UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

INTERPRES, UM PROTÓTIPO PARA RECONHECIMENTO DE PARTITURA: IDENTIFICANDO A FORMA DOS SÍMBOLOS MUSICAIS

JONATHAN MAURICENZ

BLUMENAU
2013

2013/1-19

JONATHAN MAURICENZ

INTERPRES, UM PROTÓTIPO PARA RECONHECIMENTO DE PARTITURA: IDENTIFICANDO A FORMA DOS SÍMBOLOS MUSICAIS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Aurélio Faustino Hoppe, Mestre – Orientador

BLUMENAU
2013

2013/1-19

INTERPRES, UM PROTÓTIPO PARA RECONHECIMENTO DE PARTITURA: IDENTIFICANDO A FORMA DOS SÍMBOLOS MUSICAIS

Por

JONATHAN MAURICENZ

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
	_____
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB
	_____
Membro:	Prof. Marcel Hugo, Mestre – FURB

Blumenau, 10 de julho de 2013

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

Em especial, a minha família e colegas mais próximos Alan Filipe Cardozo Fabeni por compiladores, Arno Wilson Cassaniga por Fourier, Jackson Krause, Paulo Eduardo Danker por Prolog e Thiago Pradi por IOS e aos demais amigos, pelos incentivos.

Ao SAMAE - Serviço Autônomo Municipal de Água e Esgoto de Timbó, pelo auxílio financeiro.

A FURB e a todos os professores que estiveram presentes em minha graduação.

Ao meu orientador, Aurélio Faustino Hoppe, por ter acreditado na conclusão deste trabalho.

Os chatos serão lembrados.

Jonathan Mauricenz

RESUMO

Este trabalho descreve o desenvolvimento de um protótipo em linguagem Java capaz de reconhecer símbolos musicais a partir de uma imagem de partitura. Para a classificação dos símbolos musicais fez-se necessário remover as linhas de pauta por projeção horizontal dos pixels pretos, isolar os símbolos musicais através de morfologia matemática, calcular os descritores de Fourier para caracterizar a forma e árvore de decisão como método de classificação. Os resultados obtidos demonstram que o protótipo tem uma taxa de 77% de acerto.

Palavras-chave: Descritores de Fourier. Remoção de pauta. Teoria musical. Transcrição musical.

ABSTRACT

This paper describes the development of a prototype in Java can recognize musical symbols from a picture score. For the classification of musical symbols became necessary to remove the staff lines for horizontal projection of black pixels, isolate the musical symbols using mathematical morphology, calculate the Fourier descriptors to characterize the shape and decision tree as a method of classification. The results show that the prototype has a rate of 77% accuracy.

Key-words: Fourier descriptor. Musical theory. Removing staff. Transcript musical.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

JavaCV – Java openCV

OpenCV – *Open Computer Vision library*

OMR – *Optical Music Recognition*

RGB – *Red Green Blue*

RVM – *Relevance Vector machine*

SVM – *Support Vector Machine*

UML – *Unified Modeling Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O ensino da música para deficientes visuais sempre foi um desafio. Entre preconceitos, dificuldades e necessidades, perde-se o sonho de estudar música (BONILHA, 2006, p. 17).

Segundo Bonilha (2006, p. 14) o deficiente visual ainda é visto como incapacitado, com sentimento de inferioridade, de marginalização. Um elemento fundamental para alcançar o ensino e inclusão dos deficientes visuais ao campo da música vem da percepção de alunos e professores sobre o método de ensino musical.

Bonilha (2006, p. 8) pressupõe que, no campo das artes uma pessoa com deficiência tenha direito a uma formação de qualidade, devendo ser garantido a eles o pleno acesso ao conhecimento inerente a todas as linguagens artísticas. Estas pessoas devem ser reconhecidas e legitimadas como artistas pela qualidade de seus trabalhos e não em função de suas deficiências.

Segundo Garcia, Furlan e Ribeiro (2006, p. 65) um deficiente visual possui como principais formas de auxílio à comunicação através do computador os terminais de braile e os conversores texto-fala. As duas tecnologias visam oferecer uma alternativa à saída visual do monitor do computador. Além disso, por se tratarem de softwares, assim como os conversores podem ser mais acessíveis do que os terminais braile, outros softwares que se baseiam em um sistema *Optical Music Recognition* (OMR) podem ser aplicados.

De acordo com Seufert (2010, p. 3) o objetivo de uma aplicação OMR é o reconhecimento, a representação e a gravação de partituras musicais num formato lido por máquinas. Um programa OMR tem que ser capaz de reconhecer o conteúdo musical e de fazer a análise semântica de cada símbolo musical de uma obra. No final, todas as informações musicais devem ser guardadas num formato de saída que seja facilmente lido por um computador.

Diante do exposto, este trabalho apresenta o desenvolvimento de um protótipo que faz o reconhecimento de símbolos musicais contidos em uma imagem de partitura.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de um protótipo capaz de reconhecer os símbolos contidos em uma imagem de partitura musical.

Os objetivos específicos do trabalho são:

a) reconhecer os símbolos utilizando técnica de segmentação de imagem comumente utilizadas em sistemas OMR;

b) analisar a forma do símbolo utilizando descritores de forma como método de identificação;
c) exibir e exportar o resultado do reconhecimento para um arquivo texto, que eventualmente poderá ser transcrito para uma simbologia em que deficientes visuais possam utilizar.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está disposto em quatro capítulos. O primeiro capítulo contém a introdução, objetivo do trabalho, estrutura do trabalho e observações gerais.

O segundo capítulo descreve a fundamentação teórica onde são apresentados temas relevantes ao desenvolvimento deste trabalho e, ao final, são apresentados os trabalhos correlatos.

O terceiro capítulo trata do desenvolvimento do protótipo, onde são relacionados seus requisitos, a especificação, diagramas de caso de uso e de seqüência. Também é descrito a implementação apresentando técnicas e ferramentas utilizadas, a operacionalidade e, por fim, são apresentados e discutidos os resultados obtidos.

O quarto capítulo apresenta as conclusões do presente trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os conteúdos pesquisados para auxiliar o desenvolvimento do protótipo. A seção 2.1 descreve o que é teoria musical, e fornece alguns conceitos. A seção 2.2 descreve a arquitetura OMR. A seção 2.3 descreve a segmentação. A seção 2.4 descreve a extração das características. A seção 2.5 descreve a classificação do símbolo. Na seção 2.6 são apresentados os trabalhos correlatos.

2.1 TEORIA MUSICAL

Teoria da música é o conjunto de todos os conhecimentos teóricos em música, no seu sentido original ensina os princípios básicos e fundamentais em música (MED, 1996, p. 11). É o fundamento para descrever as possibilidades de organização dos fenômenos sonoros (OLIVEIRA; MERTZIG; SCHULZ, 2006, p. 46).

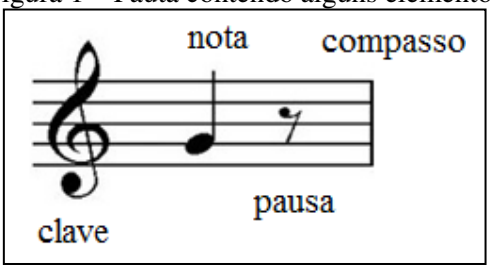
A música é a arte de combinar os sons de forma simultânea e sucessiva com ordem, equilíbrio e proporção dentro do espaço de tempo. É a arte de manifestar os diversos afetos de nossa alma mediante o som (NOBRE, 2008, p. 2).

Tradicionalmente, pode ser decomposta em melodia, harmonia, ritmo e contraponto. A melodia é representada pela combinação sucessiva dos sons. A harmonia é representada pela combinação dos sons simultâneos. O ritmo é a combinação dos valores de tempo. O contraponto é o conjunto de melodias dispostas em ordem simultânea (NOBRE, 2008, p. 2).

A música pode ser representada pelo equilíbrio de sons e silêncios. É escrita por sinais, onde cada sinal determina um valor. Os valores representam a duração de tempo de um som e silêncios. Os valores dos sons musicais são chamados de figuras. As ausências de sons são chamadas de pausas. A unidade de medida da música é o tempo. Cada tempo corresponde a uma pulsação. Esses sons musicais podem ser representados por símbolos gráficos denominados notas. São possíveis apenas sete notas, sendo elas DÓ, RÉ, MI, FÁ, SOL, LÁ e SI. Para essa escrita da música denomina-se então notação musical. Essas sete notas ouvidas sucessivamente formam uma série de sons denominada de escala (NOBRE, 2008, p. 3).

A seguir são expressos alguns conceitos fundamentais para a leitura de uma partitura musical conforme visto na Figura 1, juntamente com uma breve explicação fundamental para a compreensão deste trabalho.

Figura 1 – Pauta contendo alguns elementos



Fonte: adaptado de Nobre (2008, p.4).

Segundo Nobre (2008, p. 2) escreve-se as notas musicais sobre cinco linhas e quatro espaços horizontais paralelos e equidistantes entre si. Para estas linhas e espaços dá-se o nome de pauta ou pentagrama. Cada pauta recebe sinal colocado no início, à esquerda. Este é chamado de clave. O restante do espaço é dividido em unidades de tempo chamado de compasso. O compasso é composto por uma quantidade de tempos, ou pulsos. Os compassos são preenchidos por notas musicais e pausas que são representados com figuras. Esta escrita musical recebe o nome de notação musical (NOBRE, 2008, p. 3).

Ainda, outras características podem ser observadas, como as ligadura de prolongamento que somam os valores das notas. Essas são linhas curvas escrita sobre ou sob duas ou mais notas de mesmo nome e altura. Diferentes grafias com efeito sonoro igual são chamadas de notas enharmônia (NOBRE, 2008, p. 11). Ainda alguns ornamentos podem ser adicionados. Os ornamentos são notas ou grupos de notas acrescentadas a uma melodia com apenas a finalidade de adornar as notas reais da melodia (NOBRE, 2008, p. 18).

Através disto, se pode reconhecer os símbolos pela disposição de sua forma. Entretanto, para isto se busca isolar o símbolo pelo método de segmentação, a fim de uma melhor identificação.

2.2 SEGMENTAÇÃO

Segundo Marques Filho e Vieira Neto (1999, p. 10) a etapa de segmentação divide uma imagem em suas unidades significativas, ou seja, em objetos de interesse que a compõem.

Segundo Seufert (2010, p. 9) a segmentação dos objetos numa partitura musical consiste basicamente em identificar cada parte unitária da partitura. Este processo isola os símbolos, possibilitando a sua identificação correta. O processo de segmentação é, normalmente, fundamentado em uma decomposição hierárquica de onde a partitura é analisada e decomposta em pautas para, posteriormente, remover as linhas e identificar os componentes ligados.

O exemplo mais simples de segmentação resulta em uma imagem binária, onde os *pixels* pretos representam o fundo e as regiões de *pixels* brancos contíguos são consideradas objetos, ou inversamente (GOMES, 2001, p. 39). Esta limiarização é o método mais simples e intuitivo de segmentação de imagens. Todos os pixels que estão entre uma faixa de intensidade são classificados como pertencentes a uma mesma região (GOMES, 2001, p. 34).

2.2.1 Limiarização

Conforme Marques Filho e Vieira Neto (1999, p. 71), a limiarização divide a imagem separando as regiões de interesse através da sua similaridade. Deve ser aplicada em imagens em tons de cinza e, por isso, também é conhecida como binarização. A limiarização também modifica a representação dos tons de cinza da imagem. Para cada *pixel* é atribuído um novo valor, conforme visto no Quadro 1.

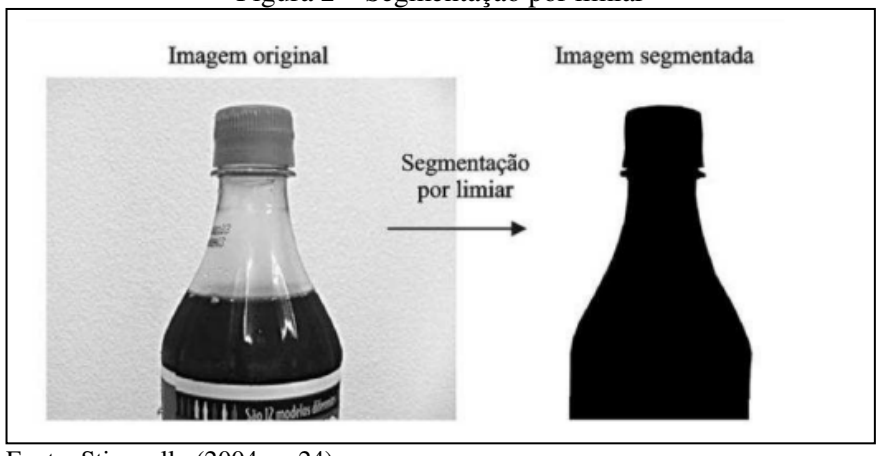
Quadro 1 – Condição do limiar

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Fonte: OpenCV (2012).

Portanto, o resultado deste processo é uma imagem binária, ou seja, a nova cor de cada *pixel* é representada por apenas um bit. Cada *pixel* da imagem, portanto, apresenta a cor preta de valor 0 ou branca de valor 255 (MARQUES FILHO; VIEIRA NETO, 1999, p. 71). A Figura 2 demonstra esta aplicação de segmentação por limiar.

Figura 2 – Segmentação por limiar



Fonte: Stivanello (2004, p. 24).

A qualidade da limiarização varia de acordo com o tom de corte escolhido, que pode ser obtido com auxílio do histograma da imagem, onde intuitivamente observa-se a quantidade de cada conjunto de tom de cinza (GOMES, 2001, p. 42). A limiarização é uma função para aumentar a automação do processo (MARGARIDA, 2009, p. 26).

2.3 EXTRAÇÃO DE CARACTERÍSTICAS

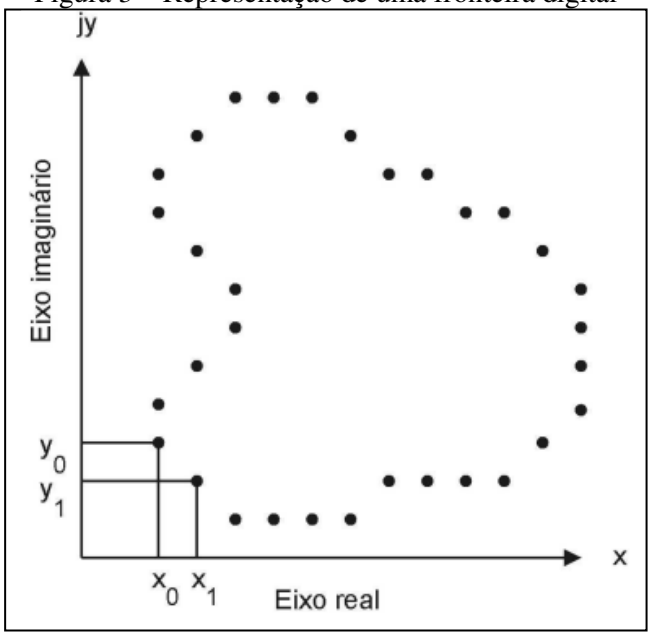
Segundo Stivanello (2004, p. 25) a análise direta sobre os *pixels* que formam os componentes da imagem não é eficiente. Estes *pixels* devem ser escritos em forma de descritores. Os descritores são um conjunto de números gerados para descrever uma forma, porém, podem não reconstruir completamente a forma descrita, mas devem ser suficientes para discriminar diferentes formas.

Segundo Marques Filho e Vieira Neto (1999, p. 10) extrair as características das imagens resultantes da segmentação através de descritores permite caracterizar com precisão cada forma encontrada, discriminando formas parecidas. Estes descritores devem ser representados por uma estrutura de dados adequada ao algoritmo de reconhecimento. É importante observar que a entrada ainda é uma imagem, mas a saída é um conjunto de dados correspondentes àquela imagem, como exemplo de saída uma lista de pontos que compõem o contorno.

