Faculdade de Engenharia da Universidade do Porto

Reconhecimento de símbolos musicais em imagens cinza de partituras manuscritas

André Castro

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
Major Automação

Orientador: Dra. Ana Rebelo (INESC Porto)
Co-orientador: Prof. Jaime Cardoso (FEUP)

1 de setembro de 2014

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Reconhecimento de Símbolos Musicais em Imagens Cinza de Partituras Manuscritas"

foi aprovada em provas realizadas em 10-10-2014

o júri

Presidente Professora Doutora Maria Teresa Magalhães da Silva Pinto de Andrade
Professora Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Maria Benedita Campos Neves Malheiro
Professora Adjunto do Departamento de Engenharia Eletrotécnica do Instituto Superior de Engenharia do Porto

Doutora Ana Maria da Silva Rebelo
Investigadora do INESC - TEC

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - André Filipe Ferreira de Castro

Faculdade de Engenharia da Universidade do Porto

Resumo

Com o decorrer dos anos a degradação de partituras antigas constitui um problema de perda de informação histórica de grande valor. Torna-se então necessário garantir a preservação destes documentos. Este procedimento é dispendioso para ser efetuado manualmente, o que torna necessária existir outra solução para o fazer. Com vista a realizar essa função surgem os sistemas de reconhecimento ótico de música (OMR), que tem como propósito a preservação de partituras. A função deste tipo de sistemas é a transformação de folhas de música num formato legível por um computador. É necessário por isso proceder-se a várias etapas até à tradução de uma imagem com caracteres musicais para uma linguagem passível de ser entendida por uma máquina. Os sistemas OMR normalmente estão divididos em 3 fases: reconhecimento dos símbolos musicais, reconstrução da notação musical e construção da representação final. O trabalho desta dissertação foca-se na primeira destas etapas, o reconhecimento de símbolos musicais. Este primeiro módulo está por sua vez dividido em 3 fases: a deteção e remoção de linhas, a segmentação e o reconhecimento de símbolos musicais. A deteção de linhas deve ser realizada numa fase inicial do processamento, para tornar o processo de reconhecimento dos símbolos musicais possível. Em seguida é necessária a sua remoção sem destruir os símbolos musicais. E por fim, o sistema deve ser capaz de segmentar e identificar os símbolos. Este trabalho foca-se nas etapas de remoção de linhas e de deteção de símbolos. O processamento foi feito utilizando imagens a tons de cinzento, de forma a aproveitar toda a informação presente na imagem que pode ser destruída caso seja binarizada.

De acordo com os resultados obtidos na remoção de linhas da pauta, o algoritmo desenvolvido prova-se eficiente em partituras que apresentem perturbações como manchas de tinta e símbolos quebrados. No entanto a sua eficácia é menor para partituras que apresentem deformações 3D, tais como as existentes em documentos de papel antigo. Os resultados obtidos na segmentação dos símbolos revelam que é possível distinguir com precisão áreas da partitura onde existem símbolos de áreas onde apenas existem manchas de tinta. No entanto, a precisão obtida na segmentação individual dos símbolos é baixa.

Abstract

Over the course of years the degradation of ancient music scores is a problem that leads to the loss of valuable historical information. So it becomes necessary to assure the preservation of these documents. This procedure is too expensive to be done manually, which makes necessary the existence of another solution to do it. In order to perform that function the Optical Music Recognition systems arise, with the purpose of preserving the musical scores. The function of these kind of systems is the transformation of musical sheets into a format readable by a computer. For that it is necessary to proceed through several stages until the translation of an image with musical characters to a language able to be understood by a machine. The OMR systems are usually divided in 3 stages: recognition of musical symbols, reconstruction of musical notation and construction of the final representation. The task of this dissertation is focused on the first of these stages, the recognition of musical symbols. This first stage is divided in 3 sub-stages: the detection and removal of the staff lines, the segmentation and the recognition of musical symbols. The detection of the staff lines must be performed at an initial stage of the processing, to make the recognition process viable. Then it is necessary to remove them without destroying the musical symbols. At last, the system must be able to segment and identify the symbols. This dissertation will focus on the stages of removal of the staff lines and the detection of the symbols. The processing was done using grayscale images, in order to take advantage of all the information present in the image that can be destroyed in case it is binarized.

According to the results obtained in the removal of the staff lines, the developed algorithm proves itself efficient in musical scores that show disturbances such as ink stains and broken symbols. However it's less effective for musical scores that show 3D disturbances such as those existing in old paper documents. The results obtained in the symbol segmentation reveal that it is possible to distinguish, quite accurately areas of the musical score where there are symbols from areas where there are only ink stains. However, the accuracy obtained in the segmentation of the individual symbols is low.

Agradecimentos

À Mariana, por todo o apoio e paciência disponibilizados sem os quais não teria terminado o curso.

À minha família, por me terem dado a oportunidade de poder fazer um curso superior e por todo o apoio dado ao longo dele.

Aos meus Orientadores de Dissertação, pela orientação dada, que foi fundamental para a conclusão deste trabalho.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e Símbolos

Lista de abreviaturas:

BLIST	*Binarization based in LIne Spacing and Thickness*
MLP	*Multi-Layer Perceptron*
OMR	*Optical Music Recognition*
PNN	*Probabilistic Neural Network* – Rede Neuronal Probabilística
RLE	*Run-Length Encoding*
SVM	*Support Vector Machines* – Máquinas de Vectores de Suporte
XOR	*Exclusive Or* – Ou Exclusivo
PCA	*Principal Components Analysis* – Análise dos Componentes Principais
IDE	*Integrated Development Environment* – Ambiente Integrado de Desenvolvimento
ICDAR	*International Conference on Document Analysis and Recognition*
GREC	*Graphics Recognition*
INESC	Instituto de Engenharia de Sistemas e Computadores
INESC TEC	Instituto de Engenharia de Sistemas e Computadores - Tecnologia e Ciência
CVC-MUSCIMA	*Computer Vision Center - Music Score Images*

Capítulo 1

Introdução

A degradação de documentos antigos devido à idade é um problema que se pode traduzir em perda de informação histórica de grande valor. Para se evitar tal perda a digitalização desses documentos é a solução mais viável. Contudo esse processo apenas permite que seja guardada a informação num formato digital de imagem, não sendo feito o reconhecimento dos caracteres presentes no documento. O processo de preservação é também demorado e dispendioso se este for efetuado manualmente. Com o propósito de colmatar essas falhas tornou-se necessária a existência de sistemas que efetuassem esse reconhecimento de modo a que toda a informação presente no documento seja preservada.

1.1 – Sistemas OMR

Os sistemas OMR surgem para transformar de folhas de música para um formato legível por um computador.

Estes sistemas estão habitualmente divididos em 3 etapas: reconhecimento dos símbolos musicais, reconstrução da notação musical e construção da representação final [20]. O trabalho desenvolvido nesta dissertação apenas se irá focar na primeira destas etapas, o reconhecimento de símbolos musicais. Esta é dividida normalmente em 3 fases: a deteção e remoção de linhas, a segmentação de símbolos e o reconhecimento destes [20]. Na fase inicial de processamento, deve ser realizada a deteção de linhas de modo a que a interpretação dos símbolos musicais detetados numa fase subsequente seja corretamente efetuada. Posteriormente é necessária a remoção das linhas sem destruir os símbolos musicais, o que pode comprometer a sua identificação se não for efetuado corretamente. Em seguida deve ser feita a segmentação e identificação dos símbolos musicais. A Figura 1 ilustra de forma simples o descrito[20]:

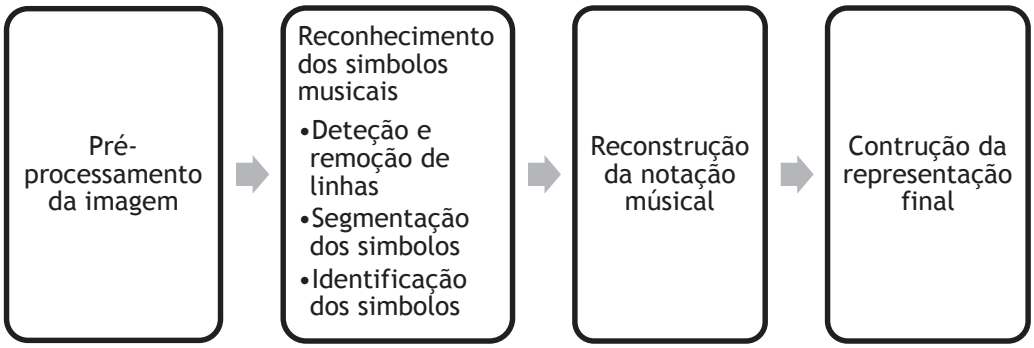

Figura 1 - Diagrama de um sistema OMR

1.2 - Objetivos

Esta dissertação vai focar-se nas etapas de remoção de linhas e de segmentação dos símbolos, no domínio das imagens a tons de cinzento, embora também seja abordada a etapa de deteção de linhas, aproveitando algoritmos desenvolvidos para esta fase (como os presentes em [21],[4] ,[8], [17] e em [31]). Nestas etapas é comum efetuar-se uma binarização das imagens quando se efetua a deteção das linhas, isto é, transformar as imagens a tons de cinzento em imagens com pixels apenas pretos e brancos representados pelos valores 0 e 1 respetivamente. Esta operação facilita em parte o processamento das imagens, mas destrói no processo muita informação a favor disso e pode criar ruído nas imagens que degradarão resultados posteriores. Um problema dos sistemas OMR é a qualidade das imagens e o estado de degradação dos documentos. Como consequência, as imagens binárias resultantes poderão não ser suficientemente satisfatórias. Pretendo então nesta dissertação realizar o processamento em imagens a tons de cinzento com vista a obter melhorias nos resultados face aos problemas descritos.

Assim os objetivos desta dissertação serão os seguintes:

- Deteção de linhas de uma partitura;
- Remoção das linhas de pauta;
- Segmentação dos símbolos;

Pretendo então, com esta dissertação, dar as seguintes contribuições para a comunidade científica e preservação cultural:

- Introduzir um novo algoritmo de remoção de linhas de partituras em imagens em cinzento;
- Introduzir um novo algoritmo para segmentar e classificar os símbolos musicais em imagens em cinzento.

- Novos estudos e análises de métodos de deteção de objetos aplicados a partituras de imagens em cinzento.

1.3 - Organização

O presente documento abordará os seguintes conceitos e estará organizado da seguinte forma:

1. Introdução.
2. Revisão de conceitos teóricos revelantes.
3. Revisão do estado de arte.
4. Ferramentas utilizadas.
5. Implementação
6. Resultados Obtidos
7. Conclusões e Trabalho Futuro

Capítulo 2

Conhecimento Teórico

Neste capítulo serão abordados alguns conceitos teóricos com vista a ter um correto entendimento dos métodos observados no estado de arte.

2.1 - Notação Musical

De forma a ter uma ideia concreta do problema a solucionar e dos métodos observados no estado de arte, é necessário primeiro ter algumas noções musicais, tais como quais os símbolos presentes, que formatos tem estes documentos, qual a informação que transmitem, e como se denominam.

Nesta dissertação irá ser usada a notação musical padrão a qual é escrita numa pauta de 5 linhas. O conjunto dessa pauta e dos símbolos musicais chama-se partitura. Um exemplo pode ser observado na Figura 2.

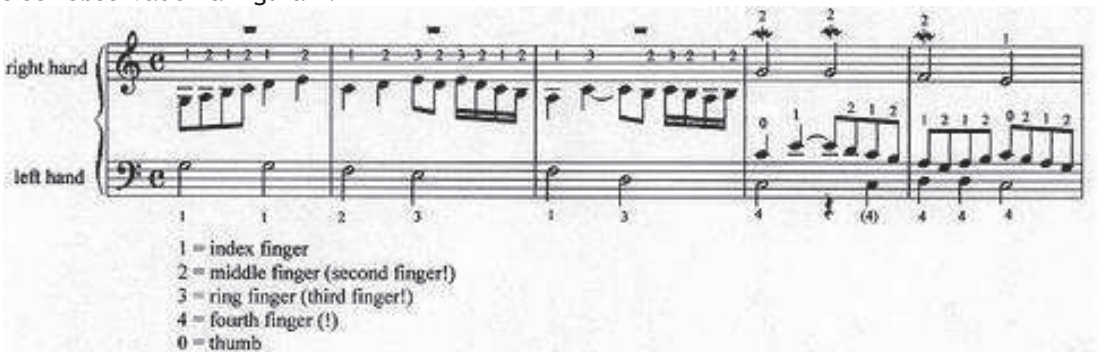

Figura 2 – Exemplo de uma partitura (retirado de [14])

Quanto aos símbolos musicais segue-se uma descrição dos utilizados nesta notação:

- Compassos - são delimitados numa partitura por linhas verticais e determinam o ritmo estrutural da música. Podem ser também representados por uma fração em que o denominador representa em quantas partes deve ser dividida uma semibreve para

obtermos um compasso e o numerador representa o número de unidades temporais desse compasso. Podem ainda ter uma altura que inclua mais do que um conjunto de 5 linhas, tal como no exemplo que pode ser observado na Figura 2.

