

Ana Maria Rebelo

Robust Optical Recognition of Handwritten Musical Scores based on Domain Knowledge

PhD Thesis



University of Porto
2012

University of Porto
Department of Electrical and Computer Engineering

*Thesis submitted to School of Engineering, University of Porto
for the Degree of Philosophy*



Robust Optical Recognition of Handwritten Musical Scores based on Domain Knowledge

Ana Maria Rebelo

(arebelo@inescporto.pt)

Thesis submitted under supervision of
Professor Doctor Jaime S. Cardoso (INESC TEC (formerly INESC Porto) e FEUP)
and co-supervised by
Professor Doctor André R. S. Marçal (CICGE e CMUP, FCUP).

Porto, 2012

Music gives a soul to the universe, wings to the mind, flight to the imagination and life to everything.

Plato

For centuries, music has been shared and remembered by two traditions: aural transmission, common to folk and popular musical genres, and in the form of written documents normally called musical scores. Prior to music typographical systems, all music was copied manually including large scores and each and every part for the players and singers. Publishers have typeset a large body of historical musical scores, principally from what is known as classical music. However, there remains a substantial and important corpus of works that exist as original hand-written manuscripts (or facsimiles of these manuscripts such as photocopies). The problem is not restricted to historical documents: many contemporary compositions also exist in hand-written or facsimile format. These important cultural artifacts are in danger of being lost through the normal ravages of time. To preserve the music (rather than the documents themselves) requires some form of typesetting or, ideally, a computer system that can automatically decode the symbolic images and create new scores. Meaning, the documents must be digitized and transformed into machine-readable format. Programs analogous to optical character recognition systems called optical music recognition (OMR) systems have been under intensive development for many years. However, the results to date are far from ideal. Each of the proposed methods emphasizes different properties and therefore makes it difficult to effectively evaluate its competitive advantages. This thesis provides a study and an overview of the literature concerning the automatic analysis of images of printed and handwritten musical score including an introduction to OMR processing systems.

After an image preprocessing stage, an OMR system is typically divided into three modules: (1) music symbol recognition, (2) musical notation reconstruction and (3) final representation construction.

For the binarization operation, several options have already been proposed in the past. However, in OMR, researchers still prefer using standard binarization procedures, such as Otsu's method. No goal-directed studies for music sheets have been carried out. In this thesis, a novel binarization algorithm is presented. The process is based on the knowledge of the content of the image. The method uses the estimation of the staff line thickness and the vertical distance between two staff lines extracted directly from the gray level music score to guide the binarization process.

The next operation in the OMR is the detection and subsequent removal of staff lines. In this work, a general-purpose, knowledge-free method for the automatic detection of music staff lines based on stable paths approach is proposed.

One of the final steps of an OMR system encompasses the segmentation and classification of the music symbols. For the detection of these various objects, two procedures are discussed: the first music symbols segmentation's method is based on contextual information and music rules; the score is first split by staves and the symbols are divided considering their graphical position and geometric features; the second method performs simultaneously the segmentation and the classification. In the music symbol classification, a comparative analysis divided into three distinct studies is described. Five different pattern recognition procedures are examined. In the first study, the database of scores is augmented with replicas of the existing patterns, transformed according to an elastic deformation

technique. Such transformations aim to introduce invariances in the prediction with respect to the known variability in the symbols, particularly relevant on handwritten works. In the second study, the classifiers are tested for three different situations: separation of composers, gradual increase of deformations and union of real and printed scores. In the third study, a distance metric learning is applied to k -NN classifier to recognize music symbols.

Syntactic and semantic music rules are also incorporated after the segmentation and classification of the music symbols as prior knowledge. Global constraints about musical rules are included as an optimization problem. The idea is to detect the best combination of symbols in order to give the indicated measure.

In this thesis the reader will find work in many stages of an OMR system with important and significant advances. Open issues and future trends will also be addressed.

Durante séculos, a música tem sido transmitida e lembrada através de duas tradições: transmissão oral, comum ao género musical popular, e na forma de documentos escritos normalmente chamados de partituras musicais. Antes dos sistemas tipográficos musicais, toda a música era copiada manualmente até mesmo grandes partituras com toda a parte relativa à letra da canção. Os editores criaram um grande corpo de partituras musicais históricas, principalmente a partir do que é conhecido como música clássica. No entanto, há ainda um substancial e significativo acervo de obras importantes que existem como manuscritos originais (ou como *fac-símile*). O problema não está restringido a documentos históricos: muitas composições contemporâneas também existem em formato manuscrito ou fotocópia. Estes importantes artefactos culturais estão em perigo de se perder pelos estragos normais do tempo. Para preservar a música (até mais do que os documentos em si) é necessário alguma forma de escrever ou, idealmente, um sistema de computador que possa automaticamente codificar as imagens simbólicas e criar novas partituras. Por outras palavras, os documentos devem ser digitalizados e transformados num formato legível por computador. Programas análogos aos sistemas de reconhecimento ótico de caracteres chamados de sistemas de reconhecimento ótico de música (OMR) têm estado em desenvolvimento intensivo por muitos anos. Contudo, os resultados até à data estão longe do ideal. Cada método proposto enfatiza diferentes propriedades e, portanto, torna difícil avaliar de forma eficaz as suas vantagens competitivas. Esta tese promove um estudo e uma análise da literatura ligada ao processamento automático de imagens impressas e manuscritas de partituras musicais incluindo uma introdução aos sistemas de processamento de OMR.

Após uma fase de pré-processamento de imagem, um sistema de OMR é normalmente dividido em três módulos: (1) reconhecimento dos símbolos musicais, (2) reconstrução da notação musical e (3) construção da representação final.

No passado já foram propostas várias opções para a operação de binarização. Contudo, em OMR, os investigadores continuam a preferir usar procedimentos de binarização standard, tal como o método de Otsu. Não foram ainda realizados nenhuns estudos diretos com partituras. Nesta tese é apresentado um novo algoritmo de binarização. O processo é baseado no conhecimento do conteúdo da imagem. O método usa as estimativas da espessura da linha e do distanciamento vertical entre duas linhas extraídas diretamente da partitura em gama de cinzentos.

A operação que se segue num sistema de OMR é a deteção e remoção das linhas de pauta. Neste trabalho é proposto um método de uso geral e conhecimento livre para a deteção automática das linhas de música baseado na aproximação do caminho estável.

Uma das etapas finais de um sistema de OMR aborda a segmentação e classificação dos símbolos musicais. Para a deteção destes vários objetos dois procedimentos são discutidos: o primeiro método de segmentação dos símbolos musicais é baseado na informação contextual e regras musicais; a partitura é primeiro dividida por pautas e os símbolos são divididos considerando as suas posições gráficas e características geométricas; o segundo método executa simultaneamente a segmentação e a classificação. Uma análise comparativa dividida em três estudos distintos é descrita na classificação dos símbolos

musicais. São examinados cinco procedimentos diferentes de reconhecimento de padrões. No primeiro estudo, a base de dados das partituras é aumentada com replicas de padrões existentes, transformados de acordo com a técnica de deformação elástica. Tais transformações pretendem introduzir invariâncias na previsão em relação à variabilidade conhecida nos símbolos, particularmente relevantes em trabalhos manuscritos. No segundo estudo, os classificadores são testados em três situações diferentes: separação de compositores, aumento gradual de deformações e junção de partituras reais e impressas. No terceiro estudo, uma aprendizagem da distância da métrica é aplicada num classificador k -NN para reconhecer os símbolos musicais.

Regras sintáticas e semânticas de música também são incorporados após a segmentação e classificação dos símbolos musicais como conhecimento prévio. Restrições globais sobre as regras musicais são incluídos como um problema de otimização. A ideia é detetar a melhor combinação de símbolos, a fim de dar a medida indicada.

Nesta tese, o leitor encontrará trabalho em várias etapas de um sistema de OMR com avanços importantes e significativos. Também serão abordadas questões em aberto e tendências futuras.

Contents	V
List of Figures	IX
List of Tables	XI
Glossary	XIII
 I Introduction	 1
<hr/>	
1 Introduction	3
1.1 OMR System Project	4
1.2 OMR Architecture	5
1.3 Objectives	7
1.4 Contributions and Related Publications	8
1.5 Thesis' Structure	9
 2 State-of-the-art	 11
2.1 Image Pre-processing	12
2.2 Staff Line Detection and Removal	15
2.3 Symbol Segmentation and Recognition	17
2.4 Musical Notation Construction and Final Representation	19
2.5 Available Datasets and Performance Evaluation	22
2.6 Open Issues	25
 3 Data Sets of Music Scores	 27
3.1 Database of Music Scores	27
3.2 Deformations in the Synthetic Scores	30
3.3 Deformations applied to each Music Symbol	33
3.4 Ground-truth Information	33
 4 Background Knowledge	 37
4.1 Recognition Process	37
4.2 Metric Learning	39
 II Preprocessing	 43

5	Staffline Thickness and Distance Estimation	45
5.1	Conventional estimation of staffline thickness and distance	45
5.2	Robust estimation of staffline thickness and spacing	45
5.3	Staffline thickness and spacing estimation in gray-level images	47
5.4	Experimental Assessment	50
5.5	Synthesis	50
6	Binarization Algorithm	51
6.1	Content aware music score binarization	51
6.2	Binarization Procedures	53
6.3	Evaluation Metrics and Results	57
6.4	Synthesis	61
III Music Symbols Recognition		63
<hr/>		
7	Staff Lines Detection and Removal	65
7.1	A Stable Path Approach for Staff Line Detection	65
7.2	Dalitz's Algorithm	71
7.3	Evaluation Metrics and Results	72
7.4	Staff Removal Competition	75
7.5	Synthesis	78
8	Musical Symbols Segmentation and Classification	79
8.1	Music Symbols Segmentation	80
8.2	Music Symbols Segmentation through Recognition	90
8.3	Syntactic Consistency	95
8.4	Evaluation Metrics and Results	97
8.5	Discussion	101
IV Conclusions and Future Work		103
<hr/>		
9	Conclusion	105
9.1	Future Trends	107
References		111
V	Appendix	123
<hr/>		
A	Fundamentals for Staff Line Detection and Removal Algorithms	125
A.1	Dynamic Programming	125

A.2	Staff Line Removal	126
A.3	Removal Error Metrics	128
B	Musical Symbols Classification	130
B.1	Musical Symbols	130
B.2	Experimental Testing	131

List of Figures

1.1	Web-based system interface.	4
1.2	Generic system architecture.	5
1.3	Typical architecture of an OMR processing system.	6
2.1	Some examples of music scores used in the state-of-art algorithms. (a) from Rebelo [104, Fig.4.4a)].)	12
2.2	Landscape of automated thresholding methods. From Pinto et al [95, Fig.1].	13
2.3	The characteristic page dimensions of staffline_height and staffspace_height. From Cardoso and Rebelo [23].	14
2.4	Example of an image where the estimation of staffline_height and staffspace_height by vertical runs fails. From Cardoso and Rebelo [23, Fig.2].	15
2.5	Summary of the most used techniques in an OMR system.	26
3.1	Variability in handwritten music scores.	27
3.2	Some examples of applied deformations from the original image: a) Original; b) Curvature c) Degradation after Kanungo; d) Staff line thickness variation; e) Staff line y-variation; f) Typeset emulation; g) Rotation; h) White Speckles; i) Staff line Interruptions.	31
3.3	Example of deformations on a musical symbol. From Rebelo [106, Fig.6)].	34
3.4	Some examples of binarization music scores.	34
4.1	Schematic illustration idea behind LMNN. Left side shows traditional approach, under Euclidean distance, where x_i has three target neighbors. Right image shows the new discriminator after the Mahalanobis distance being learnt. From [139, Fig.1].	41
5.1	Illustration of the estimation of the reference values staffline_height and staffspace_height using a single column.	45
5.2	Unsuccessful estimation of staffline_height and staffspace_height by vertical runs.	46
5.3	Illustration of the estimation of the reference value line_thickness+spacing using a single column. In practice, sums of consecutive runs are accumulated over the whole image.	47
5.4	Histogram for the music score of Figure 5.2 (score #17).	47
5.5	Histogram of the pairs (black run, white run) for score #17 in Figure 5.2.	48
5.6	Original music score #01.	48
5.7	Histograms for binarized music score #01.	49
5.8	Histograms for gray-level music score #01.	49
6.1	Original music score (detail).	52
6.2	Result of binarizing the music score in Figure 6.1.	52
6.3	Result of binarizing the music score in Figure 6.1 with the adaptive method.	53

6.4	Binarization problems with DRT.	59
6.5	Binarization with Otsu and BLIST methods.	60
6.6	Binarization with BLIST and Adaptive BLIST methods.	61
7.1	An illustrative example of the methodology.	67
7.2	Stable paths on a toy example.	67
7.3	Illustration of stable paths for Figure 7.1(a).	68
7.4	Some examples of music scores from CVC-MUSCIMA database.	76
7.5	Staff removal results. Error Rate (E.R.) in % for each one of the 12 distortions. We show the Error Rate with and without rejection. In case of the Error rate without rejection (No Rej.), we also show the number # of rejected images. From Fornés et al. [44, Table I].	77
8.1	Broken, connected and overlapping symbols.	79
8.2	Variability in sizes and shapes between handwritten and printed scores.	79
8.3	Noise and zones of high density of symbols.	79
8.4	First step in the segmentation procedure: split the score by staffs (set of staff lines).	81
8.5	Diagram of the musical symbols extraction algorithm.	82
8.6	Correlation process.	83
8.7	The result after the clef detection step.	84
8.8	The result after key and time signatures steps.	85
8.9	Closed noteheads detection steps.	86
8.10	Open noteheads with stems detection steps.	86
8.11	The result after the beams detection step.	87
8.12	Insertion of a closed notehead initially eliminated.	87
8.13	Morphological operations applied to the rest symbols: (a) Original rest symbol; (b) Closed rest symbol; (c) Skeleton of the rest symbol; (d) Termination points of the rest symbol.	87
8.14	The result after the rests detection step.	88
8.15	The result after the accents detection step.	88
8.16	Examples of objects that are considered noise and musical symbol.	91
8.17	Diagram of the algorithm for music symbols extraction.	91
8.18	Examples of objects that are considered noise.	92
8.19	Search objects in connected symbols.	93
8.20	Example of broken notes.	94
8.21	The struture of a Combined Neural Network.	95
8.22	Example of measures according to the time signature. The bar lines are represented by the green lines.	96
8.23	Tree of notes representing the relation between them. At the top is the whole note, below that half notes, then quarter notes, eighth notes, and finally sixteenth notes.	97
8.24	The architecture of the syntactic consistency process.	97
A.1	A directed acyclic graphs and its linearization.	125
A.2	Length of a chord through a skeleton point at some angle φ	127
A.3	Example (from [34]).	128

List of Tables

2.1	The most relevant OMR software and programs.	22
3.1	Music notation.	28
3.2	Data set adopted in the different stages of the OMR procedure.	29
3.3	Deformations in the images.	30
3.4	Ranges of deformation parameters used in the tests: min:step:max.	32
5.1	Mean and maximum value of errors (in pixels) in the reference lengths.	50
6.1	Values for the input parameters of the binarization algorithms.	58
6.2	Test results for various global thresholding methods, using different evaluations: difference from reference thresholds values, misclassification error (in percentage), staff detection error rates for missed and false staves (in percentage) with Stable Path and Dalitz.	59
6.3	Test results for various local thresholding methods, using different evaluations (in percentage): misclassification error, Missed Object Pixels, False Object Pixels, staff detection error rates for missed and false staves with Stable Path and Dalitz.	60
7.1	Effect of different deformations on the overall staff detection error rates in percentage: average (standard deviation) of the false detection rate and miss detection rate. See [34] for parameter details.	74
7.2	Detection performance on real music scores in percentage: average (standard deviation).	75
7.3	Effect of different deformations on the overall staff removal error rates in percentage: average (standard deviation).	76
7.4	Removal performance on real music scores (in percentage): average (standard deviation).	76
8.1	Confidence degrees of each class.	89
8.2	Accuracy obtained for the classification models for the secondary classes.	90
8.3	The set of the musical symbols considered in the <i>symbol</i> class.	92
8.4	The set of the musical symbols considered in the <i>Noise and Symbol</i> class.	92
8.5	Accuracy obtained on the 99% CI (in percentage) for the classification models.	95
8.6	Accuracy obtained for the classification models using 400 features before and after the syntactic consistency model.	98
8.7	Results for music symbols extraction.	99
8.8	Confusion Matrix for the results from the Table 8.7.	99
8.9	Results for music symbols extraction through recognition.	99
8.10	Confusion Matrix for the results from the Table 8.9.	99
8.11	Results for music symbols extraction after syntactic consistency without confidence degrees.	100
8.12	Confusion matrix for the results from the Table 8.11.	100
8.13	Results for music symbols extraction after syntactic consistency with confidence degrees.	100

8.14	Confusion matrix for the results from the Table 8.13.	100
8.15	Results for music symbols extraction through recognition after syntactic consistency. . . .	100
8.16	Confusion matrix for the results from the Table 8.15.	100
8.17	Results for music symbols detection and classification after syntactic consistency with confidence degrees, knowing the true positions of barlines and the real time signature.	101
8.18	Confusion matrix for the results from the Table 8.17.	101
A.1	Staff segment extraction errors based on the number of segments in an equivalence class r	129
B.1	The set of the musical symbols considered.	131
B.2	Full set of handwritten and printed music symbols considered.	133
B.3	Accuracy obtained for the handwritten music symbols for the classifiers trained without elastically-deformed symbols.	134
B.4	Accuracy obtained for the printed music symbols for the classifiers trained without elastically-deformed symbols.	135
B.5	Accuracy obtained for the handwritten music symbols for the classifiers trained with elastically-deformed symbols.	136
B.6	Accuracy obtained for the printed music symbols for the classifiers trained with elastically-deformed symbols.	137
B.7	Accuracy on the natural and sharp symbols.	137
B.8	Accuracy obtained with a database of real and printed scores. Input data: vector of 400 binary values.	138
B.9	Accuracy obtained with a database of real and printed scores. Input data: vector of 400 binary values plus 7 music symbol features.	138
B.10	Accuracy on the 99% CI for the expected performance in percentage for separation of types of music scores. Input data: vector of 400 binary values.	138
B.11	Effect of different deformations on the 99% CI for the expected performance in percentage. Input data: vector of 400 binary values.	139
B.12	Full set of handwritten and printed music symbols considered.	139
B.13	Accuracy obtained using the k -NN and LMNN classifiers.	140
B.14	Accuracy obtained using the SVM classifier with different kernels.	141

OMR – Optical Music Recognition
BLIST – Binarization based in Line Spacing and Thickness
DRT – Difference from Reference Threshold
ME – Missclassification Error
MOPx – Missed Object Pixel
FOPx – False Object Pixel
kNN – k-Nearest Neighbour
NNs – Neural Networks
SVMs – Support Vector Machines
RVMs – Relevance Vector Machines
HMMs – Hidden Markov Models
LMNN – Large Margin Nearest Neighbor classification
CNN – Combined Neural Networks
CD – Confidence Degrees
GUI – Graphic User Interface

Part I

Introduction

Introduction

Music is an *art* form whose *medium* is sound. The word derives from Greek *μουσική (τέχνη)*, “(art) of the Muses”¹.

Music notation, the visual manifestation of the interrelated properties of musical sound – pitch, intensity, time, timbre and pace – is a combination and prolonged efforts of hundreds of musicians. They all hoped to express by written symbols the essence of their musical ideas. The final notation was a kind of alphabet, shaped by a general consensus of opinion to serve as a general expressive technique.

Over the years, music has been an important part of the culture of all societies. In the beginning of the Eras, the scholars claimed that the ancient music had its existence based on findings from a range of paleolithic sites, such as bones in which lateral holes have been pierced (flutes). India has one of the oldest musical traditions in the world. Its origins can be found in the oldest of scriptures, part of the Hindu tradition, the Vedas. The Indian classical music appears as a meditation tool for attaining self realization and has one of the most complex and complete musical systems ever developed. Bharat’s *Natyashastra* was the first treatise laying down fundamental principles of dance, music and drama.

The earliest and largest collection of prehistoric musical instruments was found in China and dates back to between 7000 and 6600 BC. In the 9th century, the Arab scholar al-Farabi, who played and invented a variety of musical instruments, devised the Arab tone system of pitch organisation, which is still used in Arabic music. In ancient Greece, music was used for entertainment (theater), celebration and spiritual ceremonies. Music was an important part of education in ancient Greece, and boys were taught music starting at age six.

During the Medieval music period (500–1400), the only European repertory which has survived from before about 800 is the monophonic liturgical plainsong of the Roman Catholic Church, currently called Gregorian chant. From the Renaissance music Era (1400–1600), much of the surviving music of 14th century Europe is secular. The Era of Baroque music (1600–1750) began when the first operas were written and when contrapuntal music became prevalent. A fabulous composer from this time was Johann Sebastian Bach. The music of the Classical period (1750–1800) is characterized by homophonic texture, often featuring a prominent melody with accompaniment. Here a new musical form was incorporated to sonata and concerto, the symphony. Joseph Haydn and Wolfgang Amadeus Mozart are among the central figures of this Classical period. In 1800 – Romantic Era (1800–1890s) – Ludwig van Beethoven and Franz Schubert introduced a more dramatic and expressive style to music. During this time, existing genres, forms, and functions of music were developed, and the emotional and expressive qualities of music came to take precedence over technique and tradition. The melody became the most significant compositional unit.

The radio, in the 20th century, contributed to the increase of the expansion of music. The music focus was characterized by exploration of new rhythms, styles, and sounds – jazz, blues, soul, and country. Igor Stravinsky was one of the composers that influenced the art music in this century. The advent of the Internet has transformed the experience of music, partly through the increased ease of ac-

¹<http://en.wikipedia.org/wiki/Music>

cess to music and the increased choice. Online communities like YouTube and MySpace are examples of these facilities in the distribution of music.

Music has expanded in many several music styles and for many different purposes, even educational or therapy. The centrality of music in the cultural heritage of any society and the importance of cultural diversity, as necessary for humankind as biodiversity is for nature, makes policies to promote and protect cultural diversity an integral part of sustainable development².

1.1 OMR System Project

Printed documents and handwritten manuscripts deteriorate over time, causing a significant amount of information to be permanently lost. Among such perishable documents, musical scores are especially problematic. Across the world, there are many dedicated national and international programs and projects which have been focused on the preservation of huge volumes of such documents. Digitization has been commonly used as a possible tool for preservation, offering easy duplications, distribution, and digital processing. However, a machine-readable symbolic format from the music scores is needed to facilitate operations such as search, retrieval and analysis. The manual transcription of music scores into an appropriate digital format is very time consuming. The development of general image processing methods for object recognition has contributed to the development of several important algorithms for optical music recognition. These algorithms have been central to the development of systems to recognize and encode music symbols for a direct transformation of sheet music into a machine-readable symbolic format.

The work developed in this thesis encompasses detection and recognition of musical symbols in handwritten scores. It aims to overcome the inherent problems of this type of scores using the most recent techniques of machine learning and artificial intelligence. This thesis is incorporated in an OMR project founded at Instituto de Engenharia de Sistemas e Computadores do Porto (INESC Porto) with a partnership with Escola Superior de Música e das Artes do Espectáculo (ESMAE). One of the project objectives is the creation of a web-based system providing generalized access to a wide *corpus* of handwritten and unpublished music encoded in digital format – see Figure 1.1.



Figure 1.1: Web-based system interface.

²<http://www.unesco.org/bpi/eng/unescopress/2001/01-112e.shtml>

A database that centralizes as much information as possible can serve to preserve the musical heritage in an innovative way [19, 20]. The architecture for the proposed system is represented in Figure 1.2.

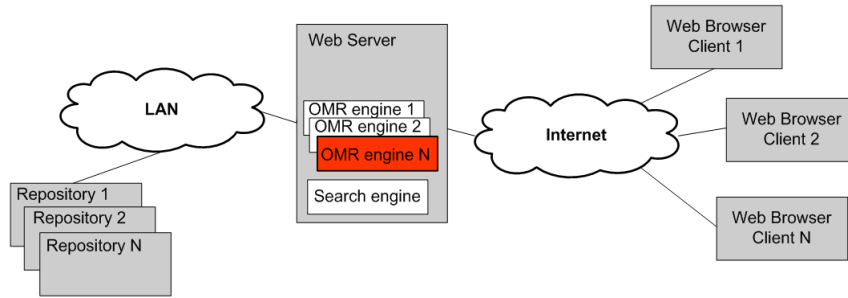


Figure 1.2: Generic system architecture.

The system is composed by three different entities: repository, web server and web browser. Briefly, the Repository module stores the original scanned score, the digital counterpart in MusicXML and all the descriptive metadata inserted by the user. All the remaining system contents, such as the user information, are also stored in this entity. The Web Server is the user access point to the system as well as to all of its processing modules run on the server, encompassing the search engine and the optical recognition engine for the musical scores. The Web Server interacts with the Repository and with the Web Browser, which establishes the interface between the user and the system. The user interface on a Web Browser allows the complete management of the musical scores and associated metadata, as well as carrying out the system administration tasks.

This thesis is part of this longer project to bring innovation in an area that needs algorithms to recognize handwritten scores and improvements in the performance of existing ones. It has the ambition to develop a formal model that covers the bidimensional structure of the musical notation, in a consistency and robust way. Moreover, the research of the potential of the machine learning techniques and artificial intelligence in order to merge knowledge and make decisions is a further goal of this thesis. The digitization and preservation of a wide *corpus* of handwritten scores in a way never before explored is also a target in mind.

1.2 OMR Architecture

Breaking down the problem of transforming a music score into a graphical music-publishing file in simpler operations is a common but complex task. This is consensual among most authors that work in the field.

The main objectives of an OMR system are the recognition, the representation and the storage of musical scores in a machine-readable format. An OMR program should thus be able to recognize the musical content and make the semantic analysis of each musical symbol of a music work. In the end, all the musical information should be saved in an output format that is easily readable by a computer.

A typical framework for the automatic recognition of a set of music sheets encompasses four main stages (see Figure 1.3):

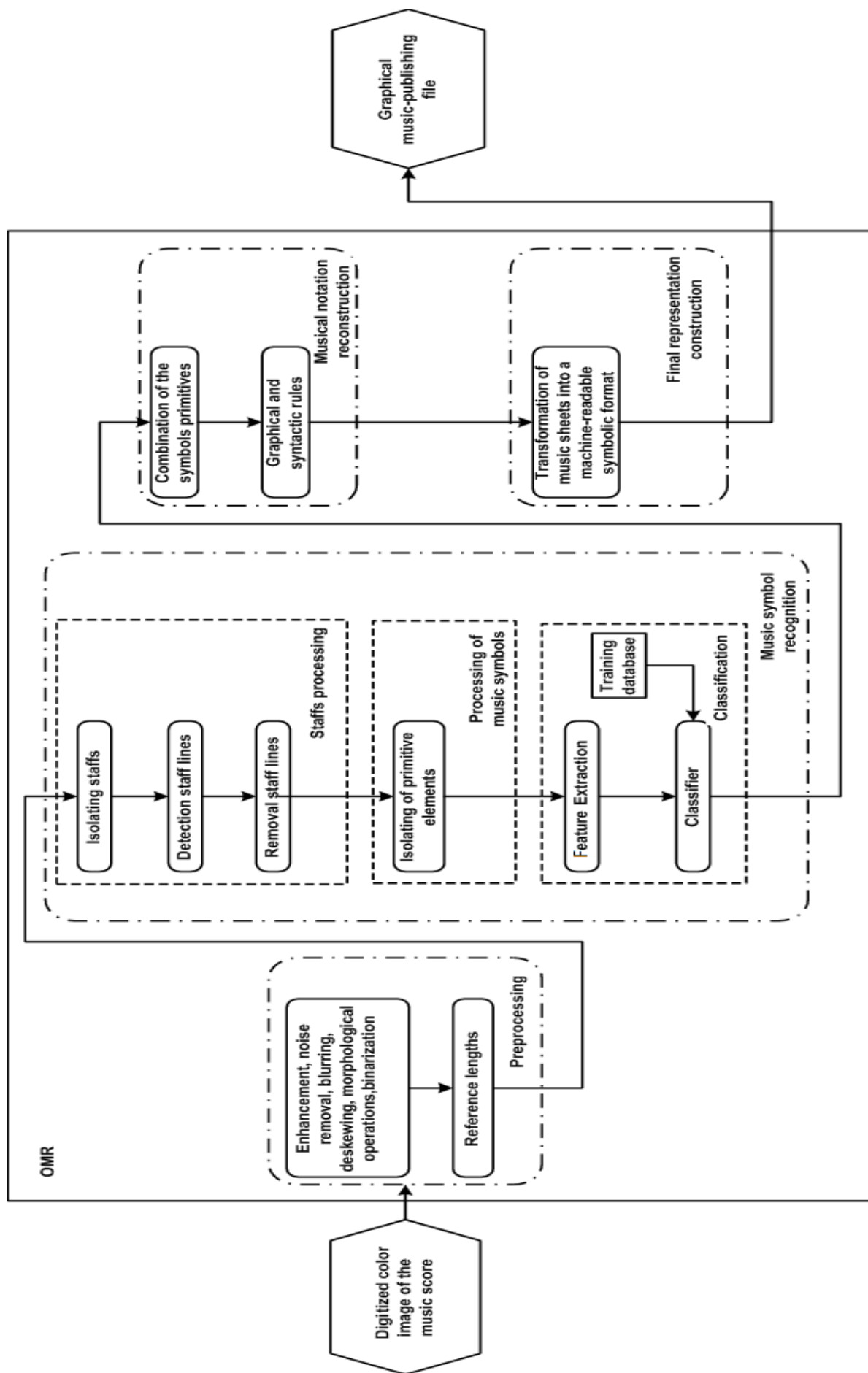


Figure 1.3: Typical architecture of an OMR processing system.

1. image preprocessing;
2. recognition of musical symbols;
3. reconstruction of the musical information in order to build a logical description of musical notation;
4. construction of a musical notation model to be represented as a symbolic description of the musical sheet.

For each of the stages described above, different methods exist to perform the respective task.

In the image preprocessing stage, several techniques – e.g. enhancement, binarization, noise removal, blurring, de-skewing – can be applied to the music score in order to make the recognition process more robust and efficient. The reference lengths staff line thickness (`staffline_height`) and vertical line distance within the same staff (`staffspace_height`) are often computed, providing the basic scale for relative size comparisons (Figure 2.3).

The output of the image preprocessing stage constitutes the input for the next stage, the recognition of musical symbols. This is typically further subdivided into three parts: (1) staff line detection and removal, to obtain an image containing only the musical symbols; (2) symbol primitive segmentation; and (3) symbol recognition. In this last stage the classifiers usually receive raw pixels as input features. However, some works also consider higher-level features, such as information about the connected components or the orientation of the symbol. Classifiers are built by taking a set of labeled examples of music symbols and randomly split them into training and test sets. The best parameterization for each model is normally found based on a cross validation scheme conducted on the training set.

The third and fourth stages (musical notation reconstruction and final representation construction) can be intrinsically intertwined. In the stage of musical notation reconstruction, the symbol primitives are merged to form musical symbols. In this step, graphical and syntactic rules are used to introduce context information to validate and solve ambiguities from the previous module (music symbol recognition). Detected symbols are interpreted and assigned a musical meaning. In the fourth and final stage (final representation construction), a format of musical description is created with the previously produced information. The system output is a graphical music-publishing file, like MIDI or MusicXML.

Some authors use several algorithms to perform different tasks in each stage, such as using an algorithm for detecting noteheads and a different one for detecting the stems. For example, Byrd and Schindele [18] and Knopke and Byrd [73] use a voting system with a comparison algorithm in order to merge the best features of several OMR algorithms in order to produce better results.

1.3 Objectives

This thesis has the ambition to achieve better results from the ones presented in the actual state-of-the-art methods that are incapable to deal in a robust way with the inconsistencies of the writing music. Hence, this work will encompass the research of new procedures for OMR for handwritten and printed scores. The study presupposes the comprehension of image processing processes, pattern recognition techniques, and machine learning. The proposed methodology wants to include in a natural way contextual information and music writing rules in the music symbols recognition. Moreover, the inclusion of syntactic and semantic musical rules as prior knowledge in the process is also an aim. The explo-

ration of global constraints associated to musical rules should allow to overcome the existing problems in the current algorithms.

1.4 Contributions and Related Publications

This dissertation presents the following contributions for the preservation and the general access to musical and cultural heritage:

1. The creation of a database of real scores with its segmented references: binary images, detection and removal of the staff lines and classes and positions of the music symbols.
2. Analysis and study of different binarization methods applied to music scores, which has never been done.
3. The introduction to the music analysis community of a robust method to estimate the staff line thickness and spacing in binary and gray-level music scores.
4. The introduction to the music analysis community of the Binarization algorithm based in LIne Spacing and Thickness (BLIST) to binarize the music scores.
5. The introduction to the music analysis community of the algorithm for staff lines detection based in the Stable Path approach.
6. Integration of graphical and syntactic rules in the automatic extraction algorithm of the music symbols.
7. New studies and analysis conducted in the phase of musical symbols classification.

The work related with this thesis already resulted in the publication of the followings journals:

- “Optical music recognition - state-of-the-art and open issues for handwritten music scores”, Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso, in *International Journal of Multimedia Information Retrieval*, 2012.
- “Optical Recognition of Music Symbols: a comparative study”, Ana Rebelo, Artur Capela and Jaime S. Cardoso, in *International Journal on Document Analysis and Recognition*, volume 13, pages 19–31, 2010.
- “Staff Detection with Stable Paths”, Jaime S. Cardoso, Artur Capela, Ana Rebelo, Carlos Guedes, Joaquim Pinto da Costa, in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, volume 31, pages 1134–1139, 2009.

And the following conference papers:

- “Music score binarization based on domain knowledge”, Telmo Pinto, Ana Rebelo, Gilson Giraldo and Jaime S. Cardoso, in *In Proceedings of 5th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2011.
- “A Method for Music Symbols Extraction based on Musical Rules”, Ana Rebelo, Filipe Paszkiewicz, Carlos Guedes, Andre R. S. Marcal, and Jaime S. Cardoso, in *In Bridges: Mathematical Connections in Art, Music, and Science (BRIDGES)*, 2011.

- “Metric Learning for Music Symbols Recognition”, Ana Rebelo, Jakub Tkaczuk, Ricardo Sousa and Jaime S. Cardoso, in *In Proceedings of 10th International Conference on Machine Learning and Applications (ICMLA)*, 2011.
- “Robust staffline thickness and distance estimation in binary and gray-level music scores”, Jaime S. Cardoso and Ana Rebelo, *In Proceedings of The Twentieth International Conference on Pattern Recognition*, pages 1856–1859, 2010.

I also participated in the following portuguese scientific encounters:

- “Music Symbols Extraction Based on Domain Knowledge”, Ana Rebelo and Jaime S. Cardoso, in *RECPAD*, 2011.
- “Content aware music score pre-processing”, Ana Rebelo and Jaime S. Cardoso, in *Proceedings of 16th Portuguese Conference on Pattern Recognition (RECPAD)*, 2010.

The following paper will also be submitted:

- “Global constraints for syntactic consistency in OMR”, Ana Rebelo, Andre R. S. Marcal and Jaime S. Cardoso, in *In Proceedings of 6th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2013.

1.5 Thesis' Structure

This thesis is organized in 9 chapters that describe the work developed over a period of four years. After the Introduction, the description of the works related with this thesis will be made in Chapter 2.

In Chapter 3 the database used is presented, while in Chapter 4 the techniques that were used in this thesis are given.

The image preprocessing stage starts in the Chapter 5. Most OMR algorithms rely on an estimation of the staffline thickness and the vertical line distance within the same staff. Subsequent operation can use these values as references, dismissing the need for some predetermined threshold values. In this work an improvement on previous conventional estimates for these two reference lengths is presented. A new method for binarized music scores extending the approach for gray-level music scores is proposed.

In Chapter 6, a description of the proposed binarization algorithm is described. The applied error metrics to test the algorithm and the obtained results are also presented.

A method for the automatic detection of staff lines is proposed in Chapter 7. This Chapter describes the first step of the music symbol recognition module. The proposed paradigm uses the image as a graph, where the staff lines are considered as connected paths between the two margins of the image. Firstly, the concept of stable path is introduced in order to improve the computational performance of the method. Secondly, the design of the weights on the graph resulting from the music score is generalized to differentiate black pixels belonging to the staff lines from black pixels resulting from the music symbols. Finally, the post processing is refined, improving the overall performance. A further development is the study of new staff removal algorithms, by incorporating the proposed staff line detection on standard staff removal algorithms. The experimental work reported at the end of the

chapter includes a thorough testing on synthetic and real scores, with the latter manually processed to be used as ground truth.

In Chapter 8 the algorithms for detection of the musical symbols are addressed. This phase consist in localizing and isolating the musical objects in order to identify them. In section 8.1 the music symbol recognition was composed by two main steps: first the image was segmented in order to detect and isolate the primitives elements and then the symbols were classified. In section 8.2 the music symbols were simultaneously segmented and recognized. A comparative study of the most common classification algorithms applied to music symbols, extending and updating previous comparative works [106] is provided in Section 8.1.2. The performances of HMMs, RVMs, SVMs, NNs and k-NNs methods are compared using both real and synthetic scores. The work presented here can open new research paths towards a novel automatic musical symbols recognition module for handwritten scores.

This thesis finishes with Chapter 9 where conclusions are presented and future work is discussed.

The musical score is the primary artifact for the transmission of musical expression for non-aural traditions. Over the centuries, musical scores have evolved dramatically in both symbolic content and quality of presentation. The appearance of musical typographical systems in the late 19th century and, more recently, the emergence of very sophisticated computer music manuscript editing and page-layout systems, illustrate the continuous absorption of new technologies into systems for the creation of musical scores and parts. Until quite recently, most composers of all genres – film, theater, concert, sacred music – continued to use the traditional “pen and paper” finding manual input to be the most efficient. Early computer music typesetting software developed in the 1970’s and 1980’s produced excellent output but was awkward to use. Even the introduction of data entry from musical keyboard (MIDI piano for example) provided only a partial solution to the rather slow keyboard and mouse GUI’s. There are many scores and parts still being “hand written”. Thus, the demand for a robust and accurate Optical Music Recognition (OMR) system remains.

The research field of OMR began with Pruslin [98] and Prerau [96] and, since then, has undergone many important advancements. Several surveys and summaries have been presented to the scientific community: Kassler [70] reviewed two of the first dissertations on OMR, Blostein and Baird [13] published an overview of OMR systems developed between 1966 and 1992, Bainbridge and Bell [4] published a generic framework for OMR (subsequently adopted by many researchers in this field), and both Homenda [60] and Rebelo et al. [106] presented pattern recognition studies applied to music notation. Jones et al. [66] presented a study in music imaging, which included digitization, recognition and restoration, and also provided a well detailed list of hardware and software in OMR together with an evaluation of three OMR systems.

Access to low-cost flat-bed digitizers during the late 1980’s contributed to an expansion of OMR research activities. Several commercial OMR software have appeared, but none with a satisfactory performance in terms of precision and robustness, in particular for handwritten music scores [9]. Until now, even the most advanced recognition products including Notescan in Nightingale², Midiscan in Finale³, Photoscore in Sibelius⁴ and others such as Smartscore⁵ and Sharpeye⁶, cannot identify every musical symbol. Furthermore, these products are focused primarily on recognition of typeset and printed music documents and while they can produce quite good results for these documents, they do not perform very well with hand-written music. The bi-dimensional structure of musical notation, revealed by the presence of the staff lines alongside the existence of several combined symbols organized around the noteheads, poses a high level of complexity in the OMR task.

In this chapter, we survey the relevant methods and models in the literature for the optical recognition of musical scores.

*Some portions of this chapter appears in [107]

²<http://www.ngale.com/>.

³<http://www.finalemusic.com/>.

⁴<http://www.neuratron.com/photoscore.htm>.

⁵<http://www.musitek.com/>.

⁶<http://www.music-scanning.com/>.

2.1 Image Pre-processing

The music scores processed by the state-of-art algorithms, described in the following sections, are mostly written in a standard modern notation (from the 20th century). However, there are also some methods proposed for 16th and 17th century printed music. Figure 2.1 shows typical music scores used for the development and testing of algorithms in the scientific literature. In most of the proposed works the music sheets were scanned at a resolution of 300 dpi [57, 85, 48, 114, 73, 34, 22, 106, 47], but other resolutions were also considered: 600 dpi [112, 75] or 400dpi [99, 124]. No studies have been carried out in order to evaluate the dependency of the proposed methods on other resolution values, thus restricting the quality of the objects presented in the music scores, and consequently the performance of OMR algorithms.



Figure 2.1: Some examples of music scores used in the state-of-art algorithms. (a) from Rebelo [104, Fig.4.4a].)

In digital image processing, as in all signal processing systems, different techniques can be applied to the input, making it ready for the detection steps. The motivation is to obtain a more robust and efficient recognition process. Enhancement [57], binarization (e.g. [53, 57, 85, 48, 22, 55, 126]), noise removal (e.g. [53, 57, 124, 126]), blurring [57], de-skewing (e.g. [53, 57, 85, 48, 126]) and morphological operations [57] are the most common techniques for preprocessing music scores.

