Optical Music Recognition System Capable of Interpreting Brass Symbols

Lisa Neale

BSc Computer Science Major with Music Minor
2005/2006

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)_____

Summary

The process of Optical Music Recognition (OMR) involves taking a piece of sheet music as an input, recognising the musical features, representing the recognised features on screen, outputting the recognised features in an interoperable output format and evaluating the recognition performance.

Currently there are over 100 available OMR software applications, including PhotoScore, SmartScore, SharpEye and Capella-Scan, none of which support 100% recognition of musical features. This is due to the extensive range of music notation used throughout the world. Common Music Notation (CMN) is widely supported by commercially available applications. However, support for other notations, Chinese and Japanese notation, for example, is limited.

This project investigates OMR, with a specific focus on brass symbols represented above the stave. Since the process of OMR can be split into five main stages: input, recognition, representation, output and evaluation, the project follows this structure.

The input stage is presented in a chapter on digitisation standards and input devices, giving an insight in data capture and the management of small scale digitisation projects. The recognition stage is broken down further into four additional phases: staff line identification, musical object location, symbol identification and musical semantics, which are explored in the background research chapter.

The representation and output stages are combined together and presented in both the symbolic music representation chapter, recommending a suitable interoperable representation for brass symbols, and the chapter discussing the design and implementation of the software prototype.

The evaluation stage is presented firstly in the evaluation of current OMR software. This chapter presents an evaluation of four commercially available OMR software applications concentrating on their recognition performance of brass symbols above the stave. The evaluation is later continued in the results analysis and evaluation chapter, which presents an evaluation of the performance of the software prototype upon recognition of individually isolated brass symbols. It also presents a comparison of the software prototype's recognition performance to the commercially available OMR software, focusing on the recognition of sheet music.

Acknowledgements

I would like to thank my project supervisor, Dr. Kia Ng, for suggesting the idea of this project and for his continued help and advice throughout it and also my assessor, Professor Roger Boyle, for his feedback and advice on my mid-project report and the progress meeting.

I would also like to thank everyone else who has helped me along the way, especially my proof readers, Sarah and Ben.

Contents

Chapter 1

Introduction

1.1 Context

The process of Optical Music Recognition (OMR) is presented in five stages: input, recognition, representation, output of musical features and an evaluation of the recognition performance. Since commercially available software applications are not capable of 100% recognition of musical features, due to the diversity of symbols used by Common Music Notation (CMN), this project focuses on the recognition of brass symbols represented above the stave.

1.2 Aim

The aim of this project is to develop an additional feature of an OMR system that is capable of interpreting brass music (performance symbols and directions) written in CMN form. This additional feature will be able to use image processing techniques to recognise specified notation and symbols. In order to do this, further investigation into existing OMR system capabilities, symbolic music representations and digitisation standards will need to be undertaken. In addition, further exploration into the methods of data capture, with particular focus on digital camera and flatbed scanner capture will be made in order to increase the versatility of the data capture for the software prototype.

1.3 Objectives

In order to achieve the aims of the project the following objectives must be fulfilled:

- A survey of currently available OMR software applications.
- A survey of symbolic music representations, with particular focus on expressive symbols for brass music.
- Implementation of a software prototype and evaluation of its recognition performance.
- A survey and report on different methods of optical document acquisition.

1.4 Minimum Requirements

In order to achieve the aims of the project the following minimum requirements have been set:

- Create a small ground truth data set for OMR.
- Survey of at least 3 currently available OMR systems.
- Report on at least 3 symbolic music representations.
- Software prototype to recognise several main features of brass music, e.g. mouth piece pops, open and close mute directions.
- Report on digitisation standards and input devices.

1.4.1 Possible Enhancements

On completion of the minimum requirements the following possible enhancements have been identified:

- Recognition of additional symbols including close-open and open-close mute symbols and inhale air from instrument symbol.
- Extension of survey of commercial OMR software.
- Extension of report on symbolic music representations.
- Research into current digitisation projects not specific to OMR.
- Investigate other methods of data capture.

1.5 Deliverables

Upon accomplishment of the minimum requirements the following deliverables will be made:

- Database containing images of ground truth data set.
- Survey of 3 commercial OMR systems.
- Report on 3 symbolic music representations.
- Software prototype.
- Report on digitisation standards and input devices.

1.6 Report Structure

This report consists of seven chapters. This first chapter introduces the context of the problem, outlining the aims of the project and how they will be successfully fulfilled. Chapter 2 discusses the recognition stage of the OMR process through background research into stave line identification, musical object location, symbol identification, musical semantics and the different approaches to OMR. Chapter 3 explores the presentation and output stage of the process through discussion of symbolic music representations, including a comparative study, thus recommending a suitable representation for brass symbols. The input stage of the process is presented in Chapter 4 through research into digitisation standards, input devices and suitable file formats. Chapter 5 presents a comparative study of commercially available software, with research into evaluation techniques for evaluating commercial software. This forms part of the evaluation stage of the OMR process. The representation and output stages are presented further in Chapter 6, which discusses the design and implementation of the software prototype. The evaluation stage is explored further in Chapter 7, which presents the results analysis and project evaluation. Firstly by discussing the software prototype's ability at recognising individually isolated symbols and secondly by comparing its recognition performance to commercially available software when reading a sheet of music. Chapter 8 presents the project conclusion and opportunities for further work. Also included are appendices presenting the project reflections, a glossary of musical terms discussed in the project, further information on symbolic music representations and digitisation standards, evaluation tables supporting the commercial OMR software comparison, an example of horizontal and vertical projections for each symbol, graphs demonstrating the feature vector clusters and project management time plans.