- Figuras musicais - representam o tempo de duração das notas e encontram-se representadas na Figura 3, com a devida duração em unidades de tempo que representam.

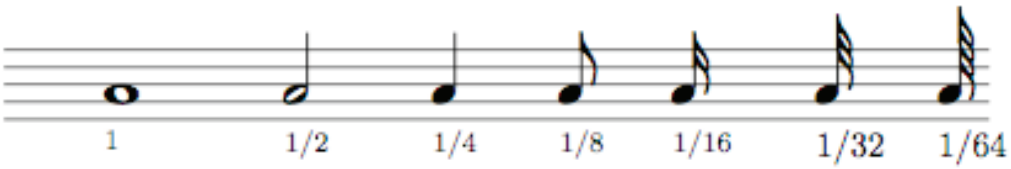

Figura 3 – Figuras musicais (retirado de [14])

Existem também notas seguidas de um ponto o qual acrescenta metade do valor da nota.

- Pausas - representam o tempo de silêncio e encontram-se representadas na Figura 4.

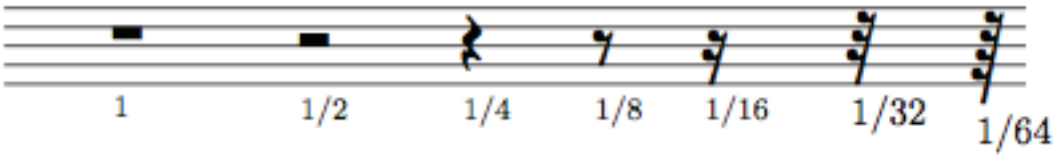

Figura 4 – Pausas (retirado de [14])

- Claves - definem qual a nota que cada sitio da pauta representa. A clave escreve-se sempre do lado esquerdo no início da pauta. Existem a Clave de Sol, Fá e Dó que se encontram representadas na Figura 5 pela ordem descrita:

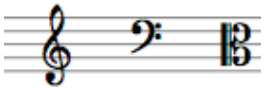

Figura 5 – Claves de Sol, Fá e Dó (retirado de [14])

- Acidentes - representam variações no tom da nota e são colocados antes de uma nota. Existem o sustenido (primeiro à esquerda na Figura 6), o dobrado sustenido (segundo na figura), o bemol (terceiro mais à esquerda na Figura 5) e o dobrado bemol (primeiro á direita na Figura 5). Encontram-se ilustrados na figura seguinte:

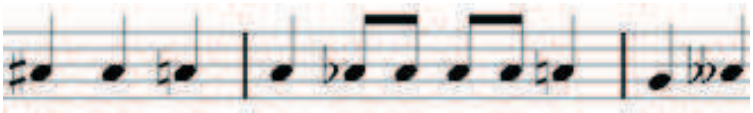

Figura 6 – Acidentes representando diferentes variações de tom (retirado de [14])

- Tonalidade - representam qual a escala da música e são colocados após a clave na pauta. É ilustrado através de um grupo de sustenidos ou bemol, como se pode ver na Figura 6.

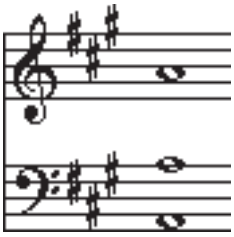

Figura 7 – Tonalidade (retirado de [14])

- Variação de volume - faz-se representar por 2 símbolos o crescendo (representado pelo sinal >) e o diminuindo (representado pelo sinal <) e significam aumento e diminuição de volume, respetivamente. Um exemplo pode ser observado na Figura 8.

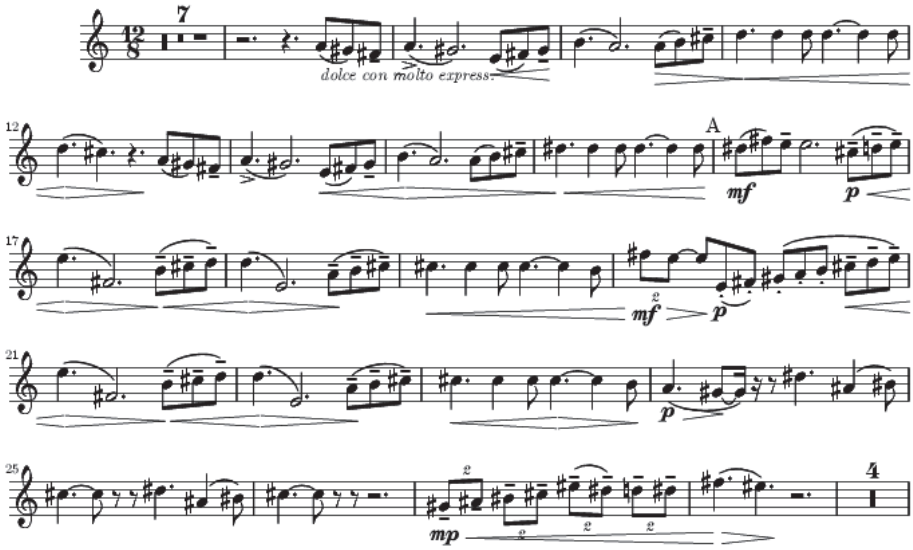

Figura 8 – Variação de Volume (retirado de [14])

2.2 - Binarização

De forma a perceber o que se ganha em evitar a binarização é necessário entender em que consiste e como se pode aplicar para deteção de símbolos. A binarização consiste em encontrar um limiar global, ou vários limiares de binarização para partes da área da imagem,

de modo que o fundo seja separado dos símbolos presentes na imagem, atribuindo aos pixels o valor 0 ou 1 conforme estes estejam abaixo ou acima do limiar [25]. As técnicas para determinar este limiar baseiam-se maioritariamente na análise do histograma da imagem em tons de cinzento [25]. O histograma mostra para cada tom do preto ao branco, normalmente 256 níveis, a quantidade de pixels que existem na imagem com essa intensidade. Assim, para imagens que apresentem apenas 2 grandes picos máximos no histograma,i.e., imagens com o histograma bimodal, encontrar o limiar consiste em encontrar um ponto entre os 2 picos que satisfaça o requerimento, por exemplo, o ponto mínimo. Se as imagens não apresentarem um histograma nessa forma é necessário que sejam utilizados métodos mais complexos de modo que sejam cumpridos os requisitos. Um exemplo de um algoritmo simples para encontrar um limiar automaticamente é o de Otsu [15]. Este algoritmo encontra o limiar, assumindo que a imagem tem um histograma maioritariamente bimodal e encontrando o ponto que minimiza a variância dos 2 máximos detetados. Esta técnica simplifica um pouco o processamento da imagem pois reduz o tamanho dos dados. A grande desvantagem desta técnica é a perda da informação, que antes possuía 256 níveis, para passar apenas a ter 2 valores.

2.3 – Aprendizagem Automática

A identificação dos símbolos é um problema que requer uma solução capaz de tratar um elevado número de diferentes caligrafias e com o facto de os símbolos poderem não estar desenhados na perfeição. Assim, e de modo a proceder à identificação correta dos símbolos, é bastante eficaz recorrer a algoritmos de aprendizagem automática, tal como será demonstrado na secção de identificação de símbolos. Estes algoritmos conseguem lidar com grandes quantidades de dados e detetar padrões automaticamente (um exemplo disto são as redes neuronais artificiais).

2.3.1 Treino

A maior parte destes algoritmos necessita de treino. Os tipos de treino podem ser divididos maioritariamente em 2 classes, supervisionado ou não supervisionado [25]. No treino supervisionado é fornecido um conjunto de dados de entrada e é fornecida qual a saída desejada para aqueles dados. É esperado com este tipo de treino que o conjunto de dados de exemplo caracterize bem as classes de dados que pretendemos identificar, de modo a que as generalize. Isto permitirá que classifique os dados com baixa taxa de erro. No treino não supervisionado, é fornecido um conjunto de dados de entrada e o algoritmo de aprendizagem tenta correlacioná-los de forma a classificá-los em diferentes categorias tendo como base as

caraterísticas dos dados. É importante referir que o treino de um algoritmo de aprendizagem deve realizar-se até que uma determinada condição de erro seja satisfeita ou que o erro tenho convergido para um mínimo global [7].

2.3.2 Redes Neuronais

As redes neuronais são algoritmos que constroem um modelo baseado em ligações de elementos simples [23]. Dos vários tipos de redes neuronais existentes, apenas se analisam as redes do tipo *feed-forward* nas quais a informação se desloca das entradas para a saída. Estas redes são compostas por vários elementos simples, neurónios, formando no mínimo 2 camadas, uma de entrada e uma de saída, embora possam conter camadas escondidas adicionais. Cada neurónio artificial é composto pelos seguintes elementos [23],[7]:

- Entradas, vindas de fora da rede ou de um outro neurónio, e sujeitos a um determinado peso, consoante a entrada;
- Limiar, utilizado como se fosse uma entrada, com o seu determinado peso, mas que provém de um neurónio em que a saída é sempre 1.
- Função de combinação, que calcula a magnitude dos sinais que entram no neurónio artificial, e está exibida na equação 1.1.

$$C_j = \sum_{i=o}^{n} W_{ij} X_i \,, \tag{1.1}$$

  onde C é a saída da função de combinação com n entradas, X_i são os valores de entrada e W_{ji} o seu peso correspondente.
- Função de ativação, que determina o valor da saída do neurónio artificial quando este dispara. Exemplos de função de ativação são o degrau ou a sigmoide. Deste modo a saída do neurónio será a mostrada pela equação 1.2.

$$y_i = \varphi\left(\sum_{i=0}^{n} w_{ij} x_{ij} \right), \tag{1.2}$$

2.3.3 Máquinas de Vetores de Suporte

As Máquinas de Vetores de Suporte (SVM) são um método de classificação que pertencem à classe métodos de núcleo, dependendo dos dados apenas através de funções de núcleo, permitindo assim gerar fronteiras de decisão não lineares através de métodos de classificadores lineares [1]. Um exemplo de uma função de núcleo é dada pela equação 1.3, que representa uma função de base radial,

$$k\left(x\ , x_i\right) = \exp\left(-\gamma \left\| x - x_i \right\|^2\right),\ \gamma \geq 0 \tag{1.3}$$

onde γ é o parâmetro que controla a largura da superfície. Neste algoritmo o objetivo passa pela construção de um hiperplano que atuará como uma superfície de decisão, uma fronteira entre os exemplos positivos e negativos. Essa superfície terá de ser maximizada de modo a

que as classes fiquem o mais separado possível. Assim isto torna-se num problema de maximização da largura da margem. Considerando um vetor *w*, um hiperplano *f* e um conjunto de dados *D*, a margem é definida pela equação 1.4.

$$m_D(f) = \frac{1}{\|w\|}, \tag{1.4}$$

Maximizar a equação 1.4 é equivalente a minimizar $\|w\|^2$. Deste modo, o problema simplifica-se nas seguintes expressões:

$$\text{Minimizar } w \text{ e } b\text{: } \frac{1}{2}\|w\|^2 + C\sum_{i=0}^{n}\xi_i\,, \tag{1.5}$$

$$\text{Sujeito a: } y_i\left(w^T x_i + b\right) \geq 1 - \xi_i\,, \quad \xi_i \geq 0 \tag{1.6}$$

Em que x_i são entradas, y_i são as saídas, ξ_i são variáveis de folga que penalizam os exemplos incorretamente classificados, *b* é a variável de *bias* que desloca o hiperplano da origem, e *C* define a folga que se pode dar na margem de modo a maximizá-la permitindo algumas classificações incorretas.