2.3.1 Descritores de Fourier

César e Costa (2001 apud STIVANELLO, 2004, p. 26) afirmam que os descritores de Fourier são formas de representação de imagens para aplicações de visão computacional e reconhecimento de padrões. Estes descritores não constituem um método simples, mas uma classe de métodos com diferentes maneiras para defini-los. Podem ser obtidos a partir de uma fronteira digital como visto na Figura 3.

Figura 3 – Representação de uma fronteira digital



Fonte: Stivanello (2004, p. 27).

Conforme Gonzalez e Woods (2002, p. 840), partindo de uma fronteira digital de `N` pontos no plano `xy`, e iniciando a partir de um ponto arbitrário `(x0, y0)`, encontra-se pares ordenados `(x0, y0), (x1, y1),..., (xn-1, yn-1)` ao longo da fronteira. Cada par pode ser tratado como número complexo da forma exibida no Quadro 2, para `k=0,1,2,...,N-1`, onde `N` é a quantidade de pontos que compõe a fronteira.

Quadro 2 – Formato de número complexo

$$s(k) = x(k) + jy(k)$$

Fonte: Gonzalez e Woods (2002, p. 840).

Erpen (2004, p. 25) afirma que ao considerar o eixo real (`x`) e o eixo imaginário (`y`) de uma sequência de números complexos, reformula-se a interpretação da sequência, entretanto a natureza da fronteira não se altera. Esta representação reduz o problema de duas para uma dimensão.

Gonzalez e Woods (2002, p. 840) afirmam que os coeficientes complexos `a(u)` chamados de descritores de Fourier são obtidos através da transformada discreta de Fourier, sendo `u = 0, 1, 2,..., N-1`, onde `N` é a quantidade de pontos que compõe a fronteira, como observado no Quadro 3.

Quadro 3 - Transformada discreta de Fourier

$$a(u) = \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K}$$

Fonte: Gonzalez e Woods (2002, p. 840).

A transformada inversa de Fourier reconstrói s(k) a partir dos coeficientes a(u), é definida no Quadro 4, sendo u = 0, 1, 2,..., N-1 onde N é a quantidade de pontos que compõe a fronteira (GONZALEZ; WOODS, 2002, p. 840).

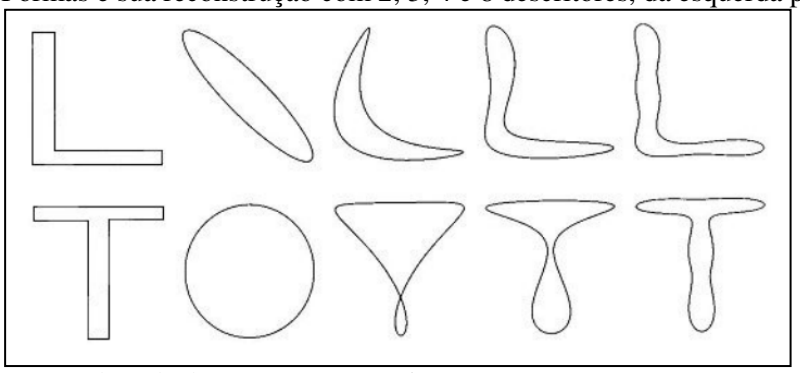
Quadro 4 – Transformada inversa de Fourier

$$s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) e^{j2\pi uk/K}$$

Fonte: Gonzalez e Woods (2002, p. 840).

Segundo Falcão (2008) estes descritores sofrem transformações de rotação, escala, translação e até sobre o ponto de partida do contorno, no entanto se a partida for sempre de um mesmo ponto sobre o contorno pode-se evitar os efeitos da rotação e escalonamento. Na Figura 4 são exibidas as fronteiras obtidas pela reconstrução da fronteira original, geradas a partir de quantidades diferentes de coeficientes.

Figura 4 – Formas e sua reconstrução com 2, 3, 4 e 8 descritores, da esquerda para direita



Fonte: adaptado de Erpen (2004, p. 26).

Bowman, Soga e Drummond (2001 apud ERPEN, 2004, p. 26) explicam que a invariância quanto à translação pode ser obtida com a não utilização do descritor onde `N=0`, e apresentam uma equação para a obtenção de descritores invariantes à rotação, sendo esta exibida no Quadro 5.

Quadro 5 – Fórmula para obtenção de invariância a rotação

$$a_r(u) = \sqrt{a_u^2 + b_u^2}$$

Fonte: Stivanello (2004, p. 27).

Através do uso da fórmula de transformação exibida no Quadro 5, os descritores obtidos de uma mesma borda em diferentes ângulos, tornam-se iguais.

2.4 CLASSIFICAÇÃO DO SÍMBOLO

A classificação consiste em reconhecer um objeto, uma forma ou uma característica marcante contida em um grupo de informações a serem observados. Esse processo agrupa estas informações por sua similaridade. O resultado pode ser um percentual de chance de acerto, ou uma imagem com algumas características enfatizadas. O problema da classificação é decidir a que grupo a amostra pertence. Deve também haver como alternativa, a rejeição da amostra (DAHMER, 1998; GONZALEZ; WOODS, 2000; LIBERMAN, 1997; WHELAN; MOLLOY, 2001 apud ERPEN, 2004, p. 21).

A árvore de decisão é um modelo de classificação que associa as classes e os conjuntos de atributos caracterizando os objetos a serem classificados. Ela é uma alternativa de expressar as mesmas regras que são obtidas quando ao construir-se uma tabela. Emprega-se em processamento simbólico e numérico. Um dos modelos de árvore de decisão é o *top-down*, que modela a classificação a partir de informações fornecidas por especialistas. A tarefa de uma árvore de decisão é realizar o mapeamento dos atributos para indicar a categorias a que pertence (ZUBEN, 2012, p. 27-28).

2.5 TRABALHOS CORRELATOS

Esta seção destina-se a investigar na literatura alguns dos principais trabalhos que envolvem pesquisa em reconhecimento de partituras. Dentre as obras disponíveis, foram selecionados três trabalhos: o trabalho de investigação e aperfeiçoamento de algoritmos de reconhecimento de símbolos musicais de Seufert (2010), o trabalho de pré-processamento e nível médio de classificação para um sistema de leitura automática de símbolos musicais de Perrotti (1993) e o trabalho de reconhecimento de imagens aplicado a partituras musicais de Margarida (2009).

2.5.1 Investigação e aperfeiçoamento de algoritmos de reconhecimento de símbolos musicais

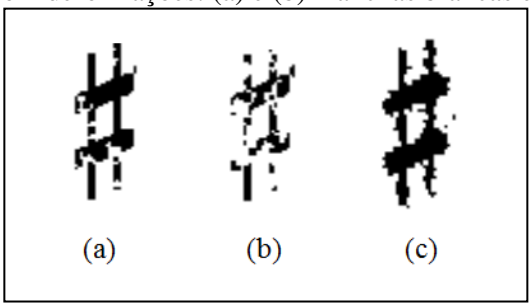
Segundo Seufert (2010, p. 1) em Portugal existe uma grande falha em relação à publicação musical. Não houve nenhum repositório de informação musical que tenha sido criado durante o último século. A ferramenta mais utilizada para a preservação tem sido a digitalização de partituras que oferece possibilidade para duplicações, distribuições e o processamento digital.

O objetivo de Seufert (2010, p. 1) foi desenvolver um estudo sobre a forma de reconhecer as notas musicais. O trabalho utilizou em seus métodos a detecção e remoção de

linhas de pauta, segmentação e, tendo como principal método, os classificadores.

Seufert (2010, p. 19) cria uma base de dados própria que contém símbolos individuais extraídos de partituras musicais de diferentes autores. Para validar seu estudo sobre partituras impressas, aplica padrões de deformação que imitam a degradação do papel impresso conhecidas como deformações *white speckles* (manchas brancas) e deformações segundo Kanungo (Figura 5). Tapas Kanungo (2000 apud SEUFERT, 2010, p. 21) estudou a degradação de documentos ajudando a criar um modelo de deformações demonstrado no documento *"A statistical, non parametric methodology for document degradation model validation"*.

Figura 5 – Sustenido com deformações: (a) e (b) manchas brancas e (c) segundo Kanungo



Fonte: adaptado de Seufert (2010, p. 21).

Após a fase de segmentação, são extraídas as características da forma encontrada, dentre as quais as principais são: a área de *pixels* pretos, a orientação e ângulo, o número de buracos ou espaços em branco, a densidade, o número de pontos finais com algoritmo de esqueletização e o número de intersecções que também é gerado a partir de um algoritmo de esqueletização.

Com isso criou uma base de dados para treinar os classificadores de rede neural, vizinhos mais próximos, máquina de vetores de suporte (SVM) e máquina de vetor de relevância (RVM), obtendo melhores resultados de detecção com SVM que reconheceu em torno de 97% e RVM com 94%.

Com relação a obras manuscritas Seufert (2010, p. 39) obteve símbolos musicais de determinados autores com resultados pouco satisfatórios, sendo a variedade de escrita entre os diversos autores a maior dificuldade para obtenção dos resultados.

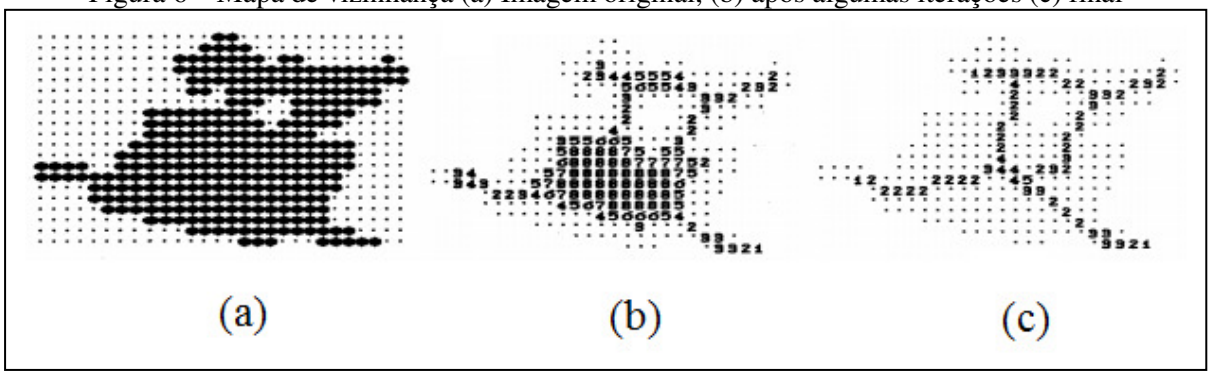
2.5.2 Pré-processamento e nível médio de classificação para um sistema de leitura automática de símbolos musicais.

Perrotti (1993, p. 67) propõe um sistema de reconhecimentos de símbolos musicais com ênfase ao pré-processamento, com aplicação de afinamento e vetorização. Partindo disto,

localiza o pentagrama e extrai as características classificando-as conforme um banco de dados próprio.

Na fase de afinamento utiliza um algoritmo próprio de mapa de vizinhança onde, a partir de uma imagem binária para cada ponto, recebe-se um valor, conforme a quantidade de pontos vizinhos. Após várias iterações gera uma esqueletização da forma, conforme mostra a Figura 6.

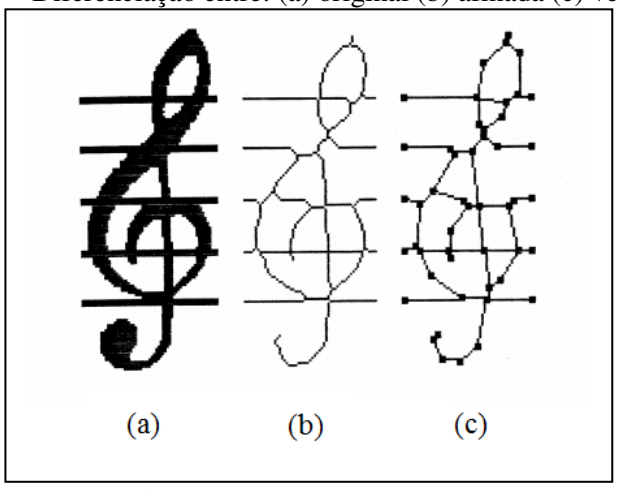
Figura 6 – Mapa de vizinhança (a) Imagem original, (b) após algumas iterações (c) final



Fonte: adaptado de Perroti (2010, p. 27).

A vetorização, como mostra a Figura 7, é o processo de aproximar as curvas do esqueleto por segmentos de retas obtendo assim os pontos críticos da imagem, que são pontos terminais e pontos de cruzamento.

Figura 7 – Diferenciação entre: (a) original (b) afinada (c) vetorizada



Fonte: Perroti (2010, p. 33).

Após isso, a classificação procura uma abordagem de reconhecimento sintático com o método de pontos mais próximo, onde se constrói candidatos criando um algoritmo próprio desenvolvendo, assim, um algoritmo que faz a identificação de símbolos mais complexos, com linhas e pontos de cruzamento entre os símbolos. Observa-se que as linhas de pauta da partitura não são removidas previamente.

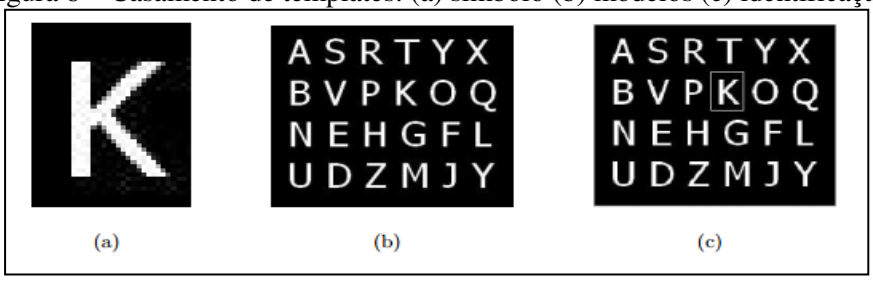
Perroti (1993, p. 67) conclui que o método desenvolvido é mais eficiente para símbolos complexos perdendo precisão para símbolos muito simples. O resultado da classificação é uma lista de candidatos. Esta lista pode ser extensa, onde cada candidato possui pesos e distâncias, onde sugere estudo mais aprofundado. Alguns símbolos menores acabam desaparecendo após processo de afinamento e vetorização, necessitando de métodos auxiliares para o reconhecimento.

2.5.3 Reconhecimento de imagens aplicado a partituras musicais.

O trabalho de Margarida (2009, p. 14) tem como objetivo criar uma ferramenta gratuita de código aberto utilizando a linguagem Python[1].

Margarida segue alguns passos de Perroti (1993, p. 26-31) e aplica afinamento, vetorização e busca pontos críticos, entretanto usa outras técnicas como morfologia matemática com elemento estruturante, classificação com casamento de *templates* conforme visto na Figura 8. O casamento de *templates* ou *Template Matching* é uma técnica para encontrar partes de uma imagem que se correlacionem com outra imagem a partir de uma base de modelos e aplicação de *Hit and Miss* que serve para detectar a presença de padrões na imagem.

Figura 8 – Casamento de templates: (a) símbolo (b) modelos (c) identificação



Fonte: Margarida (2009, p. 29).

Dentre seus objetivos ainda cita aplicações práticas de adaptação da partitura para outros instrumentos, a conversão de partituras para braile, a transposição que é a alteração de uma partitura para outro tom, possibilidade de re-impressão da partitura sem perda de qualidade e economia de espaço no arquivamento.

Em seus resultados, Margarida (2009, p. 68) alcançou taxas de acerto acima de 95% em resoluções baixa e média, havendo decréscimo de menos de 1% para o total de elementos reconhecidos corretamente ao se utilizar uma resolução mais baixa, demonstrando a não dependência da resolução ou tamanho da imagem.

