1 Obtenção do Sinal

A linguagem Java oferece, através da sua API de som (Java *Sound* API) [22][23], suporte à captura de áudio da interface de som do computador, disponibilizando uma série de classes para que se realize as operações necessárias a essa captura.

Primeiramente, deve-se escolher o formato de som que se deseja utilizar, e isso se dá através da criação de um objeto do tipo AudioFormat. Para o construtor desse objeto devem ser passados os parâmetros indicando o formato desejado, como a taxa de amostragem, a taxa de quantização e o número de canais (1 para sinal mono, 2 para sinal estéreo, por exemplo).

O formato escolhido para utilização no sistema tem taxa de amostragem de 44100 Hz, taxa de quantização de 8 bits, e 1 canal (mono). O valor de 44100 Hz foi escolhido para a taxa de amostragem porque esse valor permite calcular a FFT para períodos curtos da peça, possibilitando assim um cálculo mais preciso das durações das notas, principalmente das notas de menor duração (notas mais rápidas). A taxa de quantização com valor de 8 bits se mostrou suficiente para a aplicação.

É preciso também criar um objeto do tipo DataLine.Info, que oferece algumas informações adicionais necessárias para a abertura da linha de dados que será utilizada para a leitura do áudio.

A leitura do áudio é feita a partir de um objeto do tipo TargetDataLine, que pode ser instanciado chamando-se o método estático AudioSystem.getLine, que recebe como parâmetro o objeto DataLine.Info criado anteriormente. Esse objeto (TargetDataLine) permite que se faça a leitura da interface de áudio do computador. A partir das chamadas dos métodos open (que recebe um objeto AudioFormat como parâmetro), que abre a linha de dados para leitura, e em seguida do método start (que não tem parâmetros), a linha fica disponível para leitura, e pode ser lida através do método read do objeto TargetDataLine, e processada de acordo com a necessidade de cada aplicação. Um dos parâmetros que o método read recebe é um array de bytes, onde serão armazenados os dados (amostras) lidos da entrada de som.

3.1.2 Cálculo da FFT

Foi implementado um método no sistema para realizar o cálculo da FFT sobre o array lido da entrada de som do computador. Como a FFT atua sobre seqüências de números complexos, o método recebe 2 arrays como parâmetro, um contendo a

parte real de cada um dos números da seqüência, e outro contendo suas partes imaginárias.

Para implementação da transformada, são feitas inicialmente permutações com inversão de bits, para ordenamento dos elementos dos arrays de entrada, e em seguida as operações de borboleta para o cálculo efetivo da FFT, conforme descrito no Capítulo 2.

Para fins de otimização, todos os cálculos são feitos nos próprios arrays de entrada, onde os resultados ficam armazenados. Assim, após o retorno do método que calcula a FFT, os arrays recebidos como entrada contêm as partes real e imaginária da DFT de cada um dos valores da seqüência original. Esse resultado é o espectro de freqüência do sinal, e a partir dele é possível verificar quais são as componentes de freqüência existentes.

A amplitude da freqüência correspondente a cada item do espectro retornado pelo cálculo da FFT é dada pela norma do número complexo associado àquele item, e assim pode ser calculado por $\sqrt{r^2 + i^2}$, onde *r* representa a parte real do número e *i* a parte imaginária.

Após alguns testes com diferentes tamanhos dos arrays de entrada para o cálculo da FFT, o tamanho que apresentou melhores resultados para a taxa de amostragem utilizada foi de 2048 itens.

3.1.3 O Espectro de Freqüência e a Correspondência com as Notas Musicais

Como a taxa de amostragem utilizada na aplicação é de 44100 Hz e o tamanho do array é de 2048 itens, tem-se uma freqüência fundamental representada no espectro de freqüências (encontrada no elemento de índice 1 do array) com valor de $44100/2048 = 21{,}533Hz$, e portanto, essa é a freqüência mais baixa representada no espectro, e também o incremento para cada item avançado no mesmo. Por exemplo, a amplitude calculada para o décimo item do espectro corresponderá à freqüência de $10 * 21{,}533 = 215{,}33Hz$.

A correspondência das notas com os itens do espectro de freqüências é dada pela aproximação dos valores das freqüências das notas com as freqüências correspondentes aos itens. Assim, a nota que mais se aproximar (em valor de freqüência) da freqüência representada no item do espectro, é considerada como a nota correspondente àquele item.

3.1.4 Identificando as Notas

Como a freqüência fundamental contida no espectro tem o valor de 21,533 Hz, e esse é também o valor do incremento para cada item do mesmo, pode-se perceber que nem todas as notas musicais estão representadas no espectro. Por exemplo, a nota que mais aproxima em freqüência da fundamental do espectro é uma nota Fá, com freqüência de 21,827 Hz. A nota Fá# que sucede essa nota, tem a freqüência de 23,125 Hz, mas o segundo item do espectro tem a freqüência de 2 * 21,533 = 43,066 Hz, e portanto bem distante da desejada para ser possível identificar o Fá#. Na verdade, a nota que mais se aproxima em freqüência do segundo item do espectro é uma nota Fá uma oitava acima da anterior, com freqüência de 43,654 Hz. Com isso, todas as notas pertencentes a essa oitava inteira podem ser erroneamente interpretadas como uma dessas duas notas Fá.