2.4 – Conclusão

Neste capítulo foram abordados vários temas que irão permitir um correto entendimento dos métodos, técnicas e expressões utilizadas nos capítulos seguintes.

Capítulo 3

Estado da Arte

Neste capítulo serão abordados vários métodos atuais para as várias etapas de um sistema OMR.

3.1 – Deteção de Linhas

Nesta etapa é necessário identificar a posição das linhas posteriormente reconstruir a notação musical porque o tom das notas é indicado através da posição vertical dos símbolos na pauta e o tempo representado horizontalmente. Esta etapa é também importante pois permite recolher informação para caraterizar os símbolos, nomeadamente os valores da espessura das linhas e do espaçamento entre linhas. Permite também, numa fase seguinte, remover ou ignorar as linhas, o que facilitará a identificação de símbolos musicais.

O método usado em [28], para imagens binárias, deteta as linhas através da projeção dos pixels no eixo vertical e verifica que as linhas da pauta geram picos na projeção. Este método, no entanto, revela-se ineficaz para documentos manuscritos devido à distorção das linhas e ao facto de estas não serem retas.

O algoritmo usado em [16], aplicado em imagens em tons de cinzento, efetua a deteção de linhas com base no método de otimização da colónia de formigas. São espalhadas partículas pela imagem que se movem 1 pixel a cada ciclo do algoritmo e deixam um rasto que beneficia as partículas que sigam o mesmo caminho. Esse rasto desaparece ao fim de alguns ciclos. As partículas são atraídas a prosseguirem um caminho em direção aos pixels diferentes do fundo e na mesma direção do seu movimento anterior, pois este movimento corresponderá às partículas tenderem para as linhas ao fim de algum tempo. É depois, ao fim de bastantes ciclos, calculada a projeção das partículas sobre o eixo de vertical, obtendo-se as linhas da pauta. Este método revela uma elevada taxa de deteção de linhas, mas revela-se ineficaz para pautas que não sejam retas.

O método proposto em [27], aplicado em imagens binárias, propõem que se efetue um processo dividido em 3 fases para a deteção. Numa primeira fase é detetado o espaço entre linhas da pauta e a espessura destas, através de varrimentos verticais da imagem e verificando quais as maiores frequências de pixels brancos e pretos. Em seguida, são removidos todos os pixels maiores do que 2 vezes a espessura da linha ou maiores do que o espaço entre linhas mais a espessura da linha, dependendo de qual for o menor dos 2 critérios. Na segunda fase é obtida a orientação das linhas da pauta a partir da imagem da fase anterior. É calculada a orientação de cada pixel, coluna a coluna, verificando se existem pixels nas colunas seguintes com aquela orientação até a uma distância de 5 colunas. A orientação das linhas é depois calculada considerando a soma das orientações de cada pixel e o número de pixels considerados. Caso faltem orientações em algumas colunas, este valor é interpolado a partir das colunas vizinhas. Na terceira fase as linhas são segmentadas utilizando a orientação das linhas da pauta já calculada, se estiverem espaçadas de um valor idêntico ao do espaçamento entre linhas.

O método a ser utilizado para a deteção de linhas nesta dissertação será o *Strong Staff Pixels* (presente em [21]), aplicado a imagens a tons de cinzento. Será utilizada esta técnica pois, obteve bons resultados na competição "*International Conference on Document Analysis and Recognition* (ICDAR) / *Graphics Recognition* (GREC) 2013 *Competition on Music Scores: Staff Removal*"[12] para imagens binárias e pode ser aplicado também a imagens em tons de cinzento, com as quais se pretende trabalhar nas fases seguintes do reconhecimento. Este método baseia-se no conhecimento do espaço entre as linhas e na altura das linhas. Isto é realizado codificando as colunas da imagem utilizando a técnica *run-length encoding* (RLE), e selecionando a soma mais frequente de 2 colunas seguidas como uma estimativa do espaço entre linhas mais a altura da linha. Após esta operação, é efetuado o histograma 2D de modo a obter os valores individuais desses 2 parâmetros. Nas imagens a tons de cinzento isto é feito recorrendo à frequência acumulada das várias imagens geradas através de várias binarizações, desde um limiar inferior até a um superior. Utilizando esta informação e as margens verticais da imagem, é encontrado o caminho mais curto entre as duas margens verticais. Isto é conseguido através de uma função de custo que considera o caminho mais curto pelos pixels de menor valor (tipicamente o 0 representa os pixels a preto e o 255 a branco), penalizando os pixels pretos que se encontram nas margens superiores das linhas e que fazem parte de uma coluna com um elevado número de pixels pretos consecutivos e a mais de 2 vezes a altura da linha pois têm uma elevada probabilidade de pertencer a um símbolo. Assim, a função de custo para uma vizinhança de 4 pixels será a descrita na equação 1.7.

$$w(p) = b/(1+\exp(-(p-\alpha)/\beta)) + a \quad , \tag{1.7}$$

Para uma vizinhança de 8 pixels basta multiplicar a equação (1.7) por $\sqrt{2}$ .Na equação 1.7 *a* e *b* são fatores de normalização e são ambos iguais a 4, α é o valor do limiar de binarização determinado pelo método BLIST [17] e β relaciona-se inversamente com a curva descrita para sigmoide efetuada pela função de custo, podendo ser ajustado para compensar diferenças de luz ao longo da imagem. A curva descrita pela função de custo é a seguinte:

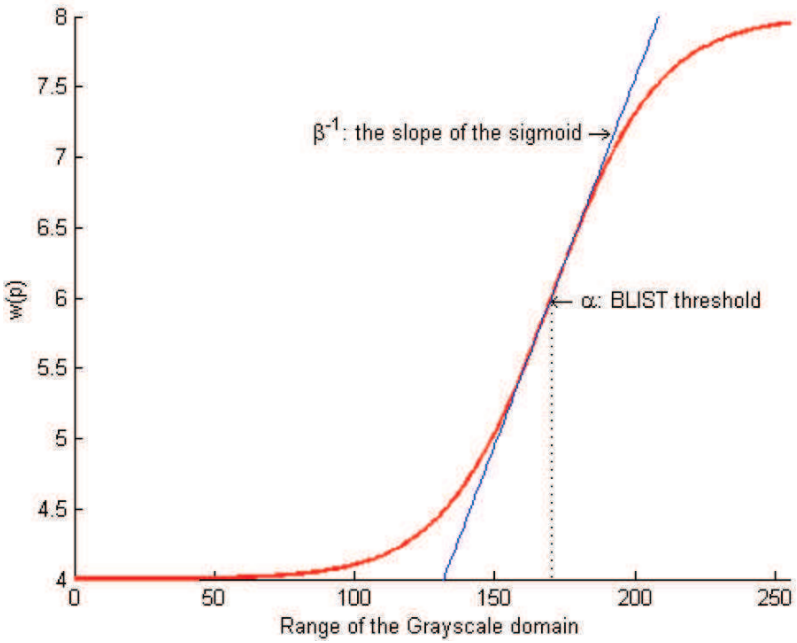

Figura 9 – Curva da função de custo (retirada de [21])

Os resultados obtidos por este método quando aplicado a imagens em tons de cinzento mostram-se mais favoráveis do que quando aplicados a imagens binarizadas, errando menos tanto na falsa deteção como na falha de deteção de linhas.

3.2 – Remoção de Linhas

Nesta etapa é pretendido remover corretamente as linhas sem danificar os símbolos musicais. . É também importante que seja possível aplicar o método escolhido a documentos em que as linhas não são perfeitamente retas, o que se sucede devido a distorção nas imagens.

O método mostrado em [21] e utilizado na competição "ICDAR / GREC 2013 *Competition on Music Scores: Staff Removal*"[12], com o nome de INESC-gray (Instituto de Engenharia de Sistemas e Computadores), propõe que a imagem seja binarizada, utilizando o limiar determinado pelo método BLIST [17], e codificada, utilizando RLE. Esta codificação permite representar a imagem através de uma lista de tamanhos sequenciais de pixels pretos e

brancos. Isto permite encontrar facilmente sequências verticais de pixels com uma espessura determinada.

São então eliminados todos os pixels que estejam numa sequência menor do que 2 vezes a altura estimada da linha. Foi ainda utilizado na mesma competição este método aplicado a imagens binárias e foi considerado o melhor para 1 de 3 subconjuntos de teste.

De um modo idêntico, o método usado em [27], em imagens binarizadas, percorre as colunas da imagem, e remove os pixels que pertençam a sequências de pixels com uma espessura menor do que o menor de dois critérios: 2 vezes a espessura da linha ou a soma da espessura da linha com o espaço entre linhas.

Outro método para imagens binarizadas seria o utilizado em [2] e [6], o qual percorre as linhas, a partir da localização de todos os pixels da linha, e não remove os que tiverem na sua vizinhança pixels pretos, considerando que fazem parte de um símbolo. Este método pode originar a fragmentação ou junção de símbolos pois podem existir símbolos colineares com as linhas e símbolos a serem unidos erradamente por uma linha.

O método LRDE-bin [12], utiliza inicialmente uma operação de morfologia matemática do tipo *hit-or-miss* com padrão de linha horizontal como elemento estruturante. Em seguida, é aplicada uma operação de dilatação com uma vizinhança horizontal de modo a aumentar o tamanho dos componentes conectados. É depois obtida uma máscara binária através da aplicação de uma operação de dilatação seguida de erosão, utilizando um retângulo como elemento estruturante. Por fim, é aplicado um filtro de mediana com uma janela vertical dentro dos maiores componentes da máscara binária, o que resulta na remoção das linhas. Este método foi considerado o melhor para 2 dos 3 conjuntos de teste da competição referida. Foi ainda utilizado em imagens em tons de cinzento (com o nome LRDE-gray) na mesma competição, procedendo-se primeiro à binarização das imagens. Para isso, inicialmente são removidas as margens da imagem através da binarização de Sauvola[24] e é aplicada uma operação morfológica de dilatação com uma vizinhança horizontal. A imagem determinada é então utilizada como máscara num método de binarização com 2 limiares e histerese, que é aplicado sobre a imagem original, resultando na imagem binarizada.

3.3 – Segmentação de Símbolos

A segmentação de símbolos é uma etapa de elevada complexidade devido à grande variedade de símbolos e de caligrafias. Durante esta fase é importante verificar se o símbolo é composto por várias regiões descontínuas, se foi fragmentado ou se foi unido a outro na etapa anterior.

Existem vários métodos propostos como, por exemplo, um algoritmo de crescimento de regiões [2], o qual a partir de pixels pretos detetados tenta definir um região propagando-a segundo uma vizinhança definida. Contudo este não se aplica, visto não ter em consideração símbolos fragmentados e unidos.

A solução apresentada em [20] propõe diferentes tipos de segmentação para várias classes dos símbolos, dividindo-os em 4 classes:

1. Símbolos com segmentos verticais maiores que um limiar, como notas;
2. Símbolos que ligam as notas, como as linhas de união;
3. Os restantes símbolos que estabelecem contacto com as linhas da pauta, como as pausas;
4. Símbolos que se encontram acima ou abaixo das linhas da pauta, como notas e ligaduras.

Estabelecendo estas classes é possível definir métodos concretos de segmentação para cada uma. Para a classe 1 é verificado se existem regiões para além de uma altura predefinida das linhas da pauta. Para a classe 2 é procurada uma região na extremidade das notas que possua uma altura maior que um limiar predefinido. Para a classe 3 foi utilizado um processo semelhante ao usado para a classe 1, mas procuraram-se regiões que tivessem valores semelhantes de altura e largura. Para a classe 4 foram procuradas regiões que tivessem valores, definidos experimentalmente, de altura e largura que os objetos detetados poderiam ter, sendo estes valores diferentes para cada símbolo desta classe. A divisão em classes dos vários tipos de símbolos permite atingir uma menor taxa de símbolos não detetados.