Chapter 2

Background Research

2.1 Context

In order to implement the recognition stage of the OMR process, concentrated background research covering the recognition process and different approaches to OMR needs to be undertaken. It is useful to firstly address the terminology used throughout this chapter.

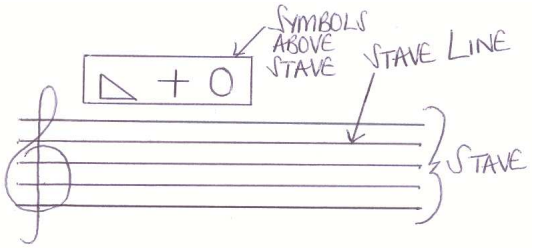


Figure 2.1: Stave, stave line and symbol definitions

Figure 2.1, defines what is meant by the terms stave, stave line and symbols above the stave. The term stave can be interchangeably used with the word staff, in the context of this project, stave will be used. Also in this context the x-coordinate refers to the horizontal axis and the y-coordinate refers to the vertical axis.

2.2 OMR

Bainbridge [2] describes the reading of music as "the visual recognition of graphical shapes and the application of musical knowledge to derive its meaning." This is an important statement when establishing the type of system to be developed and implemented. A computer paradigm that models this structure would be a vision system connected to a knowledge base.

The process of OMR is related in some ways to OCR, however, music notation is represented in a two dimensional format. While text is one dimensional, music has a vertical layout corresponding to pitch and a horizontal layout corresponding to time. OMR is a form of structured document analysis "where symbols overlaid on the conventional five line stave are isolated and identified [4]." The most significant difficulty with OMR is that CMN is a rich language, which is difficult to anticipate, as there are so many symbols that could be encountered.

As described by Bellini and Nesi [11], the recognition of note symbols can be split up into a layered "working memory" representation, as shown in Figure 2.2 [41]. It begins at the bottom layer of the memory, the "image layer," which is where the recognition is performed. The "primitive layer" is where the primitives which construct notes, rests and dots are described. The "symbol layer" is where notes and rests are synthesised from the combination of some primitives. The "meaning layer" describes the meaning of each symbol, including the pitch and beat, where rests are regarded as notes without pitch. The final layer, the "goal layer," time sequentially connects all of the hypotheses in the meaning layer, based on musical knowledge.

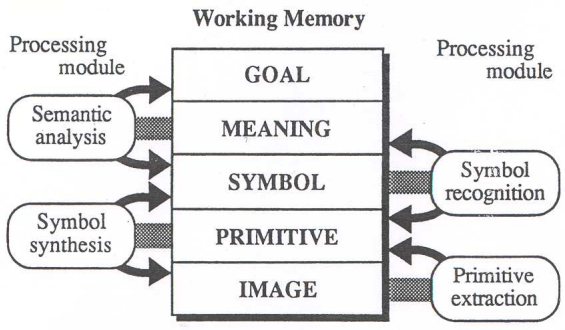


Figure 2.2: Layered Working Memory [41]

This model forms the basis and direction for further research as it provides focus on the four main stages in the OMR process. These are as follows:

- Stave line identification.
- Musical object location.
- Symbol identification.
- Musical semantics.

2.2.1 Stave Line Identification

Blostein and Baird [15], state that "stave lines play a central role in music notation." Keeping this statement in mind, Rossant [60] recognises ways in which the stave line, and its associated information, can be utilised within the OMR system. The stave line spacing, in pixel units, can be used to express the scale of the score and later used to normalise distances and sizes of objects within the score. This means that the pitch of the note can be determined from the vertical position of its head, relative to the stave lines. The stave line spacing can be computed by using the horizontal projection profile to pinpoint the stave line and determine the average stave spacing. The inclination of the stave lines allow the skew of the page to be calculated so it can be corrected if the input image is not exactly straight.

Rossant [60] also claims that stave lines can disturb recognition, because they connect and intersect with most of the musical symbols. She continues by suggesting the use of a segmentation algorithm to successfully remove the stave lines from the image. The segmentation algorithm begins by removing all of the stave line segments that do not overlap with a symbol. The second stage concerns all of the musical symbols featured by a vertical segment longer than 1.5 times the stave spacing. The longest vertical segment (longer than 1.5 times the stave spacing) is detected in each column of the stave area. These segments may belong to a vertical line or may be noise. An analysis window is then moved along the x-coordinates to analyse adjacent segments, which identifies whether the vertical segment is a primitive or noise. The main element of each line is used as a seed of a region growing and image labelling algorithm, so that the musical symbol can be isolated and precisely located by a bounding box. Eventually the stave lines are removed and only symbols with bounding boxes are left.