Esse fato impossibilita (ou dificulta) a identificação de todas as notas musicais a partir do espectro obtido. Por isso é preciso verificar em que trechos do espectro é seguro fazer uma busca comparando-se as freqüências das notas musicais com as freqüências representadas no espectro.

Como as freqüências representadas no espectro crescem de acordo com uma PA (que tem como razão a fundamental), mas as freqüências das notas musicais crescem de acordo com uma PG (de razão 1,0594631, conforme demonstrado no Capítulo 2), existe algum momento a partir do qual as freqüências das notas começarão a crescer mais rapidamente do que as freqüências no espectro, e a partir de então todas as notas musicais estarão representadas univocamente no espectro. Encontrando-se a nota a partir da qual isso acontece, pode-se identificar individualmente qualquer nota musical, sendo que as notas anteriores a essa não podem ser identificadas com exatidão no espectro.

Por conta do que foi exposto, foi implementado no sistema uma procura pela oitava a partir da qual todas as notas têm uma correspondência no espectro, e as notas anteriores a essa oitava são ignoradas, por não poderem ser identificadas com segurança.

Para a identificação das notas que estão sendo tocadas, o sistema faz uma busca no espectro de freqüência pelos picos de amplitudes, e o primeiro pico encontrado é considerado como a freqüência da nota tocada. Mas como é possível que haja alguns picos decorrentes de ruídos, é possível que a nota encontrada (primeiro pico) não seja a que de fato está sendo tocada. Para evitar isso, é calculada a média de todas as amplitudes do espectro, e um valor só é considerado como pico se ele for maior que essa média adicionada a algum fator, e também maior que essa média multiplicada por algum fator. Os valores desses fatores foram ajustados após experiências com diferentes valores, e o fator que é somado à média para o reconhecimento de um pico de freqüência tem o valor 5, e o fator que é multiplicado pela média tem o valor 30. Com isso, fica mais segura a operação de isolamento dos picos, e conseqüentemente a indicação das notas.

Contudo, com o decaimento natural dos volumes das notas tocadas (mesmo quando elas ainda estão soando), além da instabilidade do sinal no momento do ataque, é possível que, enquanto uma nota ainda esteja soando, a sua freqüência fundamental não seja identificada como pico de amplitude no espectro. Isto pode causar um erro na identificação da freqüência fundamental daquele conjunto de amostras.

Tendo isso em vista, o algoritmo desenvolvido para identificação da nota tocada não determina a nota para cada conjunto de amostras (com 2048 amostras), mas sim para alguns desses conjuntos. Mais precisamente, quando um ataque é identificado (a forma como um ataque é identificado ainda será descrita nesse documento), o sistema começa a fazer uma contagem da quantidade de vezes que cada nota foi identificada em um conjunto de amostras, e assim que o próximo ataque é identificado, os valores desses contadores são comparados, e a nota que foi identificada um maior número de vezes é então indicada como a nota que foi tocada. Dessa forma, o sistema se mostrou mais preciso na indicação das notas.

Um ponto que deve ser observado para a identificação das notas no sistema é que a relação sinal-ruído do sinal recebido da interface de som deve ser suficientemente grande para que as notas possam ser identificadas. Caso essa relação seja muito baixa, o sistema não será capaz de identificar as notas tocadas.

3.1.5 Visualizando o Sinal no Tempo e o Espectro de Freqüência

Como conveniência para visualização dos dados internos do sistema, foi implementada uma tela para visualização do sinal no tempo e o seu espectro de freqüência correspondente.

Para isso, a cada grupo de amostras lido (2048 amostras de cada vez), os dados recebidos da interface de som, ou seja, o sinal no tempo, são armazenados em um array, e esse array é copiado para um outro, onde será feito o cálculo da FFT. Depois que esse cálculo é efetuado, tem-se todos os dados do sinal no tempo e seu espectro de freqüência, e pode-se então desenhar esses sinais.

Para efetuar o desenho de cada um dos sinais, foi utilizado um objeto do tipo Canvas. Esse é um objeto visual, e oferece alguns recursos básicos de desenho, como desenho de pontos, retas, retângulos, elipses, polígonos, Strings, etc.

O sinal no tempo é desenhado iterando-se sobre o seu array e ligando-se os pontos correspondentes no Canvas. O valor de cada item do array determina a altura da linha que representa o sinal no ponto correspondente. A Figura 11 mostra um exemplo de sinal no tempo desenhado em um Canvas.

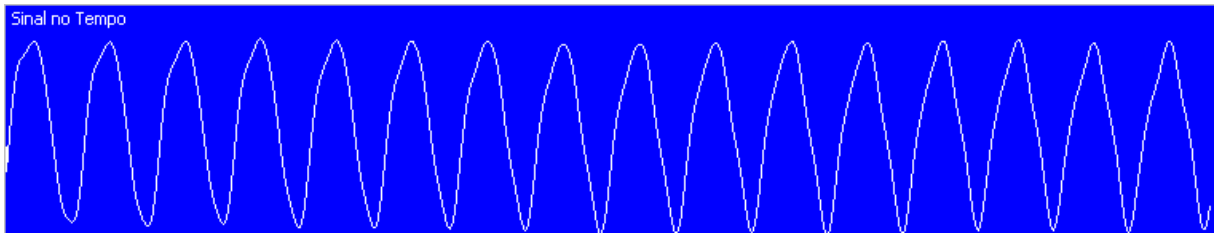

Figura 11. Exemplo de desenho de um sinal no tempo em objeto Canvas.