Outra abordagem na segmentação de símbolos, embora neste caso não musicais, é a desenvolvida em [18]. Esta consiste em detetar o retângulo mínimo de cada símbolo e, através da análise das dimensões de cada retângulo, determinar se este contém apenas um símbolo ou vários agregados. No caso de conter vários símbolos é depois efetuado, através de um algoritmo de adelgaçamento, o esqueleto dos símbolos dentro do retângulo. Em seguida é repetido o seguinte processo 6 vezes: a imagem é binarizada e os pontos de junção são encontrados e são guardados por ordem de frequência de ocorrência. No final do processo é escolhido o ponto com maior ocorrência e este é utilizado para separar os símbolos. A imagem é então transladada com o ponto de junção na origem. Seguidamente, são calculados vários vetores, partindo da origem e terminando no pixel seguinte, com o vetor seguinte a começar onde o último terminou. É calculado então o ângulo com o eixo das abcissas para cada vetor. Este processo termina quando o ângulo de 3 vetores for o mesmo, sendo depois realizada uma regressão linear entre a origem e o último vetor detetado, determinando a superfície que separa os 2 caracteres.

3.4 – Identificação de Símbolos

3.4.1 Atributos para Identificação

Partindo agora dos símbolos já segmentados, torna-se necessário classificá-los. Devido a uma elevada complexidade destes símbolos esta tarefa não é simples. É possível classificar os diversos símbolos com recurso a técnicas de reconhecimento de padrões como, por exemplo, *template matching*, a transformada de *Hough*. Também é comum a extração de caraterísticas que podem auxiliar o reconhecimento de padrões, como o centróide, a projeção nos eixos vertical e horizontal ou o número de orifícios na imagem [2; 25]. É possível representar também caracteres através do seu contorno codificado em cadeia para obter uma representação mais reduzida dos dados [19].

3.4.2 Aprendizagem Automática

A utilização de algoritmos de aprendizagem automática permite distinguir com maior facilidade os elementos da imagem que são símbolos, através da extração de características.

O método proposto para identificação de carateres musicais em [29] é a utilização de redes neuronais probabilísticas (PNN). A rede utilizada possui 4 camadas, representando a dimensão dos dados de entrada, os casos de treino, as classes em que classificar cada entrada e a dimensão da saída. As PNN constroem fronteiras não lineares entre os dados fornecidos para treino de uma forma não paramétrica. Estas fronteiras são controladas através de um parâmetro *a*, que varia de 0 a 1, e que modifica o formato das fronteiras. Foi comparado o uso de 2 tipos de entradas neste tipo de redes, imagens contendo os símbolos e as projeções horizontal e vertical dos símbolos. Concluiu-se que a solução que utilizava imagens inteiras produzia resultados ligeiramente melhores em termos de identificação dos símbolos, embora a solução baseada nas projeções seja considerada melhor para uma futura tradução para notação musical.

Outra abordagem é a utilização de redes neuronais *multi-layer perceptron* (MLP) como a utilizada em [20]. Este tipo de rede neuronal pode ter uma ou mais camadas escondidas entre a de entrada e a de saída. O uso de *perceptrons* permite que a saída seja mapeada em 1 ou 0 consoante se verifique ou não uma dada expressão matemática. Esta rede foi treinada com o algoritmo de *back-propagation* conjuntamente com o algoritmo de Levenberg–Marquardt[13], o que permite o erro convergir rapidamente para um mínimo global. Os resultados obtidos revelam que o desempenho deste tipo de rede para este problema fica aquém de outras soluções como, por exemplo, máquinas de vetores de suporte (SVM) como o de [20] com uma

função de base radial (Equação 1.3). Os resultados obtidos neste caso foram bastante superiores aos obtidos com redes neuronais e bastante satisfatórios.

3.5 – Conclusão

Neste capítulo foram abordados vários métodos atuais para as várias etapas de um sistema OMR, sendo discutidas quais as vantagens e desvantagens e também o desempenho dos mesmos.

Capítulo 4

Ferramentas

4.1 - Software

O desenvolvimento será feito nos sistemas operativos Windows 8 e Mac OSX, utilizando o *Integrated Development Environment* (IDE) Visual Studio, o *software* Matlab, uma biblioteca de funções de operações em imagens, OpenCV[3] e ainda uma biblioteca de funções que implementam uma SVM, LibSVM[5]. Estas bibliotecas foram escolhidas por implementarem as funções pretendidas, serem gratuitas para uso académico e comercial e serem recomendadas pelos orientadores.

4.2 – Bases de Dados

O treino do sistema será feito com recurso à base de dados de documentos musicais utilizada na competição "ICDAR / GREC 2013 *Competition on Music Scores: Staff Removal*"[12] que contem 6000 documentos manuscritos de pautas musicais, escritas por vários músicos (exemplo na Figura 10). Estas 6000 imagens estão divididas em 2 conjuntos, 4000 imagens para o conjunto de treino e 2000 imagens para o conjunto de teste. As imagens do conjunto de treino foram geradas a partir de 667 imagens da base de dados *Computer Vision Center – Music Score Images* (CVC-MUSCIMA) e estão divididas em 3 subconjuntos, em que cada um tem um tipo de perturbação diferente. O primeiro e segundo subconjuntos são constituídos por 1000 imagens cada, enquanto o terceiro tem 2000 imagens. Para o primeiro subconjunto foram geradas imagens com distorções 3D, imitando perturbações causadas pela digitalização de documentos unidos na margem e pequenas curvas e dobras, tal como se pode ver nas Figuras 10 e 11. Para o segundo conjunto foram geradas imagens com 3 níveis de ruído local, imitando manchas e falhas de tinta (exemplo na Figura 12) que provocam a

quebra de alguns símbolos. No terceiro subconjunto as imagens geradas possuem várias combinações das distorções apresentadas nos subconjuntos anteriores, apresentando distorções 3D e de ruído local (exemplo na Figura 13). O conjunto de teste está também dividido nos mesmos 3 subconjuntos do que as imagens de treino, com as mesmas deformações, e é composto por 500 imagens no primeiro subconjunto, 500 imagens no segundo e 1000 para o terceiro subconjunto. Vão ser também utilizadas as mesmas 6000 imagens anteriores em binário e também em *ground-truth*, que contêm apenas os símbolos das imagens binárias. Desta forma, a base de dados será composta por 6000 que possuem diversos tipos de deformações. Serão ainda utilizadas imagens fornecidas pelo INESC TEC, que totalizam 39 imagens de pautas manuscritas.

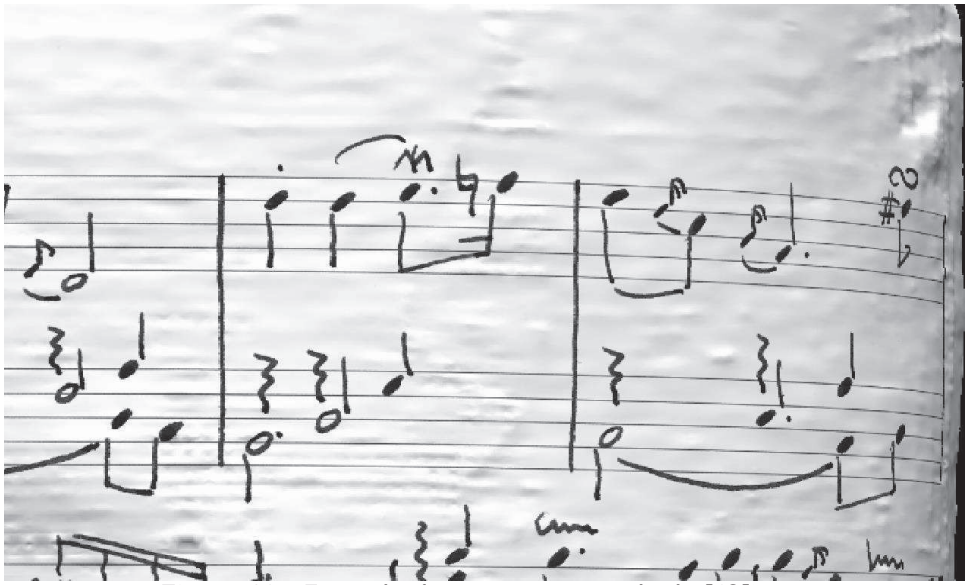

Figura 10 – Exemplo de imagem (retirada de [12])

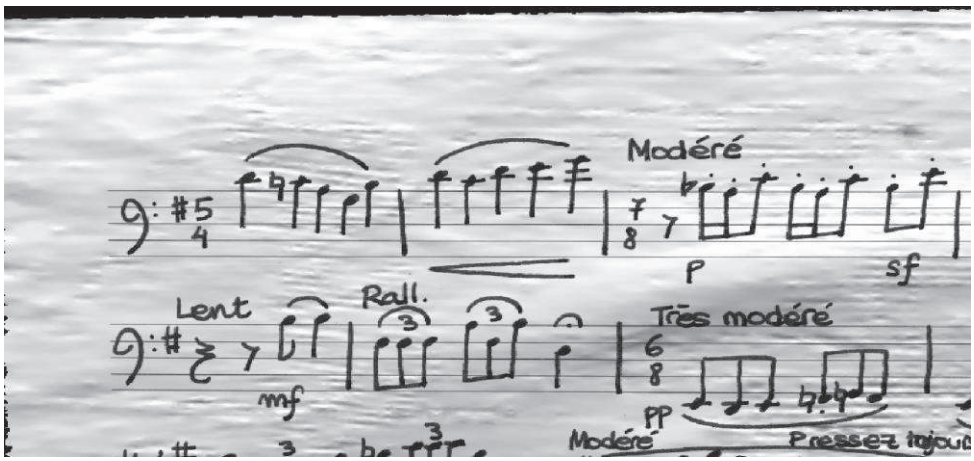

Figura 11 – Exemplo de imagem com ruído 3D (retirada de [12])

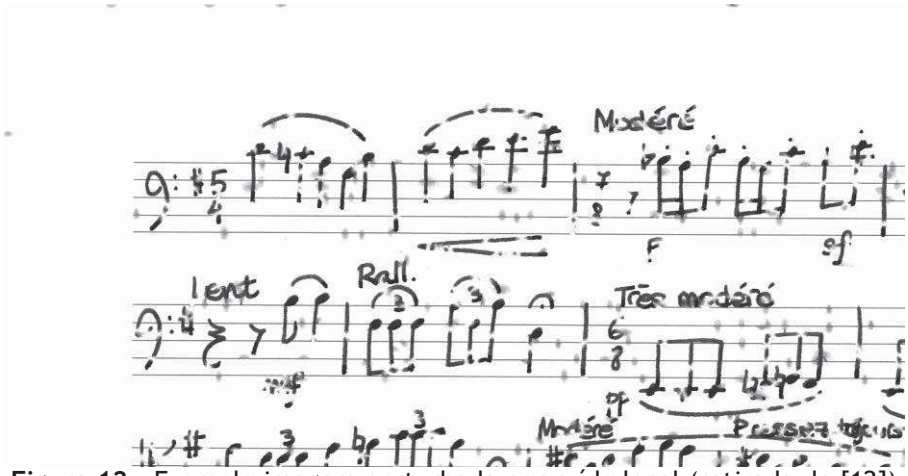

Figura 12 – Exemplo imagem perturbada por ruído local (retirada de [12])

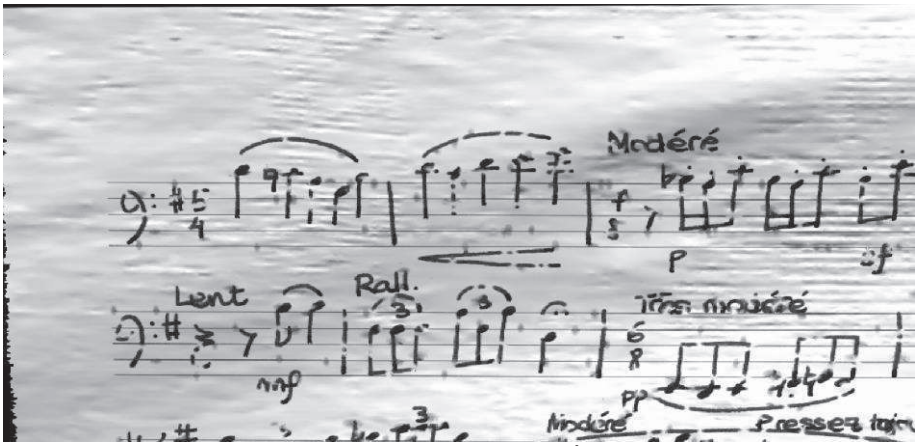

Figura 13 – Exemplo de imagem perturbada por ruído 3D e local (retirada de [12])