Bainbridge and Bell [4] suggest the use of a horizontal projection of black pixels on each line of the image, which are stored and represented in a histogram. A peak on the histogram can then be determined as each stave line. This can be adapted to automatically detect the number of lines in an input by searching for regularly spaced lines. This method is also useful in identifying the skew of the page. If the page was not straight then a deterioration of the peaks within the histogram would occur. A

simple modification to compensate for this is to restrict the horizontal projections to short sections on the left and right hand sides of the page and then connect the corresponding detected lines. Identifying the stave lines in such a manner can be used to establish the angle of rotation required to correct the skew of the scanned image.

Carter and Bacon's [20] approach is to categorise all sections of the score which remain after noise removal as either stave line or non stave line. The first stage in finding stave line sections is to extract those sections which have a high aspect ratio, low curvature and are single connected. These sections which are relatively long and thin are termed filament. Other sections that have a high aspect ratio, such as crescendo and diminuendo markings are removed from the filament list. Initially vertical links are established between filaments that overlap horizontally. By finding the closest vertical spacing between each filament the stave line spacing is established. The stave-space height is also used to threshold the vertical links between the filaments. Each set of five filaments is stored as a stave fragment in a data structure. In order to locate those sections which can be categorised as stave line sections, interpolation between all pairs of stave fragments and extrapolation beyond those at the horizontal extremes of a stave must be completed. The interpolation procedure examines each linked pair of stave fragments and attempts to find a path between corresponding stave lines which constitute the two stave fragments.

A method of improving the accuracy of stave line identification was introduced by Bainbridge and Bell [5] in the form of a post processing step known as "wobble". "Wobble" works from left to right and attempts to follow a stave line by taking slithers that are one pixel wide. Post-processing techniques were also identified but would be more suited to the restoration of musical scores and would not be an important feature for identifying symbols.

Carter and Bacon [20], in 1988, suggested the use of a Line Adjacency Graph (LAG) to locate stave lines and musical objects. The image is analysed vertically and horizontally to search single vertical paths of black pixels, called segments. Graph nodes correspond to unions of adjacent and vertically overlapped segments, while arcs define the overlapping of different segments in an adjacent column. The graph is analysed to detect stave lines and the objects on them. The graph provides identification of empty areas of stave which can be used to detect and label the sections containing single music symbols or groups of music symbols. It can also identify staves with a rotation of up to 10 degrees and staves with stave lines that are not perfectly straight.

2.2.2 Musical Object Location

Prerau, in his thesis, as referred to by Blostein and Baird [15], suggested a Fragmentation and Assemblage method for treating stave lines and isolating music symbols. The fragmentation method scans along the edges of the stave lines to identify parts of the symbols lying between, above or below the stave lines and the assemblage method recombines the symbol fragments. Prerau also describes a method of recognising the height/width symbol-space. A subset of music symbols are recognised using simple measures and the relative symbol size is used for an initial classification. Around each symbol a bounding box is applied. The bounding box of each symbol is expressed in stave-space pixel units. The height and the width of the bounding box are used to look up a list of possible symbol matches. The list is a pre-computed look up table containing the areas where each symbol can occur expressed in height/width space.

Fujinaga, in his thesis, as referred to by Blostein and Baird [15], described a basic strategy to locate symbols using projections of syntactic knowledge, and then calculate local projections for detailed symbol classification. It uses a projection of the stave nucleus, which is defined as the area between the top and the bottom of the stave lines. The stave lines themselves give a background projection value in the x-projection and a symbol location is identified whenever the x-projection value exceeds the background value by one stave space. At this point a local y-projection is taken covering the full height of the staff, which is used to determine the vertical extent of the symbol; an x-projection is then taken using these bounds. The contribution to the stave lines is subtracted from the x-projection.

Most methods remove the stave lines from the image in order to locate the musical symbols but Bainbridge and Bell [5] present an approach where pixels on the stave line are only removed if there is no evidence that the object would become fragmented by its removal. The *De Facto* test first described by Clarke *et al* in [24], traverses the length of the stave line, using a template to determine whether there are any objects superimposed on the stave line at a particular point. This template can be used for examining the top lines of the stave and a mirror image can be used to examine the bottom half of the stave. This algorithm works by examining a vertical slither of stave line starting at pixel A, if pixel B is black and at least one of the pixels in its neighbourhood is black, then this is deemed as evidence of an object and the slither of the stave line remains.

Martin and Bellissant [13] build on the work of Clarke *et al*, but instead of using a template to decide whether an object passes through a particular point on a stave line, they use chords, which encompasses the symbol. (Chord in this instance, means a straight line joining the end of an irregular shape). The

chords are computed over the angle range $[-90^0, +90^0]$ and are inputted into a histogram based on the angle. A histogram with a single distinct peak lying at approximately 0^0 is caused by a point on the stave line that has no object passing through it. A histogram with two or more distinct peaks is the result of an object passing through it.