O desenho do espectro de freqüência é feito iterando-se pelo array resultante da FFT, e a altura desenhada para cada item do array é proporcional à norma do

número complexo referente ao item. A Figura 12 mostra como exemplo o desenho em um Canvas do espectro de freqüência do sinal no tempo mostrado na Figura 11.

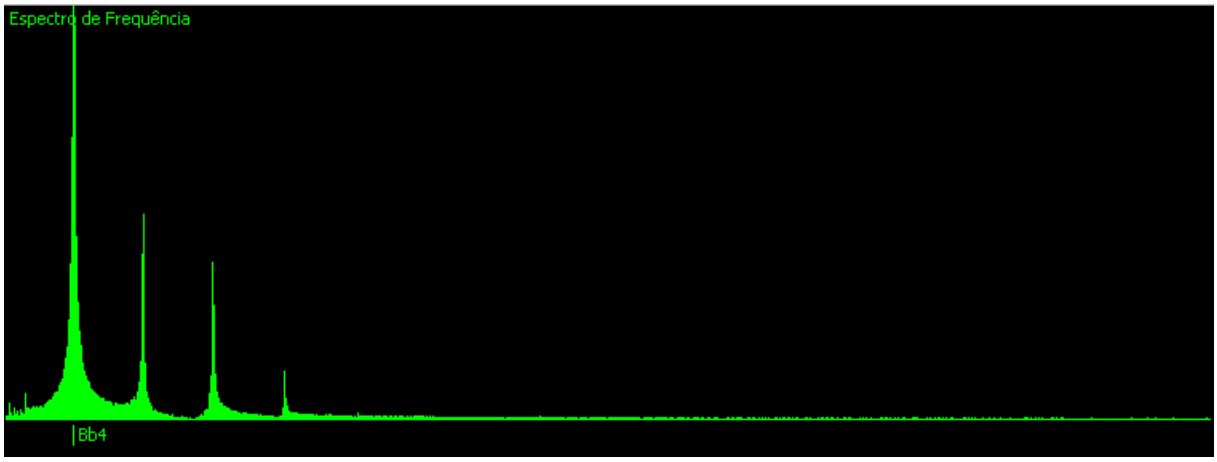

Figura 12. Exemplo de desenho do espectro de freqüência em um objeto Canvas.

Os desenhos desses sinais são feitos algumas vezes a cada segundo, pois existe uma Thread (classe da API de Java para execução de processos em paralelo) que lê os dados da entrada de áudio, chama o método para cálculo da FFT e atualiza a tela, desenhando o sinal no tempo e o seu espectro de freqüência nos Canvas correspondentes.

Caso a relação sinal-ruído seja insuficiente para o reconhecimento de uma nota no conjunto de amostras atual, será mostrada a mensagem "*Baixa SNR*" (*Signal-to-Noise Ratio* – relação sinal-ruído) no Canvas de visualização do espectro de freqüência.

3.1.6 Encontrando as Durações das Notas Tocadas

Para que se possa comparar as notas tocadas pelo usuário com as notas de uma partitura importada pelo sistema, é preciso que se calcule a duração de cada nota que foi tocada, e isso é feito utilizando-se o sinal no tempo.

Monitorando o sinal da entrada de áudio, pode-se encontrar os momentos em que as notas começam a soar (momento de ataque de cada nota). Na Figura 13 pode-se ver um desses momentos. Analisando a figura, vê-se que os valores máximos de amplitude do sinal da onda crescem rapidamente no momento do ataque, e com isso, pode-se identificar aproximadamente o instante em que este

ocorreu. Para isso, o sistema monitora o sinal, armazenando em um array os valores de amplitude das últimas cristas (amplitudes máximas das ondas) encontradas, e quando é encontrada uma onda com valor máximo maior do que todas as antecedentes contidas no array somados a algum fator (para evitar a identificação de uma instabilidade no sinal como um ataque), é indicado que houve um ataque. Esse fator tem valor 5 no sistema, e foi ajustado após experiências com diferentes valores. As experiências mostraram que para valores muito acima do utilizado, o sistema apresentava maior dificuldade para reconhecer os ataques das notas, e para valores muito abaixo, algumas vezes o sistema reconhecia o mesmo ataque mais de uma vez.

A duração de uma nota é calculada simplesmente pelo tempo decorrido desde a identificação do ataque da nota até a identificação do ataque da nota seguinte. As pausas são desconsideradas por 2 motivos: o primeiro é que é comum que os instrumentistas, principalmente os iniciantes, dêem pouca atenção às pausas, muitas vezes deixando a nota soar no momento em que deveria existir uma pausa, ou abafando a nota antes do término do tempo correspondente à sua figura. E o segundo motivo é que, com o decaimento natural das notas tocadas, torna-se difícil identificar o momento exato do início da pausa, assim como é identificado o momento de ataque.

Figura 13. Sinal no tempo no momento do ataque de uma nota.

Como foi verificado que o sinal apresenta, no momento do ataque, uma instabilidade (como pode-se ver pela Figura 13), e leva um tempo até que a onda se apresente limpa, o sistema conta um certo tempo (*timeout*) antes de permitir a identificação de um novo ataque, isso para evitar a identificação de um ataque sem que tenha realmente ocorrido.