2.1.1 Binarization

Almost all OMR systems start with a binarization process. This means that the digitized image must be analyzed, in order to determine what is useful (the objects, being the music symbols and staves) and what is not (the background, noise). To make binarization an automatic process, many algorithms have been proposed in the past, with different success rates, depending on the problem at hand. Binarization has the big virtue in OMR of facilitating the following tasks by reducing the amount of information that needs to be processed. In turn, this results in higher computational efficiency (more important in the past than nowadays) and eases the design of models to tackle the OMR task. It has been easier to propose algorithm for line detection, symbol segmentation and recognition in binary images than in grayscale or colour images. This approach is also supported by the typical binary nature of music scores. Usually, the author does not aim to portray information in different shades of gray; it is more a

consequence of the writing or of the digitization process. However, since binarization often introduces artifacts, the advantages of binarization in the complete OMR process are not clear.

Burgoyne et al. [17] and Pugin et al. [100] presented a comparative evaluation of image binarization algorithms applied to 16th-century music scores. Both works used Aruspix, a software application for OMR which provides symbol-level recall and precision rate to measure the performance of different binarization procedures. In [17] they worked with a set of 8000 images. The best result was obtained with the Brink and Pendock [15]’s method. The adaptive algorithm with the highest ranking was Gatos et al. [54]. Nonetheless, the binarization of the music score still need attention with researchers invariably using standard binarization procedures, such as the Otsu’s method (e.g. [57, 99, 22, 106]). The development of binarization methods specific to music scores could potentially provides better performances than the generic counterparts’, and leverages the performance of subsequent operations [95].

The fine-grained categorization of existing techniques presented in Figure 2.2 follows the survey in [119], where the classes were chosen according to the information extracted from the image pixels. Despite this labelling, the categories are essentially organized into two main topics: global and adaptive thresholds.

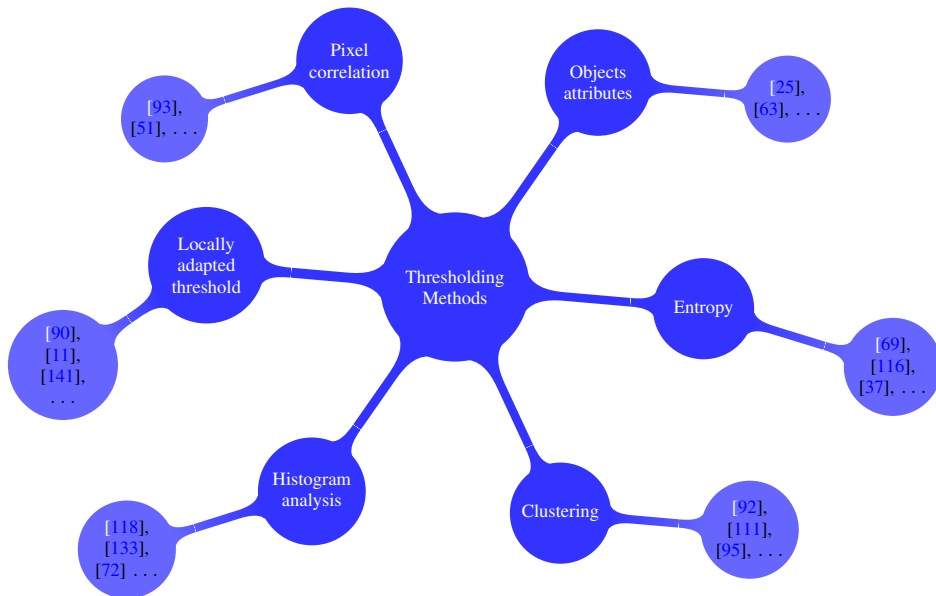


Figure 2.2: Landscape of automated thresholding methods. From Pinto et al [95, Fig.1].

Global thresholding methods apply one threshold to the entire image. Ng and Boyle [86] and Ng et al. [88] have adopted the technique developed by Ridler and Calvard [111]. This iterative method achieves the final threshold through an average of two sample means ($T = (\mu_b + \mu_f)/2$). Initially, a global threshold value is selected for the entire image and then a mean is computed for the background pixels (μ_b) and for the foreground pixels (μ_f). The process is repeated based on the new threshold computed from μ_b and μ_f , until the threshold value does not change any more. According to [131, 132], Otsu’s procedure is ranked as the best and the fastest of these methods [92]. In the OMR field, several research works have used this technique [57, 99, 22, 106, 126].

In adaptive binarization methods, a threshold is assigned to each pixel using local information from the image. Consequently, the global thresholding techniques can extract objects from uniform backgrounds at a high speed, whereas the local thresholding methods can eliminate dynamic backgrounds

although with a longer processing time. One of the most used method is Niblack [90]’s method which uses the mean and the standard deviation of the pixel’s vicinity as local information for the threshold decision. The research work carried out by [124, 46, 47] applied this technique to their OMR procedures.

Only recently the domain knowledge has been used at the binarization stage in the OMR area. The work presented in [95] proposes a new binarization method which not only uses the raw pixel information, but also considers the image content. The process extracts content related information from the grayscale image, the staff line thickness (staff- line_height) and the vertical line distance within the same staff (staffspace_height), to guide the binarization procedure. The binarization algorithm was designed to maximize the number of pairs of consecutive runs summing staffline_height + staffspace_height. The authors suggest that this maximization increases the quality of the binarized lines and consequently the subsequent operations in the OMR system.

Until now Pinto et al. [95] seems to be the only threshold method that uses content of gray-level images of music scores deliberately to perform the binarization.

2.1.2 Reference Lengths

In the presence of a binary image most OMR algorithms rely on an estimation of the staff line thickness and the distance that separates two consecutive staff lines – see Figure 2.3.



Figure 2.3: The characteristic page dimensions of staffline_height and staffspace_height. From Cardoso and Rebelo [23].

Further processing can be performed based on these values and be independent of some predetermined magic numbers. The use of fixed threshold numbers, as found in other areas, causes systems to become inflexible, making it more difficult for them to adapt to new and unexpected situations.

The well-known run-length encoding (RLE), which is a very simple form of data compression in which runs of data are represented as a single data value and count, is often used to determine these reference values (e.g. [53, 114, 34, 22, 41]) – another technique can be found in [126]. By encoding each column of a digitized score using RLE, the most common black-run represents the staffline_height and the most common white-run represents the staffspace_height.

Nonetheless, there are music scores with high levels of noise, not only because of the low quality of the original paper in which it is written, but also because of the artifacts introduced during digitization and binarization. These aspects make the results unsatisfactory, impairing the quality of subsequent operations. Figure 2.4 illustrates this problem. For this music score, we have pale staff lines that broke up during binarization providing the conventional estimation staffline_height = 1 and staffspace_height = 1 (the true values are staffline_height = 5 and staffspace_height = 19).



(a) Original music score #17.



(b) Score binarized with Otsu's method.

Figure 2.4: Example of an image where the estimation of `staffline_height` and `staffspace_height` by vertical runs fails. From Cardoso and Rebelo [23, Fig.2].

The work suggested by Cardoso and Rebelo [23], which encouraged the work proposed in [95], presents a more robust estimation of the sum of `staffline_height` and `staffspace_height` by finding the most common sum of two consecutive vertical runs (either black run followed by white run or the reverse). In this manner, to reliably estimate `staffline_height` and `staffspace_height` values, the algorithm starts by computing the 2D histogram of the pairs of consecutive vertical runs and afterwards it selects the most common pair for which the sum of the runs equals `staffline_height + staffspace_height`.

2.2 Staff Line Detection and Removal

Staff line detection and removal are fundamental stages in many optical music recognition systems. The reason to detect and remove the staff lines lies on the need to isolate the musical symbols for a more efficient and correct detection of each symbol present in the score. Notwithstanding, there are authors who suggested algorithms without the need to remove the staff lines [88, 8, 78, 57, 121, 99, 10]. In here, the decision is between simplification to facilitate the following tasks with the risk of introducing noise. For instance, symbols are often broken in this process, or bits of lines that are not removed are interpreted as part of symbols or new symbols. The issue will always be related to the preservation of as much information as possible for the next task, with the risk of increasing computational demand and the difficulty of modelling the data.

Staff detection is complicated due to a variety of reasons. Although the task of detecting and removing staff lines is completed fairly accurately in some OMR systems, it still represents a challenge.

The distorted staff lines are a common problem in both printed and handwritten scores. Staff lines are often not straight or horizontal (due to wrinkles or poor digitization), and in some cases hardly parallel to each other. Moreover, most of these works are old, which means that the quality of the paper and ink has decreased severely. Another interesting setting is the common modern case where music notation is handwritten on paper with preprinted staff lines.

The simplest approach consists of finding local maxima on the horizontal projection of the black pixels of the image [53, 102]. Assuming straight and horizontal lines, these local maxima represent line positions. Several horizontal projections can be made with different image rotation angles, keeping the image where the local maximum is higher. This eliminates the assumption that the lines are always horizontal. Miyao and Nakano [83] uses Hough Transform to detect staff lines. An alternative strategy for identifying staff lines is to use vertical scan lines [24]. This process is based on a Line Adjacency Graph (LAG). LAG searches for potential sections of lines: sections that satisfy criteria related to aspect ratio, connectedness and curvature. More recent works present a sophisticated use of projection techniques combined in order to improve the basic approach [3, 8, 114, 10].

Fujinaga [53] incorporates a set of image processing techniques in the algorithm, including run-length coding (RLC), connected-component analysis, and projections. After applying the RLC to find the thickness of staff lines and the space between the staff lines, any vertical black run that is more than twice the staff line height is removed from the original. Then, the connected components are scanned in order to eliminate any component whose width is less than the staff space height. After a global de-skewing, taller components, such as slurs and dynamic wedges are removed.

Other techniques for finding staff lines include the grouping of vertical columns based on their spacing, thickness and vertical position on the image [110], rule-based classification of thin horizontal line segments [80], and line tracing [96, 113, 126]. The methods proposed in [84, 123] operate on a set of *staff segments*, with methods for linking two segments horizontally and vertically and merging two overlapped segments. Dutta et al. [41] proposed a similar but simpler procedure than previous ones. The authors considered a staff line segment as an horizontal connection of vertical black runs with uniform height, and validating it using neighboring properties. The work by Dalitz et al. [34] is an improvement on the methods of [84, 123].

In spite of the variety of methods available for staff lines detection, they all have some limitations. In particular, lines with some curvature or discontinuities are inadequately resolved. The dash detector [77] is one of a few works that try to handle discontinuities. The dash detector is an algorithm that searches the image, pixel by pixel, finding black pixel regions that it classifies as stains or dashes. Then, it tries to unite the dashes to create lines.

A common problem to all the above mentioned techniques is that they try to build staff lines from local information, without properly incorporating global information in the detection process. None of the methods tries to define a reasonable process from the intrinsic properties of staff lines, namely the fact that they are the only extensive black objects on the music score. Usually, the most interesting techniques arise when one defines the detection process as the result of optimizing some global function. In [22], the authors proposed a graph-theoretic framework where the staff line is the result of a global optimization problem. The new staff line detection algorithm suggests using the image as a graph, where the staff lines result as connected paths between the two lateral margins of the image. A staff line can be considered a connected path from the left side to the right side of the music score. As staff lines are almost the only extensive black objects on the music score, the path to look for is the

shortest path between the two margins if paths (almost) entirely through black pixels are favoured. The performance was experimentally supported on two test sets adopted for the qualitative evaluation of the proposed method: the test set of 32 synthetic scores from [34], where several known deformations were applied, and a set of 40 real handwritten scores, with ground truth obtained manually.

2.3 Symbol Segmentation and Recognition

The extraction of music symbols is the operation following the staff line detection and removal. The segmentation process consists of locating and isolating the musical objects in order to identify them. In this stage, the major problems in obtaining individual meaningful objects are caused by printing and digitization, as well as paper degradation over time. The complexity of this operation concerns not only the distortions inherent to staff lines, but also broken and overlapping symbols, differences in sizes and shapes and zones of high density of symbols. The segmentation and classification process has been the object of study in the research community (e.g. [26, 8, 114, 130]).

The most usual approach for symbol segmentation is an hierarchical decomposition of the music image. A music sheet is first analyzed and split by staves and then the elementary graphic symbols are extracted: noteheads, rests, dots, stems, flags, etc. (e.g. [86, 110, 83, 26, 39, 57, 106, 126]). Although in some approaches [106] noteheads are joined with stems and also with flags for the classification phase, in the segmentation step these symbols are considered to be separate objects. In this manner, different methods use equivalent concepts for primitive symbols.

Usually, the primitive segmentation step is made along with the classification task [114, 130]; however there are exceptions [53, 8, 10]. Mahoney [80] builds a set of candidates to one or more symbol types and then uses descriptors to select the matching candidates. Carter [24] and Dan [36] use a LAG to extract symbols. The objects resulting from this operation are classified according to the bounding box size, the number and organization of their constituent sections. Reed and Parker [110] also uses LAGs to detect lines and curves. However, accidentals, rests and clefs are detected by a character profile method, which is a function that measures the perpendicular distance of the object's contour to reference axis, and noteheads are recognized by template matching. Other authors have chosen to apply projections to detect primitive symbols [97, 53, 8, 10]. The recognition is done using features extracted from the projection profiles. In [53], the k-nearest neighbor (kNN) rule is used in the classification phase, while neural networks (NNs) is the classifier selected in [86, 83, 8, 10]. Choudhury et al. [26] proposed the extraction of symbol features, such as width, height, area, number of holes and low-order central moments, whereas Taubman [127] preferred to extract standard moments, centralized moments, normalized moments and Hu moments. Both systems classify the music primitives using the kNN method.

Randriamahefa et al. [102] proposed a structural method based on the construction of graphs for each symbol. These are isolated by using a region growing method and thinning. In [114] a fuzzy model supported on a robust symbol detection and template matching was developed. This method is set to deal with uncertainty, flexibility and fuzziness at the level of the symbol. The segmentation process is addressed in two steps: individual analysis of musical symbols and fuzzy model. In the first step, the vertical segments are detected by a region growing method and template matching. The beams are then detected by a region growing algorithm and a modified Hough Transform. The remaining symbols are

extracted again by template matching. As a result of this first step, three recognition hypotheses occur, and the fuzzy model is then used to make a consistent decision.

Other techniques for extracting and classifying musical symbols include rule-based systems to represent the musical information, a collection of processing modules that communicate by a common working memory [113] and pixel tracking with template matching [130]. Toyama et al. [130] check for coherence in the primitive symbols detected by estimating overlapping positions. This evaluation is carried out using music writing rules. Coüasnon [29, 27] proposed a recognition process entirely controlled by grammar which formalizes the musical knowledge. Bainbridge [3] uses PRIMELA (PRIMitive Expression LAnguage) language, which was created for the CANTOR (CANTerbury Optical music Recognition) system, in order to recognise primitive objects. In [110] the segmentation process involves three stages: line and curves detection by LAGs, accidentals, rests and clefs detection by a character profile method and noteheads recognition by template matching. Fornés et al. [45] proposed a classifier procedure for handwritten symbols using the Adaboost method with a Blurred Shape Model descriptor.

It is worth mentioning that in some works, we assist to a new line of approaches that avoid the prior segmentation phase in favour of methods that simultaneously segment and recognize, thus segmenting through recognition. In [99, 101] the segmentation task is based on Hidden Markov Models (HMMs). This process performs segmentation and classification simultaneously. The extraction of features directly from the image frames has advantages. Particularly, it avoids the need to segment and track the objects of interest, a process with a high degree of difficulty and prone to errors. However, this work applied this technique only in very simple scores, that is, scores without slurs or more than one symbol in the same column and staff.

In [88] a framework based on a mathematical morphological approach commonly used in document imaging is proposed. The authors applied a skeletonization technique with an edge detection algorithm and a stroke direction operation to segment the music score. Goecke [57] applies template matching to extract musical symbols. In [127] the symbols are recognized using statistical moments. This way, the proposed OMR system is trained with strokes of musical symbols and a statistical moment is calculated for each one of them; the class for an unknown symbol is assigned based on the closest match. In [48] the authors start by using median filters with a vertical structuring element to detect vertical lines. Then they apply a morphological opening using an elliptical structuring element to detect noteheads. The bar lines are detected considering its height and the absence of noteheads in its extremities. Clef symbols are extracted using Zernike moments and Zoning, which code shapes based on the statistical distribution of points. Although a good performance was verified in the detection of these specific symbols, the authors did not extract the other symbols that were also present on a music score and are indispensable for a complete optical music recognition. In [106] the segmentation of the objects is based on an hierarchical decomposition of a music image. A music sheet is first analyzed and split by staves. Subsequently, the connected components are identified. To extract only the symbols with appropriate size, the connected components detected in the previous step are selected. Since a bounding box of a connected component can contain multiple connected components, care is taken in order to avoid duplicate detections or failure to detect any connected component. In the end, all music symbols are extracted based on their shape. In [126] the symbols are extracted using a connected components process and small elements are removed based on their size and position on the score. The classifiers adopted were the kNN, the Mahalanobis distance and the Fisher discriminant.

Some studies were conducted in the music symbols classification phase, more precisely the comparison of results between different recognition algorithms. Homenda and Luckner [61] studied decision trees and clustering methods. The symbols were distorted by noise, printing defects, different fonts, skew and curvature of scanning. The study starts with the extraction of some symbols features. Five classes of music symbols were considered. Each class had 300 symbols extracted from 90 scores. This investigation encompassed two different classification approaches: classification with and without rejection. In the later case, every symbol belongs to one of the given classes, while in the classification with rejection, not every symbol belongs to a class. Thus, the classifier should decide if the symbol belongs to a given class or if it is an extraneous symbol and should not be classified. Rebelo et al [106] carried out an investigation on four classification methods, namely Support Vector Machines (SVMs), NNs, kNN and HMMs. The performances of these methods were compared using both real and synthetic scores. In [109] the authors proposed a method to learn a Mahalanobis distance for the kNN and extended it to SVMs. The idea was to take advantage of distance learning to improve the classification accuracy. The learnt distance metric was directly connected with the application domain and the adopted symbol representation.

A more recent procedure for pattern recognition is the use of classifiers with a reject option [32, 58, 122]. The method integrates a confidence measure in the classification model in order to reject uncertain patterns, namely broken and touching symbols. The advantage of this approach is the minimization of misclassification errors in the sense that it chooses not to classify certain symbols (which are then manually processed).

Lyrics recognition is also an important issue in the OMR field, since lyrics make the music document even more complex. In [16] techniques for lyric editor and lyric lines extraction were developed. After staff lines removal, the authors computed baselines for both lyrics and notes, stressing that baselines for lyrics would be highly curved and undulating. The baselines are extracted based on local minima of the connected components of the foreground pixels. This technique was tested on a set of 40 images from the Digital Image Archive of Medieval Music. In [56] an overview of existing solutions to recognize the lyrics in Christian music sheets is described. The authors stress the importance of associating the lyrics with notes and melodic parts in order to provide more information to the recognition process. Resolutions for page segmentation, character recognition and final representation of symbols are presented.

Despite the number of techniques already available in the literature, research on improving symbol segmentation and recognition is still important and necessary. All OMR systems depend on this step.

2.4 Musical Notation Construction and Final Representation

The final stage in a music notation construction engine is to extract the musical semantics from the graphically recognised shapes, and store them in a musical data structure. Essentially, this involves combining the graphically recognized musical features with the staff systems to produce a musical data structure representing the meaning of the scanned image. This is accomplished by interpreting the spatial relationships between the detected primitives found in the score. If we are dealing with optical character recognition (OCR) this is a simple task, because the layout is predominantly one-dimensional. However, in music recognition, the layout is much more complex. The music is essentially two-dimensional, with pitch represented vertically and time horizontally. Consequently,

positional information is extremely important. The same graphical shape can mean different things in different situations. For instance, to determine if a curved line between two notes is a slur or a tie, it is necessary to consider the pitch of the two notes. Moreover, musical rules involve a large number of symbols that can be spatially far from each other in the score.

Several research works have suggested the introduction of the musical context in the OMR process by a formalization of musical knowledge using a grammar (e.g. [97, 96, 28, 6, 110, 10]). The grammar rules can play an important role in music creation. They specify how the primitives are processed, how a valid musical event should be made, and even how graphical shapes should be segmented. Andronico and Ciampa [1] and Prerau [97] were pioneers in this area. One of Fujinaga's first works focused on the characterization of music notation by means of a *context-free* and $LL(k)$ grammar. Coüasnon [28, 30] also based their works on a grammar, which is essentially a description of the relations between the graphical objects and a parser, which is the introduction of musical context with syntactic or semantic information. The author claims that this approach will reduce the risk of generating errors imposed during the symbols extraction, using only very local information. The proposed grammar is implemented in λ Prolog, a higher dialect of Prolog with more expressive power, with semantic attributes connected to *C* libraries for pattern recognition and decomposition. The grammar is directly implemented in λ Prolog using Definite Clause Grammars (DCG's) techniques. It has two levels of parsing: a graphical one corresponding to the physical level and a syntactic one corresponding to the logical level. The parser structure is a list composed of segments (non-labeled) and connected components, which do not necessarily represent a symbol. The first step of the parser is the labeling process, the second is the error detection. Both operations are supported by the context introduced in the grammar. However, no statistical results are available for this system.

Bainbridge [3] also implemented a grammar-based approach using DCG's to specify the relationships between the recognized musical shapes. This work describes the CANTOR system, which has been designed to be as general as possible by allowing the user to define the rules that describe the music notation. Consequently, the system is readily adaptable to different publishing styles in Common Music Notation (CMN). The authors argue that their method overcame the complexity imposed in the parser development operation proposed in [28, 30]. CANTOR avoids such drawbacks by using a *bag*⁷ of tokens instead of using a *list* of tokens. For instance, instead of getting a unique next symbol, the grammar can "request" a token, e.g. a notehead, from the bag, and if its position does not fit in with the current musical feature that is being parsed, then the grammar can backtrack and request the "next" notehead from the bag. To deal with complexity time, the process uses derivation trees of the assembled musical features during the parse execution. In a more recent work Bainbridge and Bell [6] incorporated a basic graph in CANTOR system according to each musical feature's position (x, y) . The result is a lattice-like structure of musical feature nodes that are linked horizontally and vertically. This final structure is the musical interpretation of the scanned image. Consequently, additional routines can be incorporated in the system to convert this graph into audio application files (such as MIDI and CSound) or music editor application files (such as Tilia or NIFF).

Prerau [96] makes a distinction between notational grammars and higher-level grammars for music. While notation grammars allow the computer to recognize important music relationships between the symbols, the higher-level grammars deal with phrases and larger units of music.

⁷A bag is a one-dimensional data structure which is a cross between a list and a set; it is implemented in Prolog as a predicate that extracts elements from a list, with unrestricted backtracking.

Other techniques to construct the musical notation are based on fusion of musical rules and heuristics (e.g. [36, 88, 39, 114]) and common parts on the row and column histograms for each pair of symbols [126]. Rossant and Bloch [114] proposed an optical music recognition (OMR) system with two stages: detection of the isolated objects and computation of hypotheses, both using low-level pre-processing, and final correct decision based on high-level processing which includes contextual information and music writing rules. In the graphical consistency (low-level processing) stage, the purpose is to compute the compatibility degree between each object and all the surrounding objects, according to their classes. The graphical rules used by the authors were:

- Accidentals and notehead: an accidental is placed before a notehead and at same height.
- Noteheads and dots: the dot is placed after or above a notehead in a variable distance.
- Between any other pair of symbols: they cannot overlapped.

In the syntactic consistency (high-level processing) stage, the aim is to introduce rules related to tonality, accidentals, and meter. Here, the key signature is a relevant parameter. This group of symbols is placed in the score as an ordered sequence of accidentals placed just after the clef. In the end, the score meter (number of beats per bar) is checked. In [85, 87, 88] the process is also based on a low- and high-level approaches to recognize music scores. Once again, the reconstruction of primitives is done using basic musical syntax. Therefore, extensive heuristics and musical rules are applied to re-confirm the recognition. After this operation, the correct detection of key and time signature becomes crucial. They provide a global information about the music score that can be used to detect and correct possible recognition errors. The developed system also incorporates a module to output the result into a *expMIDI* (expressive MIDI) format. This was an attempt to surmount the limitations of MIDI for expressive symbols and other notations details, such as slurs and beaming information.

More research works produced in the past use Abductive Constraint Logic Programming (ACLP) [43] and sorted lists that connect all inter-related symbols [26]. In [43] an ACLP system, which integrates into a single framework Abductive Logic Programming (ALP) and Constraint Logic programming (CLP), is proposed. This system allows feedback between the high-level phase (musical symbols interpretation) and the low-level phase (musical symbols recognition). The recognition module is carried out through object feature analysis and graphical primitive analysis, while the interpretation module is composed of music notation rules in order to reconstruct the music semantics. The system output is a graphical music-publishing file, such as MIDI. No practical results are known for this architecture's implementation.

Other procedures try to automatically synchronize sheet music scanned with a corresponding CD audio recording [75, 35, 50] by using a matching between OMR algorithms and digital signal processing. Based on an automated mapping procedure, the authors identify scanned pages of music score by means of a given audio collection. Both scanned score and audio recording are turned into a common mid-level representation – chroma-based features, where the *chroma* corresponds to the twelve traditional pitch classes of the equal-tempered scale – whose sequences are time-aligned using algorithms based on dynamic time warping (DTW). In the end, a combination of this alignment with OMR results is performed in order to connect spatial positions within audio recording to regions within scanned images.

2.4.1 Summary

Most notation systems make it possible to import and export the final representation of a musical score for MIDI. However, several other music encoding formats for music have been developed over the years – see Table 2.1. The OMR systems used are non-adaptive and consequently they do not improve their performance through usage. Studies have been carried out to overcome this limitation by merging multiple OMR systems [18, 73]. Nonetheless, this remains a challenge. Furthermore, the results of most OMR systems are only for the recognition of printed music scores. This is the major gap in state-of-the-art frameworks. With the exception for PhotoScore, which works with handwritten scores, most OMR systems fail when the input image is highly degraded such as photocopies or documents with low-quality paper. The work developed in [19] is the beginning of a web-based system that will provide broad access to a wide *corpus* of handwritten unpublished music encoded in digital format. The system includes an OMR engine integrated with an archiving system and a user-friendly interface for searching, browsing and editing. The output of digitized scores is stored in MusicXML which is a recent and expanding music interchange format designed for notation, analysis, retrieval, and performance applications.

Software and Program	Output File
SmartScore ^a	Finale, MIDI, NIFF, PDF
SharpEye ^b	MIDI, MusicXML, NIFF
PhotoScore ^c	MIDI, MusicXML, NIFF, PhotoScore, WAVE
Capella-Scan ^d	Capella, MIDI, MusicXML
ScoreMaker ^e	MusicXML
Vivaldi Scan ^f	Vivaldi, XML, MIDI
Audiveris ^g	MusicXML
Gamera ^h	XML files

^a <http://www.musitek.com/>

^b <http://www.music-scanning.com/>

^c <http://www.neuratron.com/photoscore.htm>

^d <http://www.capella-software.com/capella-scan.cfm>

^e <http://www.music-notation.info/en/software/SCOREMAKER.html>

^f <http://www.vivaldistudio.com/Eng/VivaldiScan.asp>

^g <http://audiveris.kenai.com/>

^h <http://gamera.informatik.hsnr.de/>

Table 2.1: The most relevant OMR software and programs.

2.5 Available Datasets and Performance Evaluation

There are some available datasets that can be used by OMR researchers to test the different steps of an OMR processing system. Pinto et al. [95] made available the code and the database⁸ they created to estimate the results of binarization procedures in the preprocessing stage. This database is composed of 65 handwritten scores, from 6 different authors. All the scores in the dataset were reduced to gray-level

⁸<http://www.inescporto.pt/~jsc/ReproducibleResearch.html>.

information. An average value for the best possible global threshold for each image was obtained using five different people. A subset of 10 scores was manually segmented to be used as ground truth for the evaluation procedure⁹. For global thresholding processes, the authors chose three different measures: Difference from Reference Threshold (DRT); Misclassification Error (ME); and comparison between results of staff finder algorithms applied to each binarized image. For the adaptive binarization, two new error rates were included: the Missed Object Pixel rate and the False Object Pixel, dealing with loss in object pixels and excess noise, respectively.

Three datasets are accessible to evaluate the algorithms for staff lines detection and removal: the Synthetic Score Database by Christoph Dalitz¹⁰, the CVC-MUSCIMA Database by Alicia Fornés¹¹ and the Handwritten Score Database by Jaime Cardoso¹²[22]. The first consists of 32 ideal music scores where different deformations were applied covering a wide range of music types and music fonts. The deformations and the ground truth information for these syntactic images are accessible through the MusicStaves toolkit from the Gamera (Generalized Algorithms and Methods for Enhancement and Restoration of Archives) framework¹³. Dalitz et al. [34] put their database available together with their source code. Three error metrics based on individual pixels, staff-segment regions and staff interruption location were created to measure the performance of the algorithms for staff lines removal. The CVC-MUSCIMA Database contains 1000 music sheets of the same 20 music scores which were written by 50 different musicians, using the same pen and the same kind of music paper with printed staff lines. The images of this database were distorted using the algorithms from Dalitz et al. [34]. In total, the dataset has 12000 images with ground truth for the staff removal task and for writer identification. The database created by Cardoso is comprised by 50 real scores with real positions of the staff lines and music symbols obtained manually.

Two datasets are available to train a classifier for the music symbols recognition step. Desaedeleer [38]¹⁴, in his open source project to perform OMR, has created 15 classes of printed music symbols with a total of 725 objects. Rebelo et al. [106] have created a dataset with 14 classes of printed and handwritten music symbols, each of them with 2521 and 3222 symbols, respectively¹⁵.

As already mentioned in this chapter, there are several commercial OMR systems available¹⁶ and their recognition accuracy, as claimed by the distributor, is about 90 percent [9, 66]. However, this is not a reliable value. It is not specified what music score database were used and how this value was estimated. The measurement and comparison in terms of performance of different OMR algorithms is an issue that has already been widely considered and discussed (e.g. [82, 9, 125, 66]). As referred in [9] the meaning of music recognition depends on the goal in mind. For instance, some applications aim to produce an audio record from a music score through document analysis, while others only want to transcode a score into interchange data formats. These different objectives hinder the creation of a common methodology to compare the results of an OMR process. Having a way of quantifying the

⁹The process to create ground-truths is to binarize images by hand, cleaning all the noise and background, making sure nothing more than the objects remains. This process is extremely time-consuming and for this reason only 10 scores were chosen from the entire dataset.

¹⁰<http://music-staves.sourceforge.net>

¹¹<http://www.cvc.uab.es/cvcmusicima/>.

¹²The database is available upon request to the authors.

¹³<http://gamera.sourceforge.net>.

¹⁴<http://sourceforge.net/projects/openomr/>.

¹⁵The database is available upon request to the authors.

¹⁶<http://www.informatics.indiana.edu/donbyrd/OMRSystemsTable.html>.

achievement of OMR systems would be truly significant for the scientific community. On the one hand, by knowing the OMR's accuracy rate, we can predict production costs and make decisions on the whole recognition process. On the other hand, quantitative measures brings progress to the OMR field, thus making it a reference for researchers [9].

Jones et al. [66] address several important shortcomings to take into consideration when comparing different OMR systems: (1) each available commercial OMR systems has its own output interface – for instance, PhotoScore works with Sibelius, a music notation software – becoming very difficult to assess the performance of the OMR system alone, (2) the input images are not exactly the same, having differences in input format, resolution and image-depth, (3) the input and output have different format representations – for instance *.mus* format is used in the Finale software and not in Sibelius software, and (4) the differences in the results can be induced by semantic errors. In order to measure the performance of the OMR systems, Jones et al. [66] also suggest an evaluation approach with the following aspects: (1) a standard dataset for OMR or a set of standard terminology is needed in order to objectively and automatically evaluate the entire music score recognition system, (2) a definition of a set of rules and metrics, encompassing the key points to be considered in the evaluation process, and (3) the definition of different ratios for each kind of error.

Similarly, Bellini et al. [9] proposed two assessment models focused on basic and composite symbols to measure the results of the OMR algorithms. The motivation for these models was the result of opinions from copyists and OMR system builders. The former give much importance to details such as changes in primitive symbols or annotation symbols. The latter, in contrast with the copyists, give more relevance to the capability of the system to recognize the most frequently used objects. The authors defined a set of metrics to count the recognized, missed and confused music symbols in order to reach the recognition rate of basic symbols. Furthermore, since a proper identification of a primitive symbol does not mean that its composition is correct, the characterization of the relationships with other symbols is also analysed. Therefore, a set of metrics has been defined to count categories of recognized, faulty and missed composite symbols.

Szwoch [125] proposed a strategy to evaluate the result of an OMR process based on the comparison of MusicXML files from the music scores. One of the shortcomings of this methodology is in the output format of the application. Even though MusicXML is becoming more and more a standard music interchange format, it is not yet used in all music recognition software. Another concern is related to the comparison between the MusicXML files. The same score can be correctly represented by different MusicXML codes, making the one-to-one comparison difficult. Miyao and Haralick [82] proposed data formats for primitive symbols, music symbols and hierarchical score representations. The authors stress that when using these specific configurations the researchers can objectively and automatically measure the symbol extraction results and the final music output from an OMR application. Hence, the primitive symbols must include the size and position for each element, the music symbols must have the possible combinations of primitive symbols, and the hierarchical score representation must include the music interpretation. Notwithstanding, this is a difficult approach for comparisons between OMR software since most of them will not allow the implementation of these models in their algorithms.

2.6 Open Issues

This section surveyed several techniques currently available in the OMR field. Figure 2.5 summarizes the various approaches used in each stage of an OMR system. The most important open issues are related to

- the lack of robust methodologies to recognize handwritten music scores,
- a web-based system providing broad access to a wide corpus of handwritten unpublished music encoded in digital format,
- a master music dataset with different deformations to test OMR systems,
- a framework with appropriate metrics to measure the accuracy of different OMR systems.

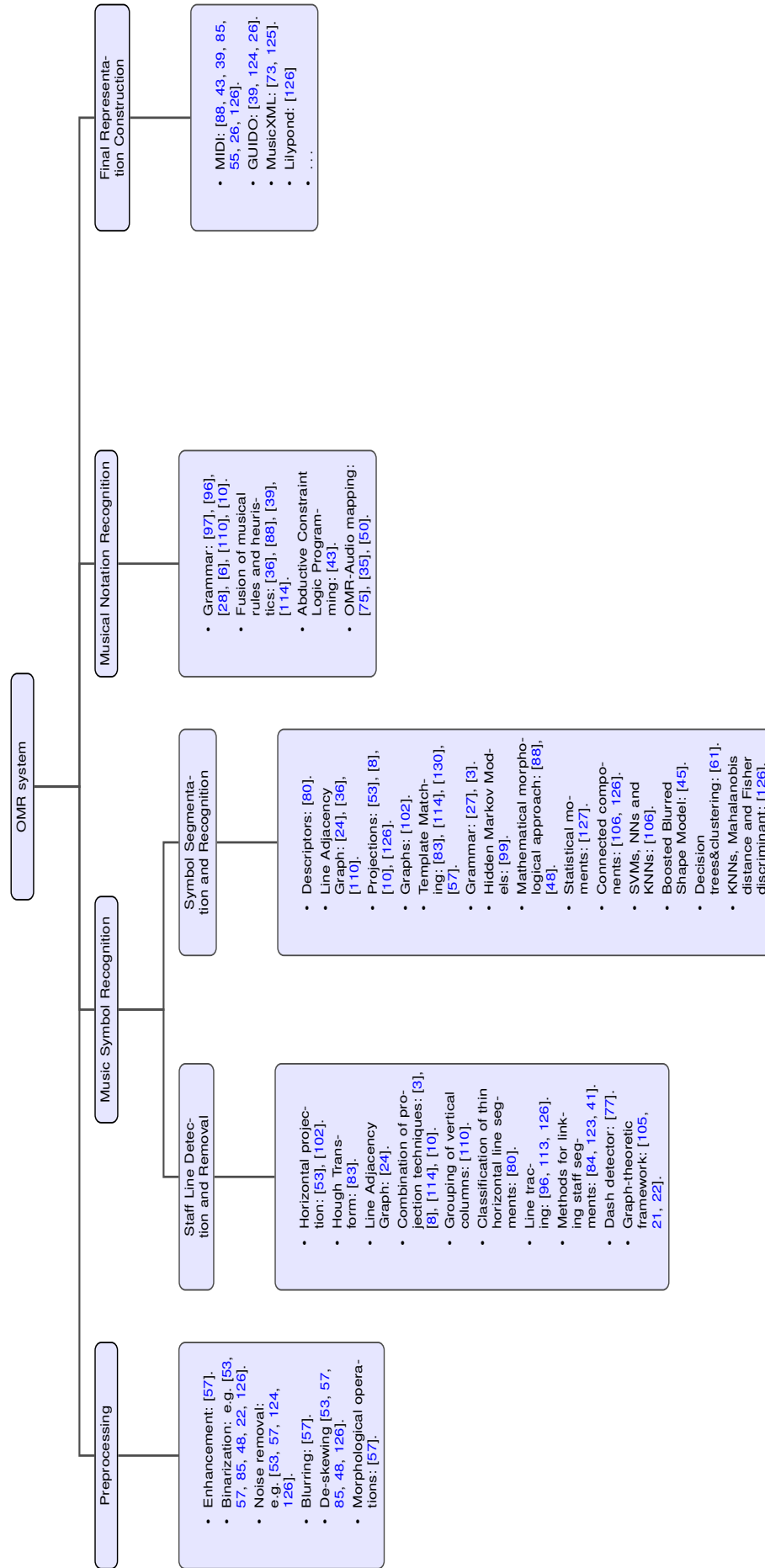


Figure 2.5: Summary of the most used techniques in an OMR system.

Data Sets of Music Scores

Music notation emerged from the combined and prolonged efforts of many musicians. They all hoped to express the essence of their musical ideas by written symbols [103]. Music notation is a kind of alphabet, shaped by a general consensus of opinion, used to express ways of interpreting a musical passage. It is the visual manifestation of interrelated properties of musical sound such as pitch, dynamics, time, and timbre. Symbols indicating the choice of tones, their duration, and the way they are performed, are important because they form this written language that we call music notation [104]. In Table 3.1, we present some common Western music notation symbols that were adopted in this thesis.

Improvements and variations in existing symbols, or the creation on new ones, came about as it was found necessary to introduce a new instrumental technique, expression or articulation. New musical symbols are still being introduced in modern music scores, to specify a certain technique or gesture. Other symbols, especially those that emerged from extended techniques, are already accepted and known by many musicians (e.g. microtonal notation) but are still not available in common music notation software. Musical notation is thus very extensive if we consider all the existing possibilities and their variations.

Moreover, the wider variability of the objects (in size and shape), found on handwritten music scores, makes the operation of music symbols extraction one of the most complex and difficult in an OMR system. Publishing variability in handwritten scores is illustrated in Figure 3.1. In this example, we can see that for the same clef symbol and beam symbol we may have different thicknesses and shapes.

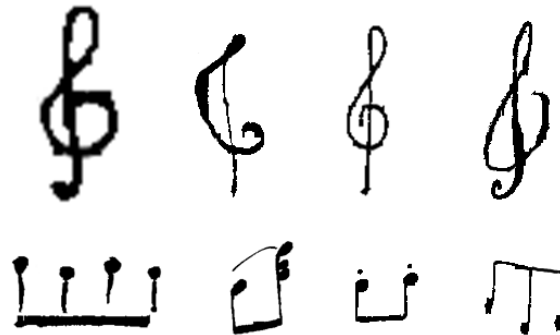


Figure 3.1: Variability in handwritten music scores.

In this chapter the data sets used to test all the proposed procedures are described. Besides, the way we obtained the ground-truth information will also be presented.

3.1 Database of Music Scores

The data set adopted for the various developed methods consists in a total of 65 real handwritten scores of 6 different composers – see Figure 3.5(b) –, 9 printed scores that were scanned (hereafter termed

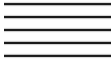



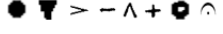

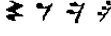


Symbols	Description
	Staff: An arrangement of parallel lines, together with the spaces between them.
	Treble, Alto and Bass clef: The first symbols that appear at the beginning of every music staff and tell us which note is found on each line or space.
	Sharp, Flat and Natural: The signs that are placed before the note to designate changes in sounding pitch.
	Beams: Used to connect notes in note-groups; they demonstrate the metrical and the rhythmic divisions.
	Staccato, Staccatissimo, Dynamic, Tenuto, Marcato, Stopped note, Harmonic and Fermata: Symbols for special or exaggerated stress upon any beat, or portion of a beat.
	Quarter, Half, Eighth, Sixteenth, Thirty-second and Sixty-fourth notes: The Quarter note (closed notehead) and Half note (open notehead) symbols indicate a pitch and the relative time duration of the musical sound. Flags (e.g. Eighth note) are employed to indicate the relative time values of the notes with closed noteheads.
	Quarter, Eighth, Sixteenth, Thirty-second and Sixty-fourth rests: These indicate the exact duration of silence in the music; each note value has its corresponding rest sign; the written position of a rest between two barlines is determined by its location in the meter.
	Ties and Slurs: Ties are a notational device used to prolong the time value of a written note into the following beat. The tie appears to be identical to slur, however, while tie almost touches the notehead centre, the slur is set somewhat above or below the notehead. Ties are normally employed to join the time value of two notes of identical pitch; Slurs affect note-groups as entities indicating that the two notes are to be played in one physical stroke, without a break between them.
	Mordent and Turn: Ornaments symbols that modify the pitch pattern of individual notes.