2.2.3 Symbol Identification

Bainbridge and Bell [6] presented that simpler geometric shapes that make up symbols are detected using pattern recognition techniques such as template matching, Hough transform and projections. Their features are then assembled into music features guided by musical knowledge. Once the primitives have been found, the challenge is to construct musical features from them. A number of suggestions have been made in order to achieve this including: Definite Clause Grammar (DCG), graph grammar, musical knowledge with constraints and decision tree classifiers.

A commonly used method of symbol analysis and identification is template matching. Template matching was adapted to OMR because it is not based on sophisticated segmentation algorithms and is sensitive in type-setting [60]. According to Bainbridge and Bell [4], a measure is made up of the similarity between an object being matched and each available template with the highest rating match being used, provided that it passes certain thresholds. There are drawbacks with this method in that the shapes of objects vary with publishers. Although bar lines, stems and note heads are similar, there are infinite varieties of beams and slurs. Also, this method is quite slow because each isolated object has to be matched with every template.

Rossant [60] uses a set of models to compute the correlation with the symbol that is to be identified. Although this method of symbol analysis may not be useful outside of type-set music sheets, it draws together some useful information which could be used in other methods. It identifies that symbols can be categorised into four main classes: notes, bar lines, accidentals and rests. A set of criteria is computed from the geometric features of the bounding box, the vertical segment and the object's relative position to the stave line. For each tested model the correlation score and the corresponding location are stored. These are used to compare with particular symbol templates already stored. The pitch and accidentals are deduced from the y-coordinate. This method does not identify performance symbols but there is the possibility that it could be adapted to include another main class if no other methods of symbol identification are deemed appropriate.

Bainbridge and Bell [4] explain some further methods of symbol identification, which may prove

useful for identifying performance symbols above the stave. Fujinaga [30] used projections to create a signature of an object and matched the signature with projections of previously identified objects. The Hough transform is used in image processing to locate straight lines and curves, and has proved useful for finding bar lines, stems and the curl of the bass clef. The pattern recognition technique of slicing involves taking a cross section through an object at a predetermined position, and counting the number of transitions from black to white pixels, which is particularly useful in the identification of accidental symbols. Like slicing, connectivity analysis is particularly suited to classifying musical features that are dynamic in their size within one piece of music, such as slurs. Connectivity analysis answers the question, 'is there a path through the object from (x_1, y_1) to (x_2, y_2) where $(x_1 < x_2)$ with x monotonically increasing?' This is particularly useful for detecting beams and slurs.

Musical symbols are often split up into their primitives, which are the most basic parts of a musical symbol. For example, a note can be split up into a stem and a note head. In order to assemble the object together again, Bainbridge and Bell [4] have identified the need for a grammar based approach, as this supports the idea of generality within the system. In the Cantor system, devised by Bainbridge and Bell, DCG is used with the replacement of the one dimensional list of tokens for a high order data structure. Tree grammars and graph grammars are used and they have been successfully employed in two dimensional document image analysis. There is one drawback with this approach, in that the increased generality increases the complexity of the parser.

Kato and Inokuchi [41] present an idea specific to recognition of piano scores, although the fundamental ideas of the paper remain the same for the recognition of any music in CMN. The recognition of a key signature consists of some sharps, flats or naturals and the time signature consists of two numbers. The particular combinations of them are defined by musical knowledge, which ensures that what is recognised is viable. The recognition of notes is the most fundamental way to describe sound and the segmentation of them is not easy. There are no constraints on the size of the symbols, and notes often have connections, overlaps and containments with other symbols. There is also the problem of symbol fragmentation when stave lines are removed, and the semantic relationship between symbols is often described ambiguously. The note heads are detected by correlating the patterning of black head and the two patterns of white heads. The size of these patterns is determined by the spacing of the stave lines used. The symbols at the left end of the staves (time signature, key signature and clef sign) are detected and discriminated. Note heads are discriminated against by checking the four types of tail and looking for three other attributes: dots, tenuto and staccato marks.

The Andronico and Ciampa's [1] method uses high level grammar to describe the organisation of music in terms of music symbols. These grammars are then used for the music symbol recognition. High level grammar is strict and very simple, describing a piece of music as a sequence of staves.

The Kato and Inokuchi [41] method describes a top-down approach designed to handle simple and complex notation. Simple, monophonic notations can be interpreted uniquely by means of simple rules. Complex notations have higher symbol density, with more connections, overlaps and complicated placement of symbols. It is difficult to devise a single method for recognising all symbols. It is instead necessary to use a collection of processing modules that communicate by operating on a common memory. The working memory represents information about the current bar of music at five levels of abstraction. The processing modules use knowledge about music notation to contain the pattern processing task.

Both Lee and Choi [47] and Fujinaga, in his thesis, referred to by Blostein and Baird [15], use a form of projection methods. Lee and Choi's method implemented a micro computer based music recognition system where x- and y-projections are used to find the bar lines. Notes are recognised using the x- and y-projections from a small window around the symbol. Fujinaga's process uses the results of the musical object location x- and y-projections in conjunction with musical syntactical knowledge for symbol classification. The symbol recognition is completed using the features extracted from the projection profile and their first and second derivatives.