3.2 Representação das Notas no Sistema

As notas no sistema estão representadas em uma classe (Nota), e nessa classe estão presentes todos os atributos necessários às operações realizadas internamente na ferramenta. Estes atributos são: a nota musical (Dó3, por exemplo), a figura da nota (se é uma semibreve ou outra figura), o número do compasso e a ordem da nota no compasso (para que seja indicada a localização quando um erro for encontrado na nota), além da duração da nota (em milissegundos), para que seja feita uma comparação entre a nota tocada e a existente na partitura.

O compasso está representado por outra classe (Compasso), que tem uma lista de notas, que são as notas contidas naquele compasso, e sua duração (para facilitar o cálculo da duração da peça como um todo).

A peça também está representada por uma classe (Peca), que tem os atributos relativos à peça como um todo, contendo uma lista de compassos, o andamento da peça (em pulsos por minuto), e a duração total da peça, para que o analisador (Thread que fica lendo a entrada de som) saiba quando deve parar de executar. A cada vez que o andamento á alterado, os valores das durações das notas é calculado, e conseqüentemente, as durações dos compassos e a duração da peça inteira.

3.3 Importação de Arquivos MusicXML

Para a importação dos arquivos *MusicXML*, foi utilizada uma biblioteca para programação musical em Java, chamada *JFugue* [24]. Essa biblioteca oferece suporte à importação de arquivos nesse formato, convertendo-os para um formato interno à biblioteca chamado *MusicString*, que é uma *String* formatada que a biblioteca utiliza para poder tocar a partitura importada. Entretanto, para a aplicação desenvolvida, é interessante que se possa importar o arquivo *MusicXML* para o formato interno da aplicação, ou seja, para objetos das classes Peca, Compasso e Nota.

A importação do arquivo *MusicXML* dentro da biblioteca *JFugue* acontece de uma forma que o *parser* do arquivo, a cada entidade encontrada (uma nota ou um novo compasso, por exemplo) no arquivo da partitura, dispara um evento. Pode-se então implementar um *Listener* (padrão de projeto muito usado em Java em que se implementa um objeto que realiza alguma operação quando “ouve” o disparo de algum evento esperado) para esses eventos, e assim construir um objeto da forma desejada a partir dos eventos disparados pelo parser do arquivo. A importação dos arquivos *MusisXML* para o formato interno da aplicação foi implementada desta forma.

3.4 Comparação das Notas da Peça Importada com as Notas Tocadas

Após a execução da peça pelo usuário do sistema, faz-se necessária uma comparação entre as notas da peça que foi importada e as notas que foram identificadas da entrada de áudio do computador.

O formato da peça importada é um objeto do tipo Peca, contendo todos os dados necessários ao sistema da partitura importada. Já as notas recebidas da entrada de áudio são dispostas em um array ordenado pela ordem em que as notas foram identificadas. Em ambos os casos, todas as notas tem sua respectiva duração em milissegundos. Essa duração é usada para fazer uma comparação entre as notas, que são portanto comparadas em sua freqüência (para verificar se a nota correta foi tocada) e em sua duração (para saber se houve algum erro de ritmo na execução).

O objeto Peca tem um método que retorna um array contendo todas as notas ordenadas por ordem de execução, formatado da mesma forma como o objeto contendo as notas que foram identificadas da interface de áudio.

As notas dos 2 arrays são então comparadas na seqüência, e cada inconsistência encontrada indica um erro na execução. Os erros podem ser de ritmo, quando uma nota que deveria ser tocada em um momento não é tocada, ou de nota,

quando a nota identificada é diferente da nota da peça importada, apesar de estar no tempo certo.

Como as notas provenientes da entrada de som não tem sua duração exata, e também por ser comum, mesmo para uma ótima execução, algumas pequenas diferenças nas durações das notas tocadas pelo instrumentista para as indicadas na partitura, existe uma flexibilidade de 20% na comparação da duração de cada nota. Ou seja, se a nota tocada tiver uma duração até 20% maior ou menor que o esperado, isso não é considerado um erro.

3.5 Classificação da Execução

Após a comparação das notas, é feita uma classificação da execução. Essa classificação é feita calculando-se a percentagem das notas que foram tocadas corretamente, e então é oferecido um resultado de acordo com a percentagem calculada. As classificações dadas para as percentagens de acertos (p) são:

- $p = 100\%$ - Execução perfeita
- $95\% <= p < 100\%$ - Execução ótima
- $85\% <= p < 95\%$ - Execução boa
- $70\% <= p < 85\%$ - Execução média
- $p < 70\%$ - Execução ruim

3.6 Apresentação dos Resultados

Depois de efetuado todo o processamento, os resultados são apresentados ao usuário. Esses resultados são exibidos em uma tela informando o número total de notas da peça executada, o número de erros, a classificação recebida e a lista dos erros, indicando a localização de cada erro na peça e o seu tipo. Para erros de ritmo, o sistema mostra a mensagem: “Foi encontrado um Erro de Ritmo”, e para erros de nota, o sistema diz que a nota encontrada foi diferente da esperada.