[1] Disponível em http://code.google.com/p/pythonomr/

Ainda, Margarida (2009, p. 69) afirma que a vantagem do algoritmo desenvolvido é a independência da resolução da imagem e o uso da morfologia matemática. Entretanto, para as imagens de entrada o programa mostra-se sensível a falhas, ruídos e inclinações, ocasionando erros ou mesmo o não reconhecimento da nota musical.

3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo detalha as etapas do desenvolvimento do protótipo. A seção 3.1 apresenta os requisitos, a seção 3.2 descreve a especificação, a seção 3.3 descreve a implementação e a seção 3.4 apresenta os resultados e discutições.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O protótipo de reconhecimento dos elementos da partitura deverá:

a) permitir que o usuário escolha uma imagem de partitura (Requisito Funcional – RF);
b) remover as linhas de pauta (RF);
c) isolar o símbolo (RF);
d) reconhecer os símbolos (RF);
e) exportar o conteúdo reconhecido para um arquivo de dados texto (RF);
f) utilizar técnicas de descritores de forma (Requisito Não-Funcional – RNF);
g) ser implementado utilizando a linguagem Java (RNF);
h) ser implementado utilizando o ambiente de desenvolvimento Eclipse ou NetBeans (RNF).

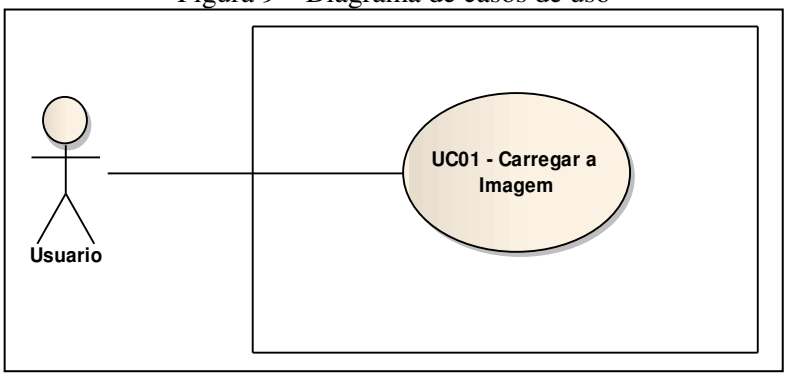
3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação do protótipo, representada por diagramas da *Unified Modeling Language* (UML) e modelada utilizando a ferramenta Enterprise Architect 8.0. Foram utilizados os diagramas de casos de uso, diagrama de classes e de sequência para representar a metodologia utilizada no desenvolvimento do protótipo.

3.2.1 Diagrama de casos de uso

A Figura 9 apresenta o único caso de uso do protótipo.

Figura 9 – Diagrama de casos de uso



O protótipo possui apenas um caso de uso, o usuário pode apenas apontar qual a partitura que será analisada e classificada.

O Quadro 6 apresenta o caso de uso `UC01 – Carregar Imagem`.

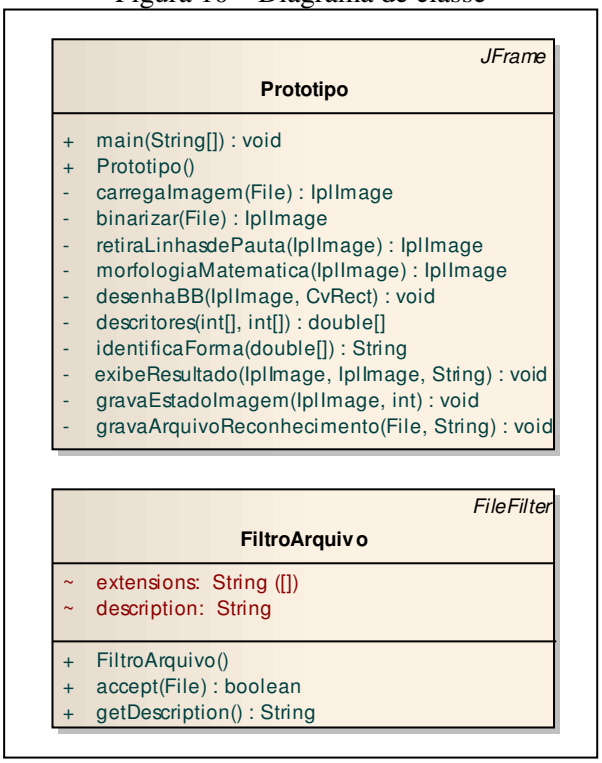
Quadro 6 – Detalhamento do caso de uso 01

UC01 – CARREGAR IMAGEM	
Pré-condições	Não há pré-condições.
Cenário principal	01) O usuário inicia o protótipo. 02) O usuário define a imagem a ser carregada. 03) O protótipo carrega a imagem. 04) O protótipo realiza o processo de análise e classificação das formas encontradas partir da imagem carregada. 05) O protótipo exibe e gera o resultado em arquivo de texto
Exceção 01	No passo 3, o arquivo pode estar em um formato com transparência ou não reconhecido pelo protótipo. O processo de análise e classificação é prejudicado, gerando um resultado incorreto.
Pós-condição	Deve ser exibido na tela e gerado arquivo com o resultado da classificação.

3.2.2 Diagrama de classes

A Figura 10 representa o diagrama de classes que compõem o protótipo assim como seus relacionamentos, omite-se as classes utilizadas pelo JavaCV.

Figura 10 – Diagrama de classe



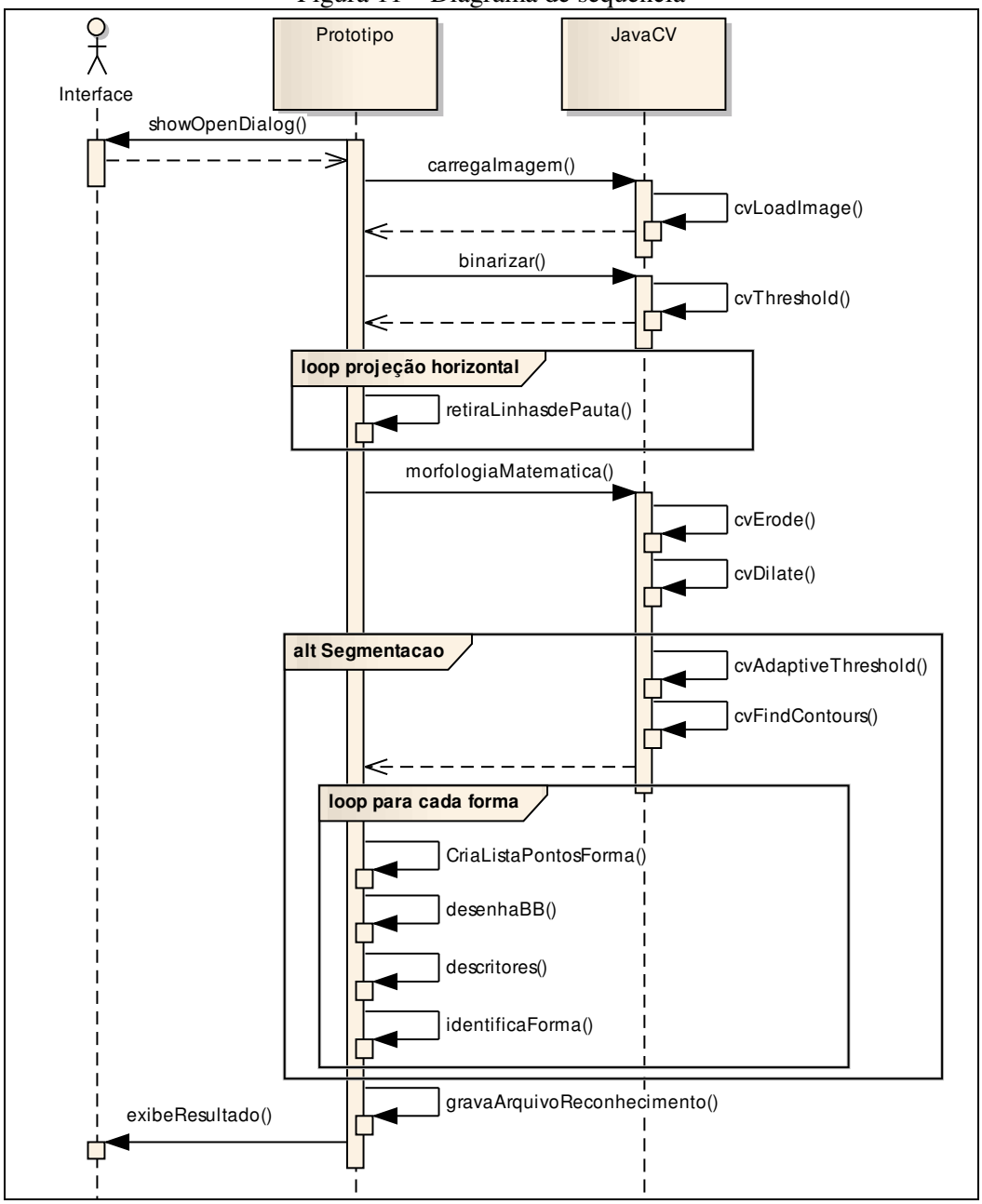
O protótipo possui uma classe principal nomeada `Prototipo` e uma classe auxiliar `FiltroArquivo`. A classe `FiltroArquivo` é usada para indicar quais arquivos são mostrados na caixa de diálogo de navegação de arquivos para o usuário. A classe `Prototipo` é responsável pela carga da imagem, pré-processamento, segmentação, valores dos descritores de Fourier geração e exibição dos resultados e geração do arquivo final.

3.2.3 Diagrama de sequência.

Nessa seção é apresentado o diagrama de sequência que representa o conjunto de ações executadas pelo protótipo. A

Figura 11 mostra o diagrama equivalente ao caso de uso `UC01 - Carregar Imagem`.

Figura 11 – Diagrama de sequência



O protótipo inicia solicitando ao usuário qual imagem a ser processada. Esta ativa a chamada do método `carregaImagem` que realizará várias chamadas de funções ao JavaCV. Depois de selecionado, o protótipo inicia a carga da imagem com o comando `cvLoadImage`.

Partindo disto, aplica-se a função `binarizar` que aplica função de limiar do JavaCV através do comando `cvThreshold`, possibilitando que o método `retiraLinhasdePauta` prossiga localizando e removendo as linhas de pauta da partitura que contiverem a maior quantidades de pontos por soma de projeção horizontal.

Após a retirada das linhas de pauta aplica-se a morfologia matemática através das funções `cvErode` e `cvDilate` que preparam a imagem para a localização das formas.

Para a segmentação, aplica-se `cvAdaptiveThreshold` que usa limiarização adaptativa de `CV_ADAPTIVE_THRESH_MEAN_C` e o tipo de limiarização `CV_THRESH_BINARY_INV`. Este comando prepara para encontrar os contornos da imagem com o comando `cvFindContours`.

A partir das coordenadas da borda, possibilita-se emoldurar uma *bounding box* e permite também criar duas listas contendo as coordenadas de pontos em `x` e `y`, utilizadas para gerar os valores dos descritores de Fourier.

Os descritores de Fourier então são calculados pela chamada da função `descritores`, onde sofrem normalização com invariância à rotação e translação, tornando-se apenas uma lista de valores, os quais são comparados e identificados.

O resultado da identificação é exibido ao usuário em tela comparativa com a imagem de entrada original, junto com a imagem segmentada e, também, é gravado em arquivo texto no mesmo diretório da imagem.

3.3 IMPLEMENTAÇÃO

A seguir, são mostradas as técnicas e ferramentas utilizadas, com ênfase na implementação do protótipo. A seção 3.3.1 descreve as técnicas e ferramentas utilizadas, a seção 3.3.2 são detalhados os processos de implementação do protótipo e a seção 3.3.3 descreve a operacionalidade da implementação.

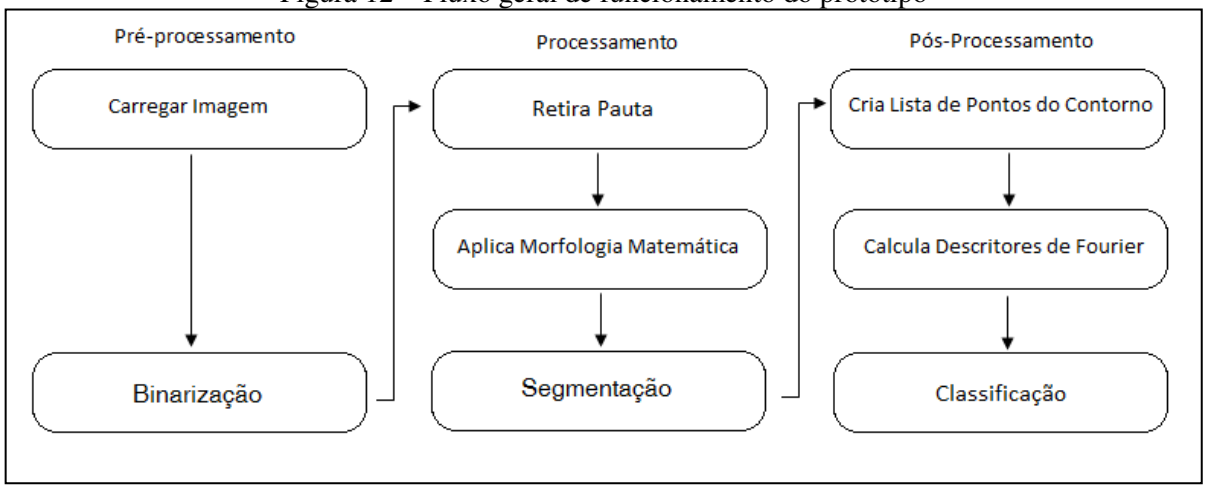
3.3.1 Técnicas e ferramentas utilizadas

O ambiente de desenvolvimento utilizado para a implementação do sistema foi o Netbeans 7.2. Esta ferramenta de desenvolvimento foi escolhida por oferecer diversos componentes necessários para a implementação e integração com o JavaCV, o que permitiu agilidade no desenvolvimento, junto com a biblioteca utilizada para manipulação de imagens. A principal técnica implementada no sistema são os descritores de Fourier, que está descrita na seção 3.3.2.3.2.

3.3.2 Implementação do protótipo

Nesta seção são apresentadas as etapas da implementação das funcionalidades do protótipo desenvolvido no presente trabalho. Pode-se dividir este protótipo em três partes principais, sendo eles o pré-processamento, processamento e pós-processamento, conforme visto na Figura 12 que apresenta o fluxo geral de funcionamento do protótipo.

Figura 12 – Fluxo geral de funcionamento do protótipo



Nas próximas seções serão descritas as etapas do fluxo geral de funcionamento do protótipo. Na seção 3.3.2.1 descreve-se as etapas envolvidas no pré-processamento detalhando os processos de carga da imagem e binarização. Na seção 3.3.2.2 descreve-se as etapas envolvidas no processamento detalhando os processos de remoção das linhas de pauta, morfologia matemática e segmentação. Na seção 3.3.2.3 descreve-se as etapas envolvidas no pós-processamento como formulação da cadeia de pontos, método de classificação e demonstração do cálculo dos descritores de Fourier.

3.3.2.1 Pré-processamento

A primeira etapa realizada pelo protótipo é dividida em dois processos, carregar imagem e binarização, e, é responsável pela correção, adaptação e preparação da imagem para ser processada conforme itens descritos a seguir.

3.3.2.1.1 Carregar imagem

Para fazer a carga da imagem é usado a função `cvLoadImage` da classe `opencv_highgui` do JavaCV, que retorna uma imagem no formato `IplImage`. Esta função precisa de dois parâmetros: o primeiro é um tipo `String` (caminho do arquivo) e o segundo um tipo `int` (quantidade de canais de cor da imagem). Os canais de cor podem ser representados pelas constantes: `CV_LOAD_IMAGE_ANYDEPTH`, `CV_LOAD_IMAGE_COLOR` e `CV_LOAD_IMAGE_GRAYSCALE`.