Mahoney [49], established that pattern primitives can be combined to form composite music symbols. The distinction between pattern primitives and composite symbols subdivides and simplifies the recognition task. The pattern primitives used are lines, dots and characters and the primitive lines accommodate the variable parameters of the composite symbols. A parameterised composite symbol such as a beamed note sequence is made up of unparameterised characters and parameterised lines. Knowledge of the structure of standard music notation is needed so the musical symbol can be deduced from the relationships between the various kinds of primitives.

Clarke's [25] idea examines the neighbourhood of each pixel to determine whether a musical symbol intersects the stave line at the particular pixel. An initial classification is obtained from the symbol height and width (as per Prerau's system). Further classification is performed by examining the pixels in a few particular rows and columns of the symbol image.

2.2.4 Musical Semantics

The idea of applying the musical semantics is to translate the graphically assembled shapes into their musical meaning [3]. Musical semantics is poorly documented compared to the other phases, as music semantics are specific to a piece of music and are often missed out from literature. This phase consists of the multiple passes of a graph like data structure applying the effect of one class of musical features on other groups of musical features i.e. key signatures on notes. It is important to note that this aspect of the system must be dynamic. The dynamic nature ensures that the semantic information can be updated if and when necessary.

The way in which the knowledge is applied enables various interpretations of the music which has been recognised [41]. Also the processing strategy which is adopted affects the way the processing modules are driven in variable thresholds and it is based on how the condition of the hypotheses in a working memory are adopted.

The result of this stage is a graph mapping the two dimensional relationship onto the assembled music features, storing information such as pitch and duration as attributes and forming links between them. Once this graph is established, it must then be traversed to generate the desired music format [7].

2.3 Different Approaches to OMR

OMR has been an active research study since the early 1960s. In 1970 Prerau, in his thesis, as referred to by Blostein and Baird [15], introduced the concept of music image segmentation to detect primitive elements of music notations with the use of fragmentation and assembling methods to identify the stave lines, isolate fragments of notation and to rebuild music symbols. Each symbol was encapsulated by a bounding box and he believed that the height and width of the symbol are sufficient features for the identification of symbols. The symbols are classified by comparing the dimensions of its bounding box with a look up table where topologic features of each symbol are stored.

In 1988, Carter [21] devoted his efforts to the coordination of existing research with the purpose of defining some reference standard. He proposed a method based on the Line Adjacency Graph (LAG) to identify the empty areas of the stave line and to isolate the symbols. In keeping with Prerau the symbols are classified according to the size of the bounding box and the number and organisation of their constituent sections.

Also in 1988, Fujinaga [30], in collaboration with Alphonce and Pennycook, developed an approach

based on an intensive use of projection methods. It was concluded that it was only necessary to identify the position of the stave and it was deemed unnecessary to remove it. The identification of symbols and their features are gathered by a sequence of projections, firstly an approximation and then a more detailed and accurate attempt is made. Music syntax knowledge is associated with the analysis of projections both in the identification and classification of symbols. A pure syntactic approach, where context is not considered, has limitations. In order to solve these limitations, it was identified that music notation can be formalised by a means of a context free and $LL(k)$ grammar. A stave is identified by analysing the y-projection of the image. A group of five peaks, giving a consistent contribution, mark the stave presence on a histogram. Symbols are identified using the x-projection values that are greater than the background noise caused by the stave lines. As support for the detection of symbols, some syntactic rules, mainly based on the position of the notation, are used. The symbol classification is performed by calculating both the x-projection and y-projection for each symbol.

Later, in 1997, Fujinaga [31], also started work on Adaptive Optical Music Recognition. The adaptive feature allows different copies of the system to evolve in different ways since each of them is influenced by the experience of different users. The system consists of a database and three inter-dependent processes encompassed by a recogniser. The recogniser detects, extracts and classifies music symbols using a k-Nearest Neighbour (k-NN) classifier. The recognition is based on learning examples.

In 1994, Roth [61] introduced a totally graphic approach to remove both horizontal and vertical lines. This helped to solve problems of objects touching each other and broken symbols caused by stave line removal, although the image has to be manually rotated to correct the skew angle. Following this stage, the statistical analysis of the vertical paths are examined. The average length of vertical paths for black and white pixels is used to find and fix the thickness of stave lines and the white spaces between two lines. Similar to Fujinaga's approach, the stave lines can then be identified by searching for groups of five peaks in the y-projection profile. The stave lines are then removed by erasing lines that have a thickness less than the calculated values in the statistical analysis stage of the process. Vertical lines are detected and removed through using the x-projection profile and morphologic operations. Objects are labelled on the basis of graphic features and the symbols are recognised using information provided by labelling steps and graphic rules. The information is saved for later use. Although the first version of this system produced modest results and only worked with a limited set of symbols it was later improved to introduce a feedback semantic control. The feedback can be used to correct results and establish zones which are not correctly recognised.