Capítulo 4

A Ferramenta Desenvolvida

Neste capítulo é apresentada uma visão geral da ferramenta que foi implementada no projeto, mostrando suas funcionalidades e a forma de acessá-las através dos seus menus.

4.1 Visualização do Espectro de Freqüência

A janela de exibição do espectro de freqüência exibe, em sua parte inferior (no Canvas inferior), o desenho do sinal no tempo referente ao conjunto de amostras mais recentemente lido da entrada de som. Na parte superior da janela (no Canvas superior), é exibido o espectro de freqüência relativo ao sinal mostrado no Canvas inferior. A nota que foi identificada para aquele conjunto de amostras é exibida um pouco abaixo do espectro de freqüência.

Na Figura 14 pode-se ver a janela de visualização do espectro. Essa janela é atualizada algumas vezes a cada segundo, conforme cada conjunto de amostras é recebido da entrada de som do computador, e a FFT relativa a cada um dos conjuntos é calculada.

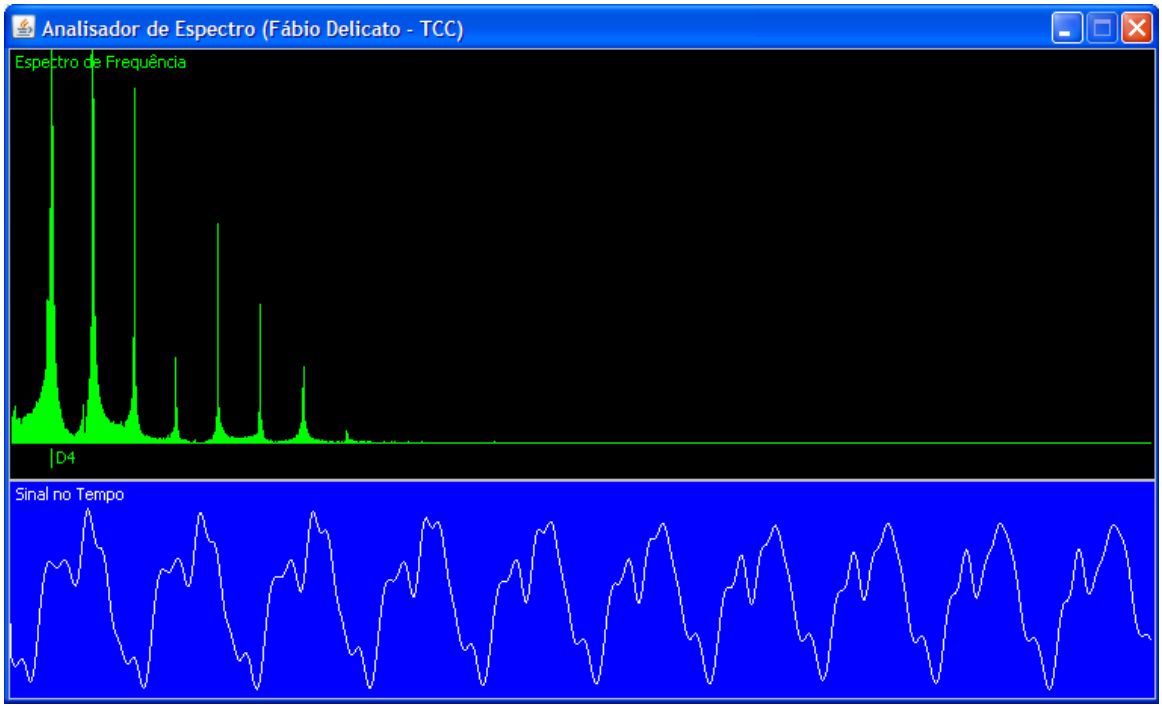

Figura 14. Janela de visualização do espectro de freqüência.

Essa Janela de exibição pode ser acessada por dois caminhos diferentes: um dos caminhos é acessando diretamente o item de menu *Espectro de Freqüência* no menu *Sistema* da ferramenta, e outra é através da análise da execução de uma peça, quando a visualização do espectro de freqüência é ativada automaticamente. Pode-se ver na Figura 15 o menu de acesso direto à visualização do espectro de freqüência. Quando essa janela de exibição é acessada diretamente através do menu, nenhuma análise da execução é feita, e os dados da interface de som do computador são lidos e processados somente para exibição.

Figura 15. Menu para acesso direto à visualização do espectro de freqüência.

4.2 Importando uma Peça (Arquivo MusicXML)

A primeira coisa a ser feita para se realizar a análise da execução de uma peça no sistema, é carregar um arquivo *MusicXML*, pois a análise é feita comparando-se o conteúdo da partitura carregada (arquivo *MusicXML*) com o que é efetivamente executado pelo usuário.

Para efetuar o carregamento de um arquivo *MusicXML*, o usuário deve acessar o item de menu *Carregar Peça*, no menu *Peça*, e em seguida escolher o arquivo a ser carregado. Após a escolha do arquivo, o sistema mostra uma mensagem de sucesso ou erro na importação do arquivo, conforme um ou outro tenha ocorrido. Na Figura 16 pode-se ver o item de menu que dá acesso ao carregamento de arquivo *MusicXML*. Alguns itens do menu *Peça* encontram-se desabilitados antes do carregamento de uma peça, pois se tratam de funcionalidade que dependem de ter uma peça carregada no sistema. Esses itens são habilitados após o carregamento, dando acesso às funcionalidades correspondentes.