Pode-se obter uma imagem em tom de cinza através de duas formas. Carregando uma imagem com a constante `CV_LOAD_IMAGE_GRAYSCALE`, ou, pela função `cvCvtColor`, que converte a imagem colorida para tons de cinza.

3.3.2.1.2 Binarização

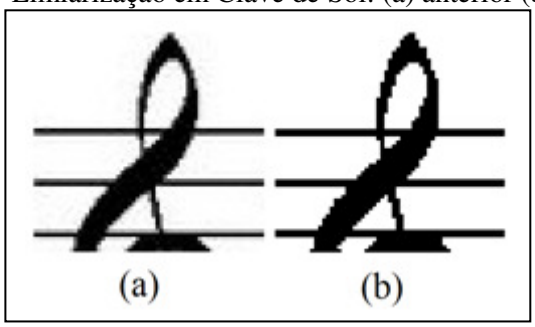
A fase de binarização é responsável por separar as áreas de interesse da imagem diferenciando assim os objetos de interesse do fundo, e para isto utilizou-se do método de limiarização, usando as funções `cvThreshold` conforme visto no Quadro 7.

Quadro 7 – Código fonte para a função de limiar

```
1: cvThreshold(cinza, cinza, limiar, 255, CV_THRESH_BINARY);
```

Para esta função utilizam-se cinco parâmetros: a imagem de referência, a imagem de destino, o valor do limiar de corte, o novo valor que será aplicado e o tipo de limiar. O resultado da aplicação deste método pode ser visto conforme Figura 13.

Figura 13 – Limiarização em Clave de Sol: (a) anterior (b) posterior



A limiarização define os contornos. Partes anteriormente esfumaçadas ou confusas são distinguidas. Com esta função pode se identificar mais claramente a parte pertencente ao fundo (área em branco) separando do objeto a ser trabalhado (área em preto).

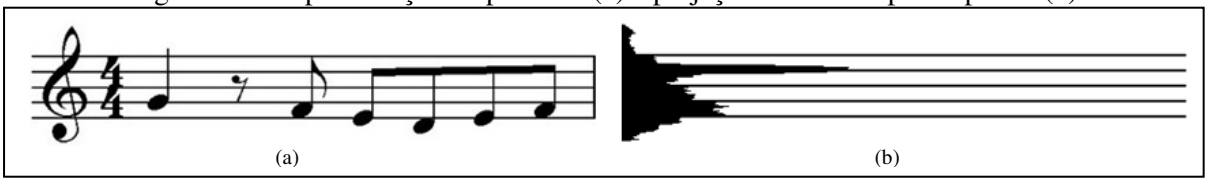
3.3.2.2 Processamento

Esta etapa foi dividida em três processos: remover as linhas de pauta da partitura, aplicação da morfologia matemática e segmentação, conforme itens descritos a seguir.

3.3.2.2.1 Remoção das linhas de pauta

Para remover as linhas de pauta foi utilizada a técnica de projeção horizontal dos *pixels* pretos. Esta técnica é aplicada tendo em vista que o protótipo considera que a imagem esteja em rotação horizontal e pode ser vista na Figura 14.

Figura 14 – Representação de partitura (a) e projeção horizontal pixels pretos (b)



Observado na Figura 14 a imagem de entrada (a) resulta na demonstração (b) da somatória de *pixels* pretos em cada linha. Esta ilustração demonstra que quando maior a linha gerada (b) maior foram à concentração dos *pixels* pretos em relação à mesma linha da imagem de entrada (a).

O método `retiraLinhasdePauta` é aplicado sobre imagens binarizadas, ou seja, que contém apenas *pixels* pretos e brancos. Seu objetivo é remover a pauta da partitura, aplicando a cor branca para a linha identificada. As linhas da imagem que contiverem as maiores quantidades de *pixels* pretos são identificadas como as linhas de pauta.

Para o método de soma dos *pixels* pretos utiliza o código do Quadro 8. Este percorre as linhas criando uma lista que retém os valores das somas, admitindo a maior soma como referência à pauta.

Quadro 8 – Código fonte para soma horizontal

```
1:  for (int a = 0; a < imagem.cvSize().height(); a++) { //linha
2:       for (int b = 0; b < imagem.cvSize().width(); b++) { //coluna
3:            if (cvGet2D(imagem, a, b).val(0) == 0) {
4:                 somas[a]++;
5:            }
6:       }
```

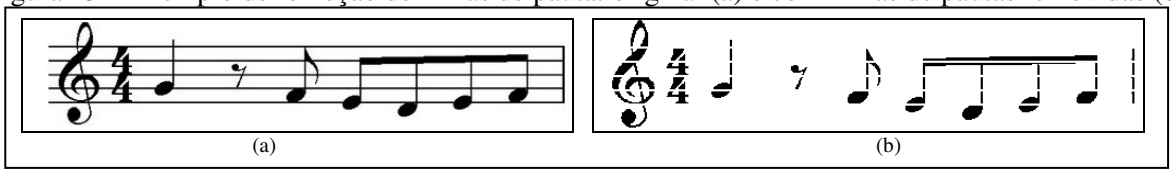
Identificadas as linhas, inicia-se a sua remoção aplicando a cor branca sobre área das linhas que obtiveram semelhança ao valor de referência conforme Quadro 9. Admite-se uma variação de até 15% para o valor de referência, para que possa ser aceito uma pequena variação no comprimento das linhas ou quando a pauta esta deformada.

Quadro 9 – Código fonte para remoção da linha

```
1:  for (int i = 0; i < somas.length; i++) {
2:       if (somas[i] >= (maior * 0.85)) {
3:            CvPoint pt1 = new CvPoint(0, i);
4:            CvPoint pt2 = new CvPoint(imagem.width(), i);
5:            cvLine(imagem, pt1, pt2, CV_RGB(255, 255, 255), 1, 4, 0);
6:       }
7:  }
```

Para mudar a cor, utiliza-se a função `cvLine` do OpenCV, modificando a cor da linha por completo para a cor branca. O resultado de todo o método pode ser observado na Figura 15.

Figura 15 – Exemplo de remoção de linhas de pauta: original (a) e com linhas de pautas removidas (b)



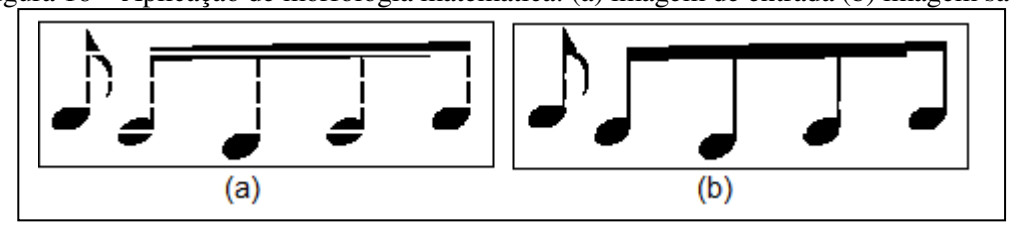
Entretanto, esta técnica de remoção de linhas de pauta deforma as notas, e em muitos casos divide-a em várias partes, necessitando a aplicação de uma função que reconstrua ou complete esta intersecção da nota com as linhas, partindo então para morfologia matemática.

3.3.2.2.2 Morfologia Matemática

Esta etapa é responsável pela reconstrução do símbolo, após ter sido deformado pela retirada das linhas de pauta e para tal o método `morfologiaMatematica` aplica as funções `cvErode` e `cvDilate` do JavaCV conforme visto na Figura 16. Ocorre primeira a erosão seguida da dilatação, preservando dessa forma o símbolo. Apenas a dilatação pode fundir símbolos muito próximos. Se a dilatação ocorrer primeiro, as hastes dos símbolos e outras partes finas poderão desaparecer.

Cada função precisa de quatro parâmetros sendo, a imagem de origem, a imagem de destino, uma instância da classe `IplConvKernel`, que é o elemento estruturante e um tipo inteiro, que é a quantidade de iterações a serem realizadas. O terceiro parâmetro de tipo `IplConvKernel` pode ser nulo e, se caso for, o JavaCV assume ser uma matriz de 3x3 dimensões.

Figura 16 – Aplicação de morfologia matemática: (a) imagem de entrada (b) imagem saída



Para o protótipo, foram utilizados como parâmetros, a imagem segmentada, sendo erodida três vezes e dilatada duas. Utilizou-se o elemento estruturante padrão de 3x3. Estes parâmetros obtidos através da documentação do OpenCV (2012) são suficientes para o protótipo. Uma matriz maior agregada a um elemento estruturante próprio demonstrou ser complexa, necessitando de um estudo específico que contempla cada possibilidade de reconstrução dos símbolos.

3.3.2.2.3 Segmentação

Esta etapa é responsável por encontrar os símbolos dispostos em *pixels* pretos em formas conexas, conforme visto na Figura 16. Para cada forma, devem-se identificar os contornos ou fronteiras digitais possibilitando o cálculo dos descritores de Fourier, que são utilizados para classificar entre formas. Para encontrar os contornos, o protótipo utiliza a função `cvFindContour` disponível no OpenCV conforme visto no Quadro 10.

Quadro 10 – Código fonte para a função de encontrar contornos

```
1: CvSeq contours = new CvContour();
2: cvFindContours(cinza, CvMemStorage.create(), contornos,
3:       Loader.sizeof(CvContour.class),
4:       CV_RETR_CCOMP, CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
```

Esta função precisa de sete parâmetros sendo estes a imagem de entrada, uma instância da classe `CvMemStorage`, uma instância da classe `CvContour`, um inteiro com o tamanho do cabeçalho, o paramento `CV_RETR_CCOMP`, o parâmetro `CV_CHAIN_APPROX_NONE` e uma instância da classe `CvPoint`, que para OpenCV é usado para compensação de deslocamento. O parâmetro `CV_RETR_CCOMP` recupera os contornos organizando em uma hierarquia. O parâmetro `CV_CHAIN_APPROX_NONE` armazena os pontos de contorno, ou seja, os pontos subsequentes do contorno será um vizinho horizontal, vertical ou diagonal.

A imagem de entrada será a referência de onde serão extraídos os contornos. Precisa ser uma imagem de 8 *bits* e de apenas 1 canal de cor. A função pede, que antes deste processo, seja aplicado na imagem que será referência, dentre algumas funções a `adaptiveThreshold` (OPENCV, 2012). Ela prepara a imagem para serem extraídos os contornos e utiliza sete parâmetros, sendo eles: a imagem de referência, a imagem de destino, um valor inteiro, que será o novo valor a que, normalmente, aplica-se o valor 255 que corresponde ao branco, a constante `CV_ADAPTIVE_THRESH_MEAN_C`, o tipo de novo limiar com a constante `CV_THRESH_BINARY_INV`,um inteiro que representa o tamanho do bloco de vizinhos e um valor constante para ser subtraído da média ou mediana (OPENCV, 2012).

Partindo disso, os segmentos de contornos são armazenados em formato `CvSeq` que é um formato de lista encadeada de objetos `CvSeq`. Sendo que cada objeto encontrado dispõem-se em horizontal e pode ser acessado por `CvSeq.h_next()` e para o contorno encontrado fica alocado na posição vertical e pode ser navegada por `CvSeq.v_next()`.

Opcionalmente, após encontrar cada forma, grava-se uma *bounding box*, que é um retângulo que envolve uma forma ou de um polígono para verificar o andamento do protótipo

e é visualizado no resultado final do protótipo.

3.3.2.3 Pós-processamento

Esta etapa foi dividida em três processos: criar a lista de pontos a partir do contorno da forma encontrada, calcular os descritores de Fourier e aplicar a classificação entre formas conforme os itens descritos a seguir.

3.3.2.3.1 Criação da lista de pontos do contorno

Esta etapa é responsável por unir os valores de coordenadas do contorno úteis à formação dos descritores de Fourier. São geradas duas listas `pontoX` e `pontoY`, que recebem os valores da instância da classe `CvPoint` obtida através da lista `CvSeq`, que é retorno direto da função `cvFindContour` visto no Quadro 11.

Quadro 11 – Parte do código fonte usado para criar a lista de pontos

```
1:   for (int i = 0; i < contornos.total(); i++) {
2:        CvPoint p = new CvPoint(cvGetSeqElem(contours, i));
3:        pontoX[i] = p.x();
4:        pontoY[i] = p.y();
5:   }
```

Assim, para obter a lista de pontos, percorrem-se os segmentos da forma encontrada recuperando os pontos e as coordenadas `x` e `y` através da função `cvGetSeqElem`.

3.3.2.3.2 Cálculo dos descritores de Fourier

Esta etapa é responsável por encontrar os valores dos descritores de Fourier. Estes descritores representam a forma do objeto através do seu contorno. A precisão da forma descrita está diretamente ligada à quantidade de descritores utilizados, ou seja, quanto mais descritores utilizados, o resultado será mais preciso e próximo a forma original. A quantidade máxima de descritores é a própria quantidade de pontos do contorno.

Inicialmente, são identificados os valores da quantidade de descritores a serem calculados e a quantidade de pontos do contorno. Partindo disto, utiliza-se a fórmula de transformada discreta de Fourier para encontrar seus coeficientes, conforme descrita no Quadro 12.

Quadro 12 – Código fonte para obtenção dos descritores de Fourier

```
01:  int U = x.length; //quantidade de descritores
02:  int K = x.length; //tamanho da lista pontos
03:
04:  double[] fReal = new double[U];
05:  double[] fImag = new double[U];
06:
07:  for (int u = 0; u < U; ++u) {//para cada descritor
08:      for (int k = 0; k < K; ++k) {//percorre cada ponto do contorno
09:          double theta = (2 * Math.PI * u * k) / K; // theta
10:          //parte real
11:          fReal[u] += x[k] * Math.cos(theta)
12:                + y[k] * Math.sin(theta);
13:          //parte imaginaria i
14:          fImag[u] += y[k] * Math.cos(theta)
15:                - x[k] * Math.sin(theta);
16:      }
17:  }// fim encontra dft's
```

Para este processo utiliza-se `U` para identificar a quantidade total de descritores, `u` apontando o descritor que está sendo calculado na iteração, `K` para identificar a quantidade de pontos do contorno, `k` identificando o índice que está sendo calculado e as listas de coordenadas descritas por `x` e `y`. Logo, a partir de `theta`, representado por `2πuk/K` encontra-se parte do somatório para os descritores guardados em parte real `fReal` e parte imaginária em `fImag`. Por definição o coeficiente encontrado é um número complexo e possui parte real e parte imaginária.

Para o próximo momento, se aplicam as invariâncias a translação e rotação conforme visto no Quadro 13.

Quadro 13 – Código fonte para aplicação de invariância

```
01:  //% invariant translacao matlab -> f([1]) = 0
02:  fReal[0] = 0;
03:  fImag[0] = 0;
04:  //fim invariacao translacao
05:
06:  //invariancia rotacao
07:  double[] normalizado = new double[fReal.length];
08:  for (int u = 1; u < fReal.length; u++) {
09:       normalizado[u] = Math.sqrt(
10:            Math.pow(fReal[u], 2)
11:            + Math.pow(fImag[u], 2));
12:  }//fim rotacao
```

Neste momento, obtém-se apenas uma lista de valores. Estes valores possuem invariância a translação e rotação das formas. Entretanto buscando uma possível quantidade menor de descritores utilizados, se aplica a rotina `novoTamanho`, que busca redimensionar a quantidade de descritores utilizada mantendo os mais significativos. Pode ser visto no Quadro 14 .