Ng and Boyle [56], in 1994, developed an Automatic Music Score Recogniser with the input, an image in bitmap format and an output of a standard MIDI file. The pre-processing system consisted of a list of automated processes; thresholding, deskewing and stave line detection. The sub-segmentation process divided the composite music symbols into lower level graphical primitives. In the classification process, the primitives are classified using the k-NN classifier based on simple features such as the aspect ratio and normalised width and height in relation to the stave line spacing.

In 1995, Cousnon and Camillerapp [26] stressed the importance of musical knowledge in the recognition process. They stated that information from knowledge should be used to control the whole OMR process. Music should be formalised by defining a grammar and the segmentation and labelling phase are controlled by the set up of the grammar. The segmentation and the labelling of primitives are driven by the music knowledge to control the correspondence of the detected graphic elements. The reconstruction and classification of music symbols is immediate since the consistency and sequence of primitives have already been tested.

Bainbridge [3] [4] focuses on the problem of automatic comprehension of music with particular attention to the human cognitive process. In the natural process of music comprehension, two main phases can be detected: the recognition of graphic shapes and the application of music knowledge to deduce the meaning. An optimal OMR system should be constituted by a drawing package, where the elementary graphic shapes defining the music symbology are described together with a specially designed music language, providing a structure to describe the abstract knowledge of music notation. Both modules should have properties of flexibility and dynamism so new music notations or new interpretations of symbols can be fitted in with ease. A special language, Primela, was incorporated to allow definitions of acceptable configurations of primitive music symbols and to represent the music semantic. A Primela description is written for each type of primitive music shape to be recognised along with the pattern recognition routine most appropriate for the task.

Currently under development by Bellini, Bruno and Nesi is the O^3MR (Object Oriented Optical Music Recognition) System. The general architecture is split up into four components. The first stage, segmentation, is the most critical as the sheet music is processed with the aim of extracting basic symbols and their positions. The second stage, basic symbol recognition, performs recognition of basic symbols using a MLP (Multi Layer Perceptions) back-propagation neural network. The third, music notation reconstruction, maps the recognised basic symbols into the elementary components of music notation symbols. The final component, music notation model, is reconstructed by refining the recogni-

tion performed in the previous component [11].

2.4 Summary

Since this project is limited to the recognition of brass symbols, with a focus on brass symbols above the stave line, there is no need to implement a segmentation algorithm, as suggested by Rossant, to remove them. However the use of an image labelling algorithm to isolate the symbols and the formation of a bounding box around a symbol would be ideal for the prototype. The isolation of a symbol could also be completed using Clarke's idea of pixel neighbourhood examination. This would ensure that there are no other neighbours and support the image labelling algorithm.

Bainbridge and Bell [4] suggested the use of horizontal projections to detect stave lines. This approach could be extended within the prototype to produce horizontal and vertical projections for each isolated symbol, which is similar to Fujinaga's proposal of using local projections for detailed symbol classification.

Fujinaga presented an approach where symbol recognition is completed by use of features extracted from the projection profile which could be implemented in the prototype. This is supported by Prerau who utilises the width and the height of the symbol as an initial classification.

The k-NN classifier, as used by Ng and Boyle and Fujinaga is a recognition tool based on learning examples. This technique is apt for this project as the ground truth data set can be split into a training data set and a test data set, thus successfully training the classifier and testing it with a different set of test data.

As previously mentioned, there are over 100 OMR software applications. Current software applications including, PhotoScore and SmartScore, boast a recognition rate of nearly 100%, although they are limited to interpreting CMN and guitar tablature. Both applications only provide recognition of basic CMN features such as notes, bar lines, clef signs and accidentals. Very few instrument specific symbols can be recognised without training the system.

Chapter 3

Symbolic Music Representations

3.1 Introduction

"A symbolic music representation is a logical structure based on symbolic elements representing audio visual events, the relationship between those events and aspects related to how those events can be rendered [12]."

Symbolic music representations are important in the context of OMR because they provide a method of representing the music symbols and notation. It is a common misconception that MIDI can be used as a symbolic music representation when in fact it has no representational capabilities at all. For further information on MIDI see Appendix C.

Symbols and their syntactic arrangement have a functional meaning within the language so they can be interpreted [59]. The main problem with symbolic music representations is that the organisation, and subsequently the interpretation of the symbols and notation have never been formalised.

Presently, there is no open standard used exclusively by the currently available commercial OMR software, therefore there is no mechanism ensuring interoperability and persistency. The two leading companies for music representation software, Sibelius and Finale, have both failed to implement an interchange format as their default. They have instead both opted to design their own, in a binary format, which is undocumented [9]. According to this source the best export formats currently available

are NIFF and MusicXML.

This chapter addresses the issue of symbolic music representations and the various formats which have been suggested. Currently, several XML compliant mark up languages have been proposed including MusicXML and WEDELMUSIC. There has also been extensive research into the use of MPEG for representing symbolic music representations. It is necessary to include previous attempts of symbolic music representation formalisations, including Notation Interchange File Format (NIFF), to understand the reasons why it was unsuccessful. There is additional information regarding previous symbolic music representation attempts included in Appendix C.