Figura 16. Item de menu para o carregamento de arquivo *MusicXML*.

4.3 Configurações do Sistema para a Execução

Existem algumas configurações no sistema que dizem respeito à análise da execução. Essas configurações são: o andamento no qual a peça será executada (em pulsos por minuto), o número de cliques iniciais (cliques que serão tocados antes que o sistema espere o ataque da primeira nota da peça) e se o sistema deve tocar cliques durante a execução (indicando o andamento esperado da peça através de cliques nos pulsos).

Essas configurações são acessadas através do item de menu *Configurações*, no menu *Peça*. Esse item de menu dá acesso a uma tela contendo as configurações do sistema, conforme mostrado na Figura 17.

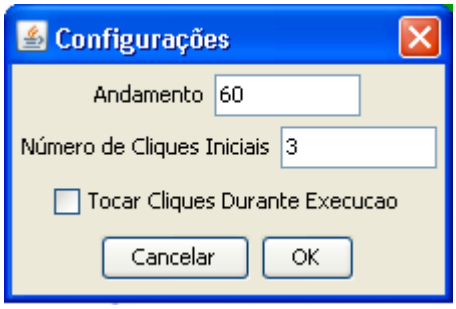

Figura 17. Tela de configurações do sistema para a execução.

4.4 Analisando uma Execução da Peça

Após o arquivo *MusicXML* estar carregado, e o sistema devidamente configurado para a execução, pode-se proceder com a análise da execução da peça carregada. Para isso, o usuário deve acessar o item de menu *Executar Verificação*, no menu *Peça*, conforme mostrado na Figura 18. Com o acionamento desse item de menu, o sistema dá início à verificação da execução.

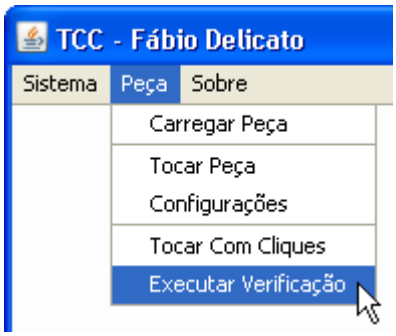

Figura 18. Item de menu que dá acesso à análise da execução.

Inicialmente, a tela do espectro de freqüência é automaticamente exibida, e os cliques iniciais (que acontecem nos pulsos antes do início da análise) são tocados, de acordo com a configuração realizada. Após os cliques iniciais, o sistema inicia a leitura da interface de áudio do computador, e as notas lidas dessa interface são armazenadas, para serem posteriormente comparadas com as da partitura

carregada. A operação pode ser cancelada a qualquer momento fechando-se a tela do espectro de freqüência.

O sistema continua a leitura até que o tempo calculado da peça carregada tenha se esgotado, momento em que ele fecha a tela do espectro de freqüência e inicia a comparação das notas, apresentando os resultados da análise em seguida.

4.5 Obtendo o Resultado

A tela mostrando o resultado da análise da execução é mostrada automaticamente após o término da execução da peça pelo usuário. Essa tela mostra em detalhes os dados da análise realizada pelo sistema.

Para exemplificar, foram escritas duas versões da música Asa Branca, de Luiz Gonzaga e Humberto Teixeira. Na primeira versão, mostrada na Figura 19, a música foi escrita corretamente, sem a adição de nenhum erro. Na segunda versão, exibida na Figura 20, foram adicionados alguns erros de notas (circulados com linha preenchida), e um erro de ritmo (circulado com linha pontilhada). O arquivo *MusicXML* carregado para a análise foi o correspondente à partitura exibida na Figura 19, onde a peça se encontra sem erros, e as duas partituras foram executadas no sistema para realização da análise.

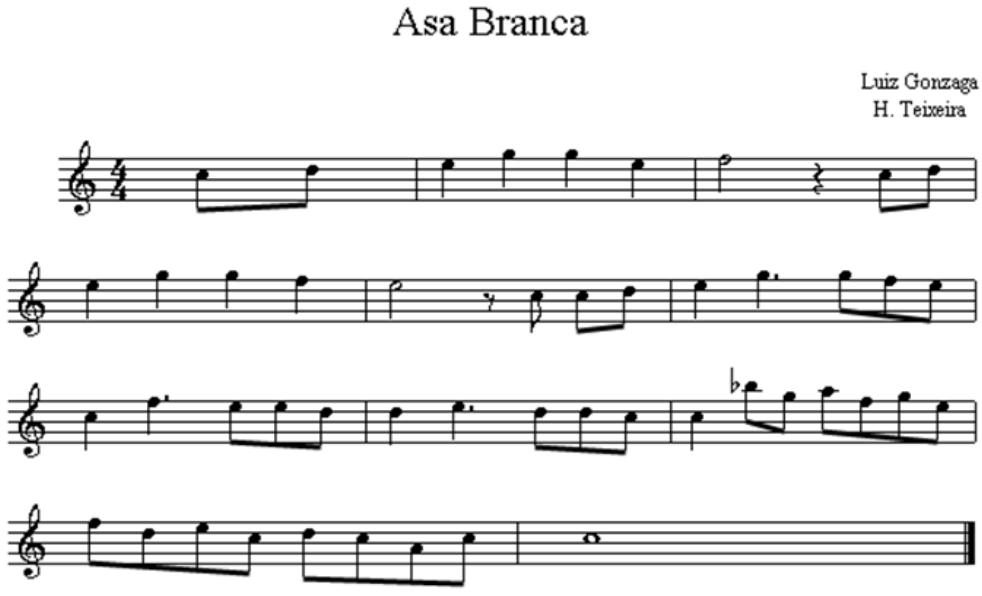

Figura 19. Partitura da música Asa Branca (sem erros).