Table 3.1: Music notation.

digitized scores) – see Figure 3.4(c) – and 19 synthetic scores, to which distortions were applied – see Figure 3.5(a). This set consists on the fraction of the dataset available from [34] written on the standard notation. In total, 2688 images were generated from these 19 perfect scores.

According with the experiences planned the portion of the data set changed as following:

Stage of OMR system	Database	Description
Staff line thickness and distance estimation	50 handwritten scores.	The objective was to test the precision and the robustness of the proposed method in the presence of irregularities in the illumination and musical works degraded, thus the synthetic and printed scores were not considered in this stage.
Binarization algorithm	65 handwritten scores.	The data set of handwritten scores was increased and the synthetic scores were not considered. This type of music sheets are already binary images. Besides, they do not have some important characteristics, such as heterogeneous light distribution, noise or back to front interference. The digitized scores were also not used because of the lack of manual references of the global threshold.
Staff line detection and removal	40 handwritten scores and 2688 images generated from 32 perfect scores.	The objective was to test the performance of the stable path based approach in detecting staff lines with several distortions. A total of 40 handwritten scores with reference of the staff lines outlined are available.
Music symbols segmentation	9 digitized scores, 26 handwritten scores and 882 images generated from 18 perfect scores.	Only the deformations that cause distortions in the music symbols were considered: rotation, curvature and typeset emulation. A total of 26 handwritten scores and 9 digitized scores with reference of the positions of the music symbols are available.
Music symbols segmentation and classification	9 digitized scores, 6 handwritten scores and 132 images generated from 12 perfect scores.	The number of music scores and deformations were restricted to the number of manual references of the positions of the musical symbols with their classes.

Table 3.2: Data set adopted in the different stages of the OMR procedure.

3.2 Deformations in the Synthetic Scores¹

As music scores may suffer from deformations, the staff lines may have discontinuities, be curved or inclined. These problems will influence the success to achieve a correct detection of lines contained on the score to recognize. Moreover, the detection of the music symbols will also be compromised. In order to simulate feasible problems of handwritten musical scores, not only to test and evaluate the algorithms, but also to increase the data set, several distortions were applied to the original set of synthetic scores. These distortions adopted from [34] can be classified in two categories:

1. Deterministic deformations, which depend of certain parameters, as for instance the *rotation*.
2. Random defects, which use various parameters about the deformation and a pseudo-random number generator.

In both cases it is necessary to apply the deformation in parallel with the original score and the ground-truth staff image.

Despite some deformations that are easy to understand by their name, there are others where this is not true (the defects *resolution*, *rotation* and *line interruption* are self explanatory). A brief explanation of these distortions will be done. In Table 3.3 the deformations considered, which are available in the MusicStaves toolkit², are listed. In Figure 3.2 the effects caused in the images is possible to see.

Deformation	Type	Parameter description
Rotation	Deterministic	Rotation angle
Curvature	Deterministic	Height ratio: width of sine curve
Typeset Emulation	Both	Range width, maximal height and variance of vertical shift
Staff Line Interruptions	Random	Interruption frequency, maximal width and variance of range width
Staff Line Thickness Variations	Random	Markov chain stationary distribution and inertia factor
Staff Line y-variation	Random	Markov chain stationary distribution and inertia factor
Degradation After Kanungo	Random	$(\eta, \alpha_0, \alpha, \beta_0, \beta, \kappa)$, see [68]
White speckles	Random	Speckles frequency, random walk length and smoothing factor

Table 3.3: Deformations in the images.

Curvature The curvature is obtained through a half sine wave over the entire score area. The intensity of the resulting curvature can be measure as the ratio of amplitude (height) and the width of the wave.

¹Some portions of this section appears in [104].

²<http://gamera.informatik.hsnr.de/>

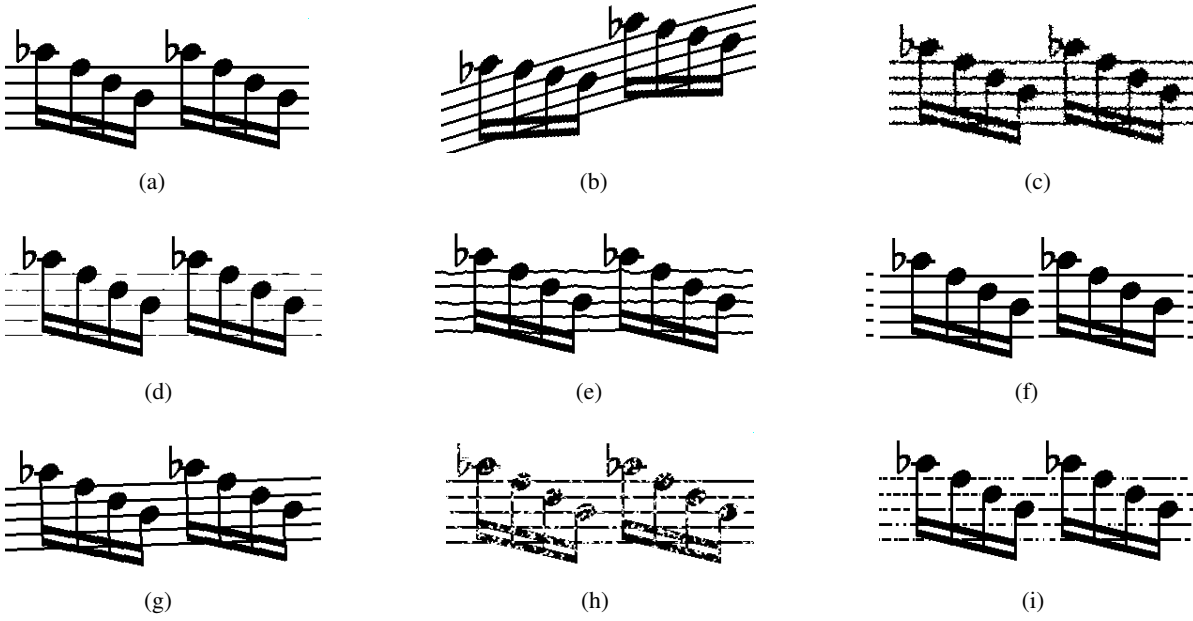


Figure 3.2: Some examples of applied deformations from the original image: a) Original; b) Curvature; c) Degradation after Kanungo; d) Staff line thickness variation; e) Staff line y -variation; f) Typeset emulation; g) Rotation; h) White Speckles; i) Staff line Interruptions.

Typeset emulation This deformation tries to reproduce the 16th-century prints, which are similar, in some ways, to typewriters, causing, therefore, interruptions in the lines between the symbols and a random vertical shift in each portion containing a symbol.

Staff line thickness variation and staff line y -variation Staff line thickness variation and staff line y -variation are obtained by a Markov chain describing the evolution of the staff line thickness from left to right. These deformations are achieved by that process because, usually, the thickness at a particular x -position depends on the thickness at the previous x -position. The parameter is the transition probability matrix \mathcal{P} with p_{ij} = probability of transition from thickness or y -deviation i to thickness or y -deviation j . The thickness or y -deviation can be of n different values (states). To the stationary distribution of the individual states a symmetric binomial distribution is assumed, that is:

$$\pi_i = \binom{n-1}{i-1} \frac{1}{2^{n-1}}$$

The mean value $(n-1)/2$ of this distribution is associated with the original value in the image without the deformation (*staffline_height* for the thickness or zero for the deviation from the original y -position). The Markov chain is generated with the Metropolis-Hastings algorithm [59]. The transition probability matrix Q , to obtain candidate transition points, is chosen to be:

$$q_{ij} = \begin{cases} c & \text{for } j = i \\ 1 - c/2 & \text{for } j = i \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

where the probability c can be considered as an inertia factor that allows smooth transitions: the closer c is to one, the slower is the state variation.

Deformation	Parameter range
Rotation	$angle = -5 : 2.5 : 5$
Curvature	$amplitude/staffwidth = 0.02 : 0.02 : 0.10$
Typeset Emulation	$n_gap = 1 : 3 : 13, n_shift = 1 : 3 : 13, p_gap = 0.5$
Staff Line Interruptions	frequency $\alpha = 0.01 : 0.02 : 0.10$, binomial parameter for width: $n = 6, p = 0.5$
Staff Line Thickness Variation	inertia $c = 0.8$, maximum deviation = $2 : 1 : 6$
Staff Line y -variation	inertia $c = 0.8$, maximum deviation = $2 : 1 : 6$
Degradation After Kanungo	$\eta = 0, \alpha_0 = 0.5, \alpha = 0.25 : 0.3 : 1.5$, $\beta_0 = 0.5, \beta = 0.25 : 0.3 : 1.5, \kappa = 2$
White Speckles	smoothing factor $k = 2$, random walk length $n = 10$, speckle frequency $p = 0.03 : 0.02 : 0.11$

Table 3.4: Ranges of deformation parameters used in the tests: **min:step:max**.

Degradation after Kanungo This deformation tries to imitate the local distortions caused during printing and scanning. The model has six parameters $(\eta, \alpha_0, \alpha, \beta_0, \beta, \kappa)$ with different meanings:

- Each black pixel in the original image is flipped with probability $\alpha_0 \exp^{-\alpha d^2} + \eta$, where d is the distance to the closest background pixel.
- Each background pixel is flipped with probability $\beta_0 \exp^{-\beta d^2} + \eta$, where d is the distance to the closest foreground pixel.
- κ is the diameter of a disk of a morphological closing operation.

White speckles This degradation model has three parameters (p, n, k) with the following meaning:

- Each black pixel in the original image is taken with probability p as a starting point for a random walk of length n .
- k is a rectangle of a morphological closing operation that will smooth an image containing the random walk.
- The image with the random walks is subtracted from the original: an image with white speckles at the random walk positions is obtained.

Therefore, p can be interpreted as the speckle frequency, n as a measure for the speckle and k as a smoothing factor.

The range of deformations parameters in our test set is given in Table 3.4. The values were restricted with the aim of obtaining realistic deformations. Even so, when real scores are treated, generally, the deformations do not occur in its pure form; a combination of several deformations is more prone to happen. However, because the evaluation of the detection algorithms is our objective it is more adequate to research the performance in each isolated deformation.

3.3 Deformations applied to each Music Symbol

The full set of training patterns extracted from the database of scores was augmented with replicas of the existing patterns, transformed according to the elastic deformation technique detailed next. Such transformations try to introduce robustness in the prediction with respect to the known variability in the symbols.

The research in deformable template fields applied on handwritten digits and printed characters recognition is well established (e.g. [76, 136, 91, 65]). Lam [76], one of the first works in this area, proposed a method of recognition in two-stages. The images are first recognized by a tree classifier; those that cannot be satisfactory assigned to a class are passed to a matching algorithm, which deforms the image to match with a template. In [91] a grammar-like model for applying deformations in primitive strokes was developed, while Wakahara [136] proposed a shape-matching approach to recognize number manuscripts. The method uses successive local affine transformation (LAT) operations to gradually deform the image. The aim is to yield the best match to an input binary image. LAT on each point at one location is optimized using locations of other points by means of least-squares data fitting using Gaussian window functions. In document degradation models, Baird [7] presented an overview in techniques that parameterized models of image defects that occur during printing, photocopying, and scanning processes. In this same line, Kanungo [67, 68] also proposed a statistical methodology of these deterioration processes in order to validate local degradation models.

The deformation technique used to deform the musical symbols is based in Jain [64, 65]. In this approach, the image is mapped on a unit square $S = [0, 1] \times [0, 1]$. The points in this square are mapped by the function $(x, y) \rightarrow (x, y) + D(x, y)$. The space of displacement functions are given by

$$\mathbf{e}_{mn}^x(x, y) = (2 \sin(\pi n x) \cos(\pi m y), 0) \quad (3.1)$$

$$\mathbf{e}_{mn}^y(x, y) = (0, 2 \sin(\pi n y) \cos(\pi m x)) \quad (3.2)$$

Specifically, the deformation function is chosen as follows:

$$D(x, y) = \sum_{m=1}^M \sum_{n=1}^N \frac{\xi_{mn}^x \mathbf{e}_{mn}^x + \xi_{mn}^y \mathbf{e}_{mn}^y}{\lambda_{mn}} \quad (3.3)$$

where $\underline{\xi} = \{(\xi_{mn}^x, \xi_{mn}^y), m, n = 1, 2, \dots\}$ are the projections of the deformation function on the orthogonal basis. Because $D(x, y)$ can represent complex deformations by choosing different coefficients of ξ_{mn} and different values of M and N, it is important to impose a probability density on $D(x, y)$. We assume that the ξ_{mn} 's are independent of each other and the x and y directions are independent, identically-distributed Gaussian distributions with mean zero and variance σ^2 . Figure 3.3 shows examples for several deformations using different higher-order terms. Note that the deformation is stronger when M, N and σ are increased.

3.4 Ground-truth Information

Lengths of staff line height and staff line space The reference for the `staffline_height` value and `staffspace_height` value were manually measured by three independent individuals.

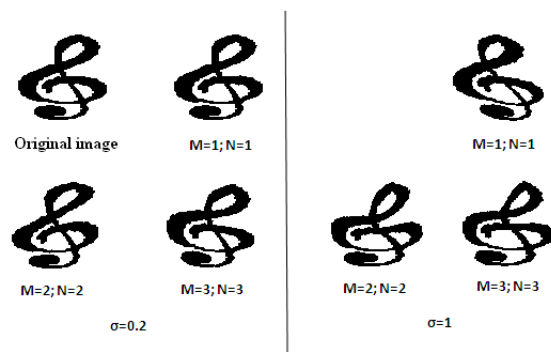


Figure 3.3: Example of deformations on a musical symbol. From Rebelo [106, Fig.6)].

Binarization A binarization ground truth of the images was also obtained manually. Since this is very time consuming, only 10 scores from different authors, chosen randomly from the complete dataset, were selected. The process consist in cleaning all the noise and background for each image, making sure nothing more than the objects remains – see figure 3.4. A supervised global



Figure 3.4: Some examples of binarization music scores.

threshold value was also selected for each of the 65 handwritten music scores. The reference thresholds were supervised independently by five different people. The threshold was chosen according with staff lines structure, the noise presence and distortion and occlusion of objects.

Staff lines positions To establish a qualitative comparison between the various staff lines detection and removal algorithms a ground-truth information is necessary. Consequently, the black pixels of the images need to be labelled as being staff lines or otherwise. The manual process has two

disadvantages: it is very time consuming and it is prone to the occurrence of errors (e.g. the existence of dubious pixels that may belong to the line or to the symbol that crosses the staff line). Therefore, ground-truth information of the syntactic images released by the authors of the MusicStaves toolkit was used. For handwritten music sheets we can only achieve the ground-truth information manually. Hence, the symbols present in the scores were deleted in order to retain just the staff lines segments.

Positions of the musical symbols For each music symbol present in the score its bounding box containing its position on the score and its class was extracted and saved.

(a) Synthetic scores.

(b) Handwritten scores.

The image displays a collage of various musical scores. At the top right, a score for 'Concerto for Bassoon and Orchestra in F-major' by Karl Stamitz (1745-1801) is shown, with the Violin II part. Below this, a score for 'Chromatics' is visible, featuring a 'Solo' section. To the left, a score for 'Allegro moderato' is shown, with a 'Solo' section. At the bottom right, a score for 'PASSEPIED' by Leopold Mozart (1719-1789) is displayed, featuring a 'Solo' section. The scores are arranged in a grid-like fashion, with some overlapping. The notation includes various musical symbols such as notes, rests, and dynamic markings.

(c) Printed Scores.

Figure 3.4: Some examples of music scores used in this thesis.

Optical Music Recognition involves many research areas. The processing of handwritten musical scores is not yet fully explored largely due to the poor results of the existing methods. One of the stages in the OMR process is the recognition of musical symbols. The efforts made to find robust symbol representations and learning methodologies have not found a similar quality in the learning of the dissimilarity concept. Simple Euclidean distances are often used to measure dissimilarity between different examples. However, such distances do not necessarily yield the best performance. In this thesis a comparative study of recognition algorithms applied to music symbols was carried out. In this chapter the various techniques used are presented, as well as the experimental testing.

4.1 Recognition Process

Towards a comparative study between classification procedures, five different approaches were evaluated: Hidden Markov Models (HMMs), Support Vector Machines (SVMs), Relevance Vector Machines (RVMs), Neural Networks (NNs) and k-Nearest Neighbor (kNN).

Classifiers are built by taking a set of labeled examples that are used to construct a rule that will assign a label to any new example. In other words, if we consider a general situation, we will have a training data set (x_i, y_i) ; where x_i is a feature vector for an object i , and y_i is the label associated with the object class. The relative costs of mislabeling each x_i are known and must be used to generate a decision criterion that can take any new observation vector (x_j) and assign it the correct class label (y_j) .

Hidden Markov Model

Hidden Markov Models (HMMs) have been used in OMR by [74, 81, 99]. The application of this technique to musical symbol classification had its origins on optical character recognition. One of the reasons for the use of HMMs lies in its capability to perform segmentation and recognition at the same time.

A HMM is a *doubly stochastic process* that generates sequence symbols, with an underlying stochastic process that is hidden and can only be detected through another process whose realizations are observable [14]. The hidden process consists of a set of states connected to each other by a transition probability. Transitions probabilities from a state i to another state j are given by $A = \{a_{ij}\}$, where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$, $1 \leq i, j \leq N$. The observed process consists of a set of outputs or observations. Each observation is contained in a state with some probability density function. The set of observations probabilities is given by $B = b_j(k)$, where $b_j(k) = P[o_t = x_k | q_t = S_j]$, $1 \leq k \leq M, j = 1, 2, \dots, N$. $b_j(k)$ represents the probability of the observation x_k in state S_j , o_j denotes the observation in time t and q_t represents the state in time t . HMM can now be concisely formulated as $\lambda = (A; B; \pi)$, where π is a set of initial probabilities of states [137].

*Some portions of this chapter appears in [106, 109].

A left-right, model discriminant HMM was adopted to construct a model for each class [2]. The learning of the models parameters ($\lambda = (A; B; \pi)$) was accomplished with the Baum-Welch algorithm. The goal of classification is to decide which class the unknown sequence belongs to, based on the model obtained in the training phase. These symbols were classified on the basis of the maximum likelihood ratio obtained by the Viterbi algorithm.

Support Vector Machine

One of the most widely adopted techniques by the pattern recognition community is the Support Vector Machines (SVM). This procedure has as its main idea the maximization of the separation margin between positive and negative examples having an hyperplane as the decision surface [135]. In a formal manner, given the training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$ with input data $\mathbf{x}_i \in \mathbf{R}^p$ and corresponding binary class labels $d_i \in \{-1, 1\}$, the maximum-margin hyperplane is defined by $g(\mathbf{x}) = \mathbf{w}^t \varphi(\mathbf{x}) + b$ where $\varphi(\mathbf{x})$ denotes a fixed-feature space transformation and b a bias parameter; \mathbf{x} is assigned to class 1 if $g(\mathbf{x}) > 0$ or to -1 if $g(\mathbf{x}) < 0$. The maximization of the margin is equivalent to solving

$$\begin{aligned} \min_{w, b, C, \xi_i} \quad & \frac{1}{2} \mathbf{w}^t \mathbf{w} + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & y_i [\mathbf{w}^t \varphi(\mathbf{x}_i) + b] \geq 1 - \xi_i, i = 1, \dots, N \\ & \xi_i \geq 0 \end{aligned} \tag{4.1}$$

where parameter $C > 0$ controls the trade-off between the classification errors and the margin. The slack variables ξ_i , $i = 1, \dots, N$ are introduced to penalize incorrectly classified data points. The dual formulation in (8.1) leads to a dependence on the data only through inner-products $\phi(\mathbf{x}_i)^t \phi(\mathbf{x}_j)$. Mercer's theorem allows us to express those inner products as a continuous, symmetric, positive semi-definite kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ defined in the input space. In this work, a radial-basis function kernel was used, given by $k(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2)$, $\gamma \geq 0$

The binary SVM classifier can be extended to multiclass scenarios. Of the multiple extensions available in the literature [62], we used the one against one methodology.

Relevance Vector Machine

A sparse kernel technique, similar to SVM but avoiding its principal limitations², is the Relevance Vector Machine (RVM). This is based on a Bayesian approach and provides posterior probabilistic outputs [129]. For a two-class classification, the method predicts the posterior probability for being one of the classes, given the input x . Formally, the logistic sigmoid function $\sigma(y) = 1/(1 + e^{-y})$ is applied to the linear model $y(x) = \sum_{n=1}^N w_n K(x, x_n) + w_0$, where $\{w_n\}$ are the model weights and $k(\cdot, \cdot)$ is a kernel function. The procedure follows a Bernoulli distribution for $P(t|x)$, so the likelihood is given by

$$P(t|w) = \prod_{n=1}^N \sigma\{y(x_n)\}^{t_n} [1 - \sigma\{y(x_n)\}]^{1-t_n} \tag{4.2}$$

where the targets $t_n \in \{0, 1\}$. The marginal likelihood, $P(t|\alpha)$, for the hyperparameters is analytically obtained using the Laplace approximation procedure:

²The predictions of an SVM are not probabilistic, the kernel function must satisfy Mercer's condition and the error/margin trade-off C parameter needs to be estimated in the training stage.

1. For the current values of α , the most probable weights w_{MP} are calculated using the following equation

$$\log\{P(t|w)p(w|\alpha)\} = \sum_{n=1}^N [t_n \log y_n + (1 - t_n) \log(1 - y_n)] - \frac{1}{2} w^T A w \quad (4.3)$$

with $y_n = \sigma\{y(x_n; w)\}$.

2. The Hessian matrix at w_{MP} is computed from

$$\nabla \nabla \log p(t, w|\alpha)|_{w_{MP}} = -(\Phi^T B \Phi + A) \quad (4.4)$$

where $B = \text{diag}(\beta_1, \beta_2, \dots, \beta_N)$ is a diagonal matrix with

$$\beta_N = \sigma\{y(x_n)\} [1 - \sigma\{y(x_n)\}].$$

Neural Networks

A neural network is a parametric approximation technique that has found several applications in diverse fields, including pattern recognition and signal processing. The process typically approximates a vector function f of some input x with a series of layers. Each layer forms a vector of outputs, each of which is obtained by applying the same non-linear function to different affine functions of the inputs [49].

A multi-layer perceptron (MLP), one type of a feed-forward network, was used in our evaluation. A MLP is a layered structure consisting of nodes or units (called neurons) and one-way connections or links between the nodes of successive layers. The training of the networks was carried out under Matlab[®] and was done using back-propagation together with the Levenberg-Marquardt algorithm. We used a network with K outputs, one corresponding to each class, and target values of 1 for the correct class and 0 otherwise. The network had one hidden layer and sigmoid functions as activation functions.

K-Nearest Neighbor

The k -nearest neighbor algorithm is amongst the simplest of all machine learning algorithms [31, 40]. This method belongs to a set of techniques called Instance-based Learning. It requires no training effort and critically depends on the quality of the distances measures among each instances. It uses the heuristic that sample points near an unclassified point should indicate the class of that point. In this manner, a k -nearest neighbor classifier finds the k data points from the training set closest to the point being considered, and classifies the object with the most frequent class amongst its k -nearest neighbors.

4.2 Metric Learning

Recently, the distance metric problem has received much attention in the machine learning community [142, 140, 138]. The performance of all machine learning algorithms depend critically on the metric that is used over input space. Some learning algorithms, such as K-means and k -nearest neighbors, require a metric that will reflect important relationships between each classes in data and will allow to discriminate instances belonging to one class from others. Depending on the availability of training examples, distance metric learning algorithms can be divided into two main categories: supervised distance metric learning and unsupervised distance metric learning. Supervised distance metric learning can be further divided into global distance metric learning and local distance metric learning.

In global case constraints are applied to all pairwise of examples, while in local approach only local pairwise are taken into consideration.

4.2.1 Metric Learning for k -NN

As already mentioned previously, in the k -nearest neighbor algorithm the decision about classifying a new query is determined by the labels of the k training examples with shortest distance. Conventionally those distances are defined by the Euclidean distance between examples. Decision rule classifies unlabeled inputs by the majority label of their k -nearest neighbor in the training set.

Our approach is based on work conducted by [138]. The authors proposed a distance metric learning algorithm for Large Margin Nearest Neighbor classification (LMNN). The main idea behind is to learn a Mahalanobis distance function by minimizing an objective function that is set up with local and global constraints. This optimization results in bringing k -nearest neighbors from the same class closer (i.e. shrinks the distances between nearby examples from the same class) and to separate examples from other classes by a large margin (expands the distances between examples from different classes). The idea of LMNN can be introduced as follows:

Let the $\{\mathbf{x}_i, y_i\}_{i=1}^N$ denotes a training set of N labeled examples with inputs $\mathbf{x}_i \in \mathbf{R}^p$ and discrete class labels y_i . The Authors also introduced a binary matrix $\tau_{ij} \in \{0, 1\}$, which indicates wherever or not the labels y_i and y_j match. The main goal is to learn a linear transformation L , which will optimize k -NN classification: $L : \mathbf{R}^d \rightarrow \mathbf{R}^d$. Transformation L is used to calculate squared distances as:

$$D(\mathbf{x}_i, \mathbf{x}_j) = \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 \quad (4.5)$$

For each input \mathbf{x}_i authors introduced k other inputs, called *target neighbors*, that share the same label y_i . These target neighbors after transformation will have minimal distance to \mathbf{x}_i . In the situation where any prior knowledge is not available, target inputs can be identified as k -nearest neighbor, determined by the well known Euclidean distance. Authors also introduced $\eta_{ij} \in \{0, 1\}$ in order to indicate whether \mathbf{x}_j is a target neighbor of \mathbf{x}_i . Both matrices τ_{ij} and η_{ij} are fixed during the learning stage. Formally, the cost function is defined as:

$$\epsilon(L) = \sum_{ij} \eta_{ij} \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 + c \sum_{ijl} \eta_{ij} (1 - \tau_{il}) [1 + \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 - \|L(\mathbf{x}_i - \mathbf{x}_l)\|^2]_+ \quad (4.6)$$

where $[z]_+ = \max(z, 0)$ denotes the standard hinge loss and $c > 0$ is a positive constant. There are two competing terms in this equation. The first one penalizes large distances between each input and its target neighbors. The second term penalizes small distances between each input and all other inputs that do not share the same label. For each input \mathbf{x}_i , the hinge loss is incurred by differently labeled inputs by one absolute unit of distance. The learning idea behind this approach is presented on Fig. 4.1.

The Equation (4.6) can be reformulated as an instance of semidefinite programming (SDP). As SDP are convex (linear costs and constraints are replaced by convex costs and constraints) the global minimum can be efficiently computed [134]. In order to obtain SDP Equation (4.5) needs to be rewritten as:

$$D(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T M (\mathbf{x}_i - \mathbf{x}_j) \quad (4.7)$$

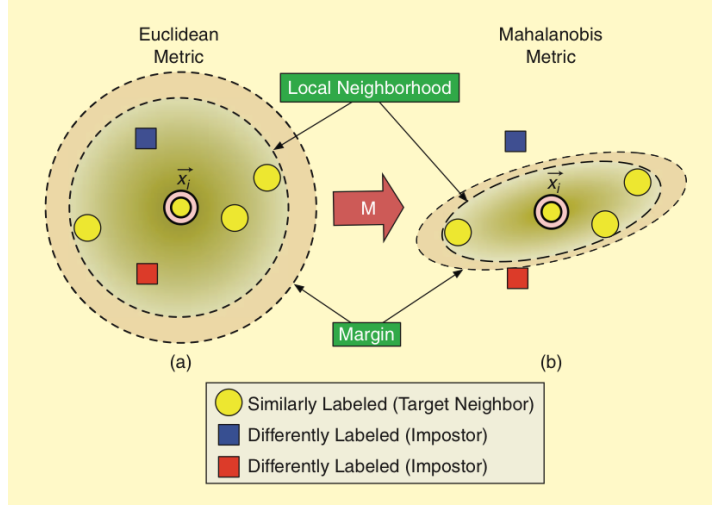


Figure 4.1: Schematic illustration idea behind LMNN. Left side shows traditional approach, under Euclidean distance, where \mathbf{x}_i has three target neighbors. Right image shows the new discriminator after the Mahalanobis distance being learnt. From [139, Fig.1].

where the matrix $M = L^T L$ parametrizes the Mahalanobis distance induced by the linear transform L . In order to *mimick* the hinge loss, slack variables ξ_{ij} were introduced for all pairs of differently labeled inputs (i.e. $\forall(i, j)$ s.t. $y_{ij} = 0$). Finally, the resulting SDP is given by:

$$\begin{aligned} \min \quad & \sum_{ij} \eta_{ij} (\mathbf{x}_i - \mathbf{x}_j)^T M (\mathbf{x}_i - \mathbf{x}_j) + c \sum_{ijl} \eta_{ij} (1 - \tau_{il}) \xi_{ijl} \\ \text{s.t.} \quad & \begin{cases} z \geq 1 - \xi_{ijl} & \forall(i, j, l) \in T \\ \xi_{ijl} \geq 0 & \forall(i, j, l) \in T \\ M \succeq 0 \end{cases} \end{aligned} \quad (4.8)$$

where $z = (\mathbf{x}_i - \mathbf{x}_l)^T M (\mathbf{x}_i - \mathbf{x}_l) - (\mathbf{x}_i - \mathbf{x}_j)^T M (\mathbf{x}_i - \mathbf{x}_j)$ and $T = \{(i, j, l) | i, j, l \in \{1, \dots, n\}, j \rightsquigarrow i, y_i \neq y_l\}$.

4.2.2 Metric Learning for SVM

An intuitive extension of the metric learning on k -NN previous described is its application to SVMs. The kernel trick, i.e., the mapping of patterns of the input feature space to a higher dimensional space, can be considered as a (dis)similarity measure. Thus, kernels can be seen as a way of a general metric learning approach [117, 89].

In this thesis we apply the concept introduced in [139] in order to assess the benefit of LMNN to SVMs on OMR. Our approach consisted on the one-against-one strategy where a matrix L is learnt for each discriminant. We used a kernel derived from the RBF presented in Equation (4.5) which resulted in $D(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma \|L(\mathbf{x} - \mathbf{x}_i)\|^2)$, $\gamma \geq 0$, dRBF.

Part II

Preprocessing

Staffline Thickness and Distance Estimation*

In this Chapter two main contributions are introduced: a robust method to estimate the staffline thickness (`staffline_height`) and the distance between stafflines (`staffspace_height`) on binarized music scores and the generalization to gray-level music scores.

5.1 Conventional estimation of staffline thickness and distance

Run-length encoding (RLE) is a very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in consecutive data elements) are represented as a single data value and count. In a binary image, used as input for the recognition process here, there are only two values: one and zero. In such a case, the run-length coding is even more compact, because only the lengths of the runs are needed. For example, the sequence {1 1 0 1 1 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1} can be coded as 2, 1, 3, 2, 4, 2, 4, 1, 5, 2, 6, assuming 1 starts a sequence (if a sequence starts with a 0, the length of zero is used at start). By encoding each column of a digitized score using RLE, the most common black-runs represents the `staffline_height` and the most common white-runs represents the `staffspace_height` – see Figure 5.1. Even in music scores with different staff sizes, there will be prominent peaks at the most frequent staffspaces. These estimates are also immune to severe rotation of the image [71, 53].

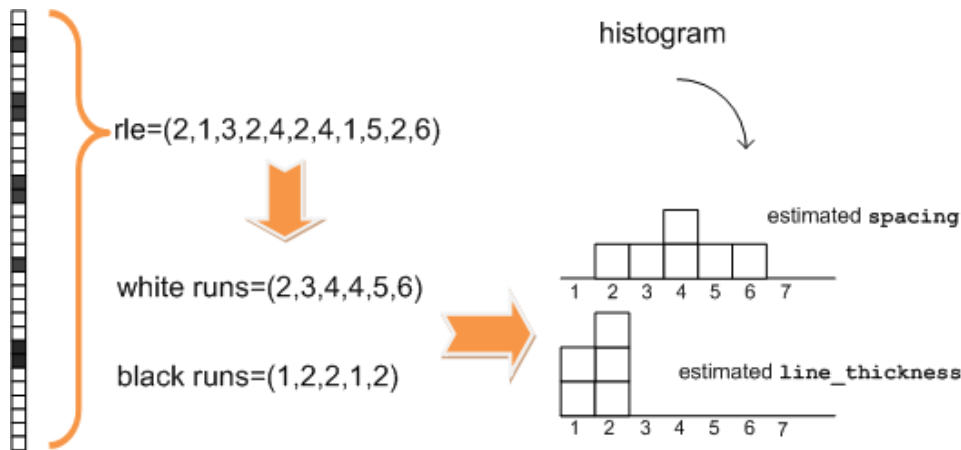


Figure 5.1: Illustration of the estimation of the reference values `staffline_height` and `staffspace_height` using a single column.

5.2 Robust estimation of staffline thickness and spacing

Although the performance of this conventional method is excellent in printed music scores and very good in handwritten scores, the estimation fails under severe degradation of the scores, as illustrated

*Some portions of this chapter appears in [23].

in Figure 5.2. For this score the conventional estimation provides `staffline_height` = 1 and `staffspace_height` = 1 (the true values are `staffline_height` = 5 and `staffspace_height` = 19).



(a) Original music score #17.



(b) Score binarized with Otsu's method.

Figure 5.2: Unsuccessful estimation of `staffline_height` and `staffspace_height` by vertical runs.

Isolated black pixels and fluctuations in the thickness and distance between lines pose challenging difficulties. Two observations are decisive for an improved estimation of the reference lengths: firstly, the length of the run of white pixels before and after the isolated black pixels will vary a lot, ‘randomly’; secondly, a local fluctuation in the thickness of the staffline (due to noise) is often compensated by a variation with an opposite sign of the local distance between lines—this effect is visible in Figure 2.3.

Therefore, it seems that the estimation of the sum of `staffline_height` and `staffspace_height` can be done much more robustly by finding the most common sum of two consecutive vertical runs (either black run followed by white run or the reverse) – see Figure 5.3. This is illustrated in Figure 5.4, with the histograms of the black runs, white runs and the sum of two consecutive runs. The prominent peak at the black and white runs histograms is at `run` = 1, due to the noise on the binarization process. Nevertheless, the prominent peak at the sum two consecutive runs is at 24, consistent with the true values of `staffline_height` and `staffspace_height`.

We propose that, when possible, any subsequent operation should be based on this new reference length, `staffline_space_height`. Nevertheless, robust estimates of `staffline_height` and `staffspace_height` can now be obtained knowing that their sum should equal `staffline_space_height`. To reliably estimate these values the process starts by computing the 2D histogram of the pairs of consecutive vertical runs. Next, it selects the most common pair for which the sum of the runs equals `staffline_space_height`.

Figure 5.5 shows the 2D histogram for the music score in Figure 5.2. As visible, restricting ourselves to the pairs summing 24 (the estimated value for the sum), the procedure is able to recover the

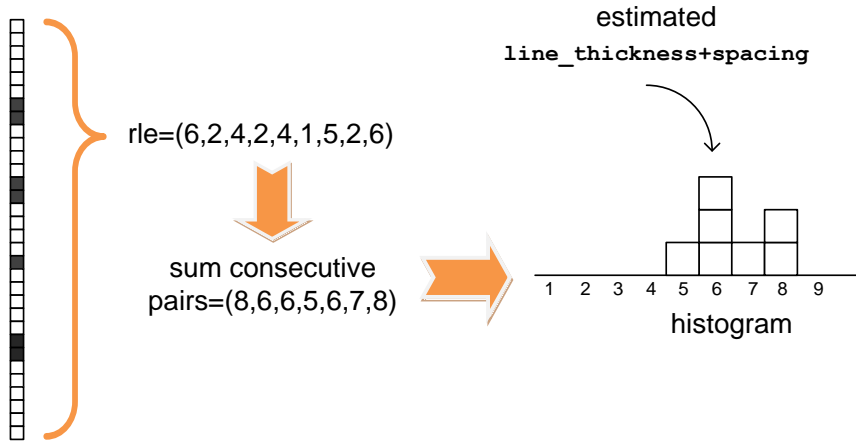


Figure 5.3: Illustration of the estimation of the reference value `line_thickness+spacing` using a single column. In practice, sums of consecutive runs are accumulated over the whole image.

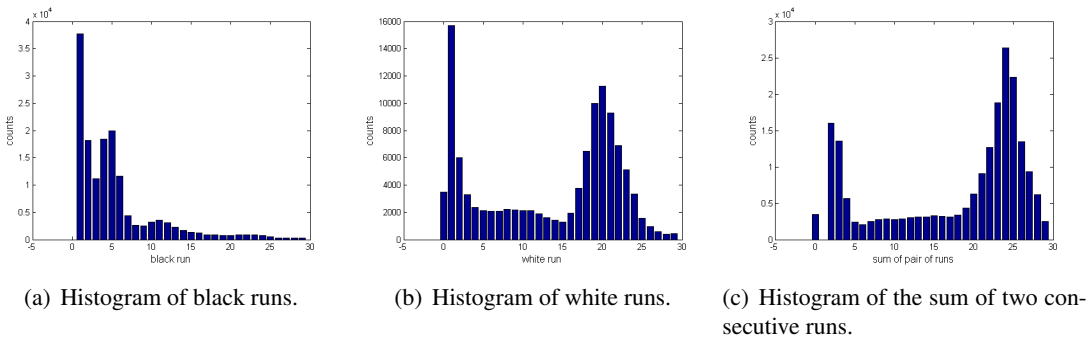


Figure 5.4: Histogram for the music score of Figure 5.2 (score #17).

correct value for the line thickness and space.

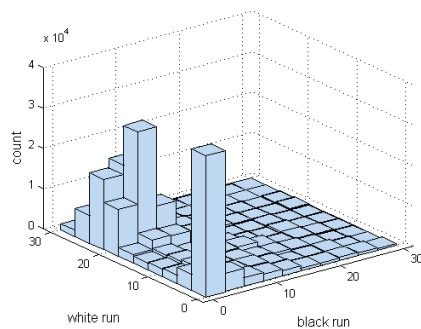
5.3 Staffline thickness and spacing estimation in gray-level images

In some binarized images, the noise level is such that even the new method is unable to correctly estimate `staffline_height` and `staffspace_height`. An example is presented in Figure 5.6 (original image). Figure 5.7 shows the result for this score. Even though the sum of `staffline_height` and `staffspace_height` is correctly estimated, the individual values for `staffline_height` and `staffspace_height` are not.

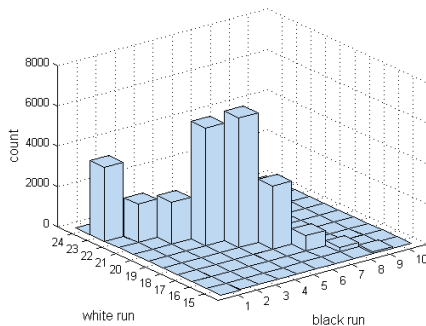
A source of difficulties is the binarization algorithm itself. State-of-the-art methods fail to correctly binarize the score under conditions such as low paper quality, gradient effect on the illumination, etc. It seems that better results could be achieved directly on the gray-level score.

Instead of computing the histogram of the runs for a single binarized score, using a threshold computed by a state-of-the-art binarization method, a procedure to compute the histogram of the runs for ‘every’ possible binary image by varying the threshold from a low to a high limit are proposed.

The rationale is that, although binarization algorithms have difficulties in finding a proper threshold, there is an interval of values that produce a proper binarized image and that will contribute to a robust histogram; threshold values outside this interval will likely produce ‘random’ runs that will disperse



(a) Complete histogram.



(b) Histogram of the pairs summing 24 (the peak value observed in Figure 5.4(c)).

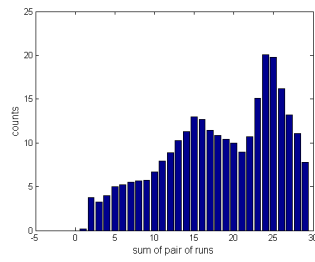
Figure 5.5: Histogram of the pairs (black run, white run) for score #17 in Figure 5.2.



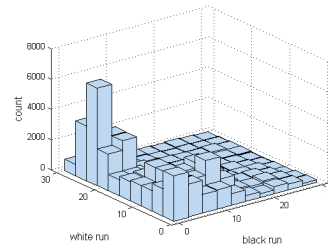
Figure 5.6: Original music score #01.

over the histogram. Figure 5.8 shows the histograms accumulating the runs for threshold from 1 to the median value of the image (it is assumed that in a music score most of the pixels are background and therefore the median pixel value is background).

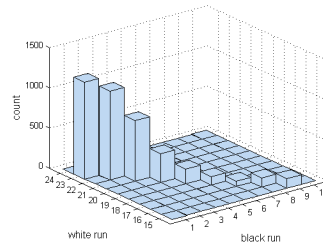
Now, the algorithm is able to correctly estimate not only the sum of `staffline_height` and `staffspace_height` but also the individual values. Moreover, the histogram for the sum shows a much more prominent peak, suggesting a more robust estimate of this length.



(a) Histogram of the sum of two consecutive runs. The most frequent value is 24.

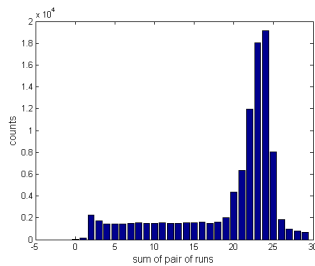


(b) Histogram of the pairs (b_run, w_run).

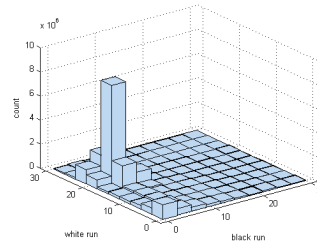


(c) Histogram of the pairs summing 24.