3.1.1 Notation Interchange File Format (NIFF)

NIFF was a file format designed to allow the interchange of music notation data between music recognition programs. It used a tree data structure to store the music notation information, implementing chunks, which formed the basic structure of the NIFF file. The chunk structure is an extension of RIFF, the Microsoft Resource Interchange File Format developed for Microsoft Windows. There were two types of chunk; the setup chunk which defined the context of the description and the symbol chunk which described the musical score [46]. The file format was structured in such a way as to maximise its extensibility and its flexibility [68]. Its structure could fully accommodate music publishing systems, simpler music displays, logical definition languages (e.g. DARMS, as discussed in Appendix C) and music scanning programs. It allowed representation of the most common situations occurring in conventional music notation, while making provision for software developers to define their own extensions to handle more unusual situations [34].

This standard was never adopted because the software publishers produced their own file formats. PhotoScore, SharpEye and SmartScore can export NIFF files. Sibelius could import from NIFF but in current versions this is not supported due to its lack of demand.

Gerd Castan initiated an implementation of NIFFML, which was an XML implementation of NIFF. The conversion, however, was not true to the original, as the NIFF file format was implemented as a tree structure which contains children and XML descriptions have no children. NIFFML, on the other hand, provided coded examples which could be discussed. It also provided a formal way to describe and validate data, as it could be modelled through use of a Document Type Definition (DTD) or schema. There was value in steering NIFF data towards an XML implementation because XML adds external schemas and validation which improves consistency [22]. As of 26th February 2006, according to the

NIFF coordinator Alan Belkin, the NIFF file format has been superseded by MusicXML [10].

3.1.2 MusicXML

XML is a subset of the standard generalised mark-up language (SGML) designed to simplify the process of structured document interchange over the Internet. The first version was designed in 1998 by W3C and it was defined as a system neutral Internet format for representing structured data. Data hierarchies are as fundamental to XML syntax as they are to musical notation [23].

MusicXML was intended to represent Common Western Music Notation from the 17th century onwards to support the interchange between musical notation, performance, analysis and retrieval applications and to support the sharing of musical data between applications. MusicXML was not designed to represent presentation concepts such as pages and systems because in the XML environment, formatting is handled separately from the structure and semantics [33]. The language is a meta representation of music information for describing and processing music within a multilayered environment and achieving integration among structural, score, MIDI and digital sound levels of representation [9].

3.1.3 WEDELMUSIC

WEDELMUSIC XML notation is an XML complaint format that presents multimedia capabilities by keeping the visual/formatting and symbolic aspects of the representation separate. It can be profitably used as a format for new applications and as a format for interchanging symbolic description of music scores. Digital music objects compliant with the WEDELMUSIC format are called WEDEL objects. Each WEDEL object presents sections about its identification, classification, protection, printing, symbolic music, image score, performance, documents, lyrics, audio, video and colour image. The section containing the symbolic music describes the scoring information, musical notation, symbols and their relationships with the formatting rules formalised by MILLA, the music notation automatic formatting language [71]. WEDELMUSIC format is still in its definitive stages but since it is based on MusicXML, which has become the most successful standard for symbolic music representation according to [63], there is a strong possibility that it will become an adopted symbolic music representation format. Currently it is only supported by WEDELMUSIC Editor.

3.1.4 MPEG SMR

MPEG SMR (Symbolic Music Representation) is currently under development and it is intended as a solution for a comprehensive representation of music information integrated with other media [12].

SMR will be integrated into MPEG-4 which introduces new coding tools over previous MPEG standards, including object coding, mesh coding, still picture coding and face and body animation [70]. For the integration of SMR into MPEG the still image, or still picture coding tool will need to be utilised using wavelets or the discrete cosine transform (DCT).

SMR will be integrated into MPEG-4 by:

- Defining an XML format for text based SMR that will allow interoperability with other SMR and notation formats, and act as a source for the production of equivalent binary information stored in files or streamed by a suitable transport layer.
- Adding a new object type for delivering a binary stream containing SMR and synchronisation information, and providing an associated decoder that will let users manage the music notation model and add the necessary musical intelligence to allow human interaction.
- Specifying the interface and behaviour for the SMR decoder and its relationship with the MPEG-4 synchronisation and interaction layer.

At present, the standardised MPEG tools do not systematically support SMR. The structured audio section of the standard allows synchronisation to a standard MIDI format. It also allows structured descriptions of the audio content to be saved through a normative algorithmic description language, which is associated with a score language, providing more flexibility than the MIDI protocol. It is possible to extract SMRs from the information carried by the structured audio tools, including MIDI, but it cannot be guaranteed that the notation coding is correct due to a lack of information about the visual and graphic aspects, symbolic details, music notation modelling and necessary details for a correct human machine interaction through the SMR decoder [12].