Quadro 14 – Código fonte para aplicação de redimensionamento

```
01:  normalizado = flip(normalizado);
02:  double[] temp = new double[n];
03:  int inicio = (int) (Math.ceil(
04:      (normalizado.length - n) / 2));
05:  if (((normalizado.length % 2 == 0)
06:          && (n % 2 != 0))
07:      || ((normalizado.length % 2 != 0)
08:          && ((n % 2 == 0)))) {
09:      inicio += 1;
10:  }
11:  System.arraycopy(normalizado, inicio, temp, 0, n);
12:  temp = iflip(temp);
13:  return temp;
```

O Quadro 14 mostra o processo de redimensionamento. A lista normalizado passa pelo processo de inversão dos lados. Em um exemplo, uma lista contendo os valores [1,2,3,4] deve resultar uma lista contendo os valores [3,4,1,2]. Após, conforme a quantidade de elementos e de elementos a ser redimensionada seja impar ou par, se define o início. Após, copia-se a parte central da lista. Novamente se faz o processo de inversão. Ao final, a lista resultante será a base para comparação das formas dos símbolos.

3.3.2.3.3 Classificação

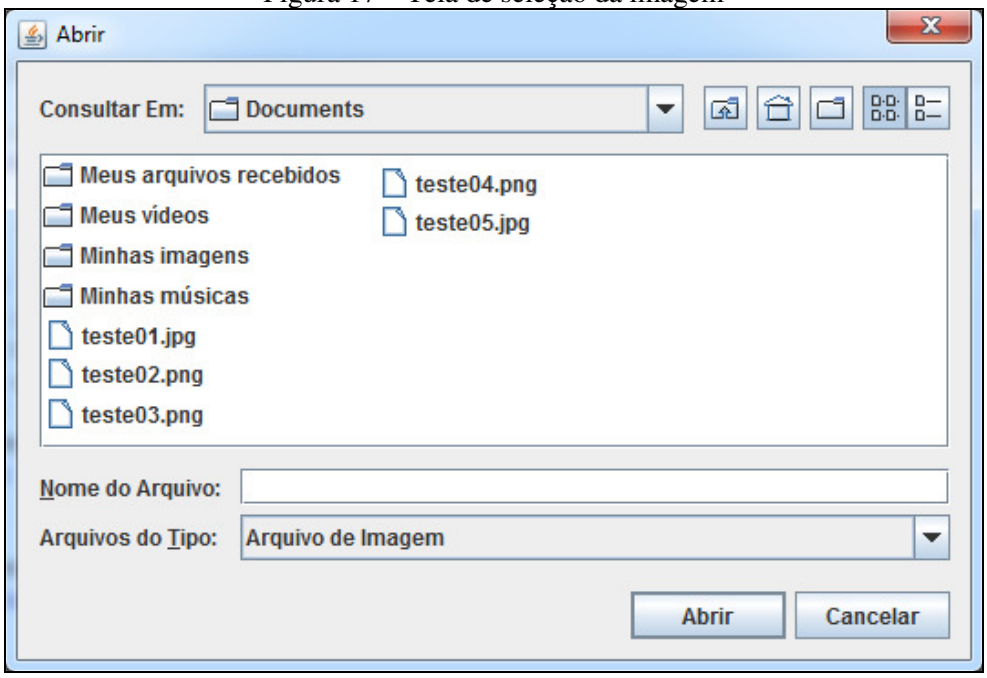
Através desta esta etapa se pode identificar e rotular (nomear) as formas encontradas. Os dados numéricos necessários para a classificação são obtidos através da extração características.

Para a classificação, se utiliza uma árvore de decisão. As regras da árvore de decisão são construídas sobre análise dos valores de formas já conhecidas criando uma árvore de decisão. Para a identificação da forma, os valores encontrados são submetidos ao método `identificaForma`. Estes valores são então comparados, resultando em apenas um rótulo. Este rótulo é adicionado à lista de formas para ao final ser exibido na tela para o usuário e ser gravado em arquivo.

3.3.3 Operacionalidade da implementação

Nesta seção será apresentada a operacionalidade do protótipo mostrando suas principais características. Após iniciado o protótipo é aberto uma tela de navegação de pastas e arquivos permitindo que seja selecionada a imagem como visto na Figura 17.

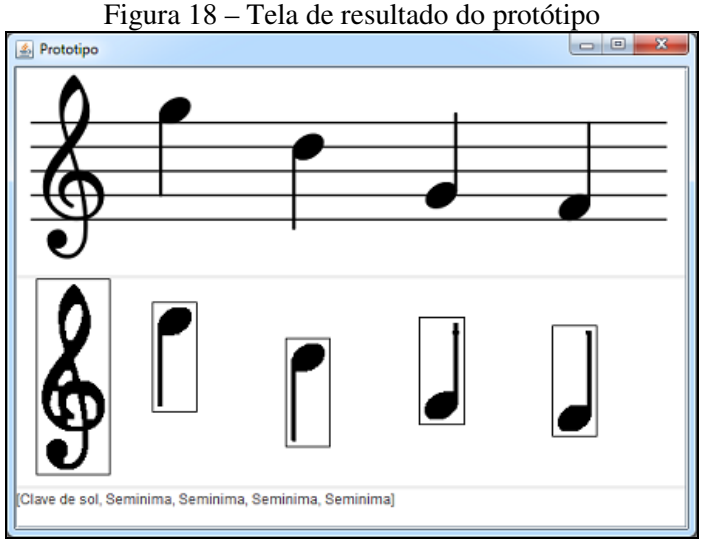
Figura 17 – Tela de seleção da imagem



Para carregar a imagem desejada pressiona-se o botão Abrir (Figura 17). Ao selecionar uma imagem inicia o processamento de análise e classificação da partitura.

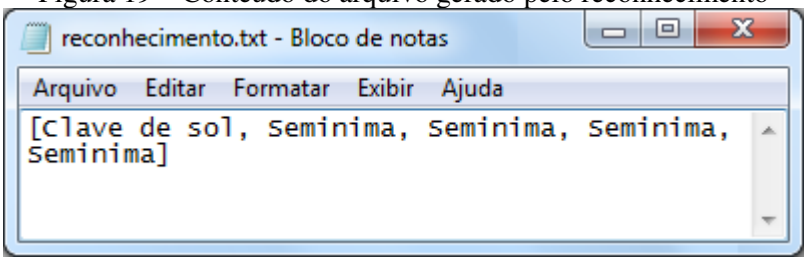
Após a realização do processamento é exibido ao usuário uma tela de resultado. Este pode ser visto na Figura 18.

Figura 18 – Tela de resultado do protótipo



Nesta tela, Figura 18, é possível comparar a imagem de entrada com uma segunda imagem que contém apenas as formas encontradas marcadas com *bounding box* e, abaixo, a representação do resultado da identificação dos símbolos. O arquivo gerado pode ser visto na Figura 19.

Figura 19 – Conteúdo do arquivo gerado pelo reconhecimento



Na Figura 19 se pode visualizar o arquivo gerado no diretório da imagem. Os valores encontrados são os mesmos contidos no rodapé da Figura 18. O nome do arquivo gerado será `reconhecimento.txt`. Através desta se podem visualizar as formas são dispostas em linhas e separadas por vírgula.

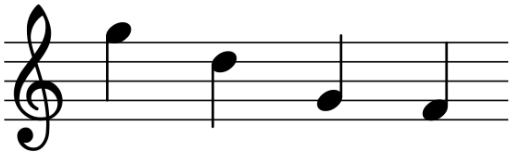
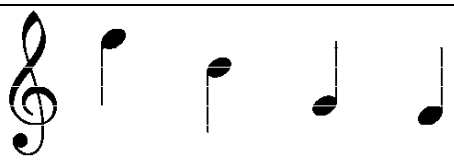
3.4 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os experimentos realizados, bem como os resultados obtidos. Foram realizados experimentos para cada fase de desenvolvimento: remoção de pauta, segmentação e representação de forma. Os trechos de partituras utilizados foram gerados pelo programa MuseScore. A seção 3.4.1 apresenta os resultados obtidos nos experimentos para a remoção de pauta. A seção 3.4.2 segmentação e na 3.4.3 os descritores.

3.4.1 Remoção da pauta

Foram realizados testes para avaliar a etapa de remoção das linhas de pauta. Para todas as imagens de entrada, o protótipo deve identificar e remover estas linhas pelo método de projeção horizontal visto no tópico 3.3.2.1.1 (remoção das linhas de pauta). As imagens de entrada são trechos de partitura confeccionados e submetidas ao método de binarização. O primeiro experimento é visto no Quadro 15.

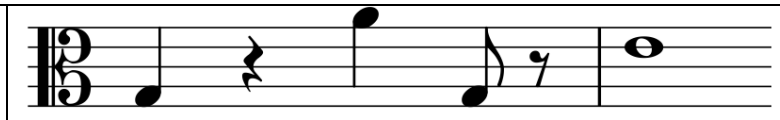
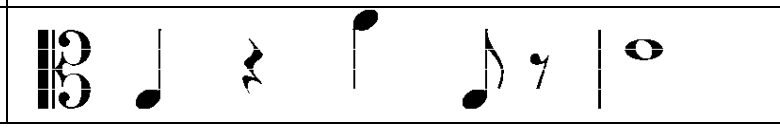
Quadro 15 - Experimento 1 de remoção de linhas de pauta

Experimento 01	
Entrada	
Saída	

O primeiro experimento que é observado no Quadro 15 é um trecho de partitura com pouca complexidade, sendo apenas cinco símbolos e contendo apenas duas formas diferentes, sendo da esquerda para a direita a clave de sol e as restantes semínimas. Para este

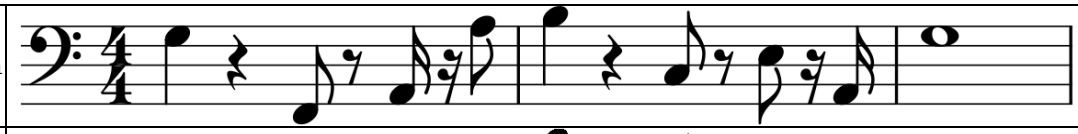
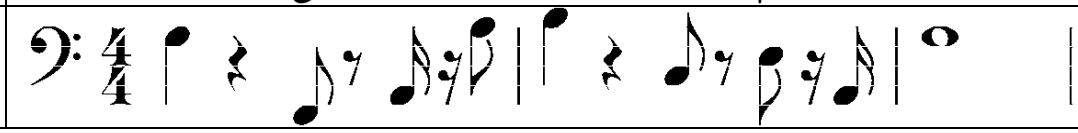
experimento, como visto na imagem de saída, o protótipo identificou e removeu as linhas horizontais por completo. Visualmente não se encontra segmentos de linhas de pauta.

Quadro 16 - Experimento 2 de remoção de linhas de pauta

Experimento 02	
Entrada	
Saída	

Para o experimento 2 visto no Quadro 16, se apresentou oito formas. Pode-se observar na imagem de saída que o protótipo identificou e removeu por completo as linhas. Para este experimento, também não se encontram segmentos de linhas de pauta.

Quadro 17 - Experimento 3 de remoção de linha de pauta

Experimento 3	
Entrada	
Saída	

Para o experimento 3 utilizou-se vinte e uma formas, sendo: onze formas diferentes. O trecho de partitura inicia com a clave de fá, fórmula de compasso, sete notas ou tempos intercalados, barra de compasso, sete notas ou tempos intercalados, barra de compasso, semibreve e por fim barra de compasso. Pode-se observar na imagem de saída, onde para todas as formas contidas no experimento 3 o protótipo identificou e removeu os segmentos de linha de pauta.

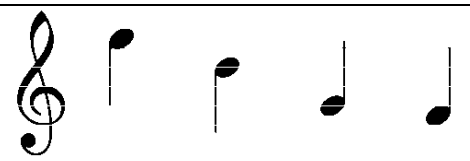
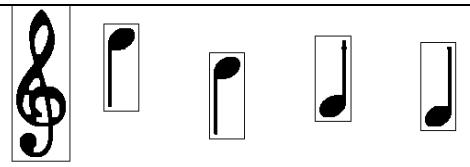
A partir dos experimentos de remoção de linha de partitura executados, o protótipo identificou e removeu completamente as linhas de pauta, sem restar algum segmento de linha, permitindo que este processo seja utilizado nos como base para os processos seguintes. Após o processo de remover as linhas de pauta, inicia-se o processo de segmentação que busca reconstruir os símbolos contidos na imagem.

3.4.2 Segmentação

Este experimento tem como objetivo analisar a etapa de segmentação onde os símbolos da partitura são isolados, possibilitando sua identificação. Para este experimento são selecionadas amostras sem as linhas de pauta, resultados do processo anterior, onde o

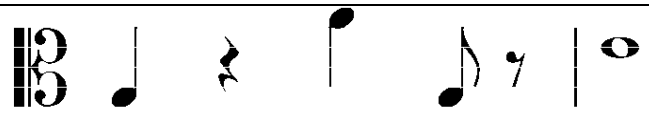
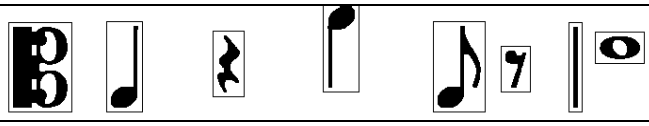
protótipo identifica e demonstra a forma encontrada com uma *bounding box*, ou seja, um caixa de marcação que envolve a forma. Para a aplicação da *bounding box*, foi adicionado uma margem, pois, não se visualizada na barra de compassos. O experimento 1 é visto no Quadro 18.

Quadro 18 – Experimento 1 de segmentação

Experimento 1	
Entrada	
Saída	

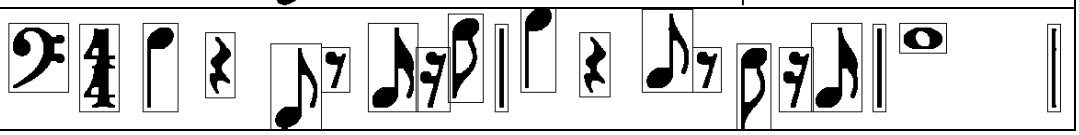
O experimento 1 observado no Quadro 18 dá seqüência ao experimento anterior de retirada das linhas de pauta. São apresentados cinco símbolos com duas formas diferentes, da esquerda para a direita a clave de sol e as restantes semínimas. Para este experimento, como visto na imagem de saída, o protótipo encontrou e marcou as formas encontradas com uma *bounding box*. Visualmente encontram-se cinco símbolos reconstruídos. Segue-se então para o experimento 2 visto no Quadro 19.

Quadro 19 – Experimento 2 de segmentação

Experimento 2	
Entrada	
Saída	

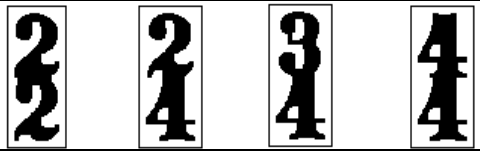
O experimento 2 observado no Quadro 19 apresenta oito símbolos, contendo uma clave de dó, semínimas, colcheia, barra de compasso, semibreve e pausas. Para este experimento, como visto na imagem de saída, o protótipo encontrou e marcou cada uma das formas com uma *bounding box*. Visualmente encontram-se oito símbolos reconstruídos. Seguindo então para o experimento 3 visto no Quadro 20.

Quadro 20 – Experimento 3 de segmentação

Experimento 3	
Entrada	
Saída	

O experimento 3 observado no Quadro 20 é um trecho de partitura maior complexidade e da seqüência ao experimento anterior de retirada das linhas de pauta. São apresentados vinte e um símbolos, contendo uma clave de fá, barras e fórmula de compasso, semibreve, semínimas, colcheias, semicolcheias e pausas. Para este experimento, como visto na imagem de saída, o protótipo encontrou e marcou cada uma das formas com uma *bounding box*. Exceto para o segundo item (fórmula de compasso quatro por quarto), onde para duas formas foi reconstruído apenas um símbolo. Observa-se então, vinte símbolos reconstruídos. Seguindo então para o experimento 4 reconstruindo apenas fórmulas de compasso, visto no Quadro 21.