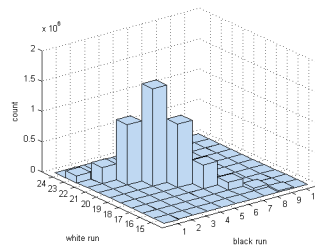
Figure 5.7: Histograms for binarized music score #01.



(a) Histogram of the sum of two consecutive runs. The most frequent value is 24.



(b) Histogram of the pairs (b_run, w_run).



(c) Histogram of the pairs summing 24.

Figure 5.8: Histograms for gray-level music score #01.

5.4 Experimental Assessment

The proposed methodology was tested on a modern set of 50 handwritten music scores from 5 different authors, as mentioned in section 3.1. The reference values of `staffline_height` and `staffspace_height` were manually measured as referred in section 3.4. The performance obtained over the dataset is summarized in Table 5.1.

Table 5.1: Mean and maximum value of errors (in pixels) in the reference lengths.

Length	Error	conventional estimation in binary images	proposed estimation in binary images	proposed estimation in gray-level images
staffline	mean	1.6	1.3	0.9
	max	4	3	2
staffspace	mean	2.7	1.3	1.0
	max	21	3	2
staffline+ staffspace	mean	2.4	0.4	0.4
	max	24	2	2

We see an improvement in the estimation of both parameters when adopting the novel approach in binary images. When applying the approach for gray-level images a further improvement is achieved. We also notice that the sum of `staffline_height` and `staffspace_height` is the most reliable estimation.

5.5 Synthesis

Studies on music recognition are under way to build up music databases and automatic performance. Since staves are crucial elements to determine the position or size of other music symbols, almost all methods extract their position in initial recognition. Before the stave candidate point is extracted, the line width and interval of the staff is usually estimated to work as reference lengths for the subsequent operations.

We presented a robust method to reliably estimate the thickness of the lines and the interline distance. We assumed that the initial image is converted, column by column, in the run-length coding. Next, we introduced the estimation of the sum of the two lengths as a more reliable estimation than the independent estimation of the two lengths. The individual lengths are then estimated as the most likely combination of runs summing to the pre-estimated total. To overcome the difficulties of binarization algorithms with low quality music scores, we propose to integrate the estimation over every possible binarization threshold.

The basic idea of estimating first the sum of quantities of interest, and then estimating the individual quantities of interest with the constraint that they sum to the estimated value, may apply in other areas of document image analysis, or in general image analysis.

Binarization Algorithm*

The work presented in Chapter 5 on the estimation of the staff line thickness and distance without binarizing the music score, working directly in the gray-scale image, opens the door to a content aware image binarization scheme applied to music scores, which we explore in this Chapter.

6.1 Content aware music score binarization

As stated in the introduction, an OMR system typically encompasses, as one of its first steps, the detection of the staff lines to facilitate the subsequent operations. An image binarization method that maximizes the ‘presence’ of the lines in the binarized image may contribute significantly to the improvement of the following operations.

A binarization method designed to maximize the number of the pairs of consecutive runs summing `line_thickness+spacing` (the peak computed over the gray-level image) will likely maximize the quality of the binarized lines. However, the direct maximization of the count of pairs of consecutive runs summing `line_thickness+spacing` could lead to a threshold value producing many, ‘noisy’, runs, and as a side effect, many runs at `line_thickness+spacing`. The use of relative histograms is also prone to problems since now one may end up choosing a threshold with a very low absolute count of runs in `line_thickness+spacing` but that, by chance, could be the highest relative count.

Therefore, we restrict the candidate thresholds to those producing a histogram of runs with the mode at `line_thickness+spacing`. If no threshold is found with this condition (note that even if the integration over all thresholds does have a mode at `line_thickness+spacing`, it is possible that no individual threshold produces a histogram with mode at `line_thickness+spacing`), we consider the minimum integer i for which there are threshold values with histogram mode at `line_thickness+spacing $\pm i$` . From the set of candidate thresholds, the proposed binarization method for music scores simply selects the threshold that maximizes the count of consecutive runs pairs on the mode.

6.1.1 Using other reference lengths to guide the binarization

The same rationale used to motivate the estimation of the sum of pair of consecutive lengths, can be used to work with sets of three or more consecutive runs. However, two problems arise when proceeding that way: there is the underlying assumption that each staff have enough lines to give meaning to the consecutive runs and one starts getting less and less values to accumulate in the histogram, potentially leading to less accurate estimations.

A potentially interesting balance is estimating the sum of two times the line thickness plus the spacing, `line_2thickness+spacing`, by working with the frequencies of triplets (black run, white run, black run). This only assumes that each staff has at least two lines, but does impact the number

*Some portions of this chapter appears in [95, 94].

of accumulated values, roughly halving it. The proposed content aware binarization method does not suffer any adaptation, besides the change of the reference length, $\text{line_thickness} + \text{spacing}$ by $\text{line_2thickness} + \text{spacing}$. Further on we will compare the two options.² In Figure 6.2 we illustrate the results obtained with the proposed approach, using the two aforementioned reference lengths (see the original score in Figure 6.1). One can observe that the resulting staff lines have good quality, with minor differences between the two results.

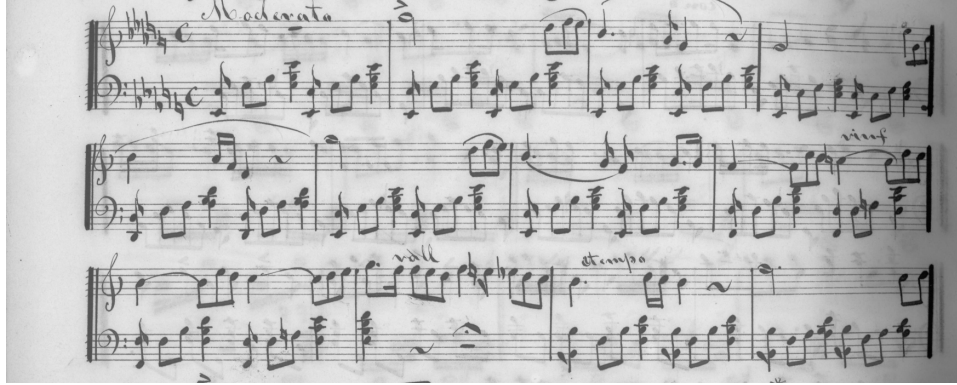


Figure 6.1: Original music score (detail).



Figure 6.2: Result of binarizing the music score in Figure 6.1.

Nevertheless, the original music score in this particular example is not correctly binarized with a global threshold. The digitization of bound documents, such as books, either performed by flatbed scanners or digital cameras often yields images that exhibit a gradient-like distortion in the average colour in the region close to the book spine. In these cases, adaptive methods can show better performance.

6.1.2 Adaptive content aware music score binarization

Despite having been presented as a global thresholding method and having been applied it to the whole image, nothing prevents the application of the ideas developed to a sampling window around a pixel p , effectively converting the proposed method to a local method.

²Any subsequent operation relying on the estimates of the individual lengths line thickness or line spacing can compute them trivially from the (robust) estimates of $\text{line_2thickness} + \text{spacing}$ and $\text{line_thickness} + \text{spacing}$: the difference of the two gives the thickness estimate and $2 \times \text{line_thickness} + \text{spacing} - \text{line_2thickness} + \text{spacing}$ provides the spacing. This procedure could be used in alternative to the method proposed in [23].

As with other adaptive methods, the size of the sampling window is a key parameter. With our approach, the sampling window should be big enough to accumulate enough information (runs) to provide a proper solution. Since the typical distortions in this kind of documents are usually vertically oriented, the local threshold should be constant along a column of the image. Therefore we suggest computing a single threshold per column, using as window a vertical strip with height equal to the height of the image and width defined by the user.

One major barrier in the application of adaptive and local models is their high computational cost. Usually, these models require the estimation of several statistical variables for each pixel of the input image (in our case for each column of the image). Depending on the size of the sampling window, the computational cost can be very high. Traditional solutions to this problem have been interpolating techniques, in which the threshold value is computed on a set of sampled columns, and then, for the rest of the columns, the threshold value is calculated by interpolating on the sampled columns.

In Figure 6.3 we illustrate the results obtained with the proposed approach, using a window width and step size of 2% of the width of the image, and cubic polynomial interpolation.



Figure 6.3: Result of binarizing the music score in Figure 6.1 with the adaptive method.

In this example, the adaptive method using the `line_thickness+spacing` reference length provided the best results, with a better staff line definition.

6.2 Binarization Procedures

The proposed method was tested against 11 state-of-art algorithms. The procedures chosen try to encompass different categories of thresholding operations: Niblack [90] Bernsen [11] and Yanowitz [141] are locally adapted threshold method, Kapur [69], Sahoo [116] and Tsallis [37] are entropy methods, Huang [63] and Chen [25] is based on object attributes, Otsu [92] is a clustering method, Tsai's moment preserving [133] and Khashman's luminance [72] are histogram based methods.

Niblack's Method

The method is based on the calculation of the local mean and of local standard deviation. The threshold is decided by the formula:

$$T(x, y) = m(x, y) + k * s(x, y) \quad (6.1)$$

where $m(x, y)$ and $s(x, y)$ are the average of a local area and standard deviation values, respectively. The value of k is used to adjust how much of the total print object boundary is taken as a part of the

given object.

Bernsen's Method

The method computes the local minimum and maximum for a neighborhood around each pixel $f(x, y) \in [0, L - 1]$, and uses the mean of the two as the threshold for the pixel in consideration:

$$g(x, y) = (F_{\max}(x, y) + F_{\min}(x, y))/2 \quad (6.2)$$

$$b(x, y) = \begin{cases} 1, & f(x, y) < g(x, y) \\ 0, & \text{otherwise} \end{cases} \quad (6.3)$$

$F_{\max}(x, y)$ and $F_{\min}(x, y)$ are the maximal and minimal values in a local neighborhood centered at pixel (x, y) .

Yanowitz's Method

The essential steps of this binarization method are the following [12]:

1. Find the *support points* $\{p_i\}$ of the image $I(x, y)$, where the image gradient is higher then some threshold value G_{th}

$$\{p_i\} = \{(x_i, y_i) : |\nabla I(x_i, y_i)| > G_{th}\} \quad (6.4)$$

2. Find the threshold surface $T(x, y)$ that equals to the image values at the support points $\{p_i\}$ and satisfies the Laplace equation at the rest of the image points:

$$\begin{aligned} T(p_i) &= I(p_i) \\ \nabla^2 T(x, y) &= 0 \quad \text{if } (x, y) \in p_i \end{aligned} \quad (6.5)$$

3. Determine the binarized image $B(x, y)$ according to (1), i.e. by comparing $I(x, y)$ with $T(x, y)$.

Kapur's Method

A threshold method based on the maximization of the entropy of the binarized image. The optimal threshold would be the one that assured the greater sum of the entropies of both classes:

$$t_{opt} = \max[E_1(t) + E_2(t)] \quad (6.6)$$

The entropy functions for each class would be:

$$E_1(t) = - \sum_{g=0}^t \frac{p(g)}{P(t)} \log \frac{p(g)}{P(t)} \quad (6.7)$$

for the objects and

$$E_2(t) = - \sum_{g=t+1}^t \frac{g_{max}}{P(t)} \log \frac{p(g)}{1 - P(t)} \quad (6.8)$$

for the background, where $p(g)$ is the probability of histogram point g and $P(t)$ is the sum of all the probabilities up to threshold t .

Sahoo's Method

This process is based in the Kapur's Method [69], explained above. The correlation method is a

weighted average of three different entropy sums, each generated with a different parameterization. The entropy function used is Renyi's entropy, defined as:

$$RE_1^\alpha = \frac{1}{1-\alpha} \ln \sum_{g=0}^t [p(g)/P(t)]^\alpha \quad (6.9)$$

for the objects and

$$RE_2^\alpha = \frac{1}{1-\alpha} \ln \sum_{g=t+1}^{g_{max}} [p(g)/(1-P(t))]^\alpha \quad (6.10)$$

for the background, where $p(g)$ and $P(t)$ are the same as above. This method finds three different thresholds, t_1 , t_2 and t_3 for $a < \alpha < 1$, $\alpha = 1$ and $\alpha > 1$, respectively. The optimum threshold is determined as the weighted average of these three values.

Tsallis' Method

This method use Tsallis entropy for the binarization process, defined as following

$$TE_1^q = \frac{1 - \sum_{g=0}^t [p(g)/P(t)]^q}{q-1} \quad (6.11)$$

for the objects and

$$TE_2^q = \frac{1 - \sum_{g=t+1}^{g_{max}} [p(g)/(1-P(t))]^q}{q-1} \quad (6.12)$$

for the background, where the real number q is the entropy index which should depend on the problem. Arguing that a pseudo-additive entropy sum function works better with the presence of non-additive information in some classes of images than the purely additive entropy sum function presented by [69], the authors presented the following decision criterion:

$$TE^q(t) = TE_1^q(t) + TE_2^q(t) + (1-q).TE_1^q(t).TE_2^q(t) \quad (6.13)$$

which should lead to the optimum threshold that maximizes the function.

Huang's Method

The process uses the distance from the gray-scale image to its binarization representing each pixel as a pair of attributes:

$$F = \{I(x, y), \mu[I(x, y)]\}, 0 \leq \mu[I(x, y)] \leq 1 \quad (6.14)$$

where $I(x, y)$ is the position of a pixel in a two-dimensional array that is the image and $\mu[I(x, y)]$ represents its fuzzy membership to the object class. Given the fuzzy memberships of all the pixels in an image, one can generate a fuzziness index FI for the whole image, using Shannon's entropy [120] in a two-dimensional space, simplifying it, with the histogram $h(g)$:

$$FI(g) = \frac{1}{g_{max} \ln(2)} \sum_{g=0}^{g_{max}} S(\mu(g)).h(g) \quad (6.15)$$

The optimum threshold will be the one that minimizes this fuzziness index.

Chen's Method

The principal steps of this algorithm are:

-
-
1. Generate the edge image using Canny edge detector.
 2. Remove the noise of the original image using the Gaussian filter, and for each edge point of the edge image, take the lowest intensity point within its 8-neighborhood as a seed.
 3. Determine the low and high intensity points: for each edge point with large gradient magnitude, choose the lowest and the highest intensity points within its 8-neighborhood as the low and the high intensity points, respectively. Take the mean of the high intensity points as the high threshold, and take the mean of the low intensity points as the low threshold. If there is obviously non-uniform illumination in the original image, the high threshold is divided by a weight $l_m \in (0, 1]$ and the low threshold is multiplied by l_m .
 4. Close the edge image generated in Step 1 by using an edge connection algorithm.
 5. Binarize the original image with the high threshold, and partition the binarized image with the closed edge image: in the high-threshold binary image generated set the pixels that belong to the edge pixels in the closed edge image as background pixels.
 6. Use the seeds to fill the partitioned high-threshold binary image generated in Step 5 with a seed-growth algorithm.
 7. Combine the primary binarized result with the low-threshold binary image: set the pixels that belong to the object regions in the low-threshold binary image as object pixels.
 8. Remove the small object region as noise where the number of pixels is less than a given threshold and obtain the final binarized image.
-
-

Otsu's Method

The method is based on homogeneous clusters from the gray-level histogram of the image. An homogeneity characteristic is the less variance between their pixels. In this manner, the optimal threshold will be the one that assures the lesser sum of the weighted variances of all the pixels in each cluster. However, because minimize the variance inside clusters is the same as maximizing the variance between them, the method computes $\sigma_{between}^2$ to each different intensity and it uses as optimal threshold the one that assures its maximization:

$$\sigma_{between}^2(t) = \sigma^2 - \sigma_{inside}^2 = p_1(t) \cdot p_2(t) \cdot [\mu_1(t) - \mu_2(t)]^2 \quad (6.16)$$

where

$$\sigma_{inside}^2(t) = p_1(t) \sigma_1^2(t) + p_2(t) \sigma_2^2(t) \quad (6.17)$$

μ_i is the mean of the cluster, p_i is the probability and σ_i^2 is the variance of cluster i with $i = 1$ for intensities below chosen threshold and $i = 2$ for intensities above.

Tsai's Method

The method considers the gray-level as a blurred version of the optimal binarization. Defining moment $m(k)$ for the gray-scale image as:

$$m(k) = \sum_{g=0}^{g_{max}} p(g) \cdot g^k \quad (6.18)$$

where $p(g)$ is the probability of histogram point g and for the binarization:

$$b(k) = p_1 \cdot m_1(k)^k + p_2 \cdot m_2(k)^k \quad (6.19)$$

where p_1 and p_2 are the probabilities and m_1 and m_2 the separate moments of each one of the classes (object and background). The method finds a threshold that guarantees moment preserving, that is, the first three moments (for $k = 1, 2, 3$) of the gray-scale are equal to the corresponding moments of the binarization.

Khashman's luminance Method

The method finds the optimum threshold t using the maximum intensity found and the mean intensity of the histogram:

$$t = |m - (g_{max} - m)| \quad (6.20)$$

where m and g_{max} are the mean and maximum intensities, respectively.

6.3 Evaluation Metrics and Results

The data set adopted to support the comparison between the different binarization procedures chosen in this work was composed by 65 handwritten scores, from 6 different authors – see section 3.1. All the music scores in this quantitative evaluation were reduced to gray level images using the `rgb2gray` function from MATLAB®. A binarization ground truth of the images was also obtained manually – see section 3.4.

The error metrics used to evaluate the performance of the proposed Binarization based in Line Spacing and Thickness (BLIST) algorithm for the binarization of the music scores were Difference from Reference Threshold (DFT), Misclassification Error (ME) and the evaluation of the results of staff finder algorithms.

Staff Finder Algorithms

This metric consist on applying staff finding algorithms to the binarizations images and measuring the percentage of false positive staff lines and the percentage of staff lines not detected. Two algorithms were tested: Stable Path algorithm³ and Dalitz algorithm [34].

The process for staff lines detection and removal usually follows the preprocessing stage in OMR systems. It was already seen in Chapter 2 that these symbols are very important for the music symbol recognition. The best binarization algorithm can be indirectly chosen showing which binarization method produce complete and non degraded staff lines.

Misclassification Error

The Misclassification Error is defined as the difference rate between the ground truth images and the resulting images from each binarization as following:

$$ME = 1 - \frac{\#(B_{bin} \cap B_{gt}) + \#(F_{bin} \cap F_{gt})}{\#B_{bin} + \#F_{bin}} \quad (6.21)$$

In Equation 6.21, B_o and F_o represents the background and foreground pixels of the original image, and B_t and F_t the background and foreground pixels in the test image, respectively. $\#$ is the number of elements in the specific set. This evaluation method was only adopted to the scores from the original

³Already introduced in Chapter 7.

dataset where ground truth is available. Nevertheless, it can be applied to global and adaptive binarizations allowing a comparison between both types of procedures.

In spite of that, for the adaptive thresholding techniques the ME results were all very similar⁴. In order to overcome this issue, further analysis was conducted based in two error rates: the Missed Object Pixel (MOPx) rate and the False Object Pixel (FOPx), dealing with loss in object pixels and excess noise, respectively:

$$MOPx = \frac{\#F_{gt} - \#(F_{bin} \cap F_{gt})}{\#F_{gt}} \quad (6.22)$$

$$FOPx = \frac{\#F_{bin} - \#(F_{bin} \cap F_{gt})}{\#F_{bin}} \quad (6.23)$$

Difference from Reference Threshold

This metric consist in comparing a supervised global threshold value with the global threshold value given by the binarization methods. The absolute difference value between the reference and tested method output values are given by

$$DRT(score) = |ref_value(score) - method_threshold(score)| \quad (6.24)$$

The DRT score indicates of how many intensity levels the two values are apart. It is important to stress that this error metric can only be applied to global thresholding methods, because adaptive methods do not use a single thresholding value for the whole image.

A frequency problem of this technique is the accuracy. This can be seen in Figure 6.4 were there is a small range of acceptable thresholds for score #02 and a wide range of possible thresholds for this score. This means that in some images if the method had a DRT value of 100 it could still produce a good binarization. In this manner, the average of the multiple supervised values was used in order to have an acceptable optimal threshold for the scores and to avoid possible outliers.

Results

Some of the algorithms tested required the input of different parameters. Table 6.1 indicates the values chosen for each method based in experimental results.

Tsallis [37]	$q = 2$
Sahoo [116]	$\alpha_1 = 0.4 ; \alpha_3 = 3$
Niblack [90]	$window = 200; k = -1$
Bernsen [11]	$window = 10; contrast = 20$
Chen [25]	$\sigma = 3; nSeedsConnected = 15; noiseRemoval = 30; radius = 10$
Ad BLIST, Ad Otsu	$windowWidth = 2$

Table 6.1: Values for the input parameters of the binarization algorithms.

The analysis will start with the results for the global methods presented in Table 6.2.

Both versions of the Binarization based in LIne Spacing and Thickness (BLIST) method, proposed in this work, performed above average. Even so, the version that uses pairs of runs instead of triplets did better in the tests. Only this version will be considered on all the following comparisons.

Entropy based binarizations and Khashman's algorithms got fairly similar results to each other. Huang and Tsai managed to top these results, with acceptable line detection rates and misclassification

⁴As will be seen on the results later in presented.

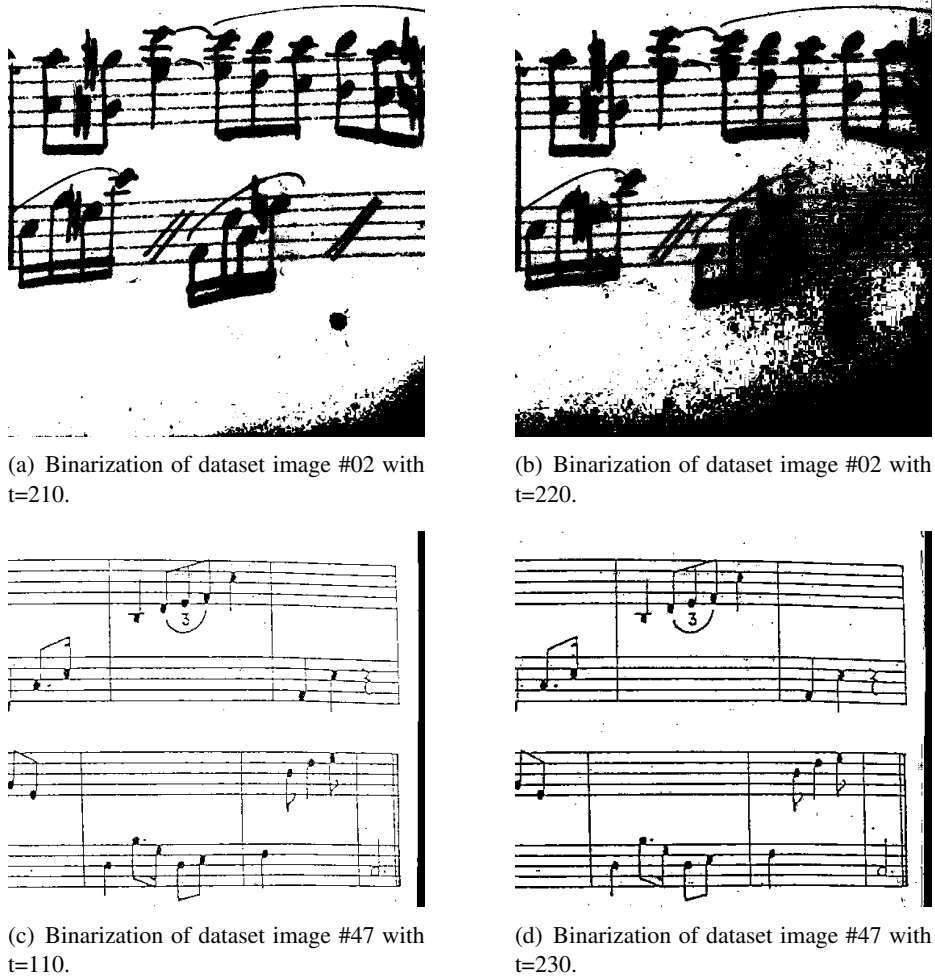


Figure 6.4: Binarization problems with DRT.

	Huang [63]	Khashman [72]	Kapur[69]	Sahoo [116]	Tsai [133]	Tsallis [37]	Otsu [92]	BLIST pairs	BLIST triplets
DRT: avg	48	33	50	50	29	50	19	19	29
ME: avg %	6.2	3.8	4.9	7.6	4.7	5.7	4.6	4.8	5.1
SP False: avg(std) %	2.6(5.5)	2.1(4.0)	1.4(3.4)	3.5(10.2)	2.1(4.1)	3.3(7.4)	2.0(3.4)	1.3(2.7)	1.7(3.7)
SP Missed: avg(std) %	18.0(34.5)	30.2(42.3)	27.1(42.3)	25.7(40.1)	17.0(30.3)	21.0(36.4)	8.6(20.5)	1.5(2.8)	2.8(6.3)
Dal False: avg(std) %	21.6(41.1)	3.2(7.8)	1.8(4.2)	5.4(25.6)	4.4(8.1)	2.4(6.0)	3.6(5.4)	3.2(5.0)	3.8(6.5)
Dal Missed: avg(std) %	39.6(36.9)	32.7(41.4)	31.2(42.0)	35.4(42.5)	25.4(35.0)	31.5(41.8)	18.8(31.0)	14.8(28.6)	14.9(27.4)

Table 6.2: Test results for various global thresholding methods, using different evaluations: difference from reference thresholds values, misclassification error (in percentage), staff detection error rates for missed and false staves (in percentage) with Stable Path and Dalitz.

error. There are, however, two binarization techniques that get consistently better results than the others: the Otsu's and BLIST methods. The only major difference is the higher missed staff detection rate for the Otsu's algorithm.

A visual analysis of the binarizations resulting from both methods shows that the BLIST process consistently results in binarized images with more connected staff lines (Figure 6.5). Although a little more noise was produced, there was not a significant loss in the connectivity of staves in any image, which can explain why BLIST also assures better results in the staff finding steps.

Global methods can generally produce good outputs. Even so, for some of the scores, like those with heterogenous light distribution resulting from the digitization process, there is no perfect thresh-

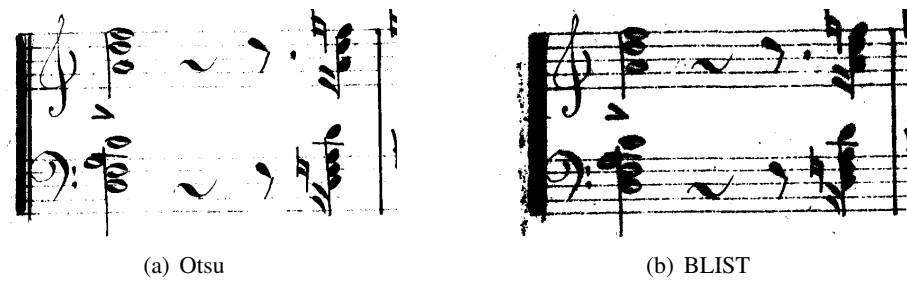


Figure 6.5: Binarization with Otsu and BLIST methods.

old. In these scores, it is not possible to find a single threshold value that produces both perfectly connected staves and no occlusion of data with noise. Although staves can be correctly found in global thresholding procedures, adaptive methods can produce results with little or no loss of information.

The adaptive version of the BLIST method was implemented as described previously. The window width used was a fixed percentage of the total image width. The interpolation of the threshold values obtained was generated with a third degree polynomial regression. Otsu's method, having good results among global methods was also implemented as adaptive, using the same reasoning.

As already mentioned, when testing local thresholding methods, the DRT evaluation cannot be performed, as there is no single threshold value for each image. In addition, because the Misclassification Errors were all very similar, the MOPx and FOPx were found for each method. The results for the adaptive methods are presented in table 6.3.

	Bernsen [11]	Chen [25]	Ad BLIST	Niblack [90]	Ad Otsu	YB [141]
ME: avg %	4.3	3.2	4.2	4.3	4.2	3.5
MOPx: avg %	24.6	22.5	15.6	22.5	21.7	12.4
FOPx: avg %	13.2	4.3	18.5	13.8	16.5	14.7
SP False: avg(std) %	1.3(3.0)	9.9(9.7)	2.1(5.6)	3.2(4.6)	2.7(5.9)	4.2(7.6)
SP Missed: avg(std) %	1.9(4.4)	33.0(32.8)	2.3(5.5)	14.2(23.2)	10.7(23.6)	7.9(13.8)
Dal False: avg(std) %	3.9(12.9)	3.4(5.6)	3.8(6.2)	3.1(5.1)	3.2(4.9)	3.5(6.1)
Dal Missed: avg(std) %	9.0(16.2)	17.3(27.0)	8.4(14.6)	10.2(14.1)	10.7(18.9)	7.7(10.1)

Table 6.3: Test results for various local thresholding methods, using different evaluations (in percentage): misclassification error, Missed Object Pixels, False Object Pixels, staff detection error rates for missed and false staves with Stable Path and Dalitz.

Ad BLIST and YB show the lowest MOPx, meaning these are the methods that find most of the correct pixels, which translates into lower missed staves rates. Even so, Ad BLIST also has a FOPx rate slightly higher than the other methods. This higher noise also translates into a slightly higher false staves rate with Dalitz method. Bernsen's binarizations, although presenting the highest missed pixel rate, seem to perform well in the staff finding steps, having both the lowest missed and false staves rates.

Comparison between global and local thresholding suggests similar results between the best of each class of techniques. Even with the increase in computational cost for adaptive methods, no significant improvement in the staff finding steps is shown.

These local methods, however, may be proven useful with further testing. In some cases where both methods were capable of correctly finding the music staves, the noise produced with the global

thresholding occludes some relevant information, as illustrated in Figure 6.6. This may prove critical in the symbol recognition steps that follow the staff line detection.



Figure 6.6: Binarization with BLIST and Adaptive BLIST methods.

6.4 Synthesis

Many binarization techniques have been proposed for digital images in the past. These methods can be applied to music scores with different rates of success, although none is based on the knowledge of the content of a music score. The main contribution of this chapter is the introduction of a content aware binarization method for music scores. The method, based on the knowledge of the staff line thickness and spacing, extracted directly from the gray-level image, tries to find the threshold that maximizes the information content of the image, as measured by these values.

Part III

Music Symbols Recognition

Staff Lines Detection and Removal*

Staff line detection and removal are one of the fundamental stages in many systems of optical music recognition. The reason for detecting and removing the staff lines lies on the need to isolate the musical symbols for a more efficient and correct detection of each symbol presented on the score.

The detection of staves is complicated due to a variety of reasons – see Chapter 2 Section 2.2. Although the problem of detecting staff lines is exacerbated with old and handwritten works, it is also present in recent printed scores. Despite the multitude of attempts to treat the problem of staff lines detection the results are not completely satisfactory yet.

In this thesis a new conceptualization to detect the staff lines is presented. The proposed paradigm begins with the work done in [105]. The main idea is to consider the staff lines as the result of the shortest path between the two margins of the music sheet, giving preference to black pixels.

7.1 A Stable Path Approach for Staff Line Detection

In the work to be detailed, the image grid is considered as a graph with pixels as nodes and arcs connecting neighbouring pixels. The weight of each arc, $w(p, q)$, is a function of pixels values and pixels relative positions. A path from vertex (pixel) v_1 to vertex (pixel) v_n is a list of unique vertices v_1, v_2, \dots, v_n , with v_i and v_{i+1} corresponding to neighbour pixels. The total cost of a path is the sum of each arc weight in the path $\sum_{i=2}^n w(v_{i-1}, v_i)$.

A path from a source vertex v to a target vertex u is said to be a *shortest path* if its total cost is minimum among all v -to- u paths. The distance between a source vertex v and a target vertex u on a graph, $d(v, u)$, is the total cost of a shortest path between v and u .

A path from a source vertex v to a sub-graph Ω is said to be a shortest path between v and Ω if its total cost is minimum among all v -to- $u \in \Omega$ paths. The distance from a node v to a sub-graph Ω , $d(v, \Omega)$, is the total cost of a shortest path between v and Ω :

$$d(v, \Omega) = \min_{u \in \Omega} d(v, u). \quad (7.1)$$

A path from a sub-graph Ω_1 to a sub-graph Ω_2 is said to be a shortest path between Ω_1 and Ω_2 if its total cost is minimum among all $v \in \Omega_1$ -to- $u \in \Omega_2$ paths. The distance from a sub-graph Ω_1 to a sub-graph Ω_2 , $d(\Omega_1, \Omega_2)$, is the total cost of a shortest path between Ω_1 and Ω_2 :

$$d(\Omega_1, \Omega_2) = \min_{v \in \Omega_1, u \in \Omega_2} d(v, u). \quad (7.2)$$

7.1.1 Algorithm outline

To a first approximation, staff lines can be considered as the only extensive objects made from black pixels in the music score, connected paths of black pixels from the left side to the right side of the music score. Assuming that paths through black pixels are preferred over paths through white pixels,

*Some portions of this chapter appears in [22, 21, 104].

staff lines can then be found among the shortest paths from the left to the right margin of the music score. Staff lines are then best modelled as paths between two regions Ω_1 and Ω_2 , the left and right margins of the score.

One may assume that staff lines do not zigzag back and forth, left and right. Therefore, one may restrict the search among connected paths containing one, and only one, pixel in each column of the image². Formally, let I be an $N_1 \times N_2$ image and define an admissible staff to be

$$\mathbf{s} = \{(x, y(x))\}_{x=1}^{N_1}, \text{ s.t. } \forall x |y(x) - y(x-1)| \leq 1,$$

where y is a mapping $y : [1, \dots, N_1] \rightarrow [1, \dots, N_2]$. That is, a staff line is an 8-connected path of pixels in the image from left to right, containing one, and only one, pixel in each column of the image.

Given the weight function $w(p, q)$, the cost of a staff can be defined as $C(\mathbf{s}) = \sum_{i=2}^{N_1} w(v_{i-1}, v_i)$. The optimal staff line that minimizes this cost can be found using dynamic programming³. The first step is to traverse the image from the second column to the last column and compute the cumulative minimum cost C for all possible connected staff lines for each entry (i, j) :

$$C(i, j) = \min \begin{cases} C(i-1, j-1) + w(p_{i-1, j-1}; p_{i, j}) \\ C(i-1, j) + w(p_{i-1, j}; p_{i, j}) \\ C(i-1, j+1) + w(p_{i-1, j+1}; p_{i, j}) \end{cases},$$

where $w(p_{i, j}; p_{l, m})$ represents the weight of the edge incident with pixels at positions (i, j) and (l, m) . At the end of this process,

$$\min_{j \in \{1, \dots, N_2\}} C(N_1, j)$$

indicates the end of the minimal connected staff. Hence, in the second step, one backtrack from this minimum entry on C to find the path of the optimal staff.

Assume one wants to find all staff lines present in a score. This can be approached by successively finding and erasing the shortest path from the left to the right margin of the score. The erase operation is required to ensure that a staff is not detected multiple times.

Consider the music score presented in Figure 7.1(a); in Figure 7.1(b) the first 11 shortest paths are traced. This example shows that music symbols placed on top of staff lines do not interfere with the detection of the staff lines. Moreover, the example also makes clear that slightly skewed scores do not pose any problem to the proposed approach.

Before presenting the complete algorithm, we introduce here for the first time the concept of a stable path in a graph, which will allow computing multiple staff lines in a single iteration, instead of sequentially computing them one at a time.

7.1.2 Stable paths on a graph

Before moving to the formal definition of a stable path on a graph, it is instructive to motivate the concept by considering a hypothetical, simplified music score, with four staff lines (rows 1, 4, 6, and 8) in a 8×9 image. The staff lines are comprised of mostly black elements; the discontinuities on some of the staff lines try to simulate the presence of noise. The graph corresponding to such score

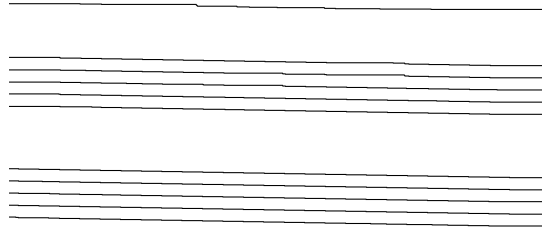
²These assumptions, 8-connectivity and one pixel per column, impose a maximum detectable 45 rotation degree.

³See Appendix A Section A.1 for details about dynamic programming.

A scale in LilyPond



(a) Skewed staff lines with music symbols.



(b) The first 11 shortest paths between left and right margins.

Figure 7.1: An illustrative example of the methodology.

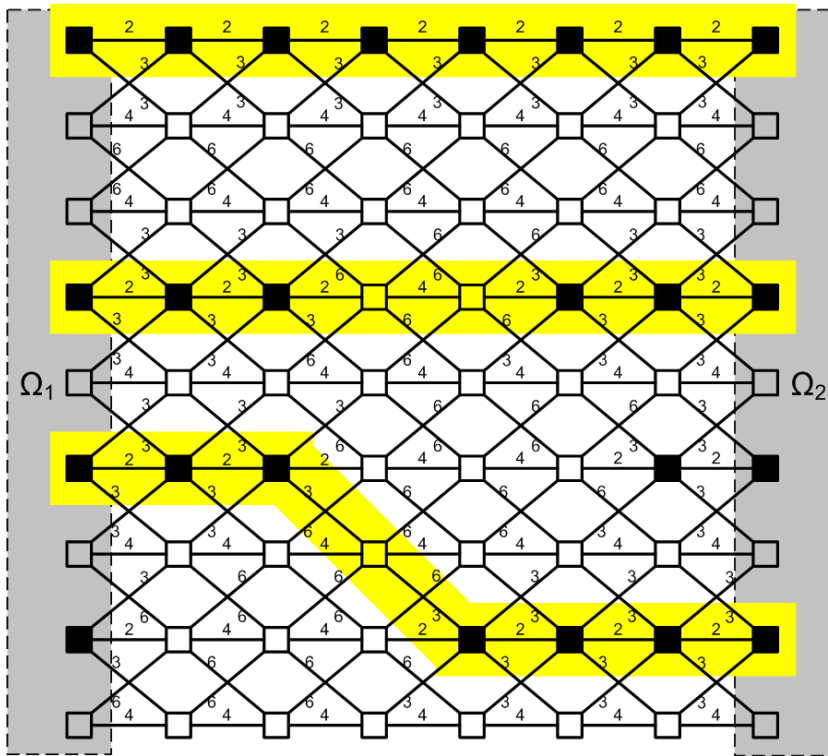


Figure 7.2: Stable paths on a toy example.

is represented in Figure 7.2. The design of the weight function will be considered next; for now, it suffices to know that it was constructed to favour paths through black pixels.

The shortest path between the left and right margins (sub-graphs Ω_1 and Ω_2) is the path corresponding to the first row, entirely through black pixels. By following the strategy just delineated, one could find the four staff lines in four iterations, sequentially. Nonetheless, although only one staff line corresponds to the shortest path, they all constitute a sort of (almost) optimal paths. The stable path concept provides a means to find all of such paths simultaneously.

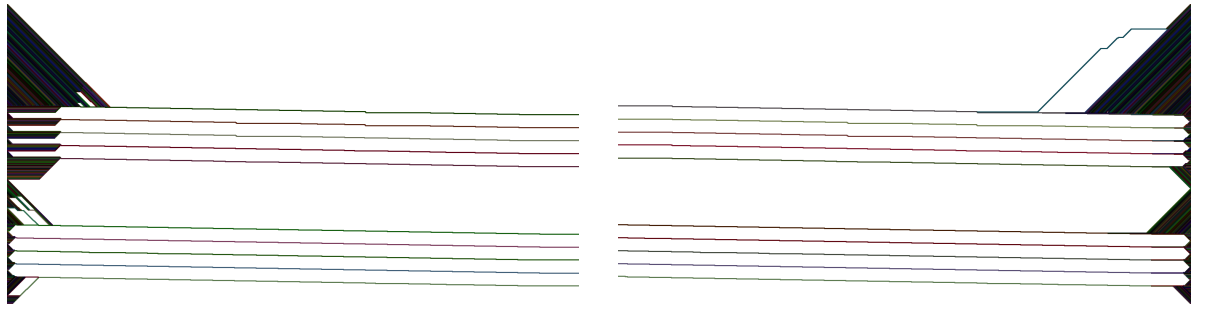
Definition. A path $\mathcal{P}_{s,t}$ is a stable path between regions Ω_1 and Ω_2 if $\mathcal{P}_{s,t}$ is the shortest path between $s \in \Omega_1$ and the whole region Ω_2 , and $\mathcal{P}_{s,t}$ is the shortest path between $t \in \Omega_2$ and the whole region Ω_1 .

The naming of stable path has its roots in dynamical systems, as it resembles stable fixed points. If one considers the function $\mathcal{F}_{\Omega_1 \rightarrow \Omega_2}()$, mapping a node $s \in \Omega_1$ to a node $t \in \Omega_2$ by finding the shortest path $\mathcal{P}_{s,t}$ between $s \in \Omega_1$ and Ω_2 , with $t = \mathcal{F}_{\Omega_1 \rightarrow \Omega_2}(s)$ as the end node of such shortest path, then

$$\mathcal{G}_{\Omega_1 \rightarrow \Omega_1}(s) = \mathcal{F}_{\Omega_2 \rightarrow \Omega_1}(\mathcal{F}_{\Omega_1 \rightarrow \Omega_2}(s)) = s$$

if and only if $\mathcal{P}_{s,t}$ is a stable path. Note that the concept of stable path is valid for any graph and any two sub-graphs in general. The computation of the stable paths on the toy example of Figure 7.2 provides the three paths yellow-highlighted in the figure.