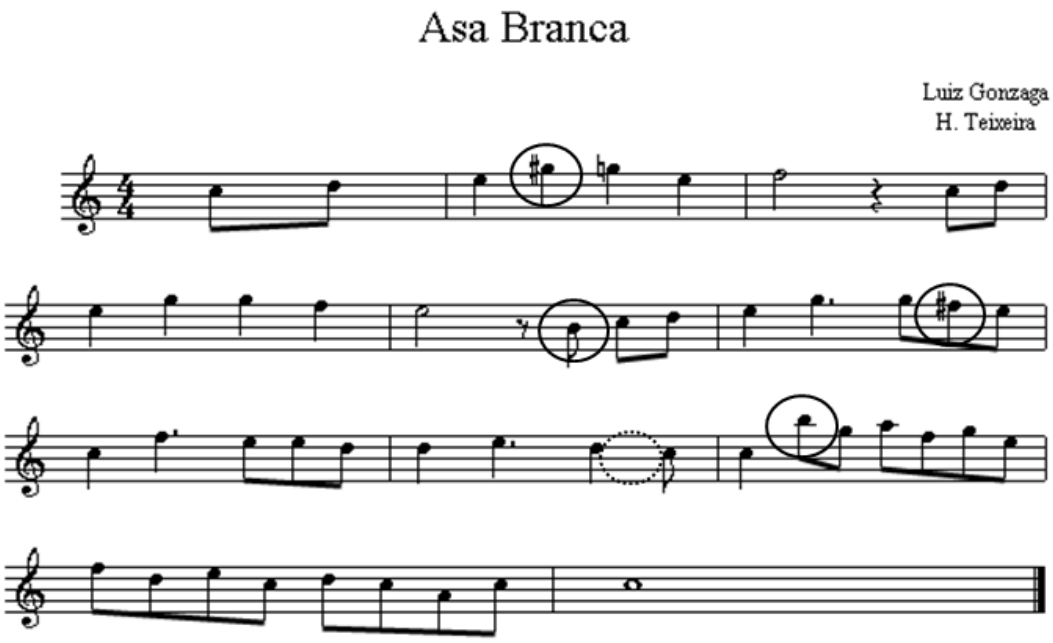

Figura 20. Partitura de Asa Branca, com erros de nota (linha preenchida) e ritmo (linha pontilhada).

A Figura 21 mostra os resultados obtidos da execução das duas versões da música, sendo que o resultado da versão sem erros é exibido à esquerda, e o da versão na qual os erros foram adicionados é exibido à direita. Pode-se ver pela figura que o sistema indicou corretamente os locais dos erros e os seus tipos, assim como não indicou nenhum erro inexistente.

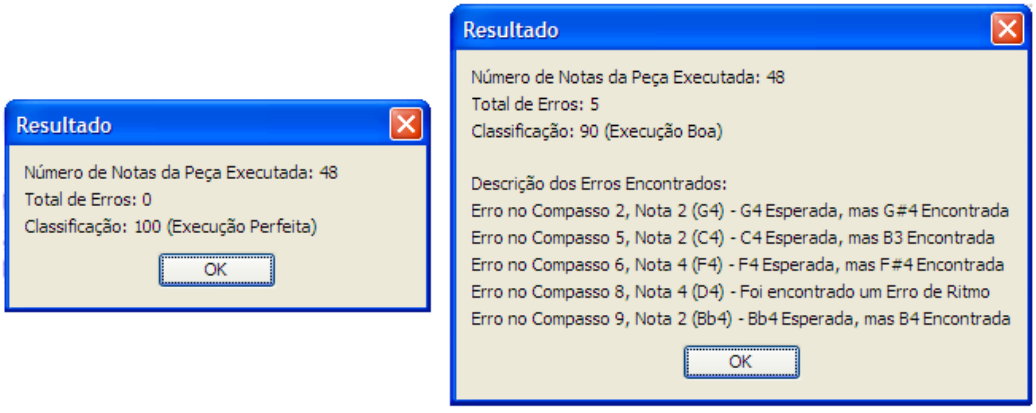

Figura 21. Exemplos de resultados obtidos na ferramenta.

Com isso, pôde-se ter uma visão geral das funcionalidades mais importantes do sistema, principalmente do processo de análise da execução de uma peça, desde o carregamento da partitura até a obtenção dos resultados da execução.

Capítulo 5

Conclusões e Trabalhos Futuros

Neste projeto foi implementada uma versão inicial de uma ferramenta de apoio ao ensino de instrumento musical, e todo o processo de desenvolvimento dessa ferramenta está descrito nesse trabalho, desde o embasamento teórico necessário ao seu desenvolvimento até as decisões tomadas em sua implementação.

O objetivo do projeto foi atendido, visto que a ferramenta apresentou bons resultados para peças simples, no nível real de um iniciante em instrumento musical.