Quadro 21 – Experimento 4 de segmentação

Experimento 4	
Entrada	
Saída	

O experimento 4 observado no Quadro 21 é um trecho editado para conter apenas exemplos de fórmulas de compasso. São apresentados quatro fórmulas de compasso sendo elas da esquerda para direita "dois por dois", "dois por quatro", "três por quatro" e "quatro por quatro". Para este experimento, como visto na imagem de saída, o protótipo encontrou e marcou com uma *bounding box* cada uma das fórmulas de compasso como formas únicas. Isto ocorre pelo fato das formas estarem muito próximas. Visualmente, cada par de símbolos foi reconstruído em uma forma.

Conforme visto nos experimentos de segmentação, o protótipo reconstruiu em segmentos os símbolos contidos nos trechos de partituras confeccionados. Entretanto, ocorre uma exceção nas fórmulas de compasso pela proximidade dos símbolos. Isto também evidencia que, formas muito próxima também poderão ser fundidas na reconstrução. Isto é

devido ao método de erosão visto anteriormente (3.3.2.2.2 morfologia matemática).

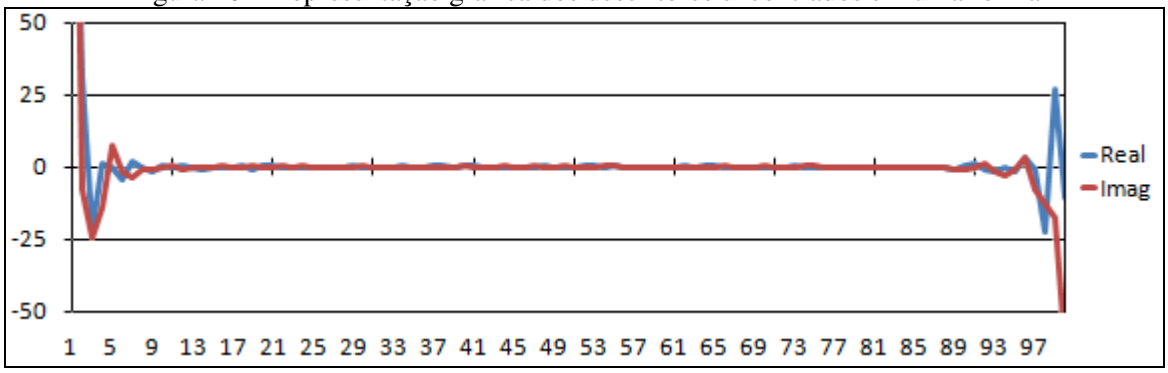
3.4.3 Geração e restauração dos descritores de Fourier

Estes experimentos têm como objetivo analisar a etapa de extração de características das formas encontradas após o processo de segmentação. De cada forma é extraído o contorno e uma lista de coordenadas no plano cartesiano. A lista é o critério para a construção dos descritores de Fourier obtidos pela transformada discreta de Fourier. Os descritores gerados são as justificativas para a comparação entre as diferentes formas.

Inicialmente, busca-se a quantidade ideal de descritores a serem utilizados pelo protótipo. Esta análise requer a geração dos descritores e uma quantidade a ser reconstruída. Para a geração dos valores utilizou-se o protótipo. Para a restauração dos descritores utilizou-se o Matlab, onde se desenvolveu uma melhor visualização do resultado em gráfico. No Matlab foram utilizadas as funções `ifft` e `plot`.

A construção dos descritores de Fourier gera uma lista contendo os coeficientes encontrados. Os valores encontrados a partir de uma colcheia podem ser vistos na Figura 20.

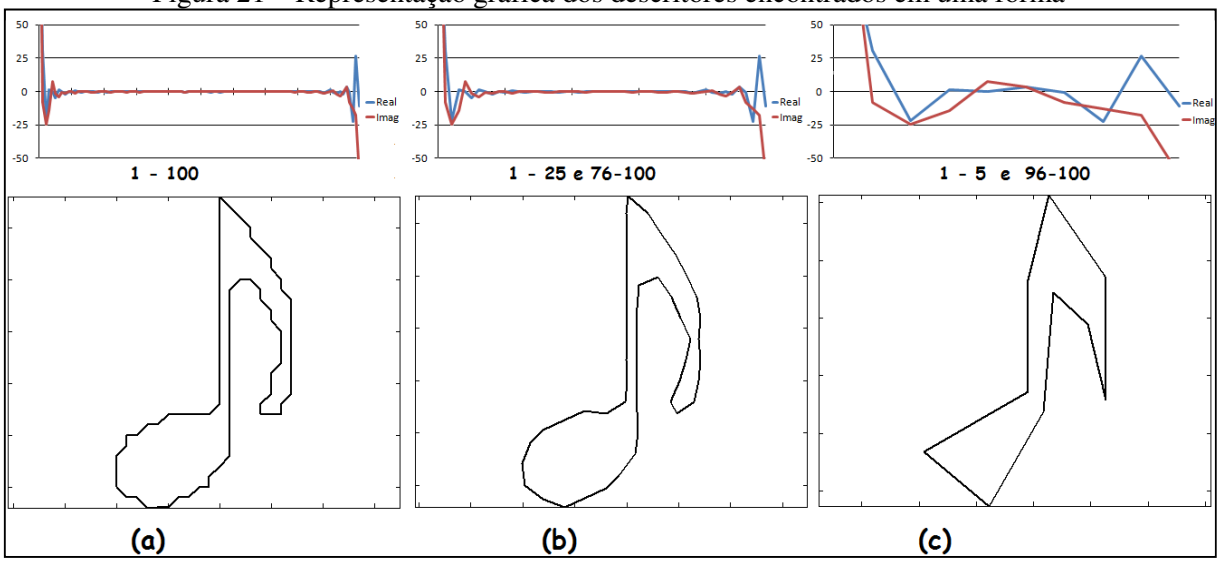
Figura 20 – Representação gráfica dos descritores encontrados em uma forma



A Figura 20 mostra a evolução dos descritores gerados em uma colcheia com cem pontos de contorno. São apresentados dois eixos, o Real e o Imaginário. Nesta, pode-se observar nas extremidades uma maior variação dos valores.

Logo, a reconstrução dos descritores de Fourier gera uma lista com as coordenadas dos pontos em um plano cartesiano. Para isto, se utiliza os descritores mais significativos, ou seja, as extremidades. A reconstrução da forma contida na Figura 20 pode ser visto na Figura 21.

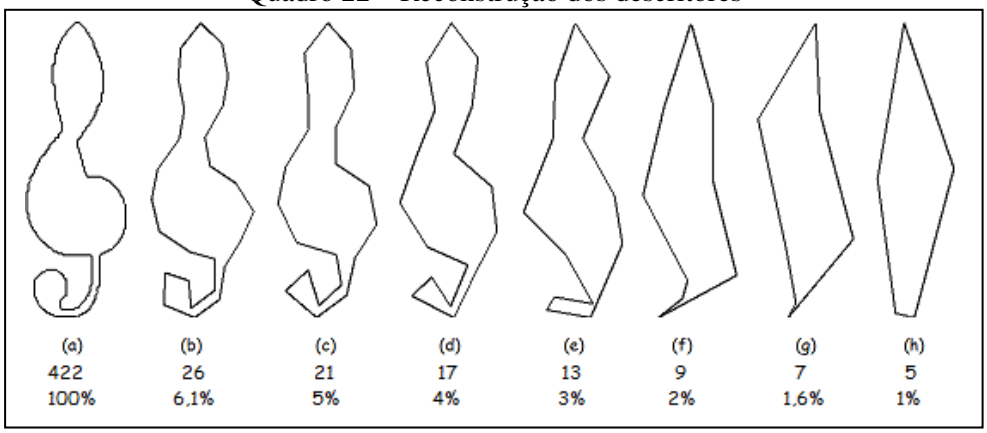
Figura 21 – Representação gráfica dos descritores encontrados em uma forma



Na Figura 21, as figuras reconstruídas são identificadas por: (a) reconstruída com todos os descritores (utilizados do primeiro ao ultimo, "1-100"); (b) reconstruída com cinqüenta descritores (utilizados dois intervalos, de 1 a 25 e 76 a 100); (c) reconstruída com dez descritores (utilizando dois intervalos, de 1 a 5 e 96 a 100). Os intervalos centrais são apenas desconsiderados. Através deste experimento se pode perceber que os descritores mais significativos permanecem nas extremidades.

Entretanto, busca-se uma quantidade ideal para esta reconstrução. Para o experimento 2, observam-se com mais detalhes a restauração dos descritores conforme visto no Quadro 22.

Quadro 22 – Reconstrução dos descritores



O Quadro 22 demonstra a reconstrução de uma clave de sol. A clave de sol utilizada foi extraída do experimento 1 de segmentação. Para cada contorno, atribuiu-se uma letra. Da esquerda para a direita: (a) contém 422 pontos de contorno sendo a forma original; (b) 26 pontos ou 6,1%; (c) 21 pontos ou 5%; (d) 17 pontos ou 4%; (e) 13 pontos ou 3%; (f) 9 pontos ou 2%; (g) 7 pontos ou 1,6%; (h) contém 5 pontos de contorno sendo apenas 1% do contorno

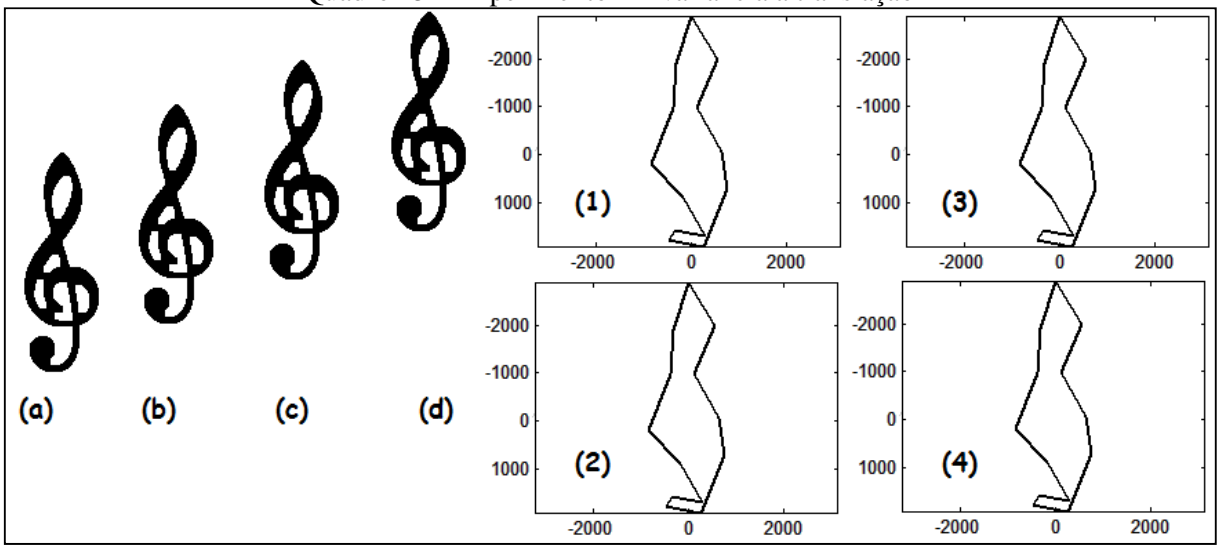
original. As reconstruções acima de 6,1% foram omitidas, por serem muito próximas a (a) ou (b). Através deste experimento observa-se que os descritores estão sendo gerados corretamente. Nesta reconstrução, se busca a quantidade ideal, pois, para comparação são utilizadas quantidades fixas. Visualmente se pode perceber a forma ideal em (e) sendo pouco disforme e suficiente para assemelhar a forma em (a). Deste ponto, para testes futuros, se utiliza a quantidade de 13 descritores para representar uma forma.

3.4.4 Invariância dos descritores

Este experimento tem como objetivo analisar o efeito da aplicação da invariância aos descritores obtidos no processo anterior, a geração. A invariância é necessária para identificar uma mesma forma mesmo contendo características diferentes.

Para o experimento 1, uma mesma forma colocada em posições diferentes de um plano deve gerar o mesmo resultado, caracterizando a invariância a translação. Para obter esta invariância se anula o primeiro descritor e o código fonte pode ser visto no Quadro 13 (seção 3.3.2.3.2 - Cálculo dos descritores de Fourier). O experimento 1 é visto no Quadro 23 e desenvolve-se a invariância a translação.

Quadro 23 – Experimento 1 invariância à translação

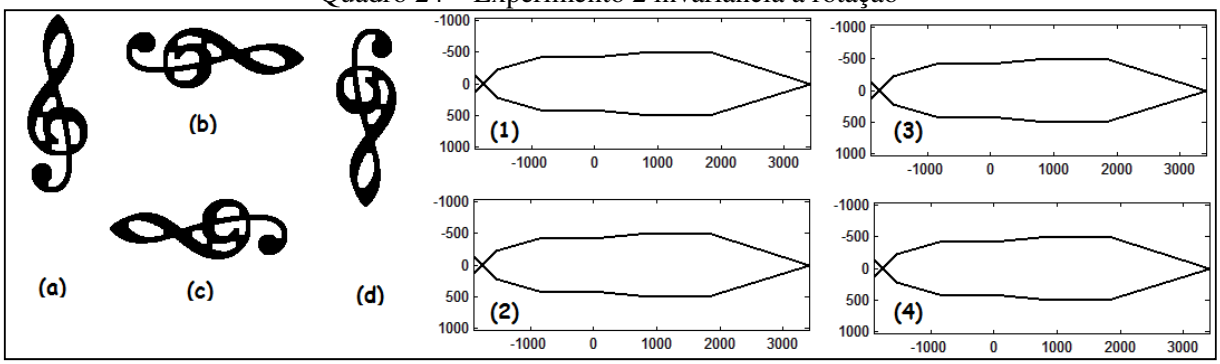


Através do Quadro 23, é possível observar ao lado esquerdo da imagem representa a imagem de entrada. A imagem de entrada possui quatro formas iguais da clave de sol em posições (x,y) diferentes no gráfico, identificadas por: (a), (b), (c) e (d). A clave de sol utilizada foi extraída do experimento 1 de segmentação. Ao lado direito contem a imagem de saída. A imagem de saída é a representação dos quatro resultados sobre a aplicação da invariância a translação, respectivamente identificados na mesma seqüência lógica: (1), (2), (3) e (4). Pode-se observar na imagem de saída o efeito da invariância a translação o centro

(x,y) da forma torna-se (0,0). Através deste experimento se pode ver a correta representação com invariância a translação, sendo que a mesma imagem em posições diferentes formou o mesmo resultado.

Para o experimento 2 acrescenta-se a invariância a rotação. Uma mesma forma colocada em diferentes rotações de seu eixo deve gerar o mesmo resultado caracterizando a invariância a rotação. A fórmula de invariância pode ser vista no Quadro 5 (página 18) e seu código fonte gerado no Quadro 13 (seção 3.3.2.3.2 - Cálculo dos descritores de Fourier). O experimento 2 é visto no Quadro 24.