As a second example, in Figure 7.3(a) the shortest paths between *each point* on the left margin and the *whole* right margin are traced for the score in Figure 7.1(a). As seen, the paths got attracted by the staff lines. Likewise, Figure 7.3(b) shows the shortest paths between *each point* on the right margin and the *whole* left margin. The set of stable paths between both margins result as the set of paths present in both figures.



(a) Shortest paths from each pixel in the left column and the whole right column, superimposed on the original image.

(b) Shortest paths from each pixel in the right column and the whole left column, superimposed on the original image.

Figure 7.3: Illustration of stable paths for Figure 7.1(a).

Although the computation of the stable paths may be expensive in general graphs, the computation in the graph derived from an image under the setting adopted in Section 7.1.1 has only roughly twice the complexity of the shortest path computation presented in the same Section. Noticing that the procedure delineated in Section 7.1.1 actually gives the shortest path between the whole left margin Ω_1 and each point on the right margin Ω_2 , the first step on the computation of the stable path corresponds verbatim to the computation of the shortest path presented on Section 7.1.1. In a second step one repeats the same procedure, traversing now the graph from the right column to the left. At the end of this process, if the two endpoints of a direct and reverse path coincide, we are in the presence of a stable path.

The stable paths on the toy example of Figure 7.2 provide only three out of the four staff lines, with the last stable path following partially through a segment of the third staff line and a segment of the fourth staff line⁴. Therefore, the computation of the stable paths is not guarantee to find all paths of interest. The application of the stable path procedure a second time, after erasing the paths found on the first iteration, yields a new set of paths containing a path joining the remaining segments of the 3rd and 4th staff lines. The advantage of this search based on stable paths over the sequential search of the shortest paths depends on the number of stable paths found simultaneously. While a score with 60 staff lines would require 60 iterations of the shortest path algorithm, it requires a few (typically between 4 to

⁴Note that the sequential search of the shortest path would suffer from the same limitations.

6) iterations with the stable path method. Next, the complete proposed algorithm for staff line detection is detailed.

7.1.3 Proposed Algorithm

The proposed algorithm can be implemented as a sequence of a few high-level operations, as presented in Listing 1.

```
BEGIN PreProcessing
    compute staffspaceheight and stafflineheight
    compute weights of the graph
END PreProcessing
CYCLE
    compute stable paths
    validate paths with blackness and shape
    remove valid paths from image
    add valid paths to list of stafflines
END OF CYCLE if no valid path is found

BEGIN PostProcessing
    uncross stafflines
    organize stafflines in staves
    smooth and trim stafflines
END PostProcessing
```

Listing 1: Main operations of the proposed method.

Preprocessing

To detect the staff lines, the proposed algorithm starts by estimating the staff space height, *staffspaceheight*, and staff line height, *stafflineheight*. These lengths are used as reference lengths on subsequent operations. Robust estimators are already in common use as described in Chapter 5. After estimating the reference lengths, the edges' weights are estimated as explained in the next section.

Main Cycle

The preprocessing is followed by the main cycle of the methodology, by successively finding the stable paths between the left and right margins, adding the paths found to the list of staff lines, and erasing them from the image. The erase operation sets to white the pixels on a vertical strip centred on the detected staff line. The erase operation is necessary to ensure that a line is not detected multiple times, even if its height is more than one pixel. The height of the strip was empirically fixed at *staffspaceheight*.

Stopping Rule

To stop the iterative staff line search, a sequence of (arguably) sensible rules is used to validate the stable paths found; if none of them passes the checking, the iterative search is stopped. Two validation rules were applied, both assessing features with respect to the median values obtained during the first iteration. A path is discarded if it does not have a percentage of black pixels above a fixed threshold.

The median percentage of blackness of all lines found in the first iteration of the main cycle provides the necessary reference (a threshold of 80% of the median value was empirically selected). Likewise, a path is discarded if its shape differs too much from the shape of the line with median blackness. A dissimilarity—measured as the average y -distance between both paths, after removing the means—above $\text{shapediff} = 4 \times \text{staffspaceheight}$ was selected as threshold.

PostProcessing

After the main search step, valid staff lines are post-processed. Although true staff lines never intersect, the above algorithm may occasionally create intersecting lines, detected on different iterations. Local discontinuities can induce a stable path to zigzag back and forth between consecutive lines; on the next iteration, the detected path is likely to connect the remaining segments, and therefore intersect with the path detected in first place. To preclude such final, undesired state, lines are post-processed to remove intersections: for each image column, sort on y the pixels of the detected lines and assign the i -pixel to the i -line. After this simple process, lines may touch but they do not intersect.

It is now possible to eliminate spurious lines and to cluster them in staves. Because lines are ordered, these operations require only iterating through the list of lines and starting a new staff whenever the distance between two consecutive lines is above a fixed threshold ($= 2 \times \text{staffspaceheight}$). Next, spurious staves can be eliminated. Although more robust rules can be designed, we are simply discarding sets with a single line.

Finally, lines are smoothed and trimmed. As noticeable in the example of Figure 7.1(b), before meeting with a staff line, a path travels through a sequence of white pixels. Likewise, after the end of the staff line, the path goes again through a sequence of white pixels until it meets the right margin of the image. In order to eliminate the undesirable segments, the trimming operation works per staff. For each staff, a sequence of median colours is computed as follows: for each column, the median of the colours (black and white values, as we are working with binary images) of the lines is added to the sequence. Next, the trimming points are found on this sequence: starting on the centre, we traverse the sequence to the left and right until a run of $\text{whiterun} = 2 \times \text{staffspaceheight}$ white pixels is found. The pixels between the left and right runs are kept in the staff lines. At the end, lines are smoothed with a standard average low-pass filter. Considering a staff line as a sequence $y(x)$ of y -positions, a one-dimensional averaging filter is applied. A window size of $2 \times \text{staffspaceheight}$ was selected empirically.

7.1.4 Design of the Weight Function

An immediate approach is to support the design of the weight function solely on the values of the incident nodes: if any of the corresponding pixels are black then a low cost is assigned to the edge; otherwise the edge assumes a high cost. In [21] we followed this straightforward approach, with already significant results. We call this the `baseWeight` in Listing 2. Now the weight function is generalized to account for other factors. To incorporate some prior knowledge about a music score into the shortest path process, we consider different alternatives to modify the weight function of the graph.

We considered two more attributes of a pixel to influence the main contribution to the edge's weight resulting from the values of the incident pixels. The prime intention of these additional features is to

discriminate black pixels in the staff lines from black pixels in the music symbols, penalizing the latter and favouring the former.

If a black pixel is part of a short vertical run of black pixels, then it is more likely to be part of a staff line rather than of a symbol. Therefore, a term benefiting such edges is included in the weight function. Another prior knowledge is that a staff line is likely to have another staff line at roughly `staffspaceheight` pixels (assuming that staves have at least two lines). As such, if the nearest vertical run of black pixels on the same column is excessively far from the vertical run of black pixels containing the current black pixel, then this pixel is more likely to belong to a symbol (probably a ligature) rather than to staff line. Consequently, a penalising term is incorporated in the weight function for these cases. The pseudo-code for the weight function is provided in Listing 2.

```
WeightFunction(pixelValue1, pixelValue2, vRun1,
               vRun2, nearestVRun1, nearestVRun2,
               NeighbourhoodType)
{
    value = min(pixelValue1, pixelValue2);
    weight = baseWeight(value, NeighbourhoodType);
    if( (vRun1<=STAFFLINEHEIGHT)
        OR(vRun2<=STAFFLINEHEIGHT))
        weight = weight - delta;
    if( (nearestVRun1>=STAFFSPACEHEIGHT+STAFFLINEHEIGHT)
        OR(nearestVRun2>=STAFFSPACEHEIGHT+STAFFLINEHEIGHT))
        weight = weight + delta;
    return weight;
}
```

Listing 2: Pseudo-code for the weight Function. The base weight was set to 4 on black pixels and 8 on white pixels for 4-neighbourhoods and to 6 and 12 on for 8-neighbourhoods. The *delta* penalizing term in the weight function was set to 1. For efficiency, weights were designed with integer values.

7.2 Dalitz's Algorithm

Another staff line detection method used in this thesis is Dalitz's algorithm. This is implemented in the MusicStaves plug-in that is incorporated in GAMERA framework⁵ (Generalized Algorithms and Methods for Enhancement and Restoration of Archives) which is a toolkit for building document image recognition systems.

Dalitz' algorithm is a generalization of the method described in [84]. It starts by estimating the values for the staff line thickness and the staff space height. These values are estimated by the technique used in Fujinaga [53]: the most frequent black-runs represents the staff line height (`staffline_height`) and the most common white-runs represents the vertical line distance within the same staff (`staffspace_height`). This technique starts by computing the vertical run-lengths representation of the image.

The process of Dalitz for finding the staff lines operates on a set of "staffsegments" and requires methods not only for linking two of those "staffsegments" horizontally and vertically, but also for

⁵See <http://ldp.library.jhu.edu/vhost-base/gamera> and [19] for more details.

merging two segments with overlapping positions into one. The algorithm will be described in the following steps :

1. Add vertical links between “staffsegments” with a vertical distance around `staffline_height+staffspace_height`.
2. Add horizontal links between adjacent “staffsegments” possibly belonging to the same staff line.
3. Partition the resulting graph into connected subgraphs; each subgraph that is wide and high enough corresponds to a staff.
4. All “staffsegments” within a system are labelled as belonging to a certain staff line. Segments of the same line at the same horizontal position are merged into one segment.
5. Due to ledger lines, ties and beams, some subgraphs will contain too many staff lines. To reduce them to a predefined number of lines per staff (typically five for modern notation, four for chant and six for tablature), the outer staff lines of each staff are subsequently removed until the predefined number of staff lines remains.

Dalitz developed the following algorithm to find the “staffsegments”:

1. Extract horizontal runs with more than 60 percent black pixels within a window of width `staffspace_height`.
2. The resulting filaments are vertically thinned by replacing each vertical black run with its middle pixel. For black runs higher than $2 \times \text{staffline_height}$, more than one skeleton point is extracted.
3. The resulting skeleton segments wider than $2 \times \text{staffline_height}$ are the “staffsegments”.

7.3 Evaluation Metrics and Results

Although the assessment of a new staff detection algorithm can be done by visually inspecting the output on a set of scores—as adopted on [105]—, here the comparison is supported on quantitative measures. In section 3.1 the adopted database was presented. To conveniently measure the performance of a staff line detection algorithm, two different error metrics are considered: the percentage of staff lines falsely detected and the percentage of staff lines missed to detect.

To evaluate these metrics, the process starts by computing the average Euclidian distance between each reference staff line – see section 3.4 for ground-truth information – and each staff line actually detected. Then, the matching problem on the resulting bipartite graph is solved by minimizing the assignment cost (= distance). Only pairs with average error-distance below `stafflineheight` are assumed to be correctly matched (the other pairs were assumed to originate from a false positive staff line being matched to an undetected true staff line and were therefore unmatched). Now the two metrics are the number of unmatched detected staff lines (false positive) and unmatched reference staff lines (failed to detect). It should be noted that these metrics only measure whether staff lines are found, not how good the match is.

The proposed algorithm was compared with the three methods considered in [34] for staff line detection. As Dalitz’s algorithm performed significantly better than the others two methods evaluated in [34], we have only considered Dalitz’s results in subsequent analysis. This performance of the staff line detection algorithms agrees with the relative performance for staff line removal reported in [34].

The parameters of the stable path algorithm were preliminary tuned with an independent set of images, yielding the thresholds already presented in the method description.

The effects of the different deformations over the respective parameter ranges are shown in Table 7.1; the shortest path method relates to the version presented in [21], including the weight function and post-processing. The curvature deformation is done with a half sine wave; the line y-variation generates random defects on the y-position of the staff line; the typeset emulation tries to imitate sixteenth-century prints with staff lines interruptions between symbols and random vertical shifts [34]. With respect to the distortions considered, the stable path based approach (and the shortest path approach) outperforms the Dalitz algorithm⁶. In fact, the performance of the proposed approach is almost independent of intensity of the deformation, for the range of values considered. This performance gain is even more noteworthy as the Dalitz algorithm is receiving as input the correct number of lines per staff. Had not this been the case, the difference between both would have been much larger. The current implementation of the stable path algorithm runs as fast as the Dalitz algorithm (and about five times faster than the shortest path version), as available in the MusicStaves Toolkit [33]. By generalizing the design of the weight function and enhancing the postprocessing, we were able to improve the detection performance on our initial results in [21]. The use of the stable paths improved the detection speed.

In a simple sensitivity analysis of the approach to the selected parameters, the experiments were repeated with values for `stafflineheight` and `staffspaceheight` one pixel above and below the true estimated values. Note that all the method parameters are scaled by one of these two estimates. In both cases, the stable path method presented a good performance and continued to yield the best results (the strongest degradation occurred for the rotation degradation, with $\text{angle} = 5^\circ$, where both error rates increased to 1.7%).

In a second experiment we evaluated the staff line detection methods on a set of 40 real music scores – see section 3.1 –, for which reference staff lines were manually outlined – see section 3.4. Images were previously binarized with the Otsu threshold algorithm, as implemented in the Gamera project⁷. The evaluation of the detection algorithms yielded the results presented in Table 7.2. Again, and although the correct number of staff lines per staff was inputted to the Dalitz' algorithm, the stable path approach obtained the best performance.

⁶For the deformations not shown, the stable path is not significantly better than Dalitz.

⁷<http://gamera.sourceforge.net>

	Angle	rotation					Runtime
		-5	-2.5	0	2.5	5	
Error	Stable path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.6 (3.5); 0.6 (3.5)	0.7 (3.5); 0.7 (3.5)	1.2 (4.0); 1.2 (4.0)	858 sec.
	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.6 (3.5); 0.6 (3.5)	0.7 (3.5); 0.7 (3.5)	1.2 (4.0); 1.2 (4.0)	6006 sec.
	Dalitz	17.8 (22.0); 51.7 (38.1)	8.6 (14.0); 15.5 (28.7)	0.0 (0.0); 0.0 (0.0)	4.2 (19.6); 9.8 (29.0)	5.5 (9.3); 37.5 (41.9)	612 sec.
Error	Amplitude/staffwidth	0.02	0.04	0.06	0.08	0.10	Runtime
	Stable path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.2 (4.0); 1.2 (4.0)	822 sec.
	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.2 (4.0); 1.2 (4.0)	5554 sec.
Error	Dalitz	12.7 (29.0); 12.6 (28.7)	53.8 (44.5); 55.7 (45.2)	83.3 (30.6); 83.9 (30.4)	95.8 (17.5); 100.0 (0.0)	96.0 (17.5); 100.0 (0.0)	639 sec.
	Rate whitened pixels	0.03	0.05	0.07	0.09	0.11	Runtime
	Stable path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.9 (3.7); 0.9 (3.7)	1.2 (3.8); 1.2 (3.8)	2.1 (4.6); 2.3 (4.8)	809 sec.
Error	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.9 (3.7); 0.9 (3.7)	1.7 (4.0); 1.9 (4.3)	5.3 (7.4); 7.0 (9.6)	5122 sec.
	Dalitz	0.0 (0.0); 0.0 (0.0)	0.3 (1.4); 0.3 (1.4)	26.7 (25.3); 29.9 (27.2)	89.3 (54.6); 86.9 (25.6)	54.5 (55.9); 95.2 (17.0)	872 sec.
Error	Max deviation, n	2	3	4	5	6	Runtime
	Stable path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.1 (3.8); 1.1 (3.8)	767 sec.
	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.1 (3.8); 1.1 (3.8)	5122 sec.
Error	Dalitz	31.0 (50.7); 31.7 (46.1)	22.1 (38.9); 32.6 (45.6)	15.7 (27.2); 33.7 (45.0)	13.0 (20.1); 33.7 (45.0)	12.8 (18.6); 34.2 (44.7)	768 sec.
	Max gap width, n_{gap}	1	4	7	10	13	Runtime
	Stable path	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	739 sec.
Error	Sortest Path	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	5085 sec.
	Dalitz	19.7 (34.1); 13.7 (21.6)	21.4 (27.4); 15.4 (17.3)	22.3 (30.0); 17.4 (19.0)	24.2 (38.9); 16.7 (22.0)	31.4 (42.3); 19.2 (20.3)	703 sec.
Error	Max vertical shift, n_{shift}	1	4	7	10	13	Runtime
	Stable path	0.6 (3.5); 0.6 (3.5)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	886 sec.
	Sortest Path	0.6 (3.5); 0.6 (3.5)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	6106 sec.
Error	Dalitz	3.3 (10.4); 2.3 (7.3)	15.9 (27.1); 12.0 (17.1)	39.0 (48.7); 24.6 (27.3)	48.7 (60.8); 29.3 (30.7)	58.7 (54.9); 37.3 (29.0)	842 sec.

Table 7.1: Effect of different deformations on the overall staff detection error rates in percentage: average (standard deviation) of the false detection rate and miss detection rate. See [34] for parameter details.

	False detection rate	Miss detection rate	Runtime
Dalitz	5.2% (10.4)	5.9% (11.3)	112 sec.
Shortest path	1.4% (3.5)	2.5% (7.3)	612 sec.
Stable path	1.3% (5.7)	1.4% (6.4)	115 sec.

Table 7.2: Detection performance on real music scores in percentage: average (standard deviation).

7.3.1 Staff line removal

Staff line detection algorithms can be used as a first step in many staff removal algorithms. To understand the potential of our algorithm to leverage the performance of existing staff removal algorithms, we conducted a series of experiments, comparing existing versions of staff line removal algorithms with modified versions of them, making use of the stable path algorithm at the staff line detection step⁸. The quantitative comparison of the different algorithms is totally in line with the comparison presented in [34]. Adopting the naming convention from [34], the following algorithms were adapted: LineTrack Height, LineTrack Chord, Roach/Tatem. The original version of the algorithms were considered as available in [33], making use of the Dalitz' algorithm in the detection phase; the modified versions use instead the stable path for detecting lines. We also adopted the same error metrics (individual pixels, staff-segment regions and staff interruption location) and conducted the comparative study on the same test set⁹. It turned out that the effects of the deformations and the insertion of the stable path as the detection algorithm are similar for all three error metrics. Hence, in Tables 7.3 and 7.4, we just present the results for the pixel error rate and for the Line Track Height algorithm (original and modified), plus the Skeleton algorithm, which exhibited a competitive performance in [34]. The Line Track Height (LTH) checks whether the vertical black run through the staff line point is longer than ($= 2 \times \text{stafflineheight}$).

A first observation is that, overall, the replacement of the Dalitz method by the stable path approach as the staff detection step improved the results in the algorithms under comparison. Additionally, the LineTrack Height algorithm with the stable path consistently outperformed the other algorithms. Nevertheless, the skeleton method [34], which does not have a clear line detection step, continues to present a competitive performance. It is worth to finalize by noticing that the skeleton algorithm is about two times slower than the modified Line Tracking Height algorithm.

7.4 Staff Removal Competition

In this thesis, we participated in a staff removal competition promoted by International Conference on Document Analysis and Recognition (ICDAR)¹⁰. This was a competition with focus in the recognition of handwritten music scores and in the identification of the authorship of an anonymous music score. The database used was the CVC-MUSCIMA by Alicia Fornés, available to download in http://www.cvc.uab.es/cvcmuscima/index_database.html. As already mentioned in section 2.5 this is a database containing 1000 music sheets written by 50 different musicians. Each

⁸See Appendix A Section A.2 for more details about the staff lines removal algorithms adopted in this thesis.

⁹See Appendix A Section A.3 for the description of the error metrics.

¹⁰<http://www.cvc.uab.es/cvcmuscima/competition/index.htm>

Angle	rotation				
	-5	-2.5	0	2.5	5
Stable path + LTH	1.7 (0.7)	1.5 (0.7)	1.4 (0.7)	1.4 (0.7)	1.6 (0.7)
Dalitz + LTH	19.4 (18.4)	5.2 (8.7)	1.4 (0.8)	4.4 (8.8)	17.5 (18.9)
Skeleton	1.9 (0.9)	1.7 (0.8)	1.5 (0.7)	1.6 (0.7)	1.7 (0.8)
Amplitude/staffwidth	curvature				
	0.02	0.04	0.06	0.08	0.10
Stable path + LTH	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.5 (0.7)	1.6 (0.7)
Dalitz + LTH	3.8 (5.8)	14.0 (12.2)	22.8 (13.7)	31.1 (11.0)	35.0 (10.6)
Skeleton	2.6 (2.4)	5.2 (5.1)	8.1 (7.2)	11.9 (8.6)	15.4 (10.4)
Rate whitened pixels	white speckle				
	0.03	0.05	0.07	0.09	0.11
Stable path + LTH	11.9 (3.1)	17.2 (4.9)	21.1 (5.9)	24.0 (6.7)	26.1 (7.2)
Dalitz + LTH	11.5 (3.2)	16.8 (4.9)	26.7 (8.0)	53.3 (14.9)	73.3 (14.6)
Skeleton	14.6 (3.2)	21.5 (4.6)	27.1 (5.6)	35.2 (12.8)	46.9 (18.7)
Max deviation, n	line y-variation				
	2	3	4	5	6
Stable path + LTH	1.2 (0.7)	1.3 (0.7)	1.3 (0.6)	1.4 (0.6)	1.4 (0.6)
Dalitz + LTH	9.0 (13.2)	10.4 (14.1)	10.9 (14.5)	10.9 (14.5)	11.0 (14.6)
Skeleton	1.5 (0.8)	1.7 (0.8)	2.2 (0.9)	3.7 (1.7)	5.2 (2.2)
Max gap width, n_{gap}	typeset emulation I				
	1	4	7	10	13
Stable path + LTH	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)
Dalitz + LTH	2.6 (1.8)	2.9 (2.0)	3.2 (1.7)	2.9 (1.7)	3.0 (1.8)
Skeleton	26.4 (9.8)	27.3 (10.1)	27.2 (11.3)	25.5 (9.8)	26.4 (10.3)
Max vert. shift, n_{shift}	typeset emulation II				
	1	4	7	10	13
Stable path + LTH	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.5 (0.7)	1.6 (0.7)
Dalitz + LTH	1.5 (0.8)	2.8 (1.6)	3.3 (2.5)	3.8 (2.4)	4.7 (3.7)
Skeleton	7.9 (8.9)	24.1 (9.1)	26.7 (11.0)	26.1 (9.6)	29.1 (10.7)

Table 7.3: Effect of different deformations on the overall staff removal error rates in percentage: average (standard deviation).

	Stable path + LTH	Dalitz + LTH	Skeleton
Pixel Error Rate	2.8 (1.2)	3.8 (2.6)	6.5 (8.2)

Table 7.4: Removal performance on real music scores (in percentage): average (standard deviation).

writer wrote the same 20 scores, using the same pen and the same kind of music paper with printed staff lines. Moreover, each music page was distorted using the algorithms from Dalitz et al.[34], yield a total of 12000 images – see Figure 7.4 – with ground truth for the staff removal task.



Figure 7.4: Some examples of music scores from CVC-MUSCIMA database.

For the staff removal competition, the entire database was equally divided into two parts: the first

50% of the images were used as training set and the other 50% were used as testing set [44]. The algorithms that competed were the following:

1. ISI01 with variants ISI01-Rob and ISI01-HA: the thinned images are analysed and grouped as (1) straight staff line or (2) non-straight or curved staff lines. The straight staff lines are further divided into horizontal staff lines and non-horizontal straight lines. The staff lines are detected based on features of each group.
2. INP02 with variants INP02-SP and INP02-SPTrim: the proposed stable path algorithm in this chapter.
3. NUS03: The method uses the staff height and staff space to predict the lines' direction and to fit an approximate staff line curve for each image. This curve is used to identify the location of staff lines. The staff lines are assumed to be parallel.
4. NUG04 with variants NUG04-Fuji, NUG04-LTr and NUG04-Skel: see [53, 34] for more details.

Figure 7.5 presents the results for the competition where it is possible to see that our method (INP02-SP) is ranked in third, with a good overall error rate (2.83%).

Distortion	Error Rate	ISI01-Rob	ISI01-HA	INP02-SP	INP02-SPTrim	NUS03	NUG04-Fuji	NUG04-LTr	NUG04-Skel
01- Ideal	No Rej.(#)	1.50 (0)	1.50 (0)	1.5 (0)	1.51 (0)	1.54 (0)	1.53(0)	2.08 (0)	2.11 (1)
	With Rej.	1.50	1.50	1.5	1.5	1.54	1.53	2.08	2.31
02- Curvature	No Rej.(#)	1.66 (0)	1.66 (0)	1.8 (0)	1.80 (0)	2.83 (0)	38.45 (3)	– (500)	13.38 (148)
	With Rej.	1.66	1.66	1.8	1.8	2.83	38.82	100	39.02
03- Interruption	No Rej.(#)	0.92 (0)	0.91 (0)	5.16 (5)	5.19 (5)	1.04 (0)	18.79 (499)	– (500)	– (500)
	With Rej.	0.92	0.91	6.10	6.14	1.04	99.84	100	100
04- Kanungo	No Rej.(#)	2.84 (0)	2.84 (0)	2.86 (0)	2.87(0)	2.91 (0)	2.84 (0)	4.33 (0)	7.93 (0)
	With Rej.	2.84	2.84	2.86	2.87	2.91	2.84	4.33	7.93
05- Rotation	No Rej.(#)	1.76 (0)	1.76 (0)	2.03 (0)	2.03 (0)	3.06 (0)	40.40 (8)	– (500)	4.60 (48)
	With Rej.	1.76	1.76	2.03	2.03	3.06	41.35	100	13.76
06- staffline thickness v1	No Rej.(#)	2.44 (0)	2.17 (0)	2.70 (0)	2.71 (0)	3.38 (0)	2.53 (0)	3.74 (0)	4.14 (0)
	With Rej.	2.44	2.17	2.70	2.71	3.38	2.53	3.74	4.14
07- staffline thickness v2	No Rej.(#)	2.18 (0)	2.15 (0)	3.01 (0)	3.02 (0)	3.41 (0)	2.20 (0)	3.74 (0)	3.72 (0)
	With Rej.	2.18	2.15	3.01	3.02	3.41	2.20	3.74	3.72
08- staffline y-variation v1	No Rej.(#)	2.00 (0)	1.89 (0)	2.43 (0)	2.45 (0)	3.01 (0)	3.21 (0)	5.56 (2)	6.34 (0)
	With Rej.	2.00	1.89	2.43	2.45	3.01	3.21	5.94	6.34
09- staffline y-variation v2	No Rej.(#)	1.92 (0)	1.83 (0)	2.27 (0)	2.28 (0)	3.02 (0)	3.28 (0)	3.34 (2)	4.98 (0)
	With Rej.	1.92	1.83	2.27	2.28	3.02	3.28	3.72	4.98
10-Thickness Ratio	No Rej.(#)	2.86 (0)	2.86 (0)	6.89 (0)	6.89 (0)	– (500)	– (500)	10.78 (0)	15.96 (0)
	With Rej.	2.86	2.86	6.89	6.89	100	100	10.78	15.96
11-TypeSet emulation	No Rej.(#)	1.61 (0)	1.60 (0)	1.60 (0)	1.61 (0)	1.70 (0)	7.95 (0)	3.29 (8)	18.41 (477)
	With Rej.	1.61	1.60	1.60	1.61	1.70	7.95	4.83	96.25
12- WhiteSpec	No Rej.(#)	1.48 (0)	1.48 (0)	1.73 (0)	1.74 (0)	2.04 (0)	1.92 (0)	1.76 (0)	6.69 (0)
	With Rej.	1.48	1.48	1.73	1.74	2.04	1.92	1.76	6.69
Overall E.R.	No Rej.(#)	1.93 (0)	1.89 (0)	2.83 (5)	2.84 (5)	2.54 (500)	10.37 (1010)	4.29 (1512)	6.87 (1174)
	With Rej.	1.93	1.89	2.91	2.92	10.66	25.46	28.41	25.09

Figure 7.5: Staff removal results. Error Rate (E.R.) in % for each one of the 12 distortions. We show the Error Rate with and without rejection. In case of the Error rate without rejection (No Rej.), we also show the number # of rejected images. From Fornés et al. [44, Table I].

7.5 Synthesis

This chapter presented a robust algorithm for the automatic detection of staff lines in music scores. The proposed method uses a very simple but fundamental principle to assist detection and avoid the difficulties typically posed by symbols superimposed on staff lines. The stable path approach for staff line detection algorithm is adaptable to a wide range of image conditions, thanks to its intrinsic robustness to skewed images, discontinuities, and curved staff lines. The proposed approach is robust to discontinuities in staff lines (due to low-quality digitization or low-quality originals) or staff lines as thin as one pixel.

In order to take full advantage of the method, existing staff line removal algorithms were enhanced by using the stable path method as their first processing step.

Musical Symbols Segmentation and Classification*

The musical symbols detection is a stage on an OMR system where operations to localize and to isolate musical objects are applied. The proposed algorithms must be robust to several problems imposed by music symbols. The complexity of this phase, caused by printing and digitization, as well as the paper degradation over time, is directly related, not only to the distortions inherent in staff lines, but also to broken, connected and overlapping symbols (see Figure 8.1), differences in sizes and shapes (see Figure 8.2), noise, and zones of high density of symbols (see Figure 8.3).

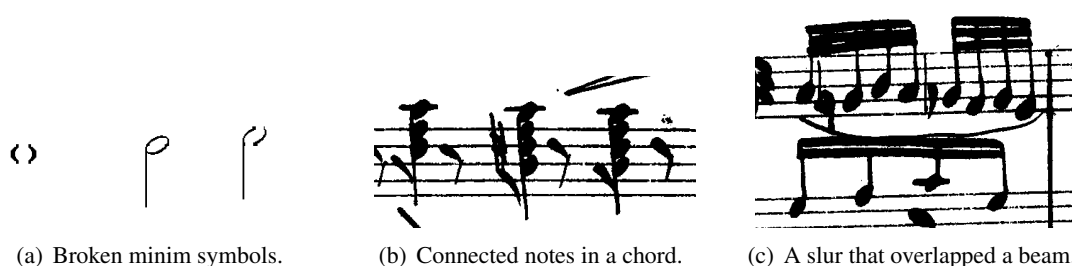


Figure 8.1: Broken, connected and overlapping symbols.



Figure 8.2: Variability in sizes and shapes between handwritten and printed scores.

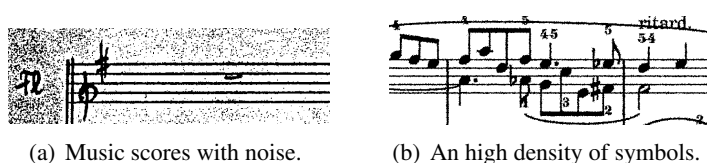


Figure 8.3: Noise and zones of high density of symbols.

As already mentioned in chapter 2 there are a lot of strategies to detect and extract the music symbols. From our point of view, some of them such as projections, descriptors and template matching

*Some portions of this chapter appears in [108, 109].






could not be suitable for handwritten music scores due to the wider variability of the objects. Other methods, such as grammars and graphs are complex because they need to cover all possible combinations of symbols. In this thesis, we choose to implement and test two processes to detect the music symbols: (1) music symbols segmentation and (2) music symbols segmentation through recognition. In both methods, information related to context, geometric features and music rules are used. The main difference between the two processes is in the segmentation step, where in the latter, machine learning techniques are used to aid the segmentation process, as we will describe.

8.1 Music Symbols Segmentation

In this section, the music symbols segmentation will first be presented and then the classification method used to classify the extracted objects will be explicated next.

The process to segment the objects was based on an hierarchical decomposition of a music image and in contextual information and music writing rules. This hierarchical decomposition choice aims the disintegration of a complex problem towards various simple problems. Hence, a music sheet is first analyzed and splitting by its staves, as yielded by the staff lines removal step – see Figure 8.4. The processing continues with the segmentation process, which is schematized in Figure 8.5. For each staff (set of staff lines) a clef detection process (step I) is applied. If this symbol exists, the process to detect key (step II) and time signatures (step III) is performed. If a clef symbol does not exist (for instance, the current music sheet can be the second sheet of that music work) the algorithm searches for stored information about the last analyzed music sheet. After step III (Time Signature Detection) a procedure to find stems is made (step IV). If these objects are present in the staff then the algorithm searches for the existence of closed note heads (step V) – crotchet –, otherwise the algorithm goes to breves and semibreves detection (step IXb). For each closed note head a beams detection (step VI) process is applied. The closed note heads with beams cannot have flags. For the other cases a flag detection step (step VII) is needed. If the algorithm does not detect any closed note head, then it goes directly to step IXa (Minim Detection). The remaining stages are accidentals detection (step X), dots detection (step XI), accents detection (step XII), rests detection (step XIII) and bar lines detection (step XIV).

During the execution of the segmentation process the symbols were divided considering their graphical position and geometric features:

1. symbols that are characterized by a vertical segment (stem) and an oval note head: crotchet (e.g. ) , notes with flags (e.g. ) and minim (e.g. ) .
2. symbols that link the notes: beams (e.g. ) .
3. the remaining symbols connected to staff lines: clefs, rests (e.g. ) , accidentals (e.g. b , \sharp , \flat) and time signature (e.g. C) .
4. symbols above and under staff lines: ties, slurs (e.g. \frown) and accents (e.g. $>$) .

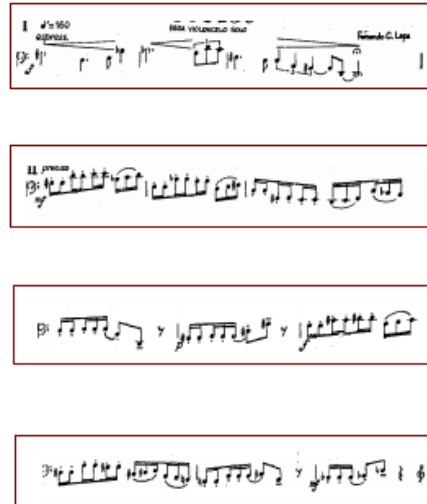
All music scores must have a clef. In this manner, the first module in the hierarchical decomposition of the music score is the clef detection. After this symbol, we can have a set of accidentals symbols composing the key signature. The time signature is placed after the clef symbol and any key signature,



(a) Music score with staff lines.



(b) Music score without staff lines.



(c) The first four staves of the music score.

Figure 8.4: First step in the segmentation procedure: split the score by staves (set of staff lines).

defining the meter of the music. Time signatures establish the number of beats in each bar line and which note value constitutes one beat.

Note heads are usually the object that appears with higher frequency in a score. Hence, it is natural the option of detecting stems after module III (Time Signature detection) in order to detect, after that, the note heads. Besides, almost all musical rules are dependent on these objects. For instance, beams only exist connecting eighth notes or smaller rhythmic values. A similar dependency can be established for the remaining symbols: accidentals only exist before each note head and at the same height, accents exist over or under a note head and dots are placed after note heads.

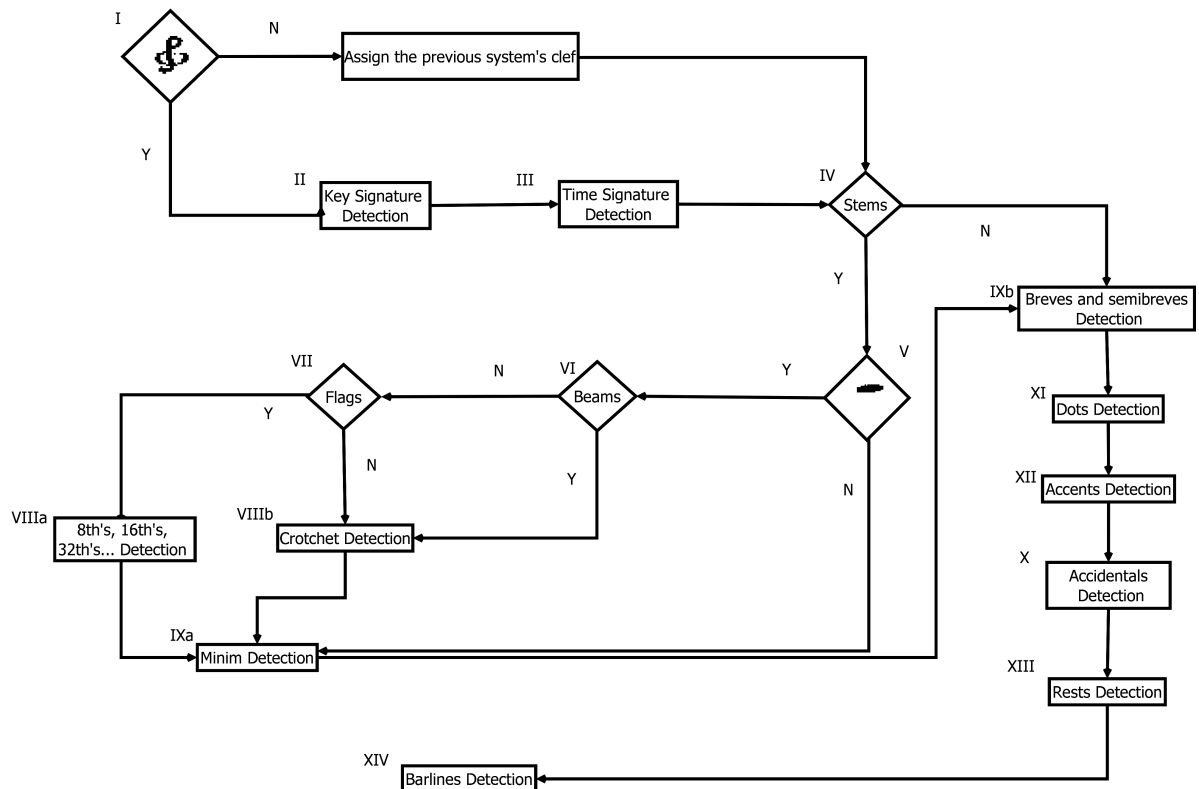


Figure 8.5: Diagram of the musical symbols extraction algorithm.

The multiple steps of the segmentation process use a common technique that is the connected components technique with a possible threshold for the distance between objects. This freedom degree for the neighboring pixels is due to the existence of broken symbols – see Figure 8.1(a). The contextual information and music rules are also used in all modules. To detect key and time signatures, the algorithm searches for connected components that are placed a certain number of pixels away from the previous detected symbol. For the key signature, distances between accidentals are also considered. Another important issue that is taken into account is the maximum number of accidentals allowed per key. Flags are symbols that are always placed to the right of the stem. As opposed to these signs, accidentals exist before each notehead and at same height. In this manner, the connected components process is only applied on these areas. It is worth noting that all the threshold values used were obtained experimentally.

Clefs detection

The clefs detection process starts by computing the connected components using a threshold for the distance between objects of $spaceHeight/2$. This value increases until the algorithm finds an object with a width of $3 \times spaceHeight$. Once again this degree for the neighboring pixels is due to the existence of broken symbols, as previously mentioned. However, in some cases when the threshold is increased, to incorporate symbols with a big distance between its segments, it finishes encompassing the symbols after the clef. In order to overcome this issue, when this procedure fails to find the clef sign and the main algorithm detects notes, other clef detection process is applied.

Matrices with 10 rows and 5 columns are created for each of the three possible clefs (treble, bass

and alto), containing in each cell the number of black pixels – see Table 8.1. These masks are used to calculate correlation degrees. This procedure starts with an image from the beginning of the staff with a certain height and width; the three masks are used. While the correlation is below a threshold the width of the image is increased – see Figure 8.6. The stop criterion, when no match symbol is found, is the maximum number of iterations. If a clef symbol is not detected the algorithm searches for the stored information about the last analyzed music sheet. It is important to stress that, since the masks were created through counting the number of black pixels and not using the exactly position of the black pixels we only used a symbol of each type of clef to compute the matrices. We believe as a result of our experiences that this technique captures the differences in the writing style in handwritten music scores.

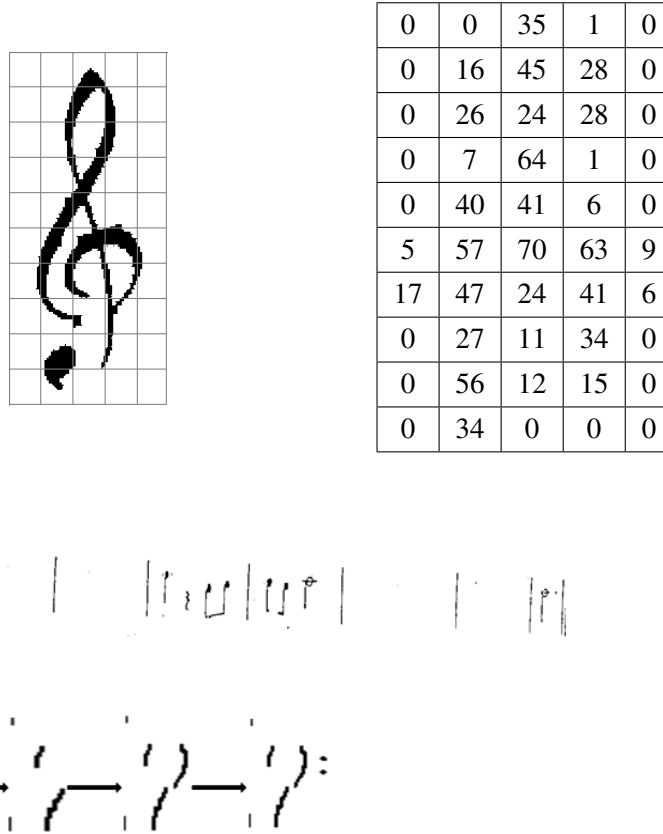


Figure 8.6: Correlation process.