3.1.5 GUIDO

GUIDO was designed to be a general purpose, formal language for representing score level music in a platform independent, plain text and human readable format with the aim to providing an adequate, universal representation, non specific to a particular application [38]. ASCII codes are used to represent

specific symbols or notation. It concentrates on general musical concepts, as opposed to simply notation. It was designed as a flexible and easily extensible open standard with syntax which does not restrict the features which can be represented. It is easily adapted and customised to support specialised music concepts. The key feature is its representational adequacy, meaning that simple musical concepts should be represented in a simple format and only complex notions should require complex representations. As result of such a design, GUIDO is able to represent incomplete musical information, such as a scale without octave or duration information. Overall, its design is focused on music information, while also supporting the exact score formatting features.

GUIDO's design is based on three layers: basic, advanced and extended. The basic layer provides the relevant aspects for representing "simple" music, from single notes and rests to complete pieces, including dynamics and text. Although at this level it does not cover the exact placement of the graphical appearance of objects. The advanced layer extends the basic layer to support more advanced concepts such as glissandos and other symbols and addresses issues of score formatting. Within this layer it is possible to specify the exact score formatting information which could be later used by professional notation software to import the file. The extended layer is still being developed and it is thought that it will reflect the experience and needs of the developers [38].

3.2 Comparative Study of Symbolic Music Representations

From the research that has been completed there are two symbolic music representations which could be used to represent brass features, GUIDO and MusicXML. In order to determine the most appropriate format, a comparative study has been completed to assess the symbolic music representation's suitability to representing brass symbols, its interoperability, ease of encoding and its flexibility and ease in which a new symbol can be defined. Since no previous attempts have been made to compare symbolic music representations, the criteria which the representations have been evaluated against have been deemed the most apt for recommending a suitable symbolic music representation for representing brass symbols.

3.2.1 Suitability

The suitability of a symbolic music representation ascertains whether its design is suitable for representing brass symbols.

GUIDO was designed to be a general purpose formal language for representing score level music,

designed in three levels. The syntax does not restrict the features which can be represented and there are three layers which the symbol could be represented in. Therefore, there should be adequate provision facilities for representing brass symbols.

MusicXML has been designed to be an Internet-friendly method of publishing musical scores and an interchange format for notation, analysis, retrieval and performance applications. It has the ability to represent new symbols and from its design it provides adequate facilities for representing brass symbols.

3.2.2 Interoperability

The interoperability of the symbolic music representation assesses whether it is capable to be interpreted by commercial OMR software, whether it is able to be saved in such a format by commercial OMR software and whether there are any platform dependancies.

GUIDO has been designed to be platform independent. From the completed research, no information has been found to determine whether GUIDO can be interpreted by the commercial OMR software applications evaluated. Also there is no evidence to suggest that commercial OMR software supports saving in GUIDO representation. According to the GUIDO music notation website [36], there is a Sibelius to Guido plug-in, and Sibelius is supported by PhotoScore, but the support for the representation is not defined in the specification of PhotoScore.

MusicXML is an open format under a royalty-free license and is supported by all of the evaluated commercial OMR software applications plus an additional 55 applications [58]. It can be interpreted and saved into the format by all of the evaluated commercial OMR software and there are no platform dependancies.

3.2.3 Ease of Encoding

The ease of encoding is assessed through presentation of examples of code needed to encode a particular symbol. Since brass symbols are specific they would need to be defined as a new symbol, therefore the rest symbol has been chosen as an example.

The rest symbol is represented by GUIDO as:

```
_*length
```

where length is the number of beats the rest will take. This is included within the code used to represent the whole bar forming a string of data.

The rest symbol is represent by MusicXML as:

```
<rest ID=" " DURATION=" " HEIGHT=" ">
```

and each symbol is defined separately.

Each representation presents an easy method of encoding. GUIDO provides a shorter, more concise method with each section of music presented as a string of symbol representations. This could prove to be complicated to interpret since it is difficult to see where each symbol falls within the bar and which symbol is which. MusicXML defines each symbol separately, thus producing more code, but it is much easier to see exactly where the symbols falls in the bar and which symbol is presented by which piece of code.

3.2.4 Flexibility and New Symbol Definition

The flexibility and the definition of a new symbol assesses whether the symbolic music representation is limited to specific symbols, whether it is able to represent new symbols, the ease with which it can represent a new symbol and the code needed to define a new symbol.

The definition of GUIDO defines that it is an extensible open standard with syntax which does not restrict the features which can be represented. Therefore it defines that the representation is not limited to specific symbols and it has the ability to represent a new symbol. Thorough research has been completed and no example code or method of defining a new symbol can be found, therefore no comment can be made to describe the ease with which it can be represented and the code which is needed.

MusicXML is not limited to specific symbols and it has the ability to represent new symbols. Each new symbol is modelled using Scalable Vector Graphics (SVG). The SVG file is then referenced in the DTD of the file and the symbol is simply called in the XML file. For example, defining the close symbol. The shape of the close symbol would be defined in SVG and referenced in the DTD using close as its ID. The close symbol in MusicXML would be reference as:

```
<close ID=" " DURATION=" " HEIGHT=" ">
```

exactly the same as the rest, as described previously.

3.2.5 Summary of Comparative Study

From the comparative study it can be concluded that both