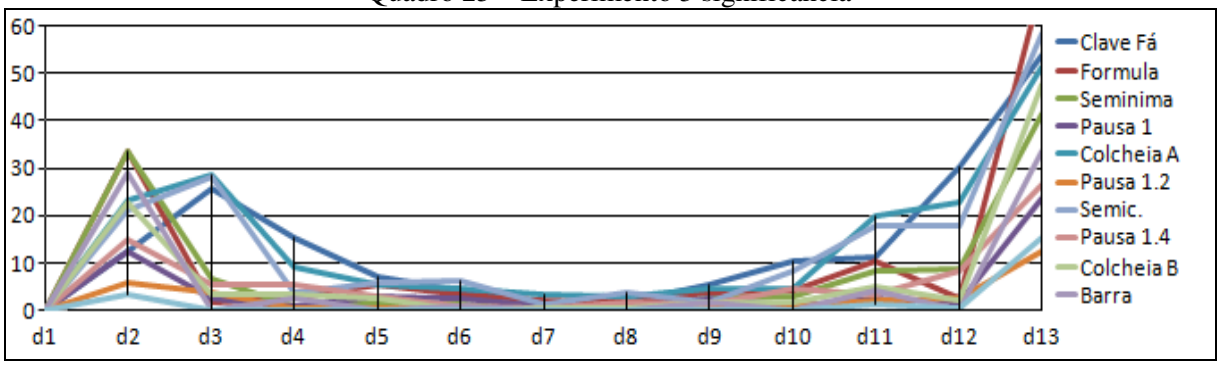
Quadro 24 – Experimento 2 invariância à rotação



Para o experimento 2 visto no Quadro 24, observa-se a ação da invariância a rotação. Ao lado esquerdo do quadro é a imagem de entrada. A imagem de entrada possui quatro formas diferentes de uma clave de sol. As rotações modificam os valores dos contornos encontrados e são identificadas por: (a), (b), (c) e (d). A clave de sol utilizada foi extraída do experimento 1 de segmentação. Ao lado direito exibe as saídas. A saída contém quatro representações do resultado sobre a aplicação da invariância a rotação e translação, sendo respectivamente identificados na mesma seqüência lógica: (1), (2), (3) e (4). Através deste experimento se pode observar a correta representação da aplicação da invariância à rotação, sendo que a mesma imagem em rotações diferentes desenvolveu o mesmo resultado.

Para o experimento 3, se elabora um gráfico que é visto no Quadro 25.

Quadro 25 – Experimento 3 significância

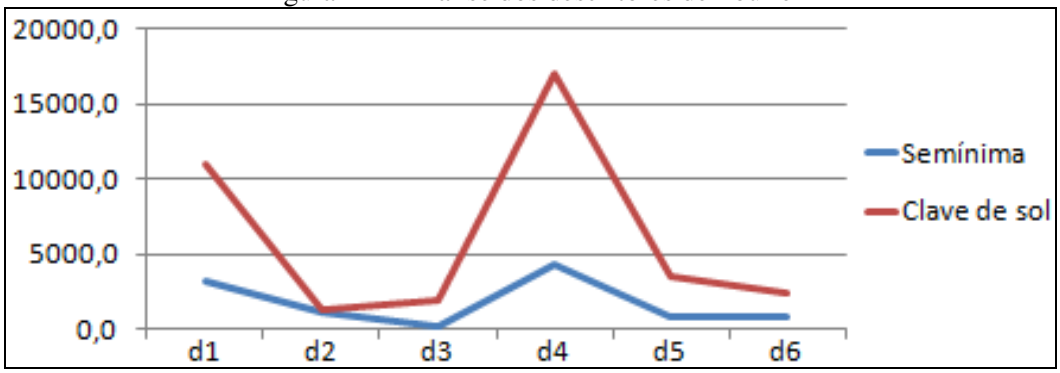


Para o experimento 3 visto no Quadro 25, contém um gráfico com valores dos descritores gerados a partir das formas não repetidas contidas no experimento 3 de segmentação. Cada forma é representada por uma linha. Através deste quadro, se pode visualizar os descritores menos significativos no centro do gráfico. Verifica-se também o primeiro descritor possui zero para todas as formas, isto é devido à invariância a translação. Então, se conclui que é possível utilizar uma quantidade menor de descritores para a etapa de comparação. Assim se identifica como mais significativos seis dos treze visto, sendo eles: d2, d3, d4, d11, d12, d13.

3.4.5 Classificação

Esta etapa identifica as forma através da comparação entre os descritores que foram gerados no processo anterior. Para os experimentos, é apresentado como entrada a imagem segmentada e como saída o próprio resultado gerado pelo protótipo. A saída contém os rótulos (nomes) das formas encontradas. Para os rótulos de saída não se utilizou o uso de acentuações. O resultado apresenta as formas em formato de texto. Os descritores são classificados conforme as regras da árvore de decisão. As regras da árvore de decisão refletem as assinaturas (descritores de Fourier gerados para a forma) das formas utilizadas. As assinaturas contidas no experimento 1 podem ser observadas em forma de gráfico na Figura 22.

Figura 22 – Análise dos descritores de Fourier



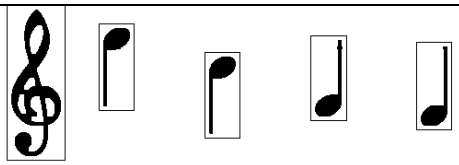
A Figura 22 permite observar a evolução dos valores dos descritores nas duas formas contidas no experimento. Através deste se observa os descritores mais significativos para a distinção das formas em d1 e d4. Se pode, então, criar regras para a árvore de decisão. Estas duas regras criadas para o experimento 1 são vistas no Quadro 26.

Quadro 26 – Regras 01 de classificação

Nome	D1 Entre	D2 Entre	D3 Entre	D4 Entre	D5 Entre	D6 Entre
Clave de sol	10900 e 10910	1360 e 1370	1890 e 1900	17095 e 17100	3515 e 3520	2390 e 2400
Semínima	2730 e 3700	610 e 1770	40 e 400	3540 e 5100	750 e 1000	520 e 1000

O Quadro 26 mostra os valores utilizados para identificar as formas contidas no experimento 1. O experimento 1 pode ser visto no Quadro 27.

Quadro 27 – Experimento 1 de classificação

Experimento 1	
Entrada	
Saída	[Clave de sol, Seminima, Seminima, Seminima, Seminima]

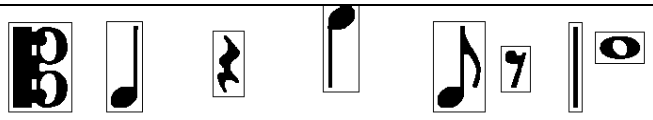
Para o Quadro 27, se pode observar na entrada, a imagem já segmentada. As formas extraídas são então identificadas. Para o experimento 1 obteve-se cinco rótulos. Cada rótulo correspondeu em sua seqüência à forma contida na imagem. Para todas as formas o protótipo identificou e classificou corretamente gerando o resultado esperado. Para o experimento 2, se adiciona seis novas regras, estas são vistas no Quadro 28.

Quadro 28 – Regras 02 de classificação

Nome	D1 Entre	D2 Entre	D3 Entre	D4 Entre	D5 Entre	D6 Entre
Clave de Dó	2100 e 2110	30 e 40	1280 e 1310	10350 e 10370	2400 e 2500	1900 e 2000
Pausa 1/1	1130 e 1250	200 e 350	40 e 110	2200 e 2370	140 e 250	200 e 300
Pausa 1/2	570 e 600	340 e 370	40 e 50	1200 e 1250	250 e 270	200 e 220
Colcheia	1800 e 2450	2730 e 2950	750 e 950	4750 e 5270	2130 e 2270	1860 e 2030
Barra de Compasso	2800 e 3000	0 e 10	250 e 300	3200 e 3400	0 e 10	350 e 450
Semibreve	300 e 350	0 e 30	5 e 30	1500 e 1580	0 e 30	130 e 150

O Quadro 28 contém os valores adicionados as regras da árvore de decisão. O experimento 2 pode ser visto no Quadro 29.

Quadro 29 – Experimento 2 de classificação

Experimento 2	
Entrada	
Saída	[Clave de Do, Seminima, Pausa 1/1, Seminima, Colcheia, Pausa 1/2, Barra Compasso, Semibreve]

O Quadro 29 apresenta o experimento 2. Para este, se obteve oito rótulos. Para todas as formas encontradas o protótipo identificou e classificou corretamente. As regras adicionadas não interferiram nas anteriores.

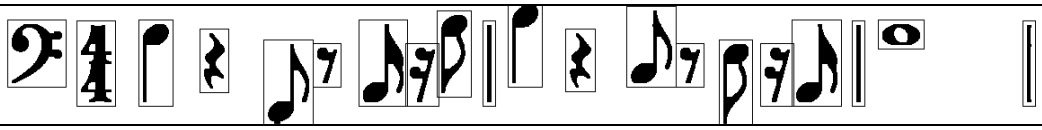
Para o experimento 3 se adiciona cinco novas regras, que podem ser vistas no Quadro 30.

Quadro 30 – Regras 03 de classificação

Nome	D1 Entre	D2 Entre	D3 Entre	D4 Entre	D5 Entre	D6 Entre
Clave de Fá	1250 e 1260	2530 e 2570	1530 e 1550	5350 e 5400	3020 e 3030	1120 e 1140
Formula 44	3300 e 3350	150 e 190	220 e 290	6820 e 6920	250 e 290	1000 e 1100
Semicolcheia	1820 e 2120	2680 e 2840	290 e 370	5550 e 5850	1650 e 1810	1660 e 1800
Colcheia (alta)	2290 e 2425	305 e 350	315 e 340	4725 e 4825	165 e 270	500 e 535
Pausa 1/4	1450 e 1515	500 e 540	530 e 570	2630 e 2700	810 e 860	290 e 330

O Quadro 30 exibe as regras e valores adicionados para execução do experimento 3. O experimento 3 pode ser visto no Quadro 31.

Quadro 31 – Experimento 3 de classificação

Experimento 3	
Entrada	
Saída	[Clave de fa, Formula 44, Seminima, Pausa 1/1, Colcheia, Pausa 1/2, Semicolcheia, Pausa 1/4, Colcheia, Barra Compasso, Seminima, Pausa 1/1, Colcheia, Pausa 1/2, Colcheia, Pausa 1/4, Semicolcheia, Barra Compasso, Semibreve, Barra Compasso]

O Quadro 31 exibe o experimento 3 de classificação. Para este experimento são apresentadas vinte formas. Para todas as formas o protótipo identificou e classificou corretamente.

Para os próximos experimentos, foram confeccionados novos trechos de partituras. Apresenta-se então: entrada, segmentação e saída. Caso a entrada seja extensa, será fracionada para melhor visualização.

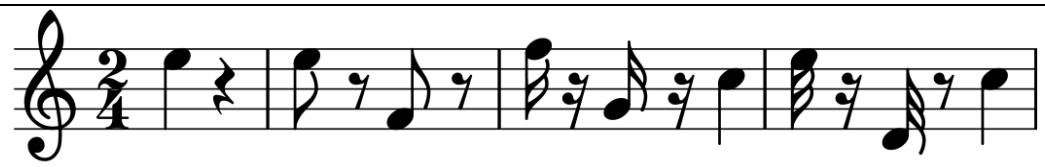
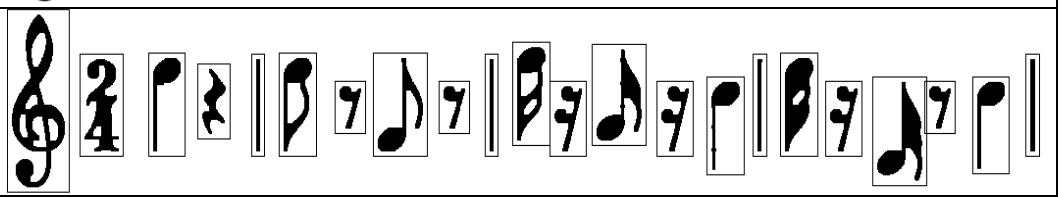
Para o próximo experimento se adiciona outras quatro regras. Estas regras podem ser vistas no Quadro 32.

Quadro 32 – Regras 04 de classificação

Nome	D1 Entre	D2 Entre	D3 Entre	D4 Entre	D5 Entre	D6 Entre
Formula 24	4200 e 4230	200 e 250	450 e 500	7500 e 7550	920 e 950	700 e 720
Semicolcheia	2270 e 2330	305 e 340	280 e 300	4800 e 4905	215 e 270	530 e 555
Fusa (Cabeça alta)	2265 e 2380	305 e 350	260 e 390	4715 e 4875	225 e 250	525 e 550
Fusa (Cabeça baixa)	2360 e 2370	2520 e 2530	550 e 560	6550 e 6570	1180 e 1190	1040 e 1050

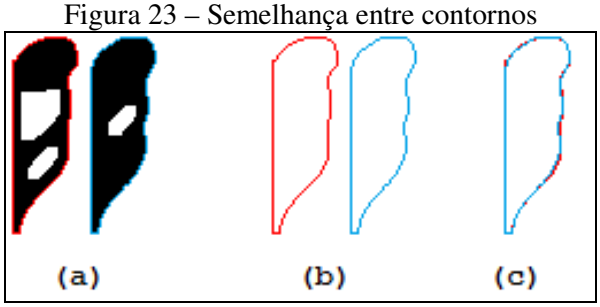
O Quadro 32 exibe as regras adicionadas para execução do experimento 4. O experimento 4 pode ser visto no Quadro 33.

Quadro 33 – Experimento 4 de classificação

Experimento 4	
Entrada	
Segmentado	
Saída	[Clave de sol, Formula 24, Seminima, Pausa 1/1, Barra Compasso, Colcheia, Pausa 1/2, Colcheia, Pausa 1/2, Barra Compasso, Fusa, Pausa 1/4, Semicolcheia, Pausa 1/4, Seminima, Barra Compasso, Semicolcheia, Pausa 1/4, Fusa, Pausa 1/2, Seminima, Barra Compasso]

O Quadro 33 exibe o experimento 4 de classificação. Para este experimento são apresentadas vinte e duas formas. Para duas formas o protótipo identificou incorretamente. A décima primeira nota que é uma semicolcheia e foi reconhecida como fusa, também a décima sétima nota que é uma fusa e foi reconhecida como semicolcheia. Estas formas foram rotuladas incorretamente. Este evento, em especial, é mais bem visto na Figura 23.

Figura 23 – Semelhança entre contornos



A Figura 23 demonstra o contorno de duas notas. No primeiro momento em (a), se pode ver: uma semicolcheia e uma fusa. As duas notas possuem a haste para baixo. No segundo momento apenas os contornos (b). No terceiro momento em (c), se pode observar a

sobreposição dos contornos. Através desta figura se pode perceber que apenas a descrição da forma não é suficiente para classificá-las, se necessita então, outra característica. O protótipo, portanto, gera um resultado incorreto. Este evento não ocorre caso a haste esteja para cima. Para todas as outras formas o protótipo identificou e classificou corretamente.

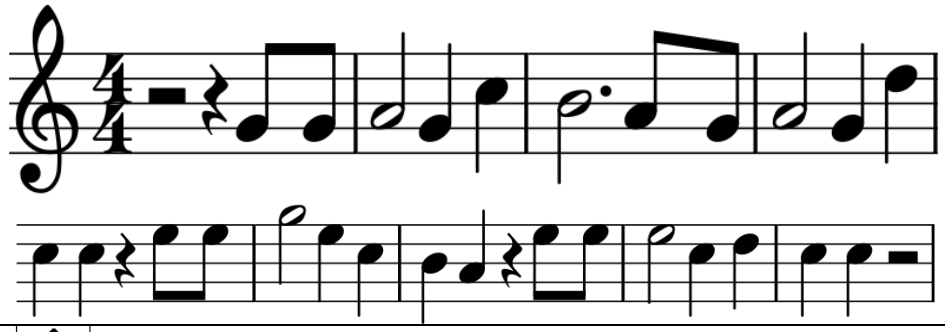
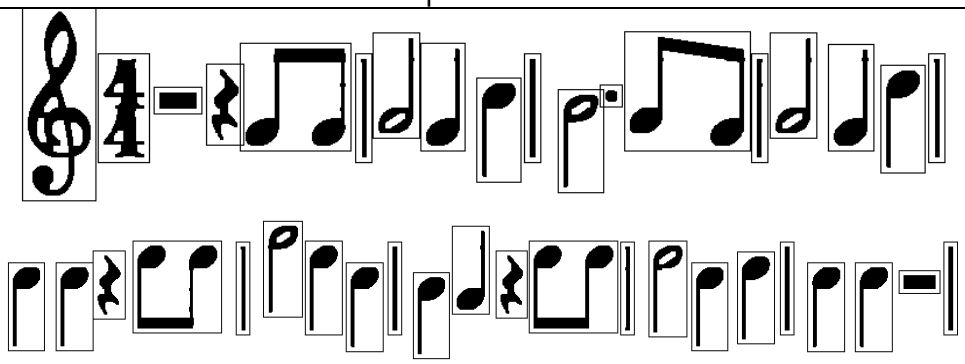
Para o próximo experimento se utiliza uma partitura da música "Parabéns pra Você". Foram adicionadas quatro novas regras, que podem ser vistam no Quadro 34.