In Figure 8.7 an example of a possible result after the clef detection step is presented.

Key and Time Signatures Detection

The key signature is a set of accidentals symbols that appears after the clef sign. The algorithm searches for connected components that are placed $3 \times spaceHeight$ pixels away from clef symbol with an height of $4 \times spaceHeight + 2 \times lineHeight$ and a width of $2 \times spaceHeight$. These thresholds were obtained experimentally. The distance allowed between each accidental is below $spaceHeight/2$. A key signature can have between 0 and a maximum of 5 accidentals – see Figure 8.8(c).

There are several possible symbols for the representation of time signatures. This type of sign is sometimes represented by two figures where the numerator occupies the two top space and the



(a) The third staff from the score 8.4(a).



(b) The score without clef symbol.

Figure 8.7: The result after the clef detection step.

denominator occupies the two bottom space, or simply it is represented by a C used to indicate $\frac{4}{4}$ or a C used to indicate $\frac{2}{2}$. The time signature is placed after the clef symbol and any key signature. The process starts to search for connected components that are a $2 \times \text{spaceHeight}$ distance from the last object detected discarding the ones with a width above $3 \times \text{spaceHeight}$ – see Figure 8.8(d).

Note heads detection

For the note heads detection, the algorithm starts with the stems removal procedure where each vertical run of black pixels above $2 \times \text{spaceHeight} - \text{lineHeight}$ is eliminated – see Figure 8.9(a). For each of the previous positions, a connected components process is applied to search for objects with an height above $2 \times \text{lineHeight}$ and width above $\text{spaceHeight} \times 0.70$ as illustrated in Figure 8.9(b). Each of these connected components must have a width and height that do not differ by more than 40% (Figure 8.9(c)). These values were chosen with the aim to preserve the geometric features of note heads. If the procedure is detecting closed note heads then the algorithm rejects the selected objects with white pixels inside – Figure 8.9(d).

The set of open note heads encompasses the minim notes, which have the symbols with stems (J), and breve and semibreve, which have the symbols without stems (O). This type of music signs are difficult to detect because their head can be degraded. Therefore, when this happen the algorithm first detects black pixels with an height above $2 \times \text{lineHeight}$ and then join these pixels with the closed left objects found. At the end its width must be above $\text{spaceHeight} \times 0.70$. The processing continues with the selection of those that preserve the geometric features of note heads – see Figure 8.10.

Beams detection

The beams are one of the symbol types that can cause problems in the detection process due to their shape and size, and the way they connect each other and to other symbols, which can happen in a huge number of different ways. Another difficulty is related to the inconsistency in their thickness. For each closed notehead previously detected, the algorithm searches for connected components with a width above $\text{spaceHeight} \times 0.70$ and a height below $4 \times \text{spaceHeight}$.

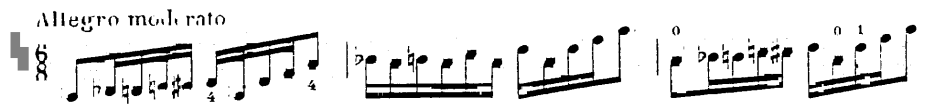
In some situations the stem detection process could remove or disintegrate closed note heads, because of their height. In order to include them, a further procedure in the beams detection module was



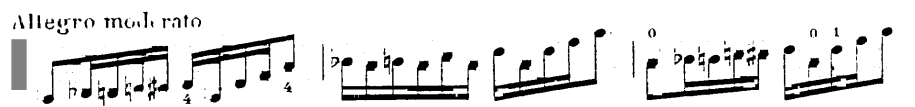
(a) The original score.



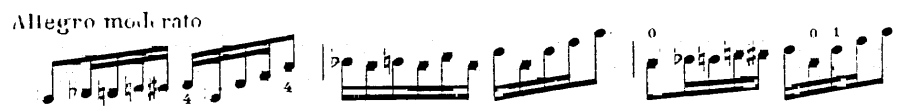
(b) The score with the detected clef.



(c) The score with the detected key signature.



(d) The score with the detected time signature.



(e) The score without the detected previous symbols.

Figure 8.8: The result after key and time signatures steps.

incorporated: to find rejected closed note heads at the beginning and at the ending of the beams selected – see Figure 8.12.

Rests detection

To detect rests symbols, the matrix correlation process already explained for clefs detection was used. Besides this, mathematical morphological procedures were also used. The algorithm has the following steps:

1. Find connected components with a distance between objects of *spaceHeight*.

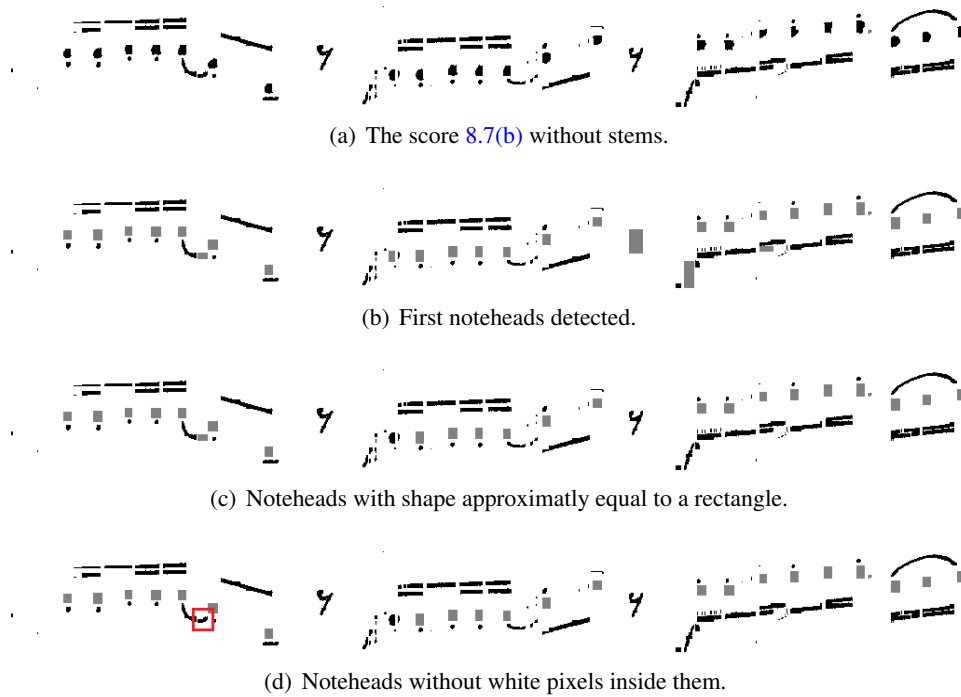


Figure 8.9: Closed noteheads detection steps.

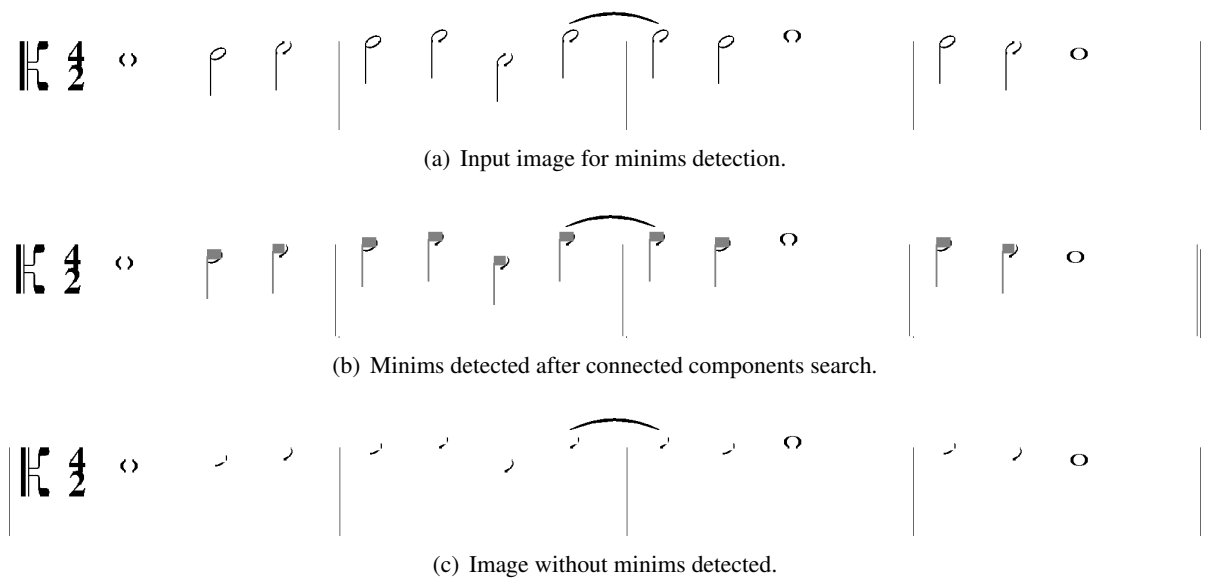


Figure 8.10: Open noteheads with stems detection steps.

2. For the symbols detected in step 1 compute its 10×5 matrix and calculate the correlation degree using the 5 models for rests symbols (quarter $\mathbb{1}$, quaver $\mathbb{7}$, semiquaver $\mathbb{7}$, demisemiquaver $\mathbb{7}$ and hemidemisemiquaver $\mathbb{7}$).
3. While the correlation degree is below 0.3 decrease the object detected width.
4. When the step 3 finds a compatible rest with the symbol detected in step 1 compute the termination points – see Figure 8.13(d) – using morphological operations: closed operation – see

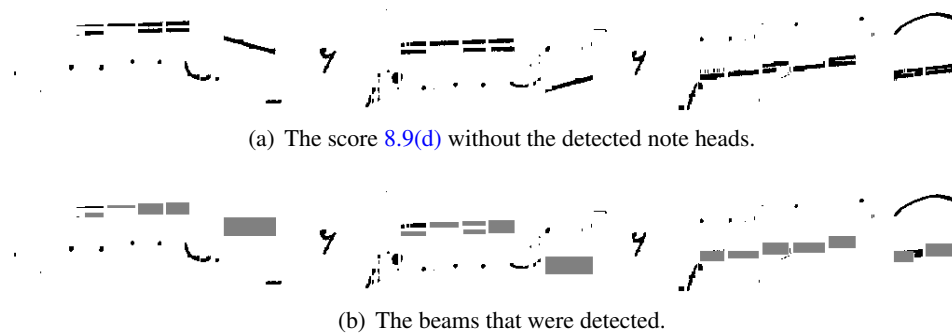


Figure 8.11: The result after the beams detection step.

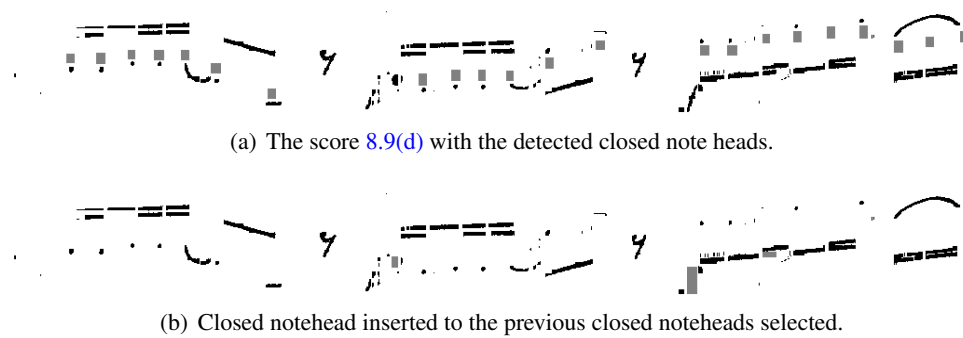


Figure 8.12: Insertion of a closed notehead initially eliminated.

Figure 8.13(b) – and skeleton – see Figure 8.13(c).

5. If the termination points coincide with termination points expected for a rest symbol then the object detected can be consider a rest symbol, otherwise the algorithm goes to another column of the image and starts step 1.

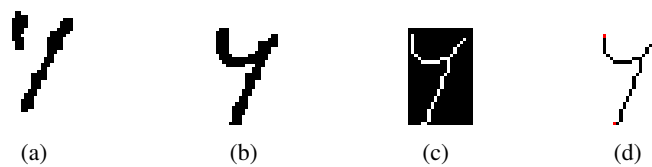


Figure 8.13: Morphological operations applied to the rest symbols: (a) Original rest symbol; (b) Closed rest symbol; (c) Skeleton of the rest symbol; (d) Termination points of the rest symbol.

In Figure 8.14 an example of a possible result after the rest detection step is presented.

Dots, accents, flags and accidentals detection

For these symbols the algorithm uses contextual information and music writing rules. Hence, as already mentioned, flags are symbols that are always placed to the right of the stem. As opposed to these signs, accidentals exist before each notehead and at same height. Dots are always placed to the right of

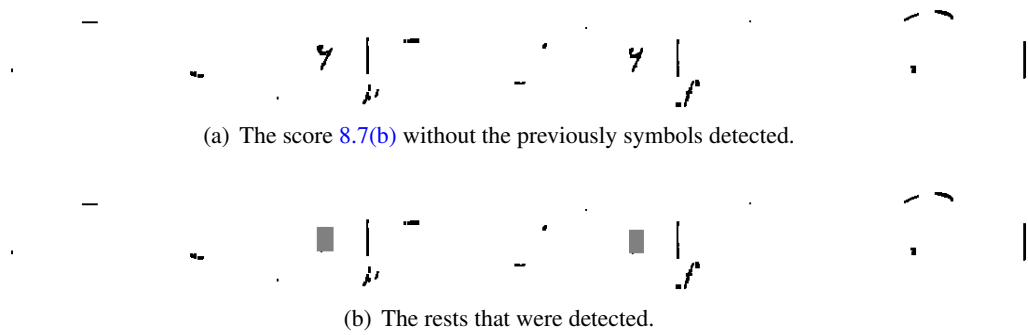


Figure 8.14: The result after the rests detection step.

noteheads and in the center of a space; these symbols can exist associated with noteheads with flags. In this situation they are located in the space above and beyond the tail. If the noteheads are placed on lines the dots are in the center of the space above. If the staff has chords (more than one notehead in the same stem) where the lower notehead is on a line its dots must go in the space beneath it. Accents exist over or under a note head. In this manner, the connect components process is only applied on these areas. Once again the objects are selected using only the size and the position of the bounding box of the detected symbol.

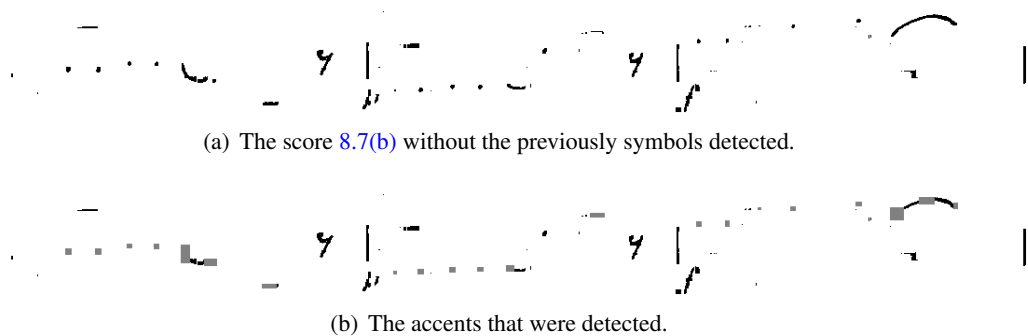


Figure 8.15: The result after the accents detection step.

Process limitations

Since in this proposed methodology the detection of the objects is dependent of the quality of the connected component procedure, connected and overlapping symbols are a problem. Besides, rotations and severe disintegration of the symbols can also cause bad results. To overcome some of these issues we proposed to:

- execute the detection process more than once until there are no more symbol on the staff, adjusting the several input parameters;
- assign recognition hypothesis to each detected symbol in order to help the classification stage;
- incorporate syntactic consistency at the end of the process.

For future work, we propose to use some existent techniques in the state-of-the-art to deal with connected and overlapping symbols:

- Multiple matching of graphical pattern recognition methods (e.g. [5]).
- Tracking contour positions of the objects (e.g. [130]).
- Template matching with recognition hypothesis (e.g. [114]).
- Horizontal and vertical projections in the bounding box of the object.

8.1.1 Confidence Degrees

The music symbols extraction is based on musical rules. Hence, as the algorithm detects the various objects it can assign a possible class for each of them with some certainty. In order to measure this level of certainty favoring the position of the object on the staff we defined Confidence Degrees (CD). These confidence degrees are probabilities of a symbol being correctly detected in its rule. The values for these degrees can be computed empirically through experimental testing for each musical rule. We have 16 musical rules according to the different classes of music symbols. The confidence degrees obtained are displayed in Table 8.1. For each music score and for each musical rule, the number of correct symbols was divided by the total number of the detected symbols. It is important to stress out that each symbol is presented in the sample. In our tests a total of 156 handwritten and synthetic music scores were used. In the end, we will have a matrix where for each detected symbol we have degrees of certainty of belonging to each class.

Accents 0.06	Accidentals 0.60	Barlines 0.95	Beams 0.91	Clefs 0.47	Dots 0.49	DoubleWhole 0.03	Half 0.53
Key 0.40	Noteheads 0.82	Minim 0.54	Semibreve&Breve 0.13	TimeN 0.01	Whole 0.13	Rests 0.19	TimeL 0.01

Table 8.1: Confidence degrees of each class.

The CD values can also show the efficiency of the algorithm in detecting the music symbols. Hence, the implemented rule to detect Barlines, Beams, Noteheads, and Accidentals presents an high value, whereas the process to detect Time, Rests, Semibreve and Breve presents a low value. This indicates that in future some procedures need to be improved.

8.1.2 Musical Symbols Classification

As mentioned on the previous section, some types of musical symbols are already divided in classes during the segmentation phase because of the musical rules. In this manner, this was an aggregation based on the graphical position and not on the symbols shape. For instance, the accidentals signs were all put in the same class. Nonetheless, they should be further divided into 3 sub-classes (sharps, naturals and flats). In order to complement the extraction stage, and to overcome possible errors, a classification phase is needed.

8.1.3 Neural Network Classifier

Although SVMs have obtained good results in our classification tests², with NNs, which are nonlinear statistical models, we can have an output with a probabilistic interpretation. Such methodology will aid the automatic analysis of syntactic recognition³ of the music sheets by providing statistical reasonings if a given pattern is a symbol or not. For this study, a Multi-Layer Perception (MLP) classifier with one hidden layer with a sigmoid activation function was used. The input of the network was the symbol bounding box resized to 20×20 pixels which was afterwards converted to a vector of binary values.

Our approach encompassed the construction of 7 classification models, one for 20 main classes and one for each of the 6 secondary classes⁴. The database consist on a set of 65 binary handwritten scores and 380 distorted synthetic scores. To obtain all the NNs the available dataset was randomly split into training and test sets, with 60% and 40% of the data, respectively. This division was repeated 20 times. No special constraint was imposed on the distribution of the categories of symbols over the training and test sets, ensuring only that at least one example of each category was present in the training set. The best parametrization of each model was found using the training and validation sets being the expected error estimated on the test set by a 4-cross validation scheme.

The results for the different models can be seen in Table 8.2. The expected performance in percentage for the group of the main classes with a 99% confidence interval was [87.6; 88.4].

	Subclasses					
	Accents	NotesFlag	Relation	Rests2	TimeSignaturesL	TimeSignaturesN
99% CI for the Expected performance in percentage:	[88; 90]	[74; 80]	[93; 96]	[92; 94]	[98; 99]	[63; 66]

Table 8.2: Accuracy obtained for the classification models for the secondary classes.

8.2 Music Symbols Segmentation through Recognition

In the previous section, the music symbol recognition procedure was composed by two main steps: first the image was segmented in order to detect and isolate the primitive elements, and then the symbols were classified. In this section a new method is presented. The idea was to perform segmentation through recognition, that is, the method simultaneously segment and recognize the image. The principal advantages regarding to the previous method are in the elimination of the heuristic approach and in the way of dealing with connected and overlapping objects. As symbols are first detected and extracted from the image and, after that, classified, various parameters related to the size, shape and position of the objects are introduced. These parameters can constitute a severe problem in handwritten music scores, because the variability in writing style of each composer. Segmenting the music sheet using classification simplifies all the process and also overcomes the issues inherent in sequential detection of the objects, becoming less prone to errors.

²See Appendix B.2 for more details.

³Please see the full explanation in section 8.3.

⁴See Appendix B Table B.1 for more details.

8.2.1 Recognition Process

The recognition process consists first in splitting by staves the music sheet without staff lines, and then analyzing each of these segments. For each staff the connected component technique is applied. Once again this technique has a threshold in order to union neighboring pixels from broken objects. The algorithm proceed doing a scanning to each connected component in order to select what is object and what is not. This analysis is carried out using classifiers. The objects detected can be noise or musical symbol (see Figure 8.16).

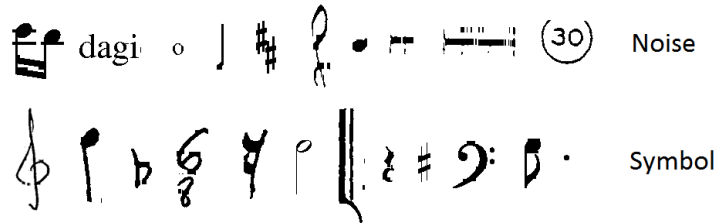


Figure 8.16: Examples of objects that are considered noise and musical symbol.

The framework to execute this scanning through each connected component is presented in Figure 8.17. This is a process based on a hierarchical classification. First the detected objects are split into *noise* and *symbol*, and then the *noise* objects are divided into four types (see Figure 8.18): (1) *connected symbol*, (2) *not symbol*, (3) *split symbol* and (4) *connected and split symbol*.

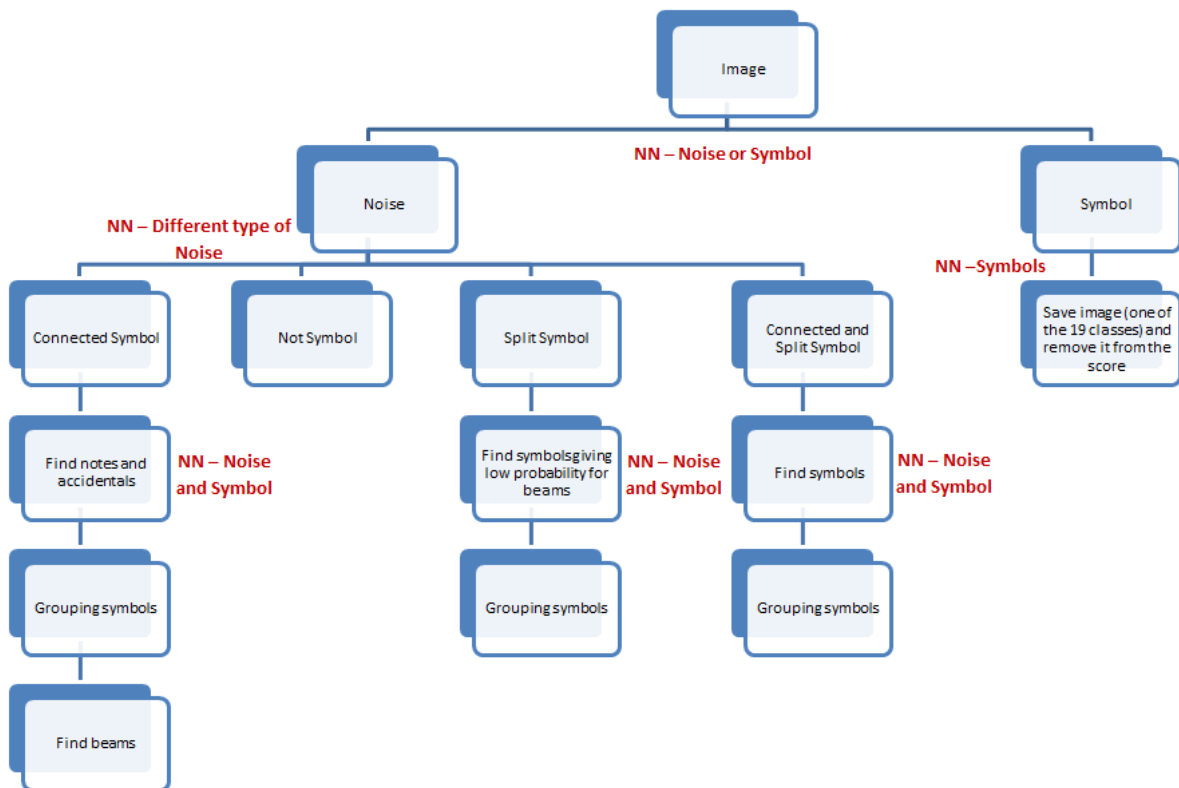


Figure 8.17: Diagram of the algorithm for music symbols extraction.

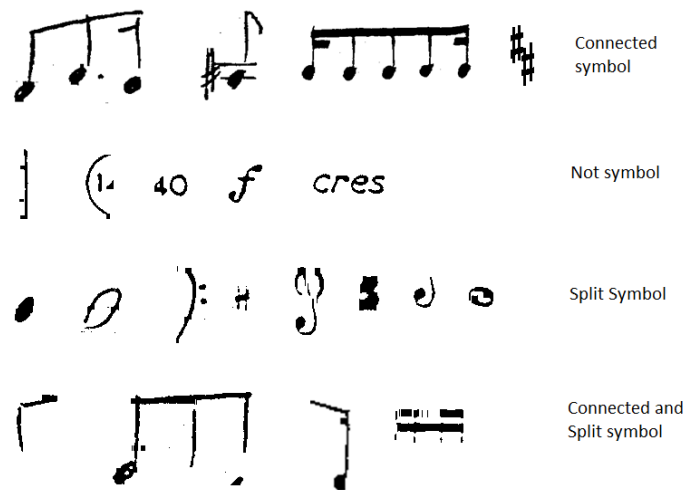


Figure 8.18: Examples of objects that are considered noise.

The objects that are classified as *symbol* can be one of the 19 possible symbols presented in Table 8.3 and the objects that are classified as *Noise and Symbol* can be one of the 15 possible symbols presented in Table 8.4. The reduction in the number of classes is related to the objects that are possible to find in each case. Note that in the *Noise and Symbol* step we are trying to split objects, usually notes connected to beams, so in this case the noise class is necessary.

>	9	≡	b	4	┐	♩	┐
Accent	BassClef	Beam	Flat	Natural	Note	NoteFlag	NoteOpen
z	7	#	♩	♩	2/4	♩	
RestI	RestII	Sharp	TrebleClef	AltoClef	TimeN	TimeL	Barlines
.	[o]	O					
Dot	Breve	Semibreve					

Table 8.3: The set of the musical symbols considered in the *symbol* class.

>	9	≡	b		┐	♩	┐
Accent	Clef	Beam	Accidentals	Barlines	Note	NoteFlag	NoteOpen
z	7	O		2/4	♩	♩	
RestI	RestII	Semibreve	Barlines	TimeN	TimeL	Noise	

Table 8.4: The set of the musical symbols considered in the *Noise and Symbol* class.

As mentioned in the beginning of this section, the analysis executed during the procedure to select

the musical symbols relies on classifiers. We built four types of classifiers relating them to each possible situation that can appear in the connected components. The CNN_1 classifier⁵ is performed in order to divide the objects detected as *noise* or *symbol*. If the objects detected are *symbol* then the CNN_3 is used, otherwise the CNN_2 is utilized. For *connected symbol*, *split symbol* and *connected and split symbol* an analysis to each bounding box of the object is carried out. The construction of these classifiers will be explained later.

Connected symbol

Connected symbol class encompasses notes connected to beams, notes connected to accidentals and notes connected to accents. In the first situation accidentals and accents can also appear in the bounding box, as illustrated in Figure 8.19. We proposed a sliding window procedure supported by the CNN_4 to detect and extract the symbols. First, the analysis window with an height equal to the height of the bounding box is moved along the columns. The window width starts equal to `staffspaceheight` and is increased three times. This value was obtained experimentally. Only the notes class is considered on this step – see Figure 8.19(a). After this the search of the window is changed to go by rows – see Figure 8.19(b). The aim is to look for accidentals and accents. Again the CNN_4 classifier is used to detect and extract the symbols. The procedure to establish the window size is the same of the previous step.

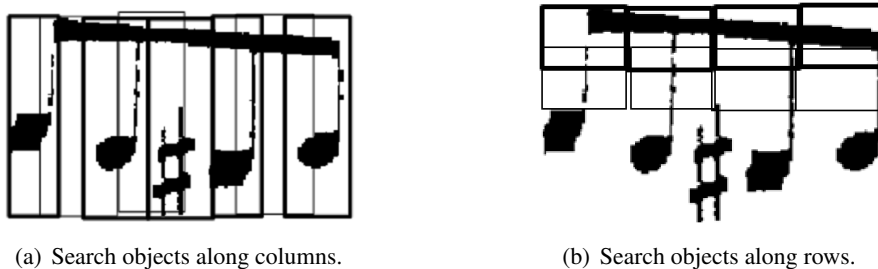


Figure 8.19: Search objects in connected symbols.

Since an overlap exists between windows, there are repeated objects that need to be removed. Hence, a process to group symbols is executed. The symbols from the same class are compared with each other; if their positions are close enough, they are saved as one symbol. All the symbols detected are removed from the image.

Now it is necessary to detect beams which are music symbols linking two notes. Towards this goal, for each image composed by two adjacent notes, the algorithm looks for black pixels. It is worth restating that notes were already removed from the image.

Split symbol

Split symbol class encompasses broken objects – see Figure 8.20. Usually they are notes separated from their stems or fragmented accidentals and clefs.

The goal is to join black pixels near to the initial object. For that the window size increases until a certain limit and the CNN_4 recognizer is used to see when we are in presence of a music symbol.

⁵The Combined Neural Network (CNN) will be explained on the next section.



Figure 8.20: Example of broken notes.

The augmentation of the window is first done in height, then in width and then in both. At the end the procedure to look for repeated symbols is again computed.

Connected and split symbol

Connected and split symbol class encompasses the two previous groups of symbols. For that reason, the techniques already described for each of the classes are applied here.

After the detection of all symbols a process to test some musical rules is executed. In here, the presence of accents only above notes and the position of accidentals before and at same height of notes is verified. If the symbols do not respect these rules then they are eliminated.

8.2.2 Combined Neural Network

To perform the various necessary classifications during the scanning procedure, as described in previous sections, we propose a music symbol recognizer based on a majority vote combination of three Multi-Layer Perception (MLP) classifiers – see figure 8.21 – named Combined Neural Networks (CNN). Two of the networks have the same architecture, being initialized with different weights. The third network is fed with a different input and with a different number of neurons in the hidden layer. In this way we expect to increase the overall performance of the classifier regarding to the usual way of only one MLP.

For two of the networks each image of a symbol was initially resized to 20×20 pixels and then converted to a vector of 400 binary values. For the third network each image of a symbol was initially resized to 60×20 pixels and then converted to a vector of 1200 binary values. Usually the images have an height larger than their width and the principle was thus to favor the height. In this manner, the problem in the classification of barlines, due to its similarity with dots after the resize, is minimized. Once again, all these networks have one hidden layer with a sigmoid transfer function.

A database of training patterns was created according to the possible objects that algorithm could find in the scanning process. Also because of that, for each CNN we have a different database. Each one of these databases was randomly split into training and test sets, with 60% and 40% of the data, respectively. This division was repeated 10 times without restricting the distribution of the categories of symbols over the training and test sets. Only two constraints were imposed: at least one example of each category should be presented in the training set and the size of the noise class should be limited to avoid unbalanced classes distributions. The best parametrization of each model was found using the

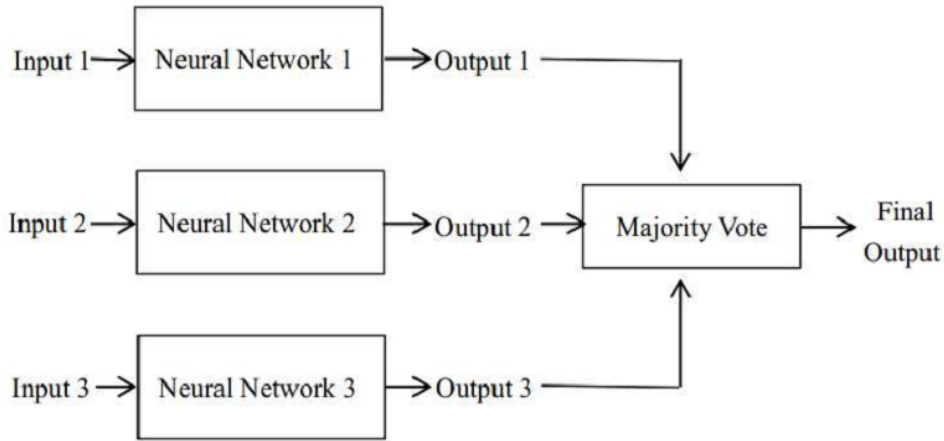


Figure 8.21: The struture of a Combined Neural Network.

training and validation sets, with the expected error estimated on the test set by a 4-cross validation scheme. The results for the different models can be seen in Table 8.5.

	Noise/Symbol	Noise	Symbols	Noise&Symbol
MLP	[91; 92]	[82; 84]	[88; 89]	[81; 84]
CNN	CNN_1: [95; 96]	CNN_2: [90; 91]	CNN_3: [95; 96]	CNN_4: [88; 89]

Table 8.5: Accuracy obtained on the 99% CI (in percentage) for the classification models.

Comparing the results with one MLP applied to the same dataset of music scores and also divided in the same way, we obtained an higher accuracy. This improvement on the performance is related to the increase (by a different order of magnitude of the input vector) of the information provided to the algorithm. Moreover, the combination of the knowledge gained independently by the three learners which is afterwards merged in a single symbol prediction is also another reason for the performance gain.

8.3 Syntactic Consistency

During the detection and classification of the music symbols errors always occur: missed, wrong clas-sified and falsely detected symbols. The purpose of the introduction of syntactic and semantic musical rules before the construction of the final musical notation model is to overcome these possible errors. In this thesis, the last two problems were treated.

The main idea behind the syntactic consistency procedure is related to the fact that in music sheet the number of symbols contained within two bar lines and the time signature must match – see Fig. 8.22. The procedure of checking the coherency is entirely related to the *measure*. The top number in the time signature indicates the number of beats to be counted in each measure/bar line, while the bottom number indicates which type of note value equals one beat. For instance, the number of beats or the number of symbols between barlines in the Fig. 8.22 is 2 and the symbol that indicates the unit is the minim symbol (♩), because the bottom number is 2, giving $\chi \times 2 = 1 \Leftrightarrow \chi = 1/2$, where χ represents the symbol type. Every measure of music in a simple time signature has the same number of beats per

measure throughout the song. Hence, symbols confusion and wrong symbols added can be mitigated by querying if the detected symbols' durations match with the amount of the value of the time signature on each bar.

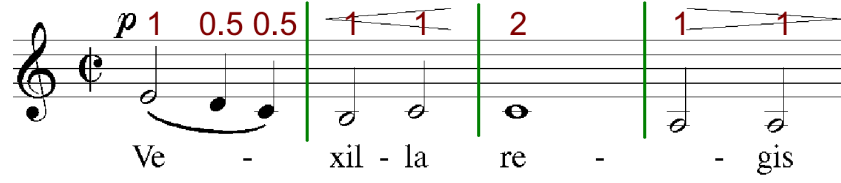


Figure 8.22: Example of measures according to the time signature. The bar lines are represented by the green lines.

8.3.1 Global Constraints

We propose to detect the best combination of symbols between two bar lines given the indicated measure in the music sheet as an optimization problem. In this manner, the syntactic and semantic music rules can be incorporated as global constraints, considering the following:

$$\begin{aligned}
 \max \quad & \sum_{i=1}^n \sum_{j=1}^k p_{ij} x_{ij} \\
 s.t \quad & \sum_{j=1}^k x_{ij} \leq 1, i = 1, \dots, n \\
 & D \sum_{j=1}^k \sum_{i=1}^n \alpha_j x_{ij} = N \\
 & \sum_{i=1}^n x_{i2} \leq \sum_{i=1}^n \sum_{j=3}^M x_{ij} \\
 & x_{i,1} \leq \sum_{j=3}^M x_{i+1,j}, i = 1, \dots, n-1 \\
 & x_{ij} \in \{0, 1\}
 \end{aligned} \tag{8.1}$$

where p_{ij} is the likelihood (matrix of probabilities) given by a classifier, of the symbol i to belong to the class j , x_{ij} represents the symbol i from class j , N and D are the numerator and denominator (as aforementioned) in the time signature, respectively, n is the total number of symbols, k is the total number of classes, M is the total number of symbols that could have associated accidentals and accents, and α_j is the music note value that represents how long each note lasts. Since notes come in different levels, each with its own note value, it is possible to associate a note value to each class j . The first constraint of the optimization problem allows symbols elimination, the second constraint is related to the time signature, the third and fourth constraints are related to the position of the symbols in the staff.

Accents are placed above beams, below and above noteheads. Accidentals exist before each notehead and at same height (placed on the same staff line or space). The third constraint incorporates the rule of accents position on the optimization problem, allowing accents only if notes are presented in the score. The fourth constraint imposes precedence of the accidentals for the notes symbols.

To better understand how α_j works in the second constraint see Fig. 8.23 which illustrates most of the notes that we can find in music arranged. For instance, the value of a half note (♩) is half of a whole note (♩), the value of a quarter note (♩) is quarter of a whole note (♩), and so on. In Fig. 8.22 the note value of ♩ is $1/2$. The rests have the same values as the notes. Occasionally, in a music sheet we can have dots placing right to the notes and rests, increasing the original value: n dots lengthen the note's

or rest's original d duration to $d \times (2 - 2^{-n})$. These α_j values will be used in the counting of the time between the barlines. The note value of accents and accidentals is 0, because they do not interfere in the meter. Depending on the time signature, the number of beats per note varies. Nevertheless, the note lengths do not change in their relationships to each other: one beat of music could indicate the length of a whole note, but two quarter notes will always be twice as fast as a half note.

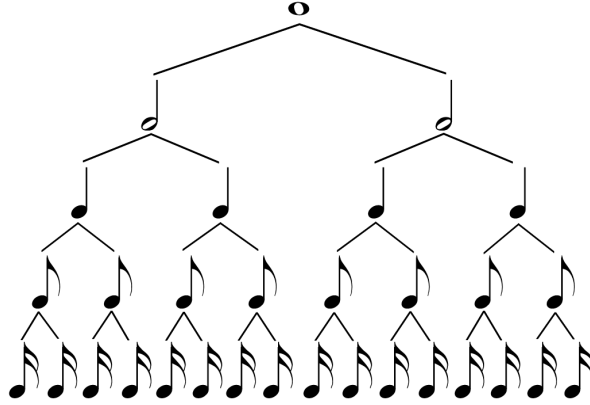


Figure 8.23: Tree of notes representing the relation between them. At the top is the whole note, below that half notes, then quarter notes, eighth notes, and finally sixteenth notes.

The matrix of probabilities given by p_{ij} is the result of one of the classifiers described in the previous sections 8.1.3 and 8.2.2. If the process encompasses confidence degrees then the p_{ij} is the multiplication of a matrix of probabilities with the matrix of the confidence degrees – see Figure 8.24. In this case, we try to minimize the probabilities of the falsely detected symbols leading the process to remove them.

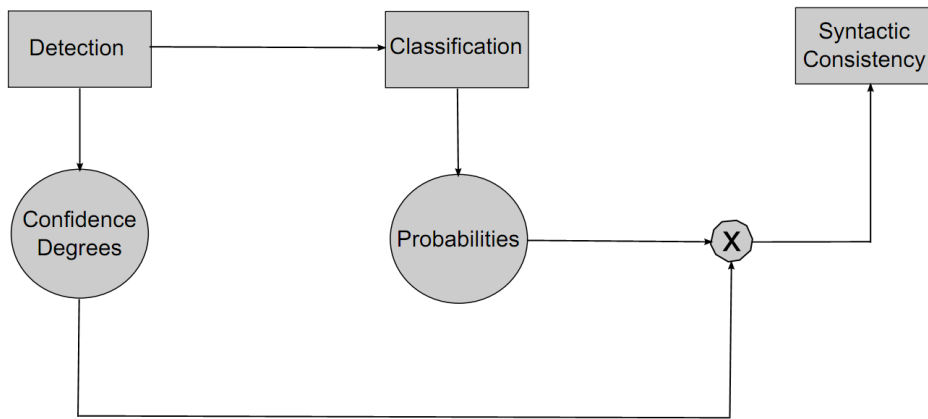


Figure 8.24: The architecture of the syntactic consistency process.

8.4 Evaluation Metrics and Results

In a first experimental testing we compute the performance of our syntactic consistency model using the reference position of the music symbols on the music score. This reference position was obtained manually and it is composed by the coordinates of the bounding box of the object plus its class. The

aim was to verify if the optimization process improves the classification results, making a good re-label of the objects without the segmentation interference. The data set adopted consisted of 6 handwritten scores from 3 different composers, 9 synthetic scores and 9 scanned printed scores, all written on the standard notation. The scanned and real scores were binarized using Otsu's method [92], while the synthetic scores were already in binary format. In total we work with 1713 handwritten music symbols, 6486 printed music symbols and 4267 synthetic music symbols. Table 8.6 presents the obtained results where TPB and TPA stands for true positive classification before and after the execution of the proposed syntactic consistency process, respectively.