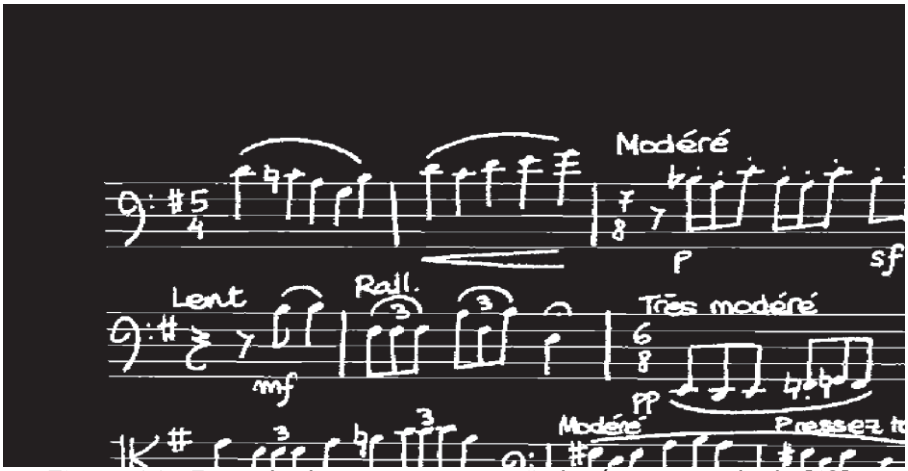

Figura 14 – Exemplo de imagem na versão binária (retirada de [12])

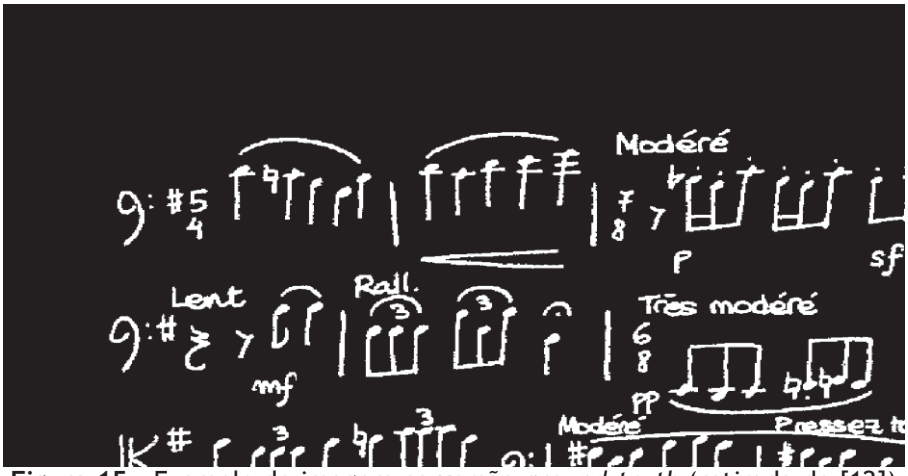

Figura 15 – Exemplo de imagem na versão *ground-truth* (retirada de [12])

4.3 – Métricas de Avaliação

4.3.1 Remoção de Linhas

A métrica escolhida para avaliar o desempenho do algoritmo de remoção foi a *F-measure* [31] porque foi a utilizada na competição "ICDAR / GREC 2013 *Competition on Music Scores: Staff Removal*"[12] para comparar o algoritmo desenvolvido com os submetidos na competição. De forma a poder avaliar a eficácia do algoritmo de remoção, foram utilizadas as imagens *ground-truth*, as imagens em binário e as imagens com apenas os pixels que foram removidos. É efetuada uma operação do tipo OU exclusivo (XOR) entre as imagens em binário e as imagens *ground-truth*, de forma a ser obtida uma imagem com apenas as linhas verdadeiras pertencentes à imagem. Estas novas imagens são utilizadas para calcular o número de pixels que foram removidos e pertencem à linha (*TP*), o número de pixels que foram removidos e não pertencem à linha (*FP*) e o número de pixels que não foram removidos e pertencem à linha (*FN*). É então calculada a *F-measure* (F_M) segundo a equação 4.1.

$$F_M = 2 \times \frac{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \tag{4.1}$$

4.3.2 Segmentação de Símbolos

Para a avaliação da segmentação dos símbolos, foram consideradas como deteções corretas as janelas que obtém uma sobreposição de área (*A1*) relativamente à área imagem de teste (*A2*) de pelo menos 50%, quando calculadas segundo a equação 4.2 A precisão foi calculada considerando as deteções corretas (*TP*), as deteções erradas (*FP*) e o número de exemplos não detetados (*MD*), segundo a equação 4.3. Apenas a deteção com maior sobreposição em relação à imagem de teste é considerada como deteção correta, sendo que em deteções múltiplas as restantes deteções são consideradas deteções erradas.

$$Sobreposição = \frac{area(A1 \cap A2)}{area(A1 \cup A2)} \tag{4.2}$$

$$\text{Precisão} = \frac{TP}{(TP + FP + MD)} \tag{4.3}$$

4.4 – Conclusão

Neste capítulo foram abordadas quais as ferramentas a utilizar para o desenvolvimento da dissertação e para a avaliação do trabalho desenvolvido.

Capítulo 5

Implementação

Neste capítulo serão apresentados os algoritmos utilizados para a realização dos objetivos propostos.

5.1 – Deteção de Linhas

O algoritmo de deteção de linhas utilizado foi o *Strong Staff Pixels* (presente em [21]). Este permite obter a posição, em forma de coordenadas, das linhas da pauta, a espessura e o espaçamento das linhas. Estes dados serão importantes nas etapas seguintes.

5.2 – Remoção de Linhas

Este algoritmo foi implementado em C e foi utilizada a biblioteca OpenCV [3].

O algoritmo de remoção de linhas desenvolvido para esta etapa faz uso da informação extraída na deteção de linhas para classificar se um pixel detetado pertence a uma linha ou a um símbolo ou ao fundo da imagem.

Este método consiste em, numa primeira fase, aplicar um filtro de mediana, com uma janela de 1 pixel de altura por 3 de largura, de forma a esbater pixels mais claros que possam estar a causar ruído sobre as linhas da pauta e símbolos. De forma a caracterizar os pixels detetados e a região adjacente aos mesmos é determinado um valor médio considerando uma janela com as seguintes dimensões de largura e altura, respetivamente, 1 pixel por 2 vezes a espessura das linhas da pauta, e centrada no pixel detetado. Estes valores da média serão usados para definir os valores dos pixels da linha em vez dos valores originais. Isto permite que toda a linha se encontre na região onde foi feita a média e que sejam incluídos também pixels da vizinhança da linha, resultando em valores mais baixos do que a média para pixels pertencentes a símbolos que intersetam a linha e valores mais altos para pixels do fundo da imagem. Na figura 16 encontra-se exemplificado o descrito, sendo o pixel detetado

representado a verde e as dimensões da janela delimitadas a vermelho.

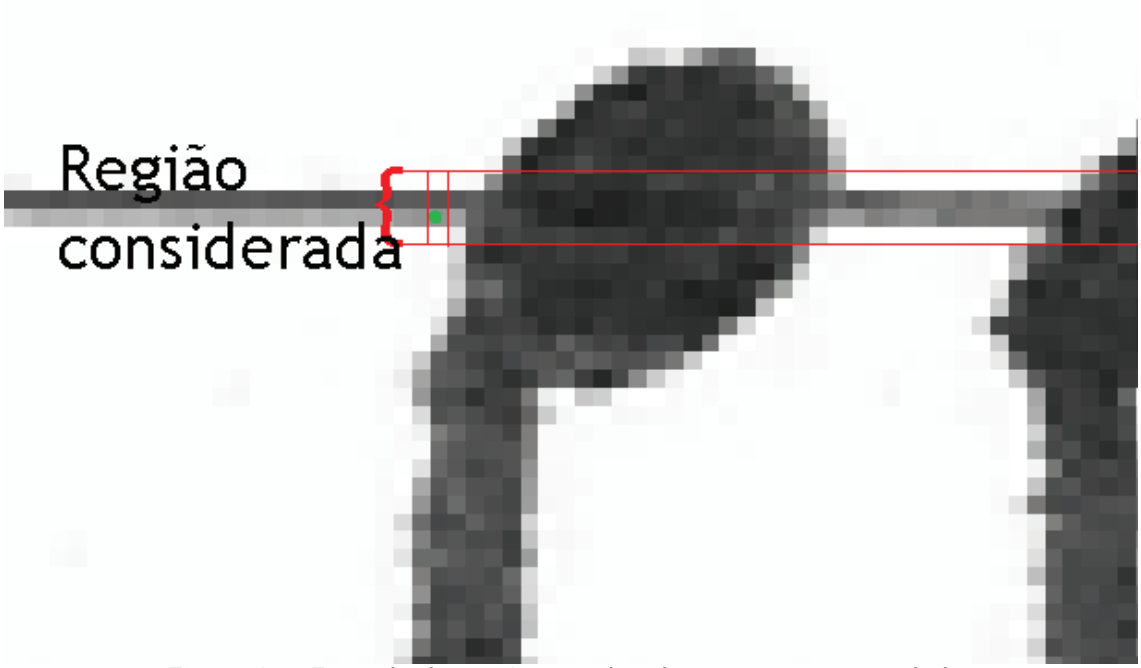

Figura 16 – Exemplo da região considerada para caraterizar a linha

Para que a classificação dos pixels seja mais exata, são definidos, para cada pixel detetado, 2 limiares t1 e t2 que irão permitir classificar os valores da média. Estes limiares são calculados obtendo um mediana centrada no pixel detetado e com uma largura igual ao espaçamento entre as linhas para t2 e com uma largura de 10 vezes o espaçamento entre as linhas para t1. Estes limiares são ainda ajustados sendo multiplicados por um coeficiente de ajustamento, alpha, obtido empiricamente e igual a 0,90. Utilizando os limiares calculados, os pixels detetados são classificados como pertencentes à linha ou não da seguinte forma:

- pixels onde a média calculada for menor do que o menor dos limiares são considerados como pertencentes a símbolos (representados a verde na Figura 19);
- pixels onde a média calculada for maior do que o maior dos limiares são considerados como pertencentes à linha (representados a vermelho na Figura 19);
- pixels que fiquem entre as 2 últimas condições e em que um dos 3 últimos pixels tiver sido classificado como símbolo são classificados como símbolo (representados a verde na Figura 19);
- os restantes pixels são classificados como linha (representados a vermelho na Figura 19), exceto se a média for maior do que o maior valor detetado na imagem multiplicado por *alpha*, sendo então classificados como fundo (representados a azul na Figura 19);

A linha é então removida alterando para 255 (branco) o valor de todos os pixels classificados como linha e que estejam numa janela de 1 pixel de largura e de 2 vezes a espessura da linha de altura centrada no pixel detetado (exemplo na Figura 17).

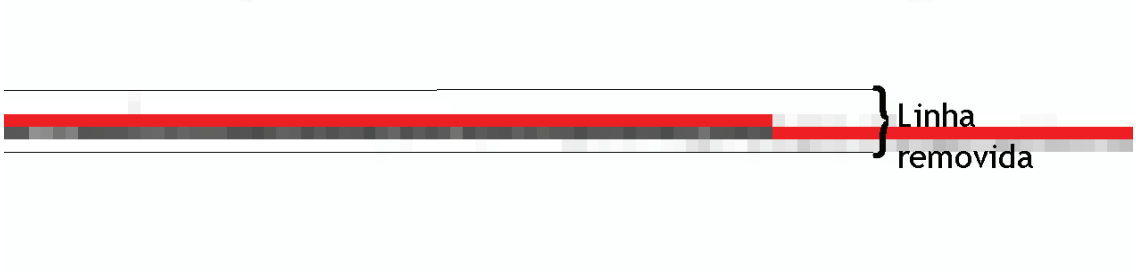

Figura 17 – Exemplo da linha removida

É também gerada uma imagem apenas com os pixels removidos para efeitos de avaliação do algoritmo (exemplo na Figura 20). Abaixo exemplifica-se uma imagem à qual vai ser aplicada o algoritmo (Figura 18), bem como a imagem classificada (Figura 19), os pixels removidos (Figura 20) e o resultado final (Figura 21).