Quadro 34 – Regras 05 de classificação

Nome	D1 Entre	D2 Entre	D3 Entre	D4 Entre	D5 Entre	D6 Entre
Pausa 2/1	350 e 390	0 e 5	50 e 80	1050 e 1100	0 e 5	100 e 110
Minima	2900 e 3590	700 e 770	50 e 100	3800 e 4450	850 e 950	750 e 900
Ponto	0 e 3	0 e 5	0 e 10	100 e 150	0 e 10	0 e 5

O Quadro 34 exibe as regras adicionadas para execução do experimento 5, este pode ser visto no Quadro 35.

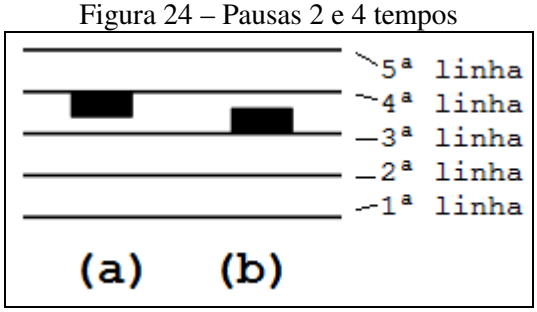
Quadro 35 – Experimento 5 de classificação

Experimento 5	
Entrada	
Segmentado	
Saída	[Clave de sol, Formula 44, Pausa 2/1, Pausa 1/1, Abreviacao 2s, Barra Compasso, Seminima, Seminima, Seminima, Barra Compasso, Seminima, Ponto, Abreviacao 2s, Barra Compasso, Seminima, Seminima, Seminima, Barra Compasso, Seminima, Seminima, Pausa 1/1, Abreviacao 2s, Barra Compasso, Seminima, Seminima, Seminima, Barra Compasso, Seminima, Seminima, Pausa 1/1, Abreviacao 2s, Barra Compasso, Seminima, Seminima, Seminima, Barra Compasso, Seminima, Seminima, Pausa 2/1, Barra Compasso]

O Quadro 35 exibe o experimento 5. Neste se apresentam trinta e nove formas. Para todas as cinco mínimas, o protótipo identificou como semínimas incorretamente. Entretanto se pode perceber que o contorno de uma mínima e de uma semínima são os mesmos, a diferença está na cabeça da nota, onde na mínima é oca. Este é um segundo exemplo onde apenas a forma não é suficiente para identificar corretamente duas formas em específico. Para todas as outras formas, o protótipo gerou o resultado correto.

Um terceiro exemplo é a pausa de 2 e 4 tempos que podem ser vistas na Figura 24.

Figura 24 – Pausas 2 e 4 tempos



A Figura 24 apresenta a pausa de 4 tempos (pausa 4/1), identificada por (a) e a pausa de 2 tempos (pausa 2/1), identificada por (b). Seus contornos são idênticos. As duas formas se posicionam entre a terceira e a quarta linha. A pausa de 4 tempos (pausa 4/1) identificada por (a) toca a quarta linha. A pausa de 2 tempos (pausa 2/1) identificada por (b) toca a terceira linha. Pode-se encontrar a forma, entretanto o protótipo não as distingue.

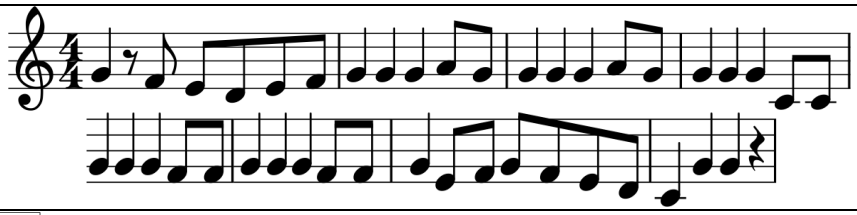
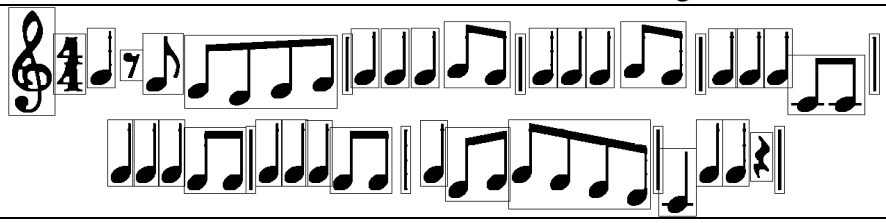
Para o experimento 6 se utiliza uma partitura da música "Atirei o pau no gato". Foram adicionadas uma nova regra, que podem ser vistam no Quadro 36.

Quadro 36 – Regras 06 de classificação

Nome	D1 Entre	D2 Entre	D3 Entre	D4 Entre	D5 Entre	D6 Entre
Abreviação (4 semínimas)	32900 e 33500	9100 e 12500	1200 e 3550	45900 e 50000	22000 e 23000	12600 e 13000

O quadro Quadro 36 exibe as regras adicionadas para execução do experimento 5 que pode ser visto no Quadro 37.

Quadro 37 – Experimento 6 de classificação

Experimento 6	
Entrada	
Segmentado	
Saída	[Clave de sol, Formula 44, Seminima, Pausa 1/2, Colcheia, Abreviacao 4s, Barra Compasso, Seminima, Seminima, Seminima, Abreviacao 2s, Barra Compasso, Seminima, Seminima, Seminima, Abreviacao 2s, Barra Compasso, Seminima, Seminima, Seminima, Abreviacao 2s, Barra Compasso, Seminima, Seminima, Seminima, Abreviacao 2s, Barra Compasso, Seminima, Abreviacao 2s, Abreviacao 4s, Barra Compasso, Seminima, Seminima, Seminima, Pausa 1/1, Barra Compasso]

O Quadro 37 exibe o experimento 6. Neste, se apresentam quarenta e uma formas. Para todas as formas, o protótipo identificou e classificou corretamente. Observa-se neste experimento o uso de formas mais complexas como a abreviação de quatro semínimas.

Para os experimentos realizados, se pode perceber que não ocorrem notas parcialmente classificadas. As notas classificadas corretamente não possuem o contorno similar. Uma tabela contendo as formas onde ocorre o acerto na classificação vista no Quadro 38.

Quadro 38 – Tabela de formas classificadas

Abreviação 2s	100%
Abreviação 4s	100%
Barra de compasso	100%
Clave de dó	100%
Clave de sol	100%
Clave de fá	100%
Colcheia Cabeça para Cima	100%
Colcheia Cabeça para Baixo	100%
Formulas de compasso	100%
Fusa Cabeça para Cima	**0%**
Fusa Cabeça para Baixo	100%
Mínima	**0%**
Pausa 4 tempos	**0%**
Pausa 2 tempos	**0%**
Pausa 1 tempo	100%
Pausa 1/2 tempo	100%
Pausa 1/4 tempo	100%
Ponto	100%
Semibreve	100%
Semicolcheia Cabeça para Cima	**0%**
Semicolcheia Cabeça para Baixo	100%
Semínima	100%

O Quadro 38 apresenta as diferentes formas utilizadas nos experimentos e sua classificação. Através dela, se pode perceber que, das vinte e duas formas utilizadas, 77% puderam ser classificadas apenas com descritores de Fourier. Entretanto a construção de características como preenchimento e posição na pauta, agregarão fatores de comparação. Através disso, se podem melhorar os resultados.

3.4.6 Outras discussões

A documentação de suporte ao JavaCV é muito escassa. Informações como requerimentos para instalação e exemplos são obtidas na página eletrônica do JavaCV (JAVACV, 2012). Enquanto referências de funções são encontradas na página de referências do OpenCV. Nota-se, em menor número, diferenças de assinaturas entre a página de referência e a implementação prática, as referências misturam-se a referências de outras linguagens como Python, C e C++. Ocorre, também, o uso de classes como a `vector`, usada para representar vetor de três dimensões. No entanto, é uma classe antiga criada nas primeiras versões do Java, atualmente obsoleta. Entende-se que há pequenos problemas e algumas funções ainda não mapeadas, considerando que o JavaCV é relativamente novo, possui uma comunidade de suporte e desenvolvimento que promove seu amadurecimento constante e contínuo.

A biblioteca OpenCV possui implementação de técnica de rede neural multi-camada perceptron, chamada classe `CvANN_MLP`. Após prévia implementação ocorriam falhas e por falta de documentação foi descontinuado. Possui também implementação à máquina de vetores de suporte, entretanto a taxa de acerto ficou baixa o método “um contra todos” não se mostrou eficiente. Em virtude de tempo e mais conhecimento demandado acabou sendo postergado.

4 CONCLUSÕES

Este trabalho abordou o desenvolvimento de um protótipo para reconhecimento de símbolos musicais em uma imagem de partitura. Devido às diversas formas de apresentação de uma partitura, para utilização no protótipo, limitou-se a partituras sintéticas, sem arranjos.

A remoção das linhas de pauta, apesar de danificar parcialmente o símbolo, compensou, permitindo isolar os símbolos. Métodos de reconhecimento que não praticam esta remoção de pauta encontram dificuldades com símbolos mais simples e menores que podem sumir.

A aplicação de descritores de Fourier para reconhecimento dos símbolos demonstra ser muito eficaz, entretanto, se fazem necessário outras características, como a classificação de *templates*, a fim de verificar o conteúdo da forma. Como visto nos resultados, o protótipo falha com símbolos muito próximos ou unidos. Apesar de ser possível corrigir em etapa futura, a quantidade de possibilidades para estas uniões é incalculável, necessitando um processo auxiliar de separação.

Contudo, os objetivos deste trabalho foram alcançados. Este trabalho também poderá servir de referência para aprimoramento de estudos sobre a análise e reconhecimento de partituras.

4.1 EXTENSÕES

Como sugestão de extensões para o protótipo propõe-se:

a) melhorar o método de remoção das linhas de pauta permitindo a entrada de uma imagem em posição não horizontal;
b) melhorar o método de segmentação para abranger maior quantidade de notas musicais como ligaduras;
c) melhorar o método de classificação utilizando SVM ou Rede Neural;
d) adicionar semântica e reconstrução dos símbolos musicais;
e) permitir a entrada de partituras digitalizadas contendo ruídos;
f) permitir exportar o conteúdo do resultado para formatos de arquivo de notação musical, como o MusicXML.

REFERÊNCIAS BIBLIOGRÁFICAS

BONILHA, Fabiana F. G. **Leitura musical na ponta dos dedos:** caminhos e desafios do ensino de musicografia Braille na perspectiva de alunos e professores. 2006. 226 f. Dissertação (Mestrado em Música) - Instituto de Artes, Universidade Estadual de Campinas, Campinas. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?down=vtls000380211>. Acesso em: 14 jun. 2012.

ERPEN, Luíz R. C. **Reconhecimento de padrões em imagens por descritor de forma.** 2004. 113 f. Dissertação (Mestrado em Ciências da Computação) - Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande do Sul. Porto Alegre. Disponível em: <www.lume.ufrgs.br/bitstream/handle/10183/27662/000766057.pdf?sequence=1>. Acesso em: 15 out. 2012.

FALCÃO, Alexandre X. **Alexandre Falção's home page.** [S.l], [2008]. Disponível em: <http://www.ic.unicamp.br/~afalcao/mo445/>. Acesso em: 30 out. 2012.

GARCIA, Leonardo; FURLAN, Matheus L; RIBEIRO Rafael. A engenharia elétrica auxiliando a comunicação de deficientes. **Revista ciências do ambiente on-line**, [S.l.], v. 2, n. 2, p. 63-68, Ago. 2006. Disponível em: <http://sistemas.ib.unicamp.br/be310/index.php/be310/article/viewFile/55/35>. Acesso em: 14 jun. 2012.

GOMES, Otávio da F. M. **Processamento de análise de imagens aplicados à caracterização automática de materiais**. 2001. 141 f. Dissertação (Mestrado em Ciências da Engenharia Metalúrgica) - Departamento de Ciência de Materiais e Metalurgia. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro. Disponível em: <http://www.dcmm.puc-rio.br/cursos/ipdi/html/ogomesmestrado.pdf>. Acesso em: 10 out. 2012.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais.** 3. ed. São Paulo: Edgard Blucher, 2002.

JAVACV. **Java interface to OpenCV and more**. [S.l.], [2012?]. Disponível em: <http://code.google.com/p/javacv/>. Acesso em: 07 jul. 2012.

MARGARIDA, Thiago. **Reconhecimento de imagens aplicado a partituras musicais**. 2009. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, Joinville. Disponível em: <http://www.pergamum.udesc.br/dados-bu/000000/00000000000D/00000D67.pdf>. Acesso em: 12 maio 2012.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento digital de imagens**. Rio de Janeiro: Brasport, 1999. Disponível em: <http://www.pessoal.utfpr.edu.br/hvieir/download/pdi99.pdf>. Acesso em: 12 maio 2012.

MED, Bohumil. **Teoria da música.** 3. ed. Brasília: Musimed, 1996.

NOBRE, Jorge. **Apostila de teoria musical**. [s.n], 2008. Disponível em: <http://www2.secult.ce.gov.br/Recursos/PublicWebBanco/Partituraacervo/Apt000002.pdf>. Acesso em: 01 nov. 2012.

OLIVEIRA, Andre L. G. de; MERTZIG, Patrícia, SCHULZ, Sabrina L. In: V REUNIÓN ANUAL DE SACCOM, 5., 2006, Corrientes. **Cognição...** Corrientes: Sociedad Argentina para las Ciencias Cognitivas de la Música, 2006. 45-54. Disponível em: <http://www.saccom.org.ar/2006_reunion5/actas/06.pdf>. Acesso em: 21 abr. 2012.

OPENCV. **Open source computer vision library**. [S.l.], [2012?]. Disponível em: <http://opencv.org/>. Acesso em: 07 jul. 2012.

PERROTTI, Francisco A. **Pré-processamento e nível médio de classificação para um sistema de leitura automática de símbolos musicais**. 1993. 76 f. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia de Computação e Automação Industrial, Universidade Estadual de Campinas, Campinas. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000076194>. Acesso em: 21 abr. 2012.

SEUFERT, Andreas D. M. **Investigação e aperfeiçoamento de algoritmos de reconhecimento de símbolos musicais**. 2010. 55 f. Dissertação (Mestrado Integrado em Engenharia Eletrotécnica e de Computadores Major Telecomunicações) – Faculdade de Engenharia, Universidade do Porto, Porto. Disponível em: <http://www.inescporto.pt/~jsc/students/2010AndreasSeufert/2010relatorioAndreasSeufert.pdf>. Acesso em: 21 abr. 2012.

STIVANELLO, Maurício E. **Inspeção industrial através de visão computacional**. 2004. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

ZUBEN, Fernando J. V. **Árvores de decisão**. [Campinas], [2012?]. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ea072_2s11/topico6_EA072_2s11.pdf>. Acesso em: 10 nov. 2012.