	M LP + 400	
	TPB	TPA
Synthetic scores	83%	82%
Real scores	74%	73%
Scanned scores	78%	79%

Table 8.6: Accuracy obtained for the classification models using 400 features before and after the syntactic consistency model.

Only with scanned scores we could improve the overall performance. It is clear the need of adjustment of the parameters of the global optimization problem. Nevertheless, the lost on the final accuracy was not significant to stop the work on the global constraints methodology. Changing the constraints, making possible to include new symbols or weigh the re-label of the symbols could improve the performance.

Despite in the first results the advantage of using syntactic consistency was not clear, we also computed the overall performance for the entire extraction process. The data set adopted to test the proposed architectures for the music symbols extraction consists of both handwritten and synthetic scores, as presented in Table 3.2 (section 3.1). To test only the detection step of the algorithm we have 9 scanned printed scores, 26 handwritten scores and 882 images generated from 18 perfect scores, while to test the classification step we only have 9 scanned printed scores, 6 handwritten scores and 132 images generated from 12 perfect scores. This decrease in music scores is related to the number of manual references of the musical symbols positions with their classes.

The metrics accuracy rate, average precision and recall were considered. They are given by

$$accuracy = \frac{\#tp + \#tn}{\#tp + \#fp + \#fn + \#tn}, \quad precision = \frac{\#tp}{\#tp + \#fp}, \quad recall = \frac{\#tp}{\#tp + \#fn}$$

The true positive rate (TPR), false positive rate (FPR), true negative rate (TNR) and false negative rate (FNR) were also considered:

$$TPR = \frac{\#tp}{\#tp + \#fn}, \quad FPR = \frac{\#fp}{\#tn + \#fp}, \quad TNR = \frac{\#tn}{\#tn + \#fp}, \quad FNR = \frac{\#fn}{\#fn + \#tp}$$

as well as the classification rate given by:

$$accuracy_class = tpc/tp$$

where tp are the true positives, tn are the true negatives, fn are the false negatives, fp are the false positives and tpc are the classes of the true positives. A false negative happens when the algorithm

identifies a musical symbol as noise; and a false positive is when the algorithm identifies noise as a music symbol. These percentages are computed using the symbols position reference and the symbols position obtained by the segmentation algorithm.

The performance of the procedures is presented in the following Tables. Table 8.7 shows the impact of the proposed technique to extract the music symbols with segmentation for all kind of music scores, while Table 8.9 presents the same study but for the algorithm where the music symbols extraction was based in the recognition of the objects. In Tables 8.8, 8.10, 8.12, 8.14, 8.16, and 8.18 the confusion matrices are displayed in order to analyse with more detail where is the gain of the method. Tables 8.11, 8.13 and 8.15 indicate the results using syntactic consistency. In Table 8.11 the algorithm was computed without confidence degrees (CD) as opposed to Table 8.13.

	Precision	Recall	Accuracy	Final classification accuracy without CD	Final classification accuracy with CD
Handwritten scores	61.0%	91.6%	97.6%	31.4%	48.2%
Printed scores	53.9%	93.0%	79.4%	29.8%	46.0%
Digitized scores	67.9%	93.8%	97.8%	29.2%	43.8%

Table 8.7: Results for music symbols extraction.

Handwritten Scores			Scanned Scores			Printed Scores		
	True	False		True	False		True	False
Positive	91.7%	2.1%	Positive	93.8%	2.7%	Positive	93.0%	18.9%
Negative	99.9%	20.5%	Negative	99.9%	30.5%	Negative	94.1%	9.0%

Table 8.8: Confusion Matrix for the results from the Table 8.7.

	Precision	Recall	Accuracy	Final classification accuracy
Handwritten scores	67.1%	78.8%	97.2%	48.4%
Printed scores	69.1%	79.9%	82.2%	45.8%
Digitized scores	66.1%	87.3%	96.7%	45.4%

Table 8.9: Results for music symbols extraction through recognition.

Handwritten Scores			Scanned Scores			Printed Scores		
	True	False		True	False		True	False
Positive	78.8%	2.0%	Positive	87.3%	2.8%	Positive	79.9%	7.6%
Negative	99.9%	41.8%	Negative	99.9%	44.4%	Negative	90.5%	33.1%

Table 8.10: Confusion Matrix for the results from the Table 8.9.

	Precision	Recall	Accuracy	Classification Accuracy
Handwritten scores	71.1%	84.5%	98.3%	40.9%
Printed scores	71.3%	84.7%	84.8%	43.6%
Digitized scores	69.5%	92.0%	97.0%	32.2%

Table 8.11: Results for music symbols extraction after syntactic consistency without confidence degrees.

Handwritten Scores			Scanned Scores			Printed Scores		
	True	False		True	False		True	False
Positive	84.5%	1.3%	Positive	92.0%	2.6%	Positive	84.8%	14.0%
Negative	99.9%	31.3%	Negative	99.9%	25.7%	Negative	98.3%	30.3%

Table 8.12: Confusion matrix for the results from the Table 8.11.

	Precision	Recall	Accuracy	Classification Accuracy
Handwritten scores	72.0%	85.2%	98.3%	42.7%
Printed scores	73.3%	93.5%	89.7%	48.3%
Digitized scores	69.0%	92.5%	96.9%	32.0%

Table 8.13: Results for music symbols extraction after syntactic consistency with confidence degrees.

Handwritten Scores			Scanned Scores			Printed Scores		
	True	False		True	False		True	False
Positive	85.2%	1.2%	Positive	92.5%	2.7%	Positive	93.5%	12.2%
Negative	99.9%	29.7%	Negative	99.9%	23.3%	Negative	98.5%	20.3%

Table 8.14: Confusion matrix for the results from the Table 8.13.

	Precision	Recall	Accuracy	Classification Accuracy
Handwritten scores	91.7%	30.7%	97.2%	28.4%
Printed scores	84.7%	51.5%	80.9%	38.8%
Digitized scores	94.0%	45.1%	96.1%	32.7%

Table 8.15: Results for music symbols extraction through recognition after syntactic consistency.

Handwritten Scores			Scanned Scores			Printed Scores		
	True	False		True	False		True	False
Positive	30.7%	0.2%	Positive	45.1%	0.5%	Positive	51.5%	1.9%
Negative	99.9%	83.1%	Negative	99.9%	84.6%	Negative	90.8%	63.1%

Table 8.16: Confusion matrix for the results from the Table 8.15.

	Precision	Recall	Accuracy	Classification Accuracy
Handwritten scores	63.7%	90.8%	98.0%	38.1%
Printed scores	90.9%	90.7%	74.1%	49.1%
Digitized scores	68.9%	94.4%	97.2%	31.7%

Table 8.17: Results for music symbols detection and classification after syntactic consistency with confidence degrees, knowing the true positions of barlines and the real time signature.

Handwritten Scores			Scanned Scores			Printed Scores		
	True	False		True	False		True	False
Positive	90.8%	1.8%	Positive	94.4%	2.6%	Positive	90.9%	9.1%
Negative	99.9%	20.7%	Negative	99.9%	24.5%	Negative	99.9%	29.2%

Table 8.18: Confusion matrix for the results from the Table 8.17.

8.5 Discussion

Looking to the various results, we can concluded that the algorithm to extract symbols through recognition detects more false negatives and less true positives than just using the algorithm for symbol extraction. This means that symbol extraction shows a reduction on the correct prediction of the symbols for the three datasets (Table 8.8 and Table 8.10). This rationale is clearly depicted on the results shown in Table 8.9 where the recall substantially decreased over the recall results shown in Table 8.7. Hence, the first process has more missed symbols, even though it detects less noise (prediction rate). Furthermore, comparing the Tables 8.7 and 8.9 the symbols extraction through recognition improves the performance in printed scores (82.2%).

It is possible to note an increase in the classification performance when the algorithm uses the confidence degrees (CD) – see Table 8.7. This shows that the prior knowledge implicitly introduced with CD lead to a clear improvement over the performance. The low results in classification for all methods could be explained by the bad quality of the extracted symbols, for instance broken objects. As the syntactic consistency procedure is dependent of classification results, this could lead to a problem.

A decrease in noise is obtained when syntactic consistency is applied to the procedures. See for example Tables 8.10 and 8.16 where the percentages of false negatives in handwritten scores decreases from 2% to 0.2%. Nevertheless, we also have a reduction in the number of true positives. This could indicate that the algorithm is discarding more symbols than it should. Moreover, in some cases with syntactic consistency the number of missed symbols growth a lot, showing the need of being very careful with the optimization method and its parameters. More research will be carried in the future to assess this.

A balance between precision and recall rates is achieved when syntactic consistency is executed. For example in Table 8.7 we have 61% of precision and 92% of recall for handwritten scores and in Table 8.13 we have 72% of precision and 85% of recall, showing a slight decrease in the number of detected symbols but also a decrease in noise. This shows that syntactic consistency procedure is performing well in the elimination of noisy objects. Moreover, by analyzing the results obtained for music symbols detection and classification after syntactic consistency with confidence degrees, knowing the true positions of barlines and the real time signature we can conclude a trend in augmenting

the number of false negatives after the use of syntactic consistency – see Tables 8.13 and 8.17. With these results we can conclude that this method could be applied to remove unknown objects, leaving to other procedure to recover or to detect the missed musical symbols. Besides, it is important to stress that the syntactic consistency algorithm always expects that the previous process detects all music symbols present in the score. In this way, the changes that occurs in the classification or the elimination of the symbols should be seen taking this constraint into account. In future work, it will be very interesting to include in the optimization algorithm the possibility of adding new symbols.

Part IV

Conclusions and Future Work

Conclusion

The work developed in this thesis had the target to overcome the several issues that affect musical symbol recognition, particularly in handwritten scores. Despite advances made in the optical musical recognition algorithms, as described in the state of the art, several open problems still exist with handwritten music sheets. On the one hand, the scores tend to be rather irregular and conditioned by the authors' own writing style and the quality of the paper in which it is written might have degraded throughout the years, making it a lot harder to correctly identify its contents. On the other hand, they are likely to have changes in size, shape and intensity of handwritten symbols by the same author into the same score and the staff lines may be tilted one way or another on the same page. They also may be curved and may have discontinuities. As a result, the detection and recognition process in the handwritten music scores becomes more complicated than in printed music sheets.

An OMR system typical starts with a preprocessing phase. The line width and interval of the staff is usually estimated to work as reference lengths for the subsequent operations. A robust method to reliably estimate the thickness of the lines and the interline distance was presented. It was assumed that the initial image is converted, column by column, in the run-length coding. Next, the estimation of the sum of the two lengths was introduced as a more reliable estimation than the independent estimation of the two lengths. The individual lengths were then estimated as the most likely combination of runs summing to the pre-estimated total. To overcome the difficulties of binarization algorithms with low quality music scores, it was proposed to integrate the estimation over every possible binarization threshold. The basic idea of estimating first the sum of quantities of interest, and then estimating the individual quantities of interest with the constraint that they sum to the estimated value, may apply in other areas of document image analysis.

Image binarization is also a common operation in the preprocessing stage in most OMR systems. The choice of an appropriate binarization method for handwritten music scores is a difficult problem. Several works have already evaluated the performance of existing binarization processes in diverse applications. However, no goal-directed studies for music sheets documents were carried out. This thesis presents a novel binarization method based in the content knowledge of the image. The method only needed the estimation of the staff line thickness and the vertical distance between two staff lines. This information is extracted directly from the gray level music score.

Another challenge faced by an OMR system was staff lines detection and removal. A robust algorithm for the automatic detection of staff lines in music scores based in stable paths was presented. The proposed method uses a very simple but fundamental principle to assist detection and avoids the difficulties typically asserted by symbols superimposed on staff lines. The main idea was to consider the staff lines as the result of the shortest path between the two margins of the music sheet, giving preference to black pixels. This approach for the staff line detection algorithm was adapted to a wide range of image conditions, thanks to its intrinsic robustness to skewed images, discontinuities, and curved staff lines. The proposed method is also robust to discontinuities in staff lines (due to low-quality digitization or low-quality originals) or staff lines as thin as one pixel. In order to take full advantage of the

method, existing staff line removal algorithms were enhanced by using the stable paths method as its first processing step. Several tests that enabled improvements in this proposed approach that induced better results were also performed.

The next operations were segmentation and classification of the music symbols. Because the project work with handwritten musical scores it was necessary to deal with several issues. Not only, the huge variability in the symbols caused inconsistency problems in the size and shape of each object, but also the presence of the complexity and ambiguity problems related to the number of possible arrangements of music primitives.

In this thesis two processes to detect the music symbols are presented and discussed. The first music symbols segmentation's method is essentially based in contextual information and music rules. The score is first split by staves and the symbols are divided considering their graphical position and geometric features: the symbols that are featured by a vertical segment, the symbols that link the notes, the remaining symbols connected to staff lines and the symbols above and under staff lines. The second method performs the extraction of music symbols without segmentation. The idea is to execute simultaneously the segmentation and recognition of the objects avoiding the issues inherent in tracking objects.

In the music symbol classification a comparative study with the most common classifiers that can be adopted for the OMR of handwritten scores was carried out. The performances of these methods were compared using both real and synthetic scores. We examined five classification methods, namely Support Vector Machines, Relevance Vector Machines, Neural Networks, Nearest Neighbour and Hidden Markov Models. In the first tests, the SVMs, RVMS, NNs and kNN received raw pixels as input features (a 400 feature vector, resulting from a 20x20 pixel image); the HMM received higher level features, like information about the connects components in a 150x30 pixel window. In the second tests, the classifiers receive raw pixels and 7 extracted characteristics as input features. All of these options tried to reflect standard practices in the literature. The performance of any classifier depends crucially on the choice of features. Therefore, results must be interpreted in light of these design options. In a global overview of the results, the performance of the SVM classifier was the best. The simple kNN also achieved a very competitive performance, better than the NNs and the HMMs. The less satisfying performance of the HMMs deserves additional exploration in the future: the number of states and the distribution assumed for the observed variable are design option that may be hampering the performance of the model.

Concerning the use of elastic deformations to increase the training set, it was interesting to observe that the performance did not improve. Our aim was to increase the size of the training set, creating controlled distorted symbols to prepare the classifier for the typical variations of symbols in handwritten music sheets. We expected that the classifiers designed with this extended data would be more robust and with improved performance. The results, in opposition to our initial thoughts, may have multiple explanations, which require further investigation: the distortions created were not the most suited for the recognition task, the initial dataset was already quite diverse in terms of symbol variety, or the raw representation adopted for features is not the most appropriate for introducing this kind of variation. The fact that the handwritten scores were authored by only five different authors may also help explaining the results: it is possible that the writing was not so diverse to benefit from the design with elastic deformation.

A distance metric learning is also successfully applied to k -NN classifier to recognize music sym-

bols. The results achieved in our experiments showed an improvement in comparison with k -NN with the simple Euclidean distance. We have also apply this method to derive a RBF SVM kernel (dRBF) which provided significant improvements. Recent works have focused in the application of metric learning in more advanced classifiers than k -NN, which is the classical and the simplest method for pattern recognition. Nguyen and Guo [89] proposed a metric learning support vector machine (MLSVM) method, where the problem of metric learning is formulated as a quadratic SDP problem for local neighbors constraints. Notwithstanding, k -NN is still the most basic application for metric learning, because we can easily demonstrate that with a proper distance metric, the process can improve the accuracy. In the future, other metric learning algorithms¹ can also be adapted, for instance Probabilistic Global Distance Metric Learning (PGDM) or Active Distance Metric Learning, to other classification methods.

In this thesis, we also propose to incorporate syntactic and semantic music rules as an optimization problem. The idea is to detect the best combination of symbols in order to give the indicated measure. The inclusion of prior knowledge in the OMR recognition process could lead to better results.

9.1 Future Trends

Over the last decades, substantial research was done in the development of systems that are able to optically recognize and understand musical scores. An overview through the number of articles produced during the past 40 years in the field of Optical Music Recognition, makes us aware of the clear increase in research in this area. The progress of in the field spans many areas of computer science: from image processing to graphic representation; from pattern recognition to knowledge representation; from probabilistic encodings to error detection and correction. OMR is thus an important and complex field where knowledge from several fields intersects.

An effective and robust OMR system for printed and handwritten music scores can provide several advantages to the scientific community: (1) an automated and time-saving input method to transform paper-based music scores into a machine-readable symbolic format for several music softwares, (2) enable translations, for instance to Braille notations, (3) better access to music, (4) new functionalities and capabilities with interactive multimedia technologies, for instance association of scores and video excerpts, (5) playback, musical analysis, reprinting, editing, and digital archiving, and (6) preservation of cultural heritage [66].

This thesis unveiled four challenges that should be addressed in future work on OMR as applied to manuscript scores: (1) preprocessing, (2) staff detection and removal, (3) music symbols segmentation and (4) recognition and final representation construction and comparison of results.

Preprocessing. This is one of the first steps in an OMR system. Hence, it is potentially responsible for generating errors that can propagate to the next steps on the system. Several binarization methods often produce breaks in staff connections, making the detection of staff lines harder. These methods also increase the quantity of noise significantly (see Fig. 5.2 in Section 2.1). Back-to-front interference, poor paper quality or non-uniform lighting causes these problems. A solution was proposed in [95] (BLIST). Performing a binarization process using prior knowledge about the content of the document can ensure better results, because this kind of procedure conserves the information that is important to

¹<http://www.cs.cmu.edu/~liuy/distlearn.htm>

OMR. Moreover, the work proposed in [23] encourages the further research in using gray-level images rather than using binary images. Similarly, new possibilities exist for music score segmentation by exploiting the differences in the intensity of grey pixels of the ink and the intensity of grey pixels of the paper.

Staff detection and removal. Some of the state-of-the-art algorithms are capable of performing staff line detection and removal with a good degree of success. Cardoso et al. [22] present a technique to overcome the existing problems in the staff lines of the music scores, by suggesting a graph-theoretic framework (see end of Section 2.2). The promising results promote the utilization and development of this technique for staff line detection and removal in gray-level images. In order to test the various methodologies in this step the author suggests to the researchers the participation in the staff line removal competition which in 2011 was promoted by International Conference on Document Analysis and Recognition (ICDAR)².

Music symbols segmentation and recognition. A musical document has a bidimensional structure in which staff lines are superimposed with several combined symbols organized around the noteheads. This imposes a high level of complexity in the music symbols segmentation which becomes even more challenging in handwritten music scores due to the wider variability of the objects. For printed music documents, a good methodology was proposed in [114]. The algorithm architecture consists in detecting the isolated objects and computing recognition hypotheses for each of the symbols. A final decision about the object is taken based on contextual information and music writing rules. This propitious technique was implemented in Rebelo et al. [108] and also in this thesis. The proposed procedure uses the natural existing dependency between music symbols to extract them. For instance, beams connect eighth notes or smaller rhythmic values and accidentals are placed before a notehead and at the same height. As a future trend, the importance of using global constraints to improve the results in the extraction of symbols should be addressed and further explored. Errors associated to missing symbols, symbols confusion and falsely detected symbols can be mitigated, e.g., by querying if the detected symbols' durations amount to the value of the time signature on each bar. We still believe that the inclusion of prior knowledge of syntactic and semantic musical rules may help the extraction of the handwritten music symbols and consequently it can also lead to better results in the future.

Final representation construction and comparison of results. The construction of a musical notation model to represent the musical sheet as a symbolic description. A web-based system providing broad access to a wide corpus of handwritten unpublished music encoded in digital format is also still needed. Moreover, a framework with appropriate metrics to measure the accuracy of different OMR systems does not exist, limiting the possibilities to test new algorithms.

An important issue that could also be addressed is the role of the users in the process. An automatic OMR system capable of recognizing handwritten music scores with high robustness and precision seems to be a goal difficult to achieve. Hence, interactive OMR systems may be a realistic and practical solution to this problem. MacMillan et al. [79] adopted a learning-based approach in which the process improves its results through experts users. The same active learning idea was suggested by Fujinaga [52] in which a method to learn new music symbols and handwritten music notations based on the combination of a k-nearest neighbor classifier with a genetic algorithm was proposed.

An interesting application of OMR concerns online recognition, which allows an automatic conver-

²<http://www.cvc.uab.es/cvcmuscima/competition/>.

sion of text as it is written on a special digital device. Taking into consideration the current proliferation of small electronic devices with increasing computation power, such as tablets and smartphones, it may be usefully to explore of such features applied to OMR. Besides, composers prefer the creativity which can only be entirely achieved without restrictions. This implies total freedom of use, not only of OMR softwares, but also the type of media (paper) where they can write their music. Hence, algorithms for OMR are still necessary.

References

- [1] A. Andronico and A. Ciampa. On automatic pattern recognition and acquisition of printed music. In ICMC, International Computer Music Conference, 1982. In B. Coüasnon and J. Camillerapp, *A way to separate knowledge from program in structured document analysis: Application to optical music recognition*, in *International Conference on Document Analysis and Recognition*, pp. 1092–7, 1995.
- [2] N. Arica and F. T Yarman-Vural. An overview of character recognition focused on off-line handwriting. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(2):216–233, May 2001.
- [3] D. Bainbridge. An extensible optical music recognition system. *Nineteenth Australasian Computer Science Conference*, pages 308–317, 1997.
- [4] D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.
- [5] D. Bainbridge and T.C. Bell. Dealing with superimposed objects in optical music recognition. In *Sixth International Conference on Image Processing and Its Applications*, volume 2, pages 756 –760 vol.2, jul 1997.
- [6] David Bainbridge and Tim Bell. A music notation construction engine for optical music recognition. *Software: Practice and Experience*, 33(2):173–200, 2003.
- [7] H. S. Baird. Document image defect models and their uses. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 62–67, Oct 1993.
- [8] P. Bellini, I. Bruno, and P. Nesi. Optical music sheet segmentation. *Proceedings of the First International Conference on Web Delivering of Music*, pages 183–190, Nov. 2001.
- [9] P. Bellini, I. Bruno, and P. Nesi. Assessing optical music recognition tools. *Computer Music Journal*, 31:68–93, March 2007.
- [10] P. Bellini, I. Bruno, and P. Nesi. Optical music recognition: Architecture and algorithms. In *Interactive Multimedia Music Technologies*, pages 80–110. Hershey: IGI Global, 2008.
- [11] J. Bernsen. Dynamic thresholding of grey-level images, 1986. In W. Bieniecki and Sz, Grabowski, Multi-pass approach to adaptive thresholding based image segmentation. *Proceedings of the 8th International IEEE Conference CADSM (2005)*.
- [12] I. Blayvas, A. Bruckstein, and R. Kimmel. Efficient computation of adaptive threshold surfaces for image binarization. *Pattern Recognition*, 39(1):89–101, 2006.

- [13] D. Blostein and H. S. Baird. A critical survey of music image analysis. In H. S. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 405–434. Springer-Verlag, Berlin, 1992.
- [14] M. Bojovic and M. D. Savic. Training of hidden Markov models for cursive handwritten word recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 973–976, Washington, DC, USA, 2000. IEEE Computer Society.
- [15] A. D. Brink and N. E. Pendock. Minimum cross-entropy threshold selection. *Pattern Recognition*, 29(1):179–188, 1996.
- [16] J. A. Burgoyne, Y. Ouyang, T. Himmelman, J. Devaney, L. Pugin, and I. Fujinaga. Lyric extraction and recognition on digital images of early music sources. In *Proceedings of the 10th International Society for Music Information Retrieval*, pages 723–727, 2009.
- [17] J. A. Burgoyne, L. Pugin, G. Eustace, and I. Fujinaga. A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources. In *Proceedings of the 8th International Society for Music Information Retrieval*, pages 509–512, 2007.
- [18] D. Byrd and M. Schindele. Prospects for improving OMR with multiple recognizers. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 41–47, Victoria, Canada, 2006.
- [19] A. Capela, J. S. Cardoso, A. Rebelo, and C. Guedes. Integrated recognition system for music scores. In *Proceedings of the International Computer Music Conference*, 2008.
- [20] A. Capela, A. Rebelo, J. S. Cardoso, and C. Guedes. Staff line detection and removal with stable paths. In *Proceedings of the International Conference on Signal Processing and Multimedia Applications (SIGMAP)*, pages 263–270, 2008.
- [21] J. S. Cardoso, A. Capela, A. Rebelo, and C. Guedes. A connected path approach for staff detection on a music score. In *Proceedings of the International Conference on Image Processing*, pages 1005–1008, 2008.
- [22] J. S. Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. F. Pinto da Costa. Staff detection with stable paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1134–1139, 2009.
- [23] J. S. Cardoso and A. Rebelo. Robust staffline thickness and distance estimation in binary and gray-level music scores. In *Proceedings of The Twentieth International Conference on Pattern Recognition (ICPR 2010)*, pages 1856–1859., 2010.
- [24] N. P. Carter. Automatic recognition of printed music in the context of electronic publishing, 1989. In Dorothea Blostein and Henry S. Baird, *A Critical Survey of Music Image Analysis, in Structured Document Image Analysis*, Baird, Bunke, and Yamamoto (Eds.), Eds., pp. 405–434, Springer-Verlag, Heidelberg, 1992.
- [25] Q. Chen, Q. Sun, P. Heng, and D. Xia. A double-threshold image binarization method based on edge detector. *Pattern Recognition*, 41(4):1254–1267, 2008.

- [26] G. S. Choudhury, M. Droetboom, T. DiLauro, I. Fujinaga, and B. Harrington. Optical music recognition system within a large-scale digitization project. In *International Society for Music Information Retrieval (ISMIR)*, 2000.
- [27] B. Coüasnon. *Segmentation et reconnaissance de documents guidées par la connaissance a priori: application aux partitions musicales*. PhD thesis, Université de Rennes, 1996.
- [28] B. Coüasnon, P. Brisset, and I. Stephan. Using logic programming languages for optical music recognition. *International Conference on the Practical Application of Prolog*, pages 115–34, 1995.
- [29] B. Coüasnon and J. Camillerapp. Using grammars to segment and recognize music scores. In *Proc. of DAS-94: International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 15–27, Kaiserslautern, 1993.
- [30] B. Coüasnon and J. Camillerapp. A way to separate knowledge from program in structured document analysis: Application to optical music recognition. *International Conference on Document Analysis and Recognition*, pages 1092–7, 1995.
- [31] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21 – 27, jan 1967.
- [32] C. Dalitz. Reject options and confidence measures for knn classifiers. In C. Dalitz, editor, *Schriftenreihe des Fachbereichs Elektrotechnik und Informatik Hochschule Niederrhein*, volume 8, pages 16–38. Shaker Verlag, 2009.
- [33] C. Dalitz, M. Droettboom, B. Czerwinski, and I. Fujigana. Staff removal toolkit for gamera, 2005-2007. <http://music-staves.sourceforge.net>.
- [34] C. Dalitz, M. Droettboom, B. Czerwinski, and I. Fujigana. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:753–766, 2008.
- [35] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen. Multimodal presentation and browsing of music. In *Proceedings of the 10th international conference on Multimodal interfaces*, pages 205–208, New York, NY, USA, 2008. ACM.
- [36] L. S. Dan, Second Examiner, and Dr. K. P. Chan. Final year project report automatic optical music recognition. Technical report, 1996.
- [37] M. Portes de Albuquerque, I. A. Esquef, and A. R. Gesualdi Mello. Image thresholding using tsallis entropy. *Pattern Recognition Letters*, 25(9):1059–1065, 2004.
- [38] A. F. Desaedeleer. Reading sheet music. Master’s thesis, Imperial College London, Technology and Medicine, University of London, 2006.
- [39] M. Droettboom, I. Fujinaga, and K. MacMillan. Optical music interpretation. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 378–386, London, UK, 2002. Springer-Verlag.

- [40] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.
- [41] A. Dutta, U. Pal, A. Fornés, and J. Lladós. An efficient staff removal approach from printed musical documents. In *Proceedings of the 20th International Conference on Pattern Recognition*, pages 1965–1968. IEEE Computer Society, 2010.
- [42] S. Escalera, A. Fornés, O. Pujol, P. Radeva, G. Sánchez, and J. Lladós. Blurred shape model for binary and grey-level symbol recognition. *Pattern Recognition Letters*, 30:1424–1433, November 2009.
- [43] M. Ferrand, J. A. Leite, and A. Cardoso. Hypothetical reasoning: An application to optical music recognition. In *APPIA-GULP-PRODE'99 Joint Conference on Declarative Programming (AGP'99)*, pages 367–381, L'Aquila, Italy, 1999.
- [44] A. Fornes, A. Dutta, A. Gordo, and J. Lladós. The icdar 2011 music scores competition: Staff removal and writer identification. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1511–1515, sept. 2011.
- [45] A. Fornés, S. Escalera, J. Lladós, G. Sánchez, P. Radeva, and O. Pujol. Handwritten symbol recognition by a boosted blurred shape model with error correction. In *Proceedings of the 3rd Iberian conference on Pattern Recognition and Image Analysis, Part I*, pages 13–21. Springer-Verlag, 2007.
- [46] A. Fornés, J. Lladós, G. Sánchez, and H. Bunke. Writer identification in old handwritten music scores. In *Proceedings of the 2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 347–353. IEEE Computer Society, 2008.
- [47] A. Fornés, J. Lladós, G. Sánchez, and H. Bunke. On the use of textural features for writer identification in old handwritten music scores. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 996–1000. IEEE Computer Society, 2009.
- [48] A. Fornés, J. Lladós, and G. Sánchez. Primitive segmentation in old handwritten music scores. In Wenyin Liu and Josep Lladós, editors, *GREC*, volume 3926 of *Lecture Notes in Computer Science*, pages 279–290. Springer, 2005.
- [49] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.
- [50] C. Fremerey, M. Müller, F. Kurth, and M. Clausen. Automatic mapping of scanned sheet music to audio recordings. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 413–418, 2008.
- [51] N. Friel and I.S. Molchanov. A new thresholding technique based on random sets. *Pattern Recognition*, 32(9):1507–1517, September 1999.
- [52] I. Fujinaga. Exemplar-based learning in adaptive optical music recognition system. In *Proceedings of the International Computer Music Conference*, pages 55–60, 1996.

- [53] I. Fujinaga. Staff detection and removal. In Susan George, editor, *Visual Perception of Music Notation: On-Line and Off-Line Recognition*, pages 1–39. Idea Group Inc., 2004.
- [54] B. Gatos, I. Pratikakis, and S. J. Perantonis. An adaptive binarisation technique for low quality historical documents. In *Document Analysis Systems VI*, volume 3163 of *Lecture Notes in Computer Science*, pages 102–113. Springer Berlin / Heidelberg, 2004.
- [55] C. Genfang, Z. Wenjun, and W. Qiuqiu. Pick-up the musical information from digital musical score based on mathematical morphology and music notation. In *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science*, pages 1141–1144. IEEE Computer Society, 2009.
- [56] S. George. Lyric recognition and christian music. In Susan George, editor, *Visual Perception of Music Notation: On-Line and Off-Line Recognition*, pages 198–225. Idea Group Inc., 2004.
- [57] R. Goecke. Building a system for writer identification on handwritten music scores. In *Proceedings of the IASTED International Conference on Signal Processing, Pattern Recognition, and Applications*, pages 250–255, Rhodes, Greece, 2003. Acta Press, Anaheim, USA.
- [58] Y. Grandvalet, A. Rakotomamonjy, J. Keshet, and S. Canu. Support vector machines with a reject option. In *Neural Information Processing Systems (NIPS)*, pages 537–544. MIT Press, 2008.
- [59] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.
- [60] W. Homenda. Optical music recognition: the case study of pattern recognition. In Marek Kurzynski, Edward Puchala, Michal Wozniak, and Andrzej zolnierrek, editors, *Computer Recognition Systems*, volume 30 of *Advances in Soft Computing*, pages 835–842. Springer Berlin / Heidelberg, 2005.
- [61] W. Homenda and M. Luckner. Automatic knowledge acquisition: Recognizing music notation with methods of centroids and classifications trees. In *Proceedings of the International Joint Conference on Neural Networks*, pages 3382–3388, 2006.
- [62] C. Hsu and C. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, Mar 2002.
- [63] L. K. Huang and M. J. J. Wang. Image thresholding by minimizing the measures of fuzziness. *Pattern Recognition*, 28(1):41–51, January 1995.
- [64] A. K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):267–278, 1996.
- [65] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1386–1391, 1997.

- [66] G. Jones, B. Ong, I. Bruno, and K. Ng. Optical music imaging: Music document digitisation, recognition, evaluation, and restoration. In *Interactive Multimedia Music Technologies*, pages 50–79. Hershey: IGI Global, 2008.
- [67] T. Kanungo. *Document degradation models and a methodology for degradation model validation*. PhD thesis, Seattle, WA, USA, 1996.
- [68] T. Kanungo, R.M. Haralick, H.S. Baird, W. Stuehle, and D. Madigan. A statistical, nonparametric methodology for document degradation model validation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1209–223, Nov 2000.
- [69] J.N. Kapur, P.K. Sahoo, and A.K.C. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, 29(3):273–285, March 1985.
- [70] M. Kassler. Optical character recognition of printed music: A review of two dissertations, 1972. In B. Vrist. Optical music recognition for structural information from high-quality scanned music. Technical report, 2009.
- [71] H. Kato and S. Inokuchi. *A recognition system for printed piano music*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992.
- [72] A. Khashman and B. Sekeroglu. A novel thresholding method for text separation and document enhancement. *Proceedings of the 11th Panhellenic Conference on Informatics (PCI 2007)*, May 2007.
- [73] I. Knopke and D. Byrd. Towards musicdiff: A foundation for improved optical music recognition using multiple recognizers. In *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 123–124, Vienna, Austria, 2007.
- [74] G. E. Kopec, P. Parc, and D. A. Maltzcarnege. Markov source model for printed music decoding. *Journal of Electronic Imaging*, pages 7–14, 1996.
- [75] F. Kurth, M. Müller, C. Fremerey, Y. Chang, and M. Clausen. Automated synchronization of scanned sheet music with audio recordings. In *Proceedings of the 8th International Conference on Music Information Retrieval*, Vienna, Austria, 2007.
- [76] L. Lam and Ching Y. Suen. Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers. *Pattern Recognition*, 21(1):19–32, 1988.
- [77] I. Leplumey, J. Camillerapp, and G. Lorette. A robust detector for music staves. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 902–905, 1993.
- [78] N. Luth. Automatic identification of music notations. In *Proceedings of the First International Symposium on Cyber Worlds*, page 203, Washington, DC, USA, 2002. IEEE Computer Society.
- [79] K. MacMillan, M. Droettboom, and I. Fujinaga. Gamera: Optical music recognition in a new shell. In *Proceedings of the International Computer Music Conference*, pages 482–485, 2002.

- [80] J. V. Mahoney. Automatic analysis of music score images. B.Sc thesis, Department of Computer Science and Engineering, MIT, 1982. In Dorothea Blostein and Henry S. Baird, *A Critical Survey of Music Image Analysis*, in *Structured Document Image Analysis*, Baird, Bunke, and Yamamoto (Eds.), Eds., pp. 405–434, Springer-Verlag, Heidelberg, 1992.
- [81] Y. Mitobe, H. Miyao, and M. Maruyama. A fast HMM algorithm based on stroke lengths for on-line recognition of handwritten music scores. In *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 521–526, Washington, DC, USA, 2004. IEEE Computer Society.
- [82] H. Miyao and R. M. Haralick. Format of ground truth data used in the evaluation of the results of an optical music recognition system. In *IAPR Workshop on document Analysis Systems*, pages 497–506, 2000.
- [83] H. Miyao and Y. Nakano. Note symbol extraction for printed piano scores using neural networks. *IEICE TRANSACTIONS on Information and Systems*, E79-D:548–554, 1996.
- [84] H. Miyao and M. Okamoto. Stave extraction for printed music scores using DP matching. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 8:208–215, 2004.
- [85] K. Ng. Optical music analysis for printed music score and handwritten manuscript. In Susan George, editor, *Visual Perception of Music Notation: On-Line and Off-Line Recognition*, pages 1–39. Idea Group Inc., 2004.
- [86] K. Ng and R. Boyle. Recognition and reconstruction of primitives in music scores. *Image and Vision Computing*, 14(1):39–46, 1996.
- [87] K. Ng, R. Boyle, and D. Cooper. Domain knowledge enhancement of optical music score recognition. Technical report, 1995.
- [88] K. Ng, D. Cooper, E. Stefani, R. Boyle, and N. Bailey. Embracing the composer: Optical recognition of handwritten manuscripts. In *Proceedings of the 1999 International Computer Music Conference*, 1999.
- [89] N. Nguyen and Y. Guo. Metric learning: A support vector approach. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 125–136, 2008.
- [90] W. Niblack. An introduction to digital image processing, 1986. In Graham Leedham and Chen Yan and Kalyan Takru and Joie Hadi Nata Tan and Li Mian, Comparison of Some Thresholding Algorithms for Text/Background Segmentation in Difficult Document Images. Proceedings of the Seventh International Conference on Document Analysis and Recognition (2003).
- [91] H. Nishida. A structural model of shape deformation. *Pattern Recognition*, 28(10):1611–1620, 1995.
- [92] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

- [93] N. R. Pal and S. K. Pal. Entropic thresholding, 1989. In Sezgin M. and Sankur B, Survey over Image Thresholding Techniques and Quantitative Performance Evaluation. *Journal of Electronic Imaging* 13 (2004) 146–165.
- [94] T. Pinto. Music score binarization based on content knowledge. Master's thesis, 2010.
- [95] T. Pinto, A. Rebelo, G. Giraldi, and J. S. Cardoso. Music score binarization based on domain knowledge. In *Pattern Recognition and Image Analysis*, volume 6669 of *Lecture Notes in Computer Science*, pages 700–708. Springer Berlin / Heidelberg, 2011.
- [96] D. Prerau. Computer pattern recognition of standard engraved music notation, 1970. In Dorothea Blostein and Henry S. Baird, *A Critical Survey of Music Image Analysis*, in *Structured Document Image Analysis*, Baird, Bunke, and Yamamoto (Eds.), Eds., pp. 405–434, Springer-Verlag, Heidelberg, 1992.
- [97] D. Prerau. Optical music recognition using projections, 1988. In Dorothea Blostein and Henry S. Baird, *A Critical Survey of Music Image Analysis*, in *Structured Document Image Analysis*, Baird, Bunke, and Yamamoto (Eds.), Eds., pp. 405–434, Springer-Verlag, Heidelberg, 1992.
- [98] D. Pruslin. Automatic recognition of sheet music, 1966. In Dorothea Blostein and Henry S. Baird, *A Critical Survey of Music Image Analysis*, in *Structured Document Image Analysis*, Baird, Bunke, and Yamamoto (Eds.), Eds., pp. 405–434, Springer-Verlag, Heidelberg, 1992.
- [99] L. Pugin. Optical music recognition of early typographic prints using Hidden Markov Models. In *International Society for Music Information Retrieval*, pages 53–56, 2006.
- [100] L. Pugin, J. Burgoyne, and I. Fujinaga. Goal-directed evaluation for the improvement of optical music recognition on early music prints. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 303–304. ACM, 2007.
- [101] L. Pugin, J.A. Burgoyne, and I. Fujinaga. MAP adaptation to improve optical music recognition of early music documents using Hidden Markov Models. In *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 513–16, Vienna, Austria., 2007.
- [102] R. Randriamahefa, J.P. Cocquerez, C. Fluhr, F. Pepin, and S. Philipp. Printed music recognition. *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 898–901, Oct 1993.
- [103] G. Read. *Music Notation: A Manual of Modern Practice (2nd ed.)*. Taplinger, New York, 1969.
- [104] A. Rebelo. New methodologies towards an automatic optical recognition of handwritten musical scores. Master's thesis, 2008.
- [105] A. Rebelo, A. Capela, J. F. Pinto da Costa, C. Guedes, E. Carrapatoso, and J. S. Cardoso. A shortest path approach for staff line detection. *Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, 2007, pages 79–85, Nov. 2007.

- [106] A. Rebelo, G. Capela, and J. S. Cardoso. Optical recognition of music symbols: A comparative study. *International Journal on Document Analysis and Recognition*, 13:19–31, 2009.
- [107] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. Marcal, C. Guedes, and J. S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, pages 1–18, 2012. 10.1007/s13735-012-0004-6.
- [108] A. Rebelo, F. Paszkiewicz, C. Guedes, A. Marcal, and J. S. Cardoso. A method for music symbols extraction based on musical rules. In *Bridges: Mathematical Connections in Art, Music, and Science*, pages 81–88, 2011.
- [109] A. Rebelo, J. Tkaczuk, R. Sousa, and J. S. Cardoso. Metric Learning for Music Symbol Recognition. In *The tenth International Conference on Machine Learning and Applications (ICMLA'11)*, 2011.
- [110] K. T. Reed and J. R. Parker. Automatic computer recognition of printed music. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 3, pages 803–807 vol.3, Aug 1996.
- [111] T. W. Ridler and S. Calvard. Picture thresholding using an iterative selection method, 1978. In N. B. Venkateswarlu, Implementation of some image thresholding algorithms on a Connection Machine-200. *Pattern Recognition Letters* 16 759–768 (1995).
- [112] J. Riley and I. Fujinaga. Recommended best practices for digital image capture of musical scores. *OCLC Systems & Services*, 19(2):62–69, 2003.
- [113] J. W. Roach and J. E. Tatem. Using domain knowledge in low-level visual processing to interpret handwritten music: an experiment. In Dorothea Blostein and Henry S. Baird, *A Critical Survey of Music Image Analysis*, in *Structured Document Image Analysis*, Baird, Bunke, and Yamamoto (Eds.), Eds., pp. 405–434, Springer-Verlag, Heidelberg, 1992.
- [114] F. Rossant and I. Bloch. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP Journal on Advances in Signal Processing*, 2007(1):160–160, 2007.
- [115] C.H. Papadimitriou S. Dasgupta and U.V. Vazirani. *Algorithms*, pages 169–179. McGraw-Hill Higher Education, 2006.
- [116] P. K. Sahoo, C. Wilkins, and J. Yeager. Threshold selection using renyi’s entropy. *Pattern Recognition*, 30(1):71–84, 1997.
- [117] B. Schölkopf. The kernel trick for distances. In *Advances in neural information processing systems (NIPS)*, pages 5–3, 1993.
- [118] M. I. Sezan. A peak detection algorithm and its application to histogram-based image data reduction, 1985. In Sezgin M. and Sankur B, Survey over Image Thresholding Techniques and Quantitative Performance Evaluation. *Journal of Electronic Imaging* 13 (2004) 146–165.