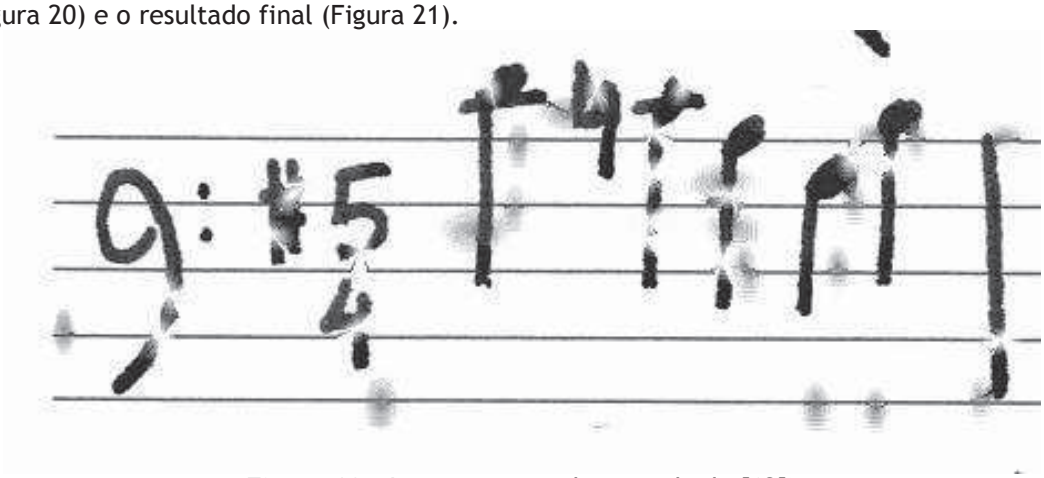

Figura 18 – Imagem original (retirada de [12])

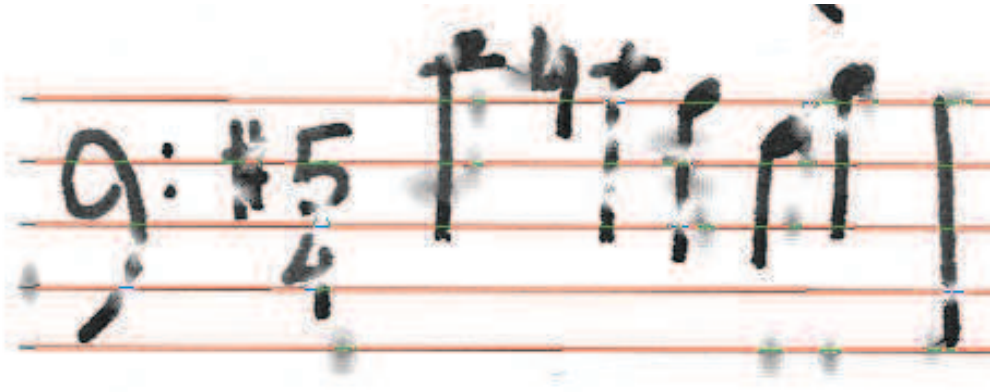

Figura 19 – Imagem classificada

Figura 20 – Pixels de linha removidos

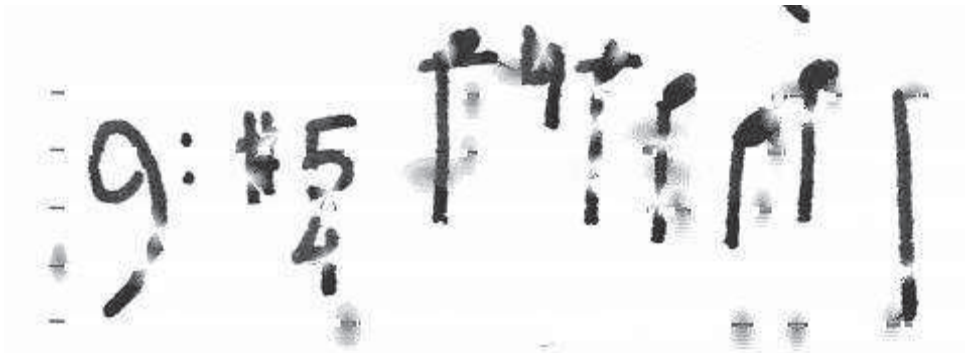

Figura 21 – Imagem com as linhas removidas

O algoritmo pode ser descrito na seguinte forma:

Algoritmo: Remoção de Linhas (imagem, linhas, espessura, espaçamento)

Variáveis:

média – valor da média com janela de 1 pixel por 2x espessura

alpha – coeficiente de ajustamento(igual a 0,90)

t1,t2 – limiares para classificação

max_val – máximo valor detetado na imagem

class – etiqueta de classificação

Pseudocódigo:

```
1:  Para todos os pixels de todas as linhas
2:      Calcular media
3:  Até todos os pixels de todas as linhas
4:  Para todos os pixels de todas as linhas
5:      Calcular calcular t1 e t2
6:      Se media <= min(t1,t2)
7:          class = símbolo
```

```
8:      Senão Se media > max(t1,t2)
9:          class = símbolo
10:     Senão Se um dos 3 últimos pixels for símbolo
11:         class = símbolo
12:     Senão class = linha
13:     Se media>max_val x alpha
14:         class = fundo
15: Até todos os pixels de todas as linhas
16: Para todos os pixels de todas as linhas
17:     Se class == linha
18:         Pixels( de –espessura até espessura)=branco;
19: Até todos os pixels de todas as linhas
```

5.3 – Segmentação de Símbolos

Esta etapa foi implementada em Matlab e utilizando a biblioteca de funções LibSVM [5].

Devido a diversos tipos de ruído que podem ser confundidos com símbolos como, por exemplo, manchas de tinta, e, de forma a facilitar a etapa de identificação de símbolos, torna-se necessário proceder à sua segmentação. Para isso, e considerando a complexidade da tarefa, é necessário recorrer a algoritmos de aprendizagem automática. O método escolhido para este caso será uma Máquina de Vectores de Suporte, pois obtém melhores resultados quando comparado com outros métodos [20].

5.3.1 Caraterísticas para Treino

As características escolhidas para treinar a máquina de vectores de suporte foram as seguintes:

- Momentos de *Zernike*, pois permitem representar propriedades que são invariantes à rotação e propriedades que não são invariantes à rotação [9],[11].
- Momentos invariantes de *Hu*, porque são invariantes à translação, rotação e escala do símbolo [9].
- Projeções horizontal e vertical da imagem, porque podem ser invariantes á inclinação dos caracteres [9].

A invariância à rotação é uma característica importante pois na partitura as notas podem surgir invertidas. Atributos como a invariância à translação, escala e inclinação são importantes, porque numa pauta manuscrita as figuras musicais não serão sempre do mesmo tamanho e podem surgir inclinadas devido à caligrafia do músico. Para escalar as imagens para a realização das projeções e os momentos de *Zernike*, que necessitam de uma imagem quadrada, as imagens são redimensionadas para uma imagem com 60 por 60 pixels, o que se considerou ser suficiente para representar os símbolos sem perda de informação apesar de existir deformação. Os momentos de *Zernike* foram extraídos até à ordem 10 pois não se verificavam diferenças significativas a partir dessa ordem. Deste modo, para cada exemplo ou subimagem de teste, é extraído um vetor com 163 caraterísticas.

5.3.2 Treino de SVM

Para treinar e testar a SVM foram utilizados cerca de 8600 exemplos de treino, metade positivos e metade negativos. Cerca de 4300 foram utilizados para treino e outros 4300 para teste do modelo. Os exemplos positivos foram selecionados manualmente de um conjunto de cerca de 50 imagens onde a remoção obteve menos erro. As imagens para estes dados incluem caligrafia de vários autores. Os exemplos negativos foram aleatoriamente retirados, com diversos tamanhos de janelas, das mesmas imagens mas retirando os símbolos presentes. Devido ao elevado número de caraterísticas extraídas e de exemplos fornecidos, considerou-se necessário recorrer a uma análise dos componentes principais (PCA) [10] de modo a reduzir o tempo necessário para treinar a SVM. Isto é realizado considerando todos os dados de treino e construindo uma transformação linear onde as várias colunas estão ordenadas da maior para a menor variância. Depois eliminaram-se as colunas para que os dados resultantes retenham pelo menos uma percentagem da variância total. Neste caso pretendeu-se reter 99% da variância total, o que resultou numa redução no tamanho do vetor de caraterísticas de 163 para 57. De modo a testar a SVM para outras imagens é necessário guardar a transformação linear aplicada para que tenham o tamanho correto. Para além desta transformação, os dados são também normalizados para que todas as caraterísticas contribuam aproximadamente da mesma forma para o cálculo dos vetores de suporte. São também guardados os valores máximo e mínimo de cada caraterística de forma a poder escalar novos dados para entre os mesmo valores. A SVM foi então treinada recorrendo a um método de validação cruzada o qual divide os dados em 3 partes iguais, utilizando um subconjunto para teste e os outros para treino. Foram então testados vários valores para os parâmetros da SVM, sendo que o

modelo final utiliza esses parâmetros e é treinado usando todos os dados. Isto evita que os parâmetros da SVM se adaptem demasiado aos dados de treino.

5.3.3 Segmentação em Tons de Cinzento

Para proceder à segmentação em tons de cinzento foi utilizado um algoritmo do tipo *sliding window* [30]. É efetuado um varrimento à imagem com janelas de vários tamanhos, de forma a poderem ser extraídas características relevantes que permitam a segmentação dos símbolos musicais. Os tamanhos das janelas foram determinados empiricamente e são baseados no espaço entre linhas da imagem, de forma a se poderem adaptar aos vários tamanhos de símbolos. Foram então escolhidos os seguintes tamanhos de janelas, com as seguintes larguras e alturas respetivamente:

1. 3x espaço entre linhas por 4,5x espaço entre linhas;
2. 2x espaço entre linhas por 4x espaço entre linhas;
3. 6x espaço entre linhas por 2x espaço entre linhas;
4. 2x espaço entre linhas por 2x espaço entre linhas.

Foram escolhidos estes tamanhos de janelas para que se adaptassem a todos os símbolos. O primeiro tamanho é mais adequado a símbolos como, por exemplo, claves, o segundo para figuras musicais, barras de compasso e pausas, o terceiro para variações de volume e o quarto tamanho para acidentes, tonalidade, e pausas e figuras musicais mais pequenas. De forma a acelerar o processo de varrimento da imagem, em vez de se deslizarem as várias janelas para cada pixel, efetuam-se saltos de 1/6 do tamanho da janela. Por exemplo, uma janela de 60 por 60 pixels, processada saltando janelas de 10 em 10 pixels. Em seguida, é aplicada a transformação linear de caraterísticas e os dados são então escalados considerando os valores máximo e mínimo de treino. É depois aplicada a deteção utilizando uma SVM com o modelo treinado como descrito anteriormente.

A segmentação resultante deste método criou múltiplas deteções e deteções parciais dos mesmos símbolos. Para eliminar algumas dessas deteções e falsas deteções foi aplicado um limiar sobre a probabilidade de uma janela conter ou não um símbolo. Essa probabilidade foi obtida através do resultado da aplicação da SVM e o limiar foi fixado empiricamente em 90%. De forma a remover as restantes janelas sobrepostas, testaram-se 2 métodos: efetuar a média das janelas sobrepostas [30] e *non-maximum suppression* [10]. Em ambos os métodos as janelas foram ordenadas das de maior para as de menor probabilidade e percorridas por essa ordem. O primeiro método consiste em efetuar a média de todas as janelas que se sobreponham mais do que uma determinada percentagem, neste caso 10%, e onde se sobreponham pelo menos um determinado número de janelas, neste caso 2. O segundo método, para cada janela que se sobreponha mais do que uma determinada percentagem com outras janelas, neste caso 10%, são eliminadas todas as janelas com menor

probabilidade. Este procedimento é realizado 2 vezes, a primeira para cada tamanho de janela e a segunda para unir os vários tamanhos de janelas.

O seguinte esquema descreve todo o processo de segmentação:

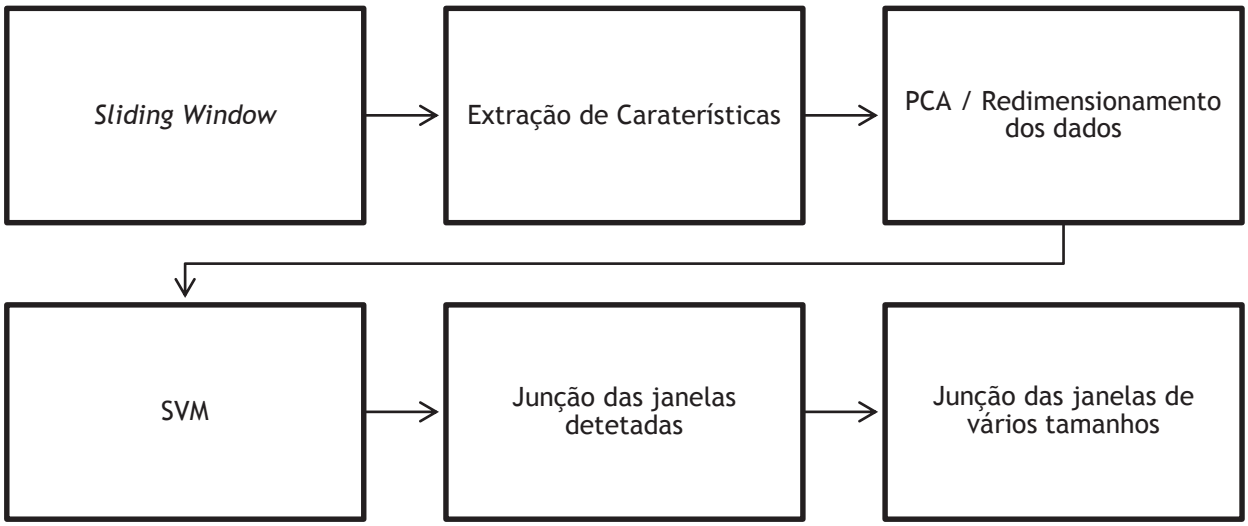

Figura 22 – Esquema do processo de segmentação

5.4 – Conclusão

Neste capítulo foram apresentados os algoritmos desenvolvidos e utilizados nesta dissertação, descrevendo e justificando todos os passos seguidos.

Capítulo 6

Resultados Obtidos

6.1 – Remoção de Linhas

Os resultados obtidos para os conjuntos de dados de teste da competição "ICDAR / GREC 2013 *Competition on Music Scores: Staff Removal*" [12], foram os seguintes (apresentados em média μ e desvio padrão σ, calculados como indicado em 4.3.1):

Tabela 1 – Resultados do algoritmo de remoção comparados com os de [12]

	ICDAR/GREC 2013 Conjunto de Teste 1	ICDAR/GREC 2013 Conjunto de Teste 2	ICDAR/GREC 2013 Conjunto de Teste 3
Algoritmo proposto	μ - 42,3% σ - 7,5%	μ - 60,3% σ – 3,6%	μ - 43,1% σ – 7,0%
INESC-gray	μ - ~38%	μ - ~52%	μ - ~40%
INESC-bin	μ - ~89%	μ - ~98%	μ - ~88%
LRDE-gray	μ - ~92%	μ - ~79%	μ - ~80%
LRDE-bin	μ - ~97%	μ - ~96%	μ - ~95%

Com base nos resultados de [31], pode-se afirmar que, quando comparado com outros algoritmos em tons de cinzento, o algoritmo desenvolvido obtém melhores resultados do que o INESC-gray, mas piores do que LRDE-gray para todos os conjuntos de teste utilizados na competição "ICDAR / GREC 2013 *Competition on Music Scores: Staff Removal*" [12]. Tal como a maioria dos outros algoritmos comparados, o algoritmo proposto apresenta melhores resultados no conjunto de teste 2, o qual só apresenta deformações com manchas de tinta e fragmentação de símbolos, e piores nos conjuntos 1 e 3, os quais apresentam deformações

3D. Isto sugere que o algoritmo tem um mau desempenho quando exposto a deformações 3D. Os motivos prendem-se com um maior número de linhas não detetadas, desvios na deteção de uma linha e uma menor robustez do algoritmo para este tipo de perturbações e para desvios na deteção da linha. Na Figura 23 (a imagem encontra-se classificada por cores de acordo com o descrito em 5.2) é possível ver exemplos do comportamento descrito, bem como situações em que a classificação foi feita corretamente.

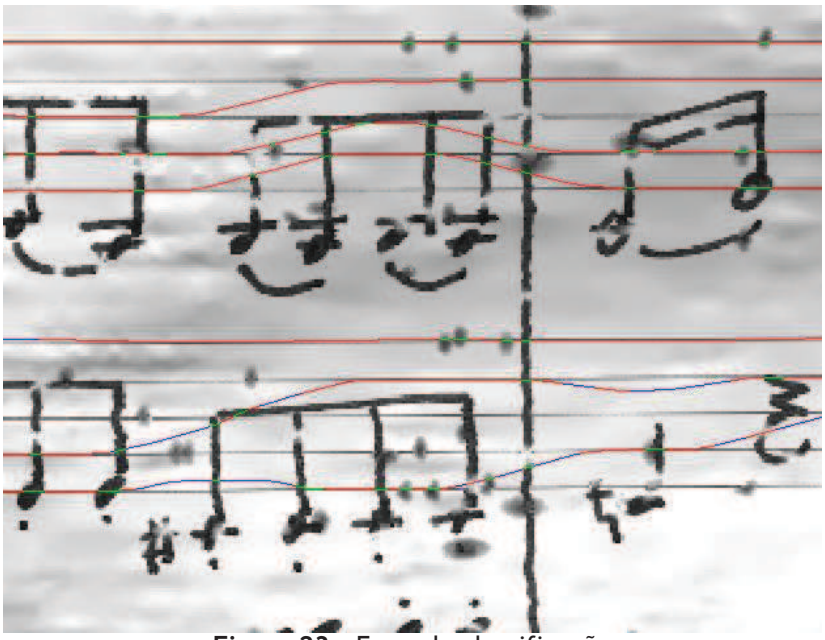

Figura 23 – Erros de classificação

Considerando a precisão obtida nos 3 conjuntos de teste e o comportamento do algoritmo face a deformações 3D, optou-se por não utilizar os dados dos conjuntos de treino 1 e 3 para a etapa de segmentação de símbolos pois seria introduzido demasiado erro no processo de treino. Calculou-se a precisão para o conjunto de treino 2 e para a base de dados fornecida pelo INESC. Os resultados obtidos encontram-se na Tabela 2.

Tabela 2 – Resultados do algoritmo de remoção

	ICDAR/GREC 2013 Conjunto de Treino 2	Base de dados INESC
Algoritmo proposto	μ - 61,5% σ – 3,4%	μ - 59,9% σ – 3,9%

Os resultados apresentados pelo conjunto de treino 2 são semelhantes aos apresentados pelo conjunto de teste 2, tal como seria de esperar visto que contem o mesmo tipo de deformações. Para a base de dados fornecida pelo INESC o resultado foi semelhante ao do conjunto anterior. O algoritmo apenas falhou na remoção em algumas imagens onde a

resolução era menor e que possuíam elevado ruído local pelo que se conclui que não é totalmente robusto para essas condições.

6.2 – Segmentação de Símbolos

6.2.1 SVM

A avaliação desta etapa foi feita recorrendo ao conjunto de exemplos reservados para teste que totalizam 4300 exemplos. A precisão obtida neste teste foi de 97,84%, o que resulta em 4207 exemplos identificados corretamente do total de 4300.

6.2.2 Sliding Window

Para avaliar esta etapa foram utilizadas 10 imagens do conjunto de teste 2, o que resultou num total de aproximadamente 2600 exemplos. As imagens foram selecionadas de modo que a que tenham sido escritas por diferentes autores e que o resultado da remoção não tenha perturbado demasiado este processo. A segmentação resultante da SVM produziu múltiplas deteções dos mesmos símbolos, não falhando símbolo nenhum, embora tenham sido produzidas algumas falsas deteções. A aplicação de um limiar na etapa após a previsão permitiu eliminar várias múltiplas deteções bem como falsas deteções, embora falhando algumas deteções, tal como se pode ver nas Figuras 24 e 25 (as deteções estão representadas por caixas a vermelho sobre a figura).

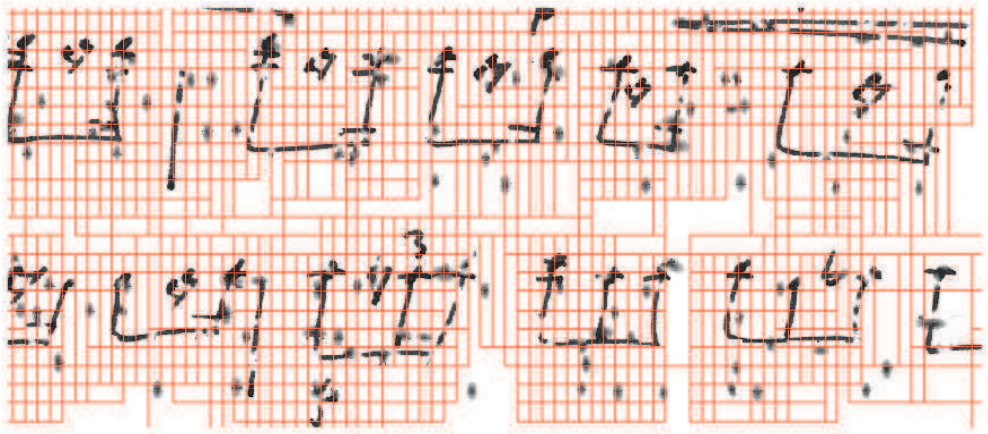

Figura 24 – Múltiplas deteções dos mesmos símbolos

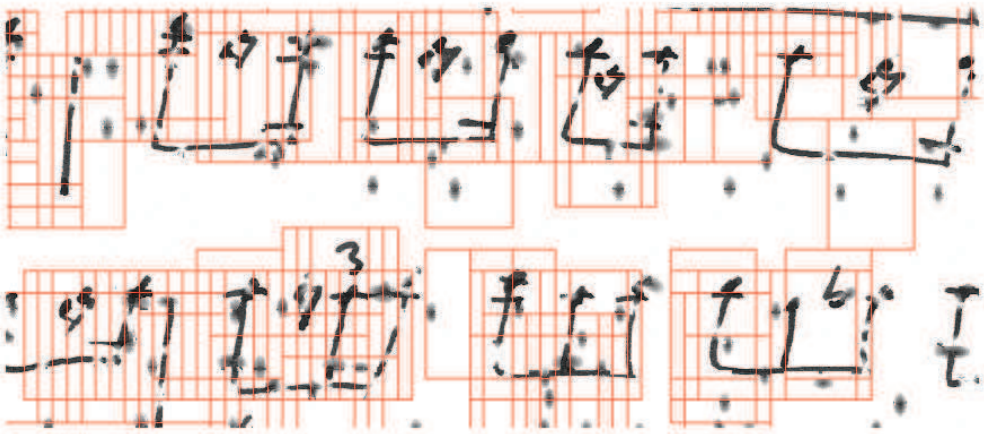

Figura 25 – Deteções após aplicação do limiar

Visto que ainda resultaram múltiplas deteções dos mesmos símbolos, como é visível na figura 25, é necessário unir as várias janelas. Foram então aplicados 2 métodos para unir as várias deteções, e os resultados obtidos estão presentes na Tabela 3.

Tabela 3 – Resultados da precisão de deteção

	non-maximum suppression	Média das janelas sobrepostas
Precisão média	4,55%	2,65%
Desvio padrão médio	1,74%	0,5%

A partir destes resultados é possível concluir que o método de *non-maximum suppression* tem um desempenho ligeiramente melhor. É também possível concluir que a precisão de deteção após a aplicação destes métodos é bastante baixa. Tal deve-se ao comportamento da SVM para as janelas com os tamanhos 2x o espaço entre linhas por 4x o espaço entre linhas e 6x o espaço entre linhas por 2x o espaço entre linhas, as quais obtêm falsas deteções bem como

falham algumas deteções, tal como se pode ver nas Figuras 26 e 27.

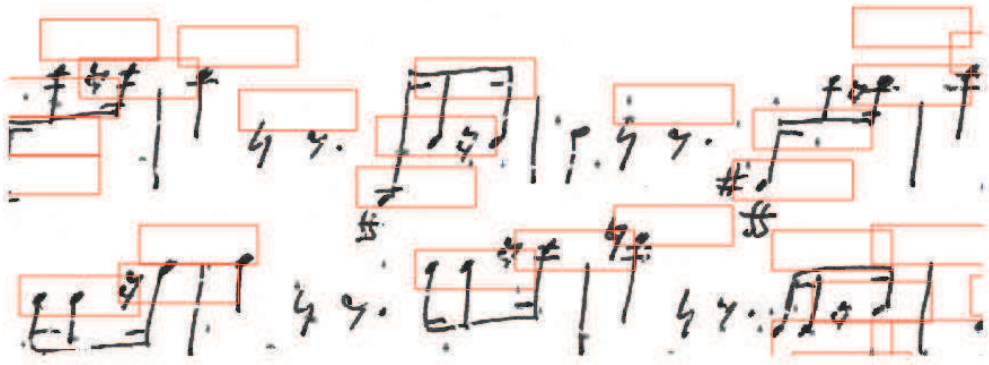

Figura 26 – Resultado da segmentação com a janela de 6x por 2x espaço entre linhas

Figura 27 - Resultado da segmentação com a janela de 2x por 4x espaço entre linhas

É também possível verificar-se que as janelas com tamanho 2x espaço entre linhas por 2x espaço entre linhas, obtêm maior probabilidade de pertencer a um símbolo, mesmo quando outras janelas são uma melhor aproximação aos exemplos de teste. Os métodos utilizados causam a eliminação das janelas com melhor aproximação, dando preferência a uma ou mais janelas mais pequenas com maior probabilidade de pertencer a um símbolo, tal como se pode ver nas Figuras 28 e 29 (na Figura 28 estão representados a azul os exemplos positivos dos símbolos e a vermelho as deteções obtidas). Isto causa um aumento do erro pois não só é eliminada uma deteção como, também, são consideradas várias falsas deteções e deteções falhadas.

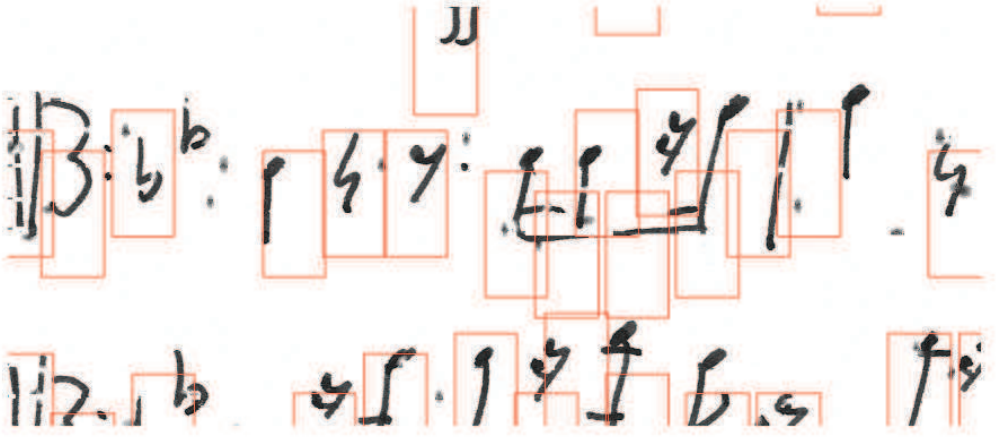

Figura 28 – Resultados da segmentação com a janela de 2x por 4x espaço entre linhas

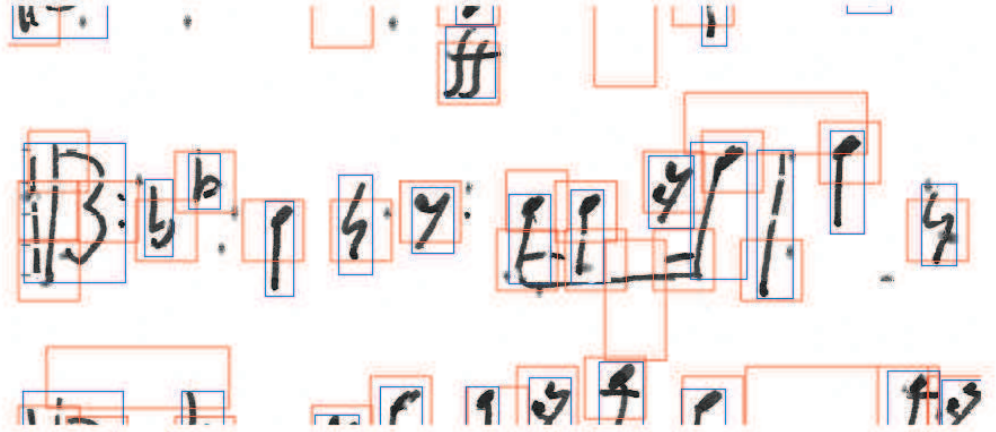

Figura 29 – Resultados finais da segmentação

6.3 – Conclusão

Neste capítulo foram apresentados e discutidos os resultados obtidos para os algoritmos desenvolvidos.

Capítulo 7

Conclusões e Trabalho Futuro

7.1 - Conclusões

Com base nos resultados, pode concluir-se que o algoritmo de remoção de linhas desenvolvido obtém resultados melhores do que o algoritmo para imagens em tons de cinzento INESC-gray, embora o seu desempenho fique sempre abaixo dos algoritmos para imagens binárias, bem outros algoritmos para tons de cinzento utilizados na competição "ICDAR / GREC 2013 *Competition on Music Scores: Staff Removal*" [12]. Ainda assim, considera-se que este algoritmo tem um desempenho favorável para as perturbações apresentadas pelo subconjunto de teste 2, sendo que a precisão apresentada só não é maior devido à remoção por excesso das linhas.

Em relação à segmentação dos símbolos, conclui-se que os resultados apresentados se revelam insatisfatórios tendo em conta a precisão média obtida. Tal deve-se ao elevado número de falsas deteções e de deteções falhadas. No entanto, considera-se que as múltiplas deteções obtidas antes da aplicação de um limiar constituem uma boa segmentação dos símbolos da imagem, visto que não deixam símbolos por detetar, embora incluam algumas falsas deteções.

7.2 – Trabalho Futuro

Tendo em conta o trabalho que foi realizado nesta dissertação, podem sugerir-se alguns pontos de partida para futuros trabalhos e investigações.

No que diz respeito ao algoritmo de remoção de linhas, sugere-se que não se realize a remoção das linhas por excesso, de modo a diminuir o erro obtido. Sugere-se ainda que se aumente a robustez do algoritmo a deformações em 3D e a desvios da linha proveniente de erros de deteção.

Relativamente à segmentação dos símbolos, sugere-se que sejam treinados vários modelos de SVM para cada tamanho de janela pretendido de forma a minorar múltiplas deteções e deteções parciais dos mesmos símbolos.

Referências

[1]. Asa Ben-Hur e Jason Weston. A User's Guide to Support Vector Machines. Colorado State University, Princeton, NJ 08540 USA.

[2]. Bainbridge, D. e T. I. M. Bell. "The challenge of optical music recognition." *Language Resources and Evaluation* no. 35 (2):95-121. 2001.

[3]. Bradski, G. "The OpenCV Library." *Dr. Dobb's Journal of Software Tools*. 2000.

[4]. Cardoso, J. S. e A. Rebelo. 2010. "Robust staffline thickness and distance estimation in binary and gray-level music scores", em Istanbul.

[5]. Chang, Chih-Chung; Lin, Chih-Jen. "LIBSVM: A library for support vector machines." *ACM Transactions on Intelligent Systems and Technology* no. 2 (3). 2011.

[6]. Cui, J., H. He e Y. Wang. 2010. "An adaptive staff line removal in music score images", em Beijing.

[7]. Desaedeleer, Arnaud F. "Reading Sheet Music", Department of Computing, University of London - Imperial College London - Technology and Medicine. 2006.

[8]. dos Santos Cardoso, J., A. Capela, A. Rebelo, C. Guedes e J. Pinto da Costa. "Staff detection with stable paths." *IEEE Transactions on Pattern Analysis and Machine Intelligence* no. 31 (6):1134-1139. 2009.

[9]. Due Trier, Øivind, Anil K. Jain e Torfinn Taxt. "Feature extraction methods for character recognition-A survey." *Pattern Recognition* no. 29 (4):641-662. 1996.

[10]. Felzenszwalb, P. F., R. B. Girshick, D. McAllester e D. Ramanan. "Object Detection with Discriminatively Trained Part-Based Models." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* no. 32 (9):1627-1645. 2010.

[11]. Hwang, Sun-Kyoo e Whoi-Yul Kim. "A novel approach to the fast computation of Zernike moments." *Pattern Recognition* no. 39 (11):2065-2076. 2006.

[12]. "ICDAR / GREC 2013 Competition on Music Scores". 2014. Acedido em 7/02/2014. http://dag.cvc.uab.es/muscima/.

[13]. Marquardt, Donald. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." *SIAM Journal on Applied Mathematics* no. 11 (2):431-441. 1963.

[14]. "Notação musical". 2014. Acedido em 7/02/2014. http://pt.wikipedia.org/wiki/Nota%C3%A7%C3%A3o_musical.

[15]. Otsu, Nobuyuki. "A Threshold Selection Method from Gray-Level Histograms." *Systems, Man and Cybernetics, IEEE Transactions on* no. 9 (1):62-66. 1979.

[16]. Piatkowska, W., L. Nowak, M. Pawłowski e M. Ogorzałek. 2012. Stafflines pattern detection using the swarm intelligence algorithm. Warsaw. http://www.scopus.com/inward/record.url?eid=2-s2.0-84868017012&partnerID=40&md5=2fe810466cd5726ee4745382d07aef79.

[17]. Pinto, Telmo, Ana Rebelo, Gilson Giraldi e Jaime S. Cardoso. "Music score binarazation based on domain knowledge."

[18]. Pravesjit, S. e A. Thammano. 2012. "Segmentation of historical lanna handwritten manuscripts". Comunicação apresentada em Intelligent Systems (IS), 2012 6th IEEE International Conference. 6-8 Sept. 2012.

[19]. Rajput, G. S. G. e R. Horakeri. 2013. "Zone based handwritten Kannada character recognition using crack code and SVM". Comunicação apresentada em Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on. 22-25 Aug. 2013.

[20]. Rebelo, A., G. Capela e J. S. Cardoso. "Optical recognition of music symbols." *International Journal on Document Analysis and Recognition* no. 13 (1):19-31. 2010.
[21]. Rebelo, A. e J. S. Cardoso. 2013. "Staff line detection and removal in the grayscale domain",Washington, DC.
[22]. Rebelo, A., J. Tkaczuk, R. Sousa e J. S. Cardoso. 2011. "Metric learning for music symbol recognition", em Honolulu, HI.
[23]. Rokach, Lior. *Pattern Classification Using Ensemble Methods*. World Scientific Publishing Co.: SGP.2009.
[24]. Sauvola, J. e M. Pietikäinen. "Adaptive document image binarization." *Pattern Recognition* no. 33 (2):225-236. 2000.
[25]. Solomon, Chris, Stuart Gibson e Toby Breckon. *Fundamentals of Digital Image Processing : A Practical Approach Using Matlab*. Hoboken, NJ, USA: Wiley.2010.
[26]. Sotoodeh, M. e F. Tajeripour. 2012. "Staff detection and removal using derivation and connected component analysis", em Shiraz.
[27]. Su, B., S. Lu, U. Pal e C. L. Tan. 2012. "An effective staff detection and removal technique for musical documents", em Gold Coast, QLD.
[28]. Tajeripour, F. e M. Sotoodeh. "A novel staff removal method for printed music image." *IEICE Electronics Express* no. 9 (7):609-615. 2012.
[29]. Tambouratzis, T. 2011. "Identification of key music symbols for optical music recognition and on-screen presentation", em San Jose, CA.
[30]. Viola, Paul e MichaelJ Jones. "Robust Real-Time Face Detection." *International Journal of Computer Vision* no. 57 (2):137-154. 2004.
[31]. Visaniy, Muriel, V. C. Kieu, Alicia Fornes e Nicholas Journet. "ICDAR 2013 Music Scores Competition: Staff Removal."1407-1411. 2013.