5.1 Contribuições

Apesar de a ferramenta desenvolvida durante o projeto ser uma versão inicial que ainda precisa ser amadurecida, a continuação do seu desenvolvimento pode resultar em um produto de grande utilidade como facilitador de aprendizado para estudantes iniciantes em um instrumento musical.

A utilidade dessa ferramenta se dá porque auxilia o aluno nos primeiros momentos de seus estudos, quando é a hora de maior dificuldade, dada a grande quantidade de novas informações a que é exposto e às adaptações iniciais necessárias para o aprendizado do instrumento.

5.2 Dificuldades Encontradas

Uma das dificuldades encontradas durante o desenvolvimento desse projeto foi decorrente da carência de disciplinas destinadas especificamente ao processamento de sinais e computação musical durante o curso de graduação. Isso fez com que uma grande quantidade de tempo fosse investida em noções básicas sobre os assuntos, visto que não foram cobertos durante o curso.

A falta de material com explicações claras sobre a implementação da FFT também trouxe alguma dificuldade ao desenvolvimento.

Além disso, a diferença dos espectros de freqüência gerados por diferentes instrumentos traz um desafio grande ao desenvolvimento de um algoritmo que funcione bem para todos, ou pelo menos uma grande quantidade de instrumentos diferentes.

5.3 Trabalhos Futuros

Para o amadurecimento da ferramenta, faz-se necessário o estudo mais detalhado dos espectros gerados por diferentes instrumentos, para que o algoritmo de tratamento do sinal seja aperfeiçoado, passando a funcionar satisfatoriamente para diferentes tipos de instrumentos.

Um outro recurso que pode ser adicionado à ferramenta em trabalhos futuros é o suporte a sons polifônicos.

Bibliografia

[1] MED, Bohumil. **Teoria da Música**. 4ªed. Brasília: Musimed, 1996.

[2] POZZOLI, Ettore. **Guia Teórico-Prático para o Ensino de Ditado Musical – 1ª e 2ª Partes**. Ricordi, 1999.

[3] PINTO, Henrique. **Iniciação ao Violão – Volume I**. São Paulo: Ricordi, 1978.

[4] PINTO, Henrique. **Iniciação ao Violão – Volume II**. São Paulo: Ricordi.

[5] PINTO, Henrique. **Curso Progressivo de Violão**. São Paulo: Ricordi, 1982.

[6] _____. **D'Accord Afinador 3.0**. Disponível em: http://www.daccord.com.br/produto.php?idProduto=2. Acesso em: 29 Set. 2009.

[7] _____. **D'Accord Dicionário Violão 3.0**. Disponível em: http://www.daccord.com.br/produto.php?idProduto=1. Acesso em: 29 Set. 2009.

[8] _____. **D'Accord Curso de Violão**. Disponível em: http://www.daccord.com.br/produto.php?idProduto=27. Acesso em: 29 Set. 2009.

[9] MADISETTI, Vijay K.; WILLIAMS, Douglas B. **Digital Signal Processing Handbook**. CRC Press: 1999.

[10] ROCCHESSO, Davide. **Introduction to Sound Processing**. Verona: Università di Verona – Dipartimento di Informatica, 2003.

[11] _____. **MusicXML Definition – Version 2.0**. Disponível em: http://www.recordare.com/xml.html. Acesso em: 30 Set. 2009.

[12] ZONST, Anders E. **Understanding the FFT**. Florida: Citrus Press, 1995.

[13] CORMEN, Thomas H.; et al. **Algoritmos: Teoria e Prática**. 2ª Ed. Rio de Janeiro: Campus, 2002.

[14] COOLEY, J. W.; TUKEY, J. W. **An Algorithm For The Machine Calculation Of Complex Fourier Series**, Mathematics of Computation, Vol. 19, Abril, 1965.

[15] BASTOS NETO, Carmelo J. A. **Notas de Aula da Disciplina de Controle de Processos**. UPE-POLI, Curso de Engenharia da Computação, 1 fev. 2007, 1 jun. 2007. Notas de Aula.

[16] CHU, Eleanor; GEORGE, Alan. **Inside the FFT Black Box**. CRC Press, 2000.

[17] BONJORNO, José R.; RAMOS, Clinton M. **Física 2: Termologia, Óptica Geométrica, Ondulatória**. São Paulo: FTC, 1992.

[18] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Quantidades, unidades e símbolos das grandezas acústicas fundamentais**. Rio de Janeiro, 1959.

[19] _____. **Extensible Markup Language (XML)**. Disponível em: http://www.w3.org/XML. Acesso em: 15 Nov. 2009.

[20] _____. **Recordare**. Disponível em: http://www.recordare.com. Acesso em: 16 Nov. 2009.

[21] _____. **The Source for Java Developers**. Disponível em: http://java.sun.com. Acesso em: 15 Nov. 2009.

[22] _____. **Java Sound API**. Disponível em: http://java.sun.com/products/java-media/sound. Acesso em: 15 Nov. 2009.

[23] _____. **Java Sound Resources**. Disponível em: http://www.jsresources.org. Acesso em: 15 Nov. 2009.

[24] _____. **JFugue – Java API for Music Programming**. Disponível em: http://www.jfugue.org. Acesso em: 28 Out. 2009.