- [119] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, 2004.
- [120] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:623–656, 1948.
- [121] S. Sheridan and S. E. George. Defacing music score for improved recognition. In Gad Abraham and Benjamin I. P. Rubinstein, editors, *Proceedings of the Second Australian Undergraduate Students' Computing Conference*, pages 1–7, Melbourne, Victoria, Australia, December 2004. Australian Undergraduate Students' Computing Conference.
- [122] R. Sousa, B. Mora, and J. S. Cardoso. An ordinal data method for the classification with reject option. In *Proceedings of The Eighth International Conference on Machine Learning and Applications*, 2009.
- [123] M. Szwoch. A robust detector for distorted music staves. In *Computer Analysis of Images and Patterns*, pages 701–708. Springer-Verlag, Heidelberg, 2005.
- [124] M. Szwoch. Guido: A musical score recognition system. *Document Analysis and Recognition, International Conference on*, 2:809–813, 2007.
- [125] M. Szwoch. Using musicxml to evaluate accuracy of omr systems. In Gem Stapleton, John Howse, and John Lee, editors, *Diagrammatic Representation and Inference*, volume 5223 of *Lecture Notes in Computer Science*, pages 419–422. Springer Berlin / Heidelberg, 2008.
- [126] L. J. Tardón, S. Sammartino, I. Barbancho, V. Gómez, and A. Oliver. Optical music recognition for scores written in white mensural notation. *EURASIP Journal on Image and Video Processing*, 2009:6:3–6:3, February 2009.
- [127] G. Taubman, A. Odest, and C. Jenkins. Musichand: A handwritten music recognition system. Technical report, 2005.
- [128] Michael Thulke, Volker Märgner, and Andreas Dengel. A general approach to quality evaluation of document segmentation results. In *DAS '98: Selected Papers from the Third IAPR Workshop on Document Analysis Systems*, pages 43–57, London, UK, 1999. Springer-Verlag.
- [129] M. E. Tipping. The relevance vector machine. In T. K. Leen In S. A. Solla and K. R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658. MIT Press, 2000.
- [130] F. Toyama, K. Shoji, and J. Miyamichi. Symbol recognition of printed piano scores with touching symbols. In *Proceedings of the International Conference on Pattern Recognition*, pages 480–483, Washington, DC, USA, 2006. IEEE Computer Society.
- [131] O. D. Trier and A. K. Jain. Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1191–1201, Dec 1995.
- [132] O. D. Trier and T. Taxt. Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):312–315, Mar 1995.

- [133] D. Tsai. A fast thresholding selection procedure for multimodal and unimodal histograms. *Pattern Recognition Letters*, 16(6):653–666, 1995.
- [134] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.
- [135] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.
- [136] T. Wakahara. Shape matching using LAT and its application to handwritten numeral recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):618–629, 1994.
- [137] Y. Wang, K. Fan, Y. Juang, and T. Chen. Using hidden Markov model for chinese business card recognition. In *IEEE International Conference on Image Processing*, pages 1106–1109, 2001.
- [138] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, June 2009.
- [139] K. Q. Weinberger, F. Sha, and L. K. Saul. Convex optimizations for distance metric learning and pattern classification. *IEEE Signal Processing Magazine*, 27(3):146–158, may 2010.
- [140] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University, 2006.
- [141] S. D. Yanowitz and A. M. Bruckstein. A new method for image segmentation. In *Computer Vision, Graphics, and Image Processing*, volume 46, pages 82–95, Apr 1989.
- [142] Z. Zhang, J. T. Kwok, and D. Yeung. Parametric distance metric learning with label information. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1450–1452, 2003.

Part V

Appendix

Fundamentals for Staff Line Detection and Removal Algorithms

A.1 Dynamic Programming

Before presenting the operations of dynamic programming techniques let us take a look to the following example [115]. Figure A.1 represents a directed acyclic graphs and its linearization (topological ordering – the nodes are arranged on a line so that all edges go from left to right). This linearization process is important for the shortest path.

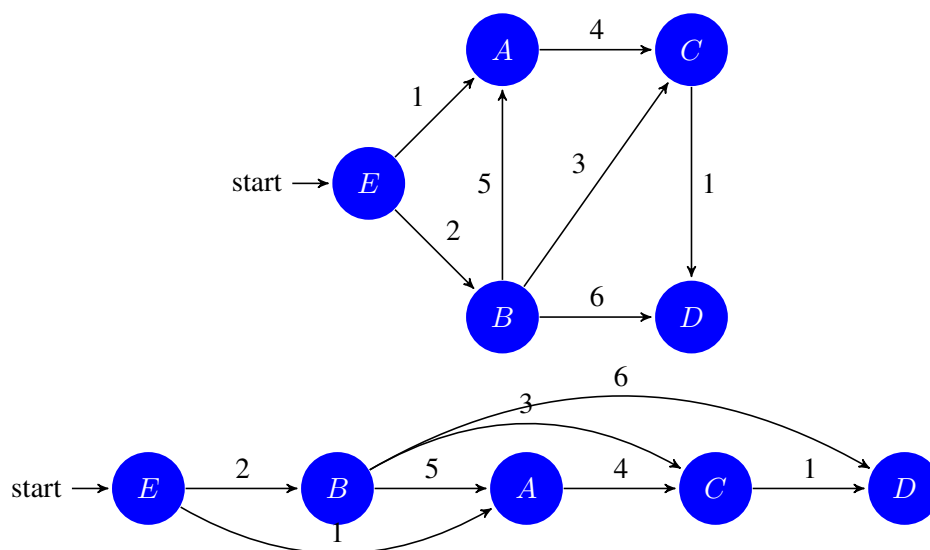


Figure A.1: A directed acyclic graphs and its linearization.

Suppose we want to figure out distances from node E to the other nodes. For concreteness, let's focus on node C . The only way to get to it is through its predecessors, B or A ; so to find the shortest path to C , we need only compare these two routes:

$$\text{dist}(C) = \min \{ \text{dist}(B) + 3, \text{dist}(A) + 4 \}.$$

A similar relation can be written for every node. If we compute these dist values in the left-to-right order of Figure A.1, we can certainly get to a node v , and then we already have the information needed to compute $\text{dist}(v)$. We are therefore able to compute all distances in a single pass:

```
initialize all  $\text{dist}(\cdot)$  values to  $\infty$ 
 $\text{dist}(s) = 0$ 
for each  $v \in V \setminus \{s\}$ , in linearized order:
     $\text{dist}(v) = \min_{(u,v) \in E} \{ \text{dist}(u) + l(u, v) \}$ 
```

In face of it, we can note that this algorithm is solving a collection of subproblems, $\{ \text{dist}(u) : u \in V \}$: it starts with the smallest of the distances, $\text{dist}(e)$, since it immediately knows its answer to be 0; then,

it proceeds with progressively “larger” subproblems – distances to vertices that are further and further along in the linearization – where it is thinking of a subproblem as large if it needs to have solved a lot of other subproblems before it can get to it. This is a very general technique. At each node, the algorithm computes some function of the values of the node’s predecessors. In this case, the particular function is a minimum of sums.

This is dynamic programming. This is a powerful algorithmic paradigm for efficiently solving a wide range of search and optimization problems which exhibit the characteristics of *overlapping subproblems* (the problem can be broken down into subproblems which are reused several times) and *optimal substructure* (optimal solutions of subproblems can be used to find the optimal solutions of the overall problem).

Now, it is crucial to know what the subproblems are when we are solving a problem by dynamic programming. Each node will represent a subproblem, and each edge will represent a precedence constraint, of the form $(i - 1, j) \rightarrow (i, j)$, $(i, j - 1) \rightarrow (i, j)$, and $(i - 1, j - 1) \rightarrow (i, j)$, on the order in which the subproblems are tackled. We can also put weights on the edges.

A.2 Staff Line Removal

Staff line detection algorithms can be used as a first step in many staff removal algorithms. In this thesis, the following algorithms were considered: LineTrack Height, LineTrack Chord, Roach/Tatem – see [34].

Line Track Height The algorithm tracks the staff lines and checks when a vertical black run is longer than a threshold (experimentally set at $2 \times \text{stafflineheight}$).

Line Track Height Modified The version modified of the Line Track Height algorithm also track the staff lines positions obtained by a detection algorithm and removes vertical run sequences of black pixels that have a value lower than a specified threshold (chosen experimentally as $2 \times \text{stafflineheight}$). In this version, a carefully attention to the deformations – staff lines may have discontinuities, be curved or inclined – that may occur in the music scores are given. These problems will influence the success to achieve a correct detection of lines contained on the score. The positions of the staff lines obtained by a staff line detection algorithm may pass slightly above or under the real staff lines positions. Therefore, if we are in presence of a white pixel when the staff lines are tracked, we search vertically for the closest black pixel. If that distance is lower than a specified tolerance – experimentally chosen as $1 + \text{ceil}(\text{stafflineheight}/3.0)$ – we move the reference position of the staff line to the position of the black pixel found.

Line Track Chord This algorithm computes, for a fixed angle resolution of three degrees, the chord length through the skeleton point (see Figure A.2). This results in a function $\text{chordlength}(\varphi)$, where φ is the chord angle, for each skeleton point. When the staff line pixel also belongs to a crossing music symbol, the function should have a second distinct peak. To detect this peak the following thresholds are used:

1. There is a local maximum when chordlength is greater than $5 \times \text{stafflineheight}$ at an angle below 30 degrees and another local maximum when chordlength is greater than $1.75 \times \text{stafflineheight} \times \sin(\varphi)$ at an angle $\varphi > 30$ degrees.
2. The valley between two maxima must have a depth greater than $1.5 \times \text{stafflineheight}$.

Concluding, this algorithm removes the staff line through the angles peaks of the chord lengths. There are two distinct peaks depending if the pixels belong to a staff line or a music symbol.

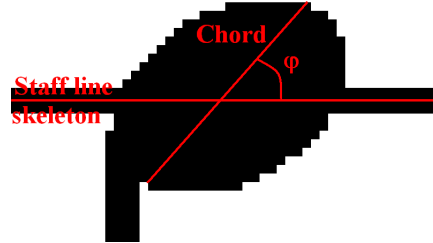


Figure A.2: Length of a chord through a skeleton point at some angle φ .

Roach/Tatem This algorithm uses a labelling scheme based on the angle information and pixel adjacency to identify the staff line pixels [113]. The chord length and the angle function described in the Line Track Chord are computed for every pixel. Consequently, the original image can be transformed into two-dimensional vector field by picking the angle and length of the longest chord for each black pixel. This will assign pixels on staff lines a high length value and an angle value of zero. To avoid the removal of symbol pixels on the staff lines, some horizontal line pixels are iteratively relabelled as non-horizontal pixels, depending on the labels of their neighboring pixels.

Skeleton This method consists of the following steps:

1. The skeleton is split at branching point and corner points with an angle below 135 degrees. Around each splitting point a number of pixels are removed – see Figure A.3(a).
2. Staff line segment candidates are picked as skeleton segments if the orientation angle (least square fitted line) is below 25 degrees, the segment is wider than tall and the *straightness* (mean square deviation from least square fitted line) is below $\text{stafflineheight}^2/2$.
3. Apply a staff-finding algorithm to the staff segment candidates. Two staff segments are horizontally linked when their extrapolations from the end points with the least square fitted angle come closer than $\text{stafflineheight}/2$.
4. Remove false positives: from each overlapping staff segment group on the same line the one that is closest to its least square fitted neighborhood is picked and the others are discarded; non-staff segments that have the same branching point as a staff segment are extrapolated by a parametric parabola: if this parabola is approximately tangential to the staff segment, the latter is considered a false positive – see Figure A.3(b).
5. Remove staff lines: all vertical black runs around the detected staff skeleton are removed.

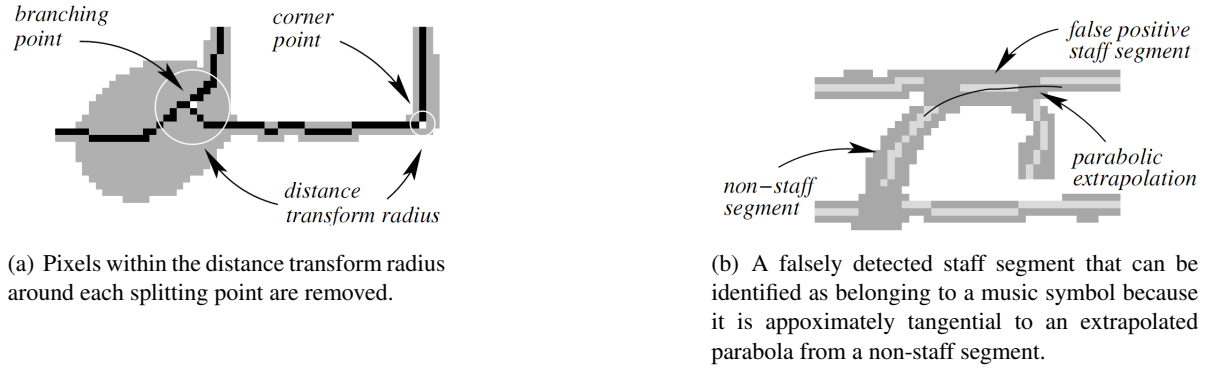


Figure A.3: Example (from [34]).

A.3 Removal Error Metrics

Individual Pixels This metric considers the staff line removal as a two-class classification problem at the pixel level, that is, one pixel can belong to a staff line or not. Therefore, a natural performance measure is the error rate for this classification, given by

$$\text{Pixel error rate} = \frac{x + y}{z}$$

x = Number of misclassified staff pixels

y = Number of misclassified non staff pixels

z = Number of all black pixels

However, this metric has one problem: little information is given about how well the staff removal algorithm separates symbols that are otherwise connected by staff lines; this error only indicates how badly the symbols are distorted when compared to the ideal staff-less images.

Segmentation Region Level This error metric has as base the regions of the line segments. The staff line removal can be considered as a segmentation problem where the staff lines segments need to be separated from the symbol segments.

In an OMR application the staff lines segments are considered “background” and the remaining symbols are taken as being the “segments of interest”. Nevertheless, when we are trying to evaluate the quality of the staff line removal the situation is reversed: our interest lies on the staff segments and the rest constitutes “background”.

Following the notation given in [128], we have two segmentations for the set of black pixels present in the test image:

1. The ground-truth segmentation $G = G_{obj} \cup \{g_{noise}\}$ with $G = \{g_1, \dots, g_M\}$
2. The segmentation detected from the algorithm $S = S_{obj} \cup \{s_{noise}\}$ with $S = \{s_1, \dots, s_N\}$, where each g_i and s_j contains the black pixels of a contiguous staff segment respectively, and g_{noise} and s_{noise} contain the remaining background black pixels, respectively.

In the set of all staff line segments from both segmentations an equivalence classes of overlapping segments is built (two segments are considered equivalent $a \simeq b$ when a sequence c_1, c_2, \dots, c_n exists with $c_1 = a$, $c_n = b$ and $c_i \cap c_{i+1} \neq \emptyset$). For each equivalence class r we count the number of the contained numbers of segments G and S and thus detect recognition errors. All possible cases are listed in Table A.1. The formula is given by

$$\text{Segmentation error rate} = \frac{x - y}{z}$$

x = Number of all classes r

y = Number of classes representing a correct recognition

z = Number of all classes r

Classes number	Segments from G_{obj}	Segments from S_{obj}	Error description
n_1	1	1	Correct
n_2	1	0	Missed segment
n_3	0	1	Falsely detected segment
n_4	1	>1	Segment split
n_5	>1	1	Segments merged
n_6	>1	>1	Both splitting and merging occurred

Table A.1: Staff segment extraction errors based on the number of segments in an equivalence class r .

Staff Line Interruption To obtain this metric a comparison between the ground-truth images, containing only the segments removed from the ideal case, with the image that contains exactly the pixels that were in fact removed by the removal algorithm under evaluation, is made. In doing so, each staff line is followed, from left to right, in the images containing only the removed staff segments; and interruptions in the staff line are looked at. Each interruption represents a detected music symbol that crosses the staff line. It follows two sets of intervals: the interrupting intervals $G = \{g_1, \dots, g_M\}$ in the ground-truth data and those in the algorithm output $S = \{s_1, \dots, s_N\}$.

With the purpose to establish an error metric, a bipartite graph is created by adding links between intervals g_i and s_j that overlap. In this manner, two types of error are revealed: intervals from G to S without a link and intervals with more than one link. In order to count the number of errors of the second type, the maximum cardinality matching in this graph is computed [?]. This cardinality matching also removes the minimal number of links leading to the second type error. As a result error rate we have

$$\text{Staff line interruption error rate} = \frac{\min \{n_3, n_1 + n_2\}}{n_3}$$














n_1 = Number of interruptions without link

n_2 = Number of removed links

n_3 = Number of ground-truth interruptions

Musical Symbols Classification

B.1 Musical Symbols

Main Classes	Secondary Classes
Accents	Staccato ·
	Staccatissimo ¸
	Dynamic >
	Fermata ^
	Tenuto –
	Marcato ^
	Stopped +
	Harmonic o
	Mordent ~
	Turn ∞
Sharps #	
Naturals ♮	
Flat ♭	
Barlines	
Beams 	
AltoClef 	
BassClef 	
TrebleClef 	
Breve    	
Dots ·	
Notes 	
TimeSignatureN	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
TimeSignatureL	Cut 
	Common 
Semibreve 	
Rests1 	
Rests2	Quaver 7
	Semiquaver 7













	Demisemiquaver 
	Hemidemisemiquaver 
	Whole 
	Half 
Relation	Tie 
	Slur 
	Glissando 
NotesOpen 	
NotesFlag	Quaver 
	Semiquaver 
	Demisemiquaver 
	Hemidemisemiquaver 

Table B.1: The set of the musical symbols considered.

B.2 Experimental Testing

In this section an analytical study of the common classification algorithms presented in chapter 4 is addressed. The work presented in [106] was a first investigation to explore these techniques. A more complete analysis, that extends the experimental results, is now presented.

Towards a comparative study between classification procedures, five different approaches were evaluated: HMMs, SVMs, RVMs, NNs and kNN. Three groups of experimental tests were carried out:

- First: the full set of training patterns extracted from the database of scores was augmented with replicas of the existing patterns, transformed according to the elastic deformation technique (section 3.3 chapter 3). Such transformations try to introduce robustness in the prediction with respect to the known variability of symbols.
- Second: the different classifiers were tested for the following situations: separation of types of music scores (handwritten and printed), gradual increase of deformations and union of real and printed scores.
- Third: a distance metric directly connected to the application domain and the adopted symbol representation was studied. The idea was to learn a Mahalanobis distance for kNN and SVMs.

For the NN, kNN, SVMs and RVMs methods, each image of a symbol was initially resized to 20×20 pixels and then converted to a vector of 400 binary values; under the HMM, the images were normalised with a height and width of 150 and 30 pixels, respectively. These approaches follow standard practices from the state of the art algorithms in the OMR field [99]. Moreover, in HMMs a 2-pixel sliding window mechanism over the symbol image was used to produce the sequence of observations (features). In doing so, dependent observations are replaced by observations depending on the horizontal position of the window. The extracted features are based on the work of Pugin [99]:

1. the number of distinct connected components of black pixels in the window;
2. the area of the largest black connected component normalized by the area of the window;
3. the area of the smallest white connected component normalized by the area of the window;

4. the position (x and y) of the centre of gravity of all black pixels in the window, normalized between 0 and 1.

For the second and third comparative study, rather than using only the image pixels we also decided to add in the classification process 7 music symbol features. The aim was to increase the final performance of the classifier by including characteristics that distinguish similar objects. The extracted features were based on the Gamera project¹:

1. The percentage of black pixels in the 20×20 pixels window of the image;
2. the orientation of the symbol;
3. the number of vertical holes;
4. the number of horizontal holes;
5. the compactness (the ratio between volume and connected components area);
6. the number of end points in the object skeleton;
7. the number of intersections in the object skeleton.

A Blurred Shape Model (BSM) descriptor [42] was also used and added to the previous features in the metric learning case (third experimental test). A BSM descriptor encodes the probability of pixel densities of image regions and hence symbols are described by a probability density function. Through the high gradient magnitude of the pixels, the shape of the symbol can be codified in terms of a set of key points. Then a grid comprised by a set of spatial regions is defined by the BSM descriptor. In the end, the spatial relations among key points from neighbor regions are established and features are computed. The output descriptor is a vector histogram where each position represents a distribution of probabilities of the symbol structure considering spatial distortions encompassing four possible sizes: 8×8 , 16×16 , 32×32 and 64×64 [42]. We opted for a feature vector histogram of the size 16×16 , not only due to the computational effort, but also because a higher definition grid would not provide a richer information (contours, structure, etc) due to the proximity to the working image size (20×20).

For the proposed evaluation of the different recognition methods, the data set of both real handwritten scores and synthetic scores, as presented in Chapter 3, was adopted.

First and second experiments

The real scores consist on a set of 50 handwritten scores from 5 Portuguese musicians. Images were previously binarized with the Otsu threshold algorithm. The deformations applied to the perfect scores were only those with significant impact on the features of symbols. For the preliminary tests, the deformations were rotation and curvature. In total, 288 images were generated from 18 perfect scores. For the posterior tests, where a gradual study of the classifiers performances was made, the deformations applied to the printed scores were curvature, rotation, Kanungo and white speckles. In total, 380 distorted images were generated from 19 original scores. The images from the real scores data set were previously binarized. Regarding the results achieved in the preliminary experiments, HMM was discarded here due to its less satisfying performance.

¹<http://gamera.informatik.hsnr.de>

The relevant classes for handwritten/printed music symbols used in the training phase of the classification models is presented in Table B.2. The symbols are grouped according to their shape. The rests symbols were divided into two groups – RestI and RestII. Besides that, an unknown class was included to classify those symbols that do not fit into any of the classes listed in Table B.2. In total, we have 3222 handwritten music symbols, 2521 printed music symbols and 14 classes for each type of music score.

Handwritten Music Symbols	>	9	≡	♭	♮	♩	♪	♫	⋈	⋈	♯	▼	♩
	Accent	BassClef	Beam	Flat	Natural	Note	NoteFlag	NoteOpen	RestI	RestII	Sharp	Staccatissimo	TrebleClef
Printed Music Symbols	⌘	^	≡	♭	♮	♩	♪	♫	⋈	⋈	♯	⌘	♩
	AltoClef	TieSlur	Beam	Flat	Natural	Note	NoteFlag	NoteOpen	RestI	RestII	Sharp	Time	TrebleClef

Table B.2: Full set of handwritten and printed music symbols considered.

For evaluation of the pattern recognition process, the available dataset was randomly split into training and test sets, with 60% and 40% of the data, respectively. This division was repeated ten times in order to obtain more stable results for accuracy by averaging and also to assess the variability of this measure. No special constraint was imposed on the distribution of the categories of symbols over the training and test sets; we only guaranteed that at least one example of each category was present in the training set. The best parameterization of each model was found based on a 4-fold cross validation scheme conducted on the training set. A confidence interval was estimated for the mean of the error as

$$\bar{X} - t^* \frac{S}{\sqrt{N}} \leq \mu \leq \bar{X} + t^* \frac{S}{\sqrt{N}} \quad (\text{B.1})$$

where t^* is the upper $(1 - C)/2$ critical value for the t distribution with $N - 1$ degrees of freedom, \bar{X} is the sample mean, S is the sample standard deviation and N is the sample size.

Results from the first experiment

From the results obtained for the handwritten music symbols – see Table B.3 – we can conclude that the classifier with the best performance was the support vector machine, with a 99% confidence interval for the expected performance [95%; 96%]. Interestingly, the performance of the simplest model – the nearest neighbour classifier – was clearly better than the performance of the HMM and the neural network model and close to the performance of the SVMs. Finally, although the neural network performed slightly better than the HMM, it exhibited strong difficulties with some classes, presenting very low accuracy values (BassClef and NoteOpen).

The results obtained for the printed music symbols – see Table B.4 – further support the superiority of the SVM model, with a 99% confidence interval for the expected performance [97%; 99%]. As expected, all models presented the best performance when processing printed musical scores.

Next, we investigated the potential of the elastic deformation to improve the performance of the classification models. The deformations as given by

$$D(x, y) = \sum_{m=1}^M \sum_{n=1}^N \frac{\xi_{mn}^x \mathbf{e}_{mn}^x + \xi_{mn}^y \mathbf{e}_{mn}^y}{\lambda_{mn}} \quad (\text{B.2})$$

(Equation (B.2) section 3.3) with $M = 1, 2, 3$ and $N = 1, 2, 3$ were applied to the training data.

	Neural network	Nearest neighbour	Support vector machines	Hidden Markov model
Accent	85%	99%	99%	91%
BassClef	13%	78%	77%	56%
Beam	85%	98%	95%	90%
Flat	84%	99%	98%	87%
Natural	93%	99%	98%	91%
Note	82%	97%	96%	73%
NoteFlag	51%	86%	89%	64%
NoteOpen	3%	75%	40%	22%
RestI	78%	100%	97%	90%
RestII	96%	100%	100%	92%
Sharp	85%	98%	98%	84%
Staccatissimo	58%	100%	100%	100%
TrebleClef	40%	92%	90%	94%
Unknown	52%	71%	89%	38%
99% CI for the Expected performance in percentage:	[81; 84]	[93; 95]	[95; 96]	[77; 81]

Table B.3: Accuracy obtained for the handwritten music symbols for the classifiers trained without elastically-deformed symbols.

The results in Tables B.5 and B.6 lead us to conclude that the application of the elastic deformation to the music symbols does not improve the performance of the classifiers. Only in two handwritten music symbols, very similar in shape, the accuracy did improve with elastically-deformed symbols – see Table B.7.

It is important to state that the features used in the SVM, nearest neighbour and neural network were raw pixels. This choice, grounded in standard practices in the literature, influences the performance of the classifiers: a slight change on the boundary of a symbol can modify the image scaling and, as a result, many pixel values may change.

Results from the second experiment

From the results obtained for the union of the handwritten and printed scores – see Table B.8 – we can conclude that the SVMs had the best performance ([95%; 96%]), outperforming the k -NNs – the simplest model – by approximately 1%, with a 99% confidence interval for the expected performance. The performance of NNs was clearly worse than the other two classifiers. Besides that, this algorithm exhibited some difficulties with a few classes, presenting low accuracy values (NoteFlag and TrebleClef).

As expected, and as it is shown in Table B.9 the performances of classifiers are slightly better than when analyzing only the raw pixels – a 400 feature vector, resulting from a 20×20 pixel image. This is due to the additional of the 7 features, so that more information is provided to the classifier, which

	Neural network	Nearest neighbour	Support vector machines	Hidden Markov model
AltoClef	94%	99%	98%	83%
Beam	92%	100%	100%	98%
Flat	97%	100%	99%	96%
Natural	94%	100%	100%	95%
Note	90%	99%	99%	91%
NoteFlag	70%	92%	96%	65%
NoteOpen	88%	98%	97%	85%
TieSlur	55%	94%	87%	81%
RestI	85%	100%	100%	83%
RestII	75%	100%	100%	69%
Sharp	97%	100%	100%	99%
Time	40%	100%	100%	27%
TrebleClef	93%	100%	100%	58%
Unknown	65%	79%	93%	74%
99% CI for the Expected performance in percentage:	[88; 89]	[96; 97]	[97; 99]	[83; 86]

Table B.4: Accuracy obtained for the printed music symbols for the classifiers trained without elastically-deformed symbols.

makes the recognition more robust. Looking at the classification of each class, almost every class has a better accuracy value, which improves the overall performance.

In an overall view of the results obtained with the gradual increase of deformations – see Table B.11 – the achievements of the classifiers do not vary much by changing the degradation degree (except White Speckles). This leads us to conclude that they do not influence the dispersion of the classes, in other words music symbols maintain their shape, although the performances decay up to 30% by rising the degradation factor of the White Speckles. The main structure of the symbol is lost with this increase. The results obtained for the degraded printed music symbols further support the superiority of the SVM model, with a 99% confidence interval for the expected performance.

From the results obtained with the separation of types of music scores – see Table B.10 – the simplest model outperformed the other, more complex, methods. On the one hand, the performance of the k-NN classifier with synthetic music sheets with a 99% confidence interval for the expected performance achieved [91%; 93%]. On the other hand, when processing handwritten musical scores, all models presented a weaker accuracy, up to 30% lower. This is due to the high variability inherent to each individual writing style.

Metric learning in OMR

Most authors use margin-based multi-class classification methods, for instance SVMs, as a benchmark for classifying music scores. This is a common approach and can be identified in the state of the art.

	Neural network	Nearest neighbour	Support vector machines	Hidden Markov model
Accent	83%	100%	100%	87%
BassClef	0%	95%	73%	44%
Beam	85%	96%	96%	87%
Flat	82%	99%	99%	71%
Natural	92%	99%	98%	84%
Note	86%	97%	97%	64%
NoteFlag	20%	83%	91%	34%
NoteOpen	2%	53%	43%	11%
RestI	59%	99%	97%	99%
RestII	93%	100%	100%	75%
Sharp	85%	99%	99%	82%
Staccatissimo	30%	100%	100%	100%
TrebleClef	33%	91%	90%	63%
Unknown	34%	64%	84%	34%
99% CI for the Expected performance in percentage:	[77; 80]	[92; 93]	[94; 96]	[69; 72]

Table B.5: Accuracy obtained for the handwritten music symbols for the classifiers trained with elastically-deformed symbols.

However, it is possible to find strong similarities between Large Margin Nearest Neighbor classification (LMNN) and SVMs [138]. The competing terms in

$$\epsilon(L) = \sum_{ij} \eta_{ij} \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 + c \sum_{ijl} \eta_{ij} (1 - \tau_{il}) [1 + \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 - \|L(\mathbf{x}_i - \mathbf{x}_l)\|^2]_+ \quad (\text{B.3})$$

Equation (4.6) are analogues to the ones presented in the cost function at SVMs. One term penalizes the norm of the *parameter* vector (linear transformation in distance metric, or (in SVM) the weight vector of the maximum margin hyperplane. The second terms are responsible for hinge loss over the examples that violate the condition of unit margin (the goal of margin maximization and a convex objective function are based on hinge loss). Moreover, LMNN has no explicit dependence on the number of classes, while in SVMs (multi-class classification) the training time scales at least linearly in the number of classes. For these reasons we decided to perform a comparative study between LMNN and SVMs.

The real scores consist on a set of 65 handwritten scores from 6 different composers. Images were previously binarized with the Otsu threshold algorithm. The deformations applied to these printed scores were curvature, rotation, Kanungo and white speckles. In total, 380 distorted images were generated from 19 original scores.

	Neural network	Nearest neighbour	Support vector machines	Hidden Markov model
AltoClef	86%	99%	97%	97%
Beam	95%	100%	100%	99%
Flat	95%	100%	99%	82%
Natural	92%	100%	98%	95%
Note	81%	100%	98%	89%
NoteFlag	43%	94%	97%	39%
NoteOpen	89%	97%	98%	78%
TieSlur	17%	91%	89%	67%
RestI	87%	100%	100%	100%
RestII	33%	100%	97%	91%
Sharp	97%	100%	100%	100%
Time	0%	100%	100%	27%
TrebleClef	89%	100%	100%	91%
Unknown	42%	76%	93%	38%
99% CI for the Expected performance in percentage:	[79; 83]	[95; 97]	[97; 99]	[81; 84]

Table B.6: Accuracy obtained for the printed music symbols for the classifiers trained with elastically-deformed symbols.

	Nearest neighbour		Support vector machines	
	With Elastic Deformation	Without Elastic Deformation	With Elastic Deformation	Without Elastic Deformation
Natural	100%	99%	98%	98%
Sharp	99%	98%	99%	98%

Table B.7: Accuracy on the natural and sharp symbols.

The relevant classes for handwritten/printed music symbols used in the training phase of the classification models are presented in Table B.12. Once again, the symbols are grouped according to their shape; the rests symbols were divided into two groups – RestI and RestII. In total the classifiers were evaluated on a database containing 7128 examples divided into 20 classes.

For evaluation of the pattern recognition processes, the available dataset was randomly split into three sub-sets: training, validation and test sets, with 25%, 25% and 50% of the data, respectively, following the state-of-the-art suggestions. This division was repeated 20 times in order to obtain more stable results for accuracy by averaging and also to assess the variability of this measure. No special constraint was imposed on the distribution of the categories of symbols over the training, validation and test sets; we only guaranteed that at least one example of each category was present in the training set. The best parametrization of each model was found using the training and validation sets being the expected error estimated on the test set by a 4-cross validation scheme. In this manner, for SVM

	Neural Network (%)	k -NN (%)	SVM (%)	RVM (%)
Beams	86	98	96	95
Flat	88	99	100	96
Natural	92	98	99	96
Note	80	97	96	89
NoteFlag	62	90	94	81
NoteOpen	86	74	95	95
RestI	82	99	98	98
RestII	92	100	100	100
Sharp	88	99	98	96
TrebleClef	73	93	99	93
Unknown	50	76	81	62
99% CI for the Expected performance in percentage:				
	[80; 82]	[93; 95]	[95; 96]	[89; 91]

Table B.8: Accuracy obtained with a database of real and printed scores. Input data: vector of 400 binary values.

	Neural Network (%)	k -NN (%)	SVM (%)	RVM (%)
Beams	90	97	96	94
Flat	90	99	98	97
Natural	93	99	99	98
Note	83	96	97	89
NoteFlag	69	90	94	85
NoteOpen	85	95	95	92
RestI	84	100	100	98
RestII	90	100	100	98
Sharp	90	98	98	97
TrebleClef	79	96	96	91
Unknown	42	70	81	63
99% CI for the Expected performance in percentage:				
	[80; 83]	[94; 95]	[95; 96]	[90; 92]

Table B.9: Accuracy obtained with a database of real and printed scores. Input data: vector of 400 binary values plus 7 music symbol features.

	SVM (%)	Neural Network (%)	k -NN (%)
Handwritten Symbols	[56; 71]	[50; 59]	[62; 66]
Printed Symbols	[86; 91]	[77; 81]	[91; 93]

Table B.10: Accuracy on the 99% CI for the expected performance in percentage for separation of types of music scores. Input data: vector of 400 binary values.

classifier C and γ values were obtained based on a grid search. In LMNN algorithm the same grid search was also conducted in order to obtain the optimal value of the nearest similar labeled vectors. One more time, a confidence interval was estimated for the mean of the error of the model on the test set as in the previous experiments.

		SVM (%)	RVM (%)	Neural Network (%)	k -NN (%)
White Speckles	0.03	[89; 91]	[82; 87]	[80; 85]	[87; 90]
	0.05	[82; 87]	[76; 81]	[73; 77]	[80; 84]
	0.07	[77; 84]	[70; 75]	[63; 70]	[68; 72]
	0.09	[69; 77]	[61; 67]	[57; 62]	[56; 61]
Curvature	0.02	[94; 96]	[88; 91]	[89; 93]	[95; 96]
	0.04	[92; 95]	[83; 87]	[87; 90]	[92; 94]
	0.06	[94; 96]	[87; 90]	[87; 91]	[93; 96]
	0.08	[93; 96]	[83; 91]	[86; 89]	[91; 94]
Rotation	3	[97; 99]	[93; 96]	[93; 96]	[96; 98]
	4	[98; 100]	[95; 98]	[95; 97]	[97; 99]
	5	[99; 100]	[95; 100]	[97; 99]	[98; 100]
	-3	[98; 99]	[94; 97]	[95; 97]	[97; 99]
	-4	[97; 98]	[94; 97]	[94; 96]	[95; 96]
	-5	[96; 100]	[93; 98]	[95; 96]	[95; 97]
Kanungo	0.25	[94; 95]	[84; 96]	[88; 91]	[95; 97]
	0.5	[93; 95]	[87; 95]	[89; 91]	[95; 97]
	0.75	[95; 97]	[88; 93]	[89; 92]	[94; 96]
	1	[95; 97]	[89; 93]	[90; 93]	[94; 96]
	1.25	[94; 97]	[88; 92]	[90; 92]	[94; 95]
	1.5	[95; 97]	[90; 94]	[91; 95]	[94; 96]

Table B.11: Effect of different deformations on the 99% CI for the expected performance in percentage. Input data: vector of 400 binary values.

Music Symbols	>	9	≡	♭	♮	♩	♯	♩	⋮	γ
	Accent	BassClef	Beam	Flat	Natural	Note	NoteFlag	NoteOpen	RestI	RestII
	#	♯	♯	♮	♮	^	⏏	○	■	↓
	Sharp	TimeN	TrebleClef	TimeL	AltoClef	Relation	Breve	Semibreve	Dots	Barlines

Table B.12: Full set of handwritten and printed music symbols considered.

Results from the third experiment

The different classifiers were tested using different sets of features extracted: 7 (features presented in section 8.1.2), 16 (BSM descriptor), and 23 (7+16) features. The vector of 400 binary values was also tested but was discarded due to its less satisfying performance. Tables B.13 and B.14 present the results obtained applying LMNN and SVMs classifiers in the OMR database, respectively.

The accuracy rates were compared using Euclidean and Mahalanobis distances. According with our expectations using information about the metric improved the results of prediction error. The first assessment is that LMNN achieved the best results, where the highest gain was obtained using 23 features with LMNN against k -NN. Moreover, within the SVM classifier, an overall improvement on using the metric learning on SVM can be stated with exception for the set using 23 features. Even though a continuous improvement is verified by consistently adding new features when using dRBF, one can assess that the overall performance is higher when using 23 features with the standard RBF, performing a cross validation on k in dRBF (due to time constraints) could be the reason behind this behavior. Furthermore, since we first optimize a distance during the metric learning phase that will be

	<i>k</i> -NN			LMNN		
	7	16	23	7	16	23
Accent	63%	73%	71%	66%	73%	77%
AltoClef	80%	95%	86%	81%	94%	88%
Barlines	77%	50%	84%	78%	50%	88%
BassClef	70%	89%	75%	72%	88%	85%
Beams	78%	75%	85%	78%	73%	87%
Breve	100%	100%	100%	100%	100%	100%
Dots	94%	94%	94%	94%	94%	96%
Flat	77%	92%	83%	79%	92%	87%
Naturals	82%	92%	86%	83%	93%	91%
Notes	66%	85%	73%	68%	83%	78%
NotesFlags	46%	83%	49%	76%	80%	59%
NoteOpen	71%	92%	76%	75%	92%	81%
Relation	66%	74%	71%	70%	74%	77%
RestsI	67%	84%	73%	71%	86%	78%
RestsII	70%	73%	75%	71%	72%	80%
Semibreve	71%	71%	73%	74%	71%	77%
Sharps	84%	86%	88%	85%	86%	93%
TimeL	57%	81%	69%	60%	81%	79%
TimeN	49%	71%	56%	52%	71%	65%
TrebleClef	81%	97%	82%	83%	96%	87%
99% CI for the Expected performance in percentage:	[72; 73]	[80; 82]	[77; 78]	[73; 74]	[80; 82]	[82; 83]

Table B.13: Accuracy obtained using the *k*-NN and LMNN classifiers.

used to construct the kernel SVM, resulting thus in a similarity kernel, this transformation could also produce performance losses.

	SVM (RBF)			SVM (dRBF, $k = 1$)		
	7	16	23	7	16	23
Accent	51%	54%	79%	67%	63%	72%
AltoClef	79%	77%	92%	84%	89%	90%
Barlines	79%	70%	90%	79%	81%	87%
BassClef	69%	78%	82%	74%	86%	80%
Beams	82%	73%	89%	81%	74%	87%
Breve	100%	0%	100%	100%	100%	100%
Dots	95%	94%	98%	95%	93%	97%
Flat	60%	90%	90%	75%	90%	86%
Naturals	76%	90%	93%	81%	88%	86%
Notes	63%	67%	81%	67%	77%	76%
NotesFlags	40%	52%	63%	46%	74%	60%
NotesOpen	59%	57%	80%	67%	85%	75%
Relation	67%	50%	78%	65%	71%	70%
RestsI	52%	73%	74%	65%	82%	70%
RestsII	76%	65%	88%	71%	68%	78%
Semibreve	65%	0%	73%	69%	65%	79%
Sharps	87%	68%	93%	82%	81%	88%
TimeL	57%	67%	84%	58%	67%	70%
TimeN	63%	64%	75%	54%	70%	69%
TrebleClef	87%	87%	91%	82%	87%	87%
99% CI for the Expected performance in percentage:	[70; 71]	[68; 69]	[85; 85]	[72; 73]	[78; 79]	[79; 81]

Table B.14: Accuracy obtained using the SVM classifier with different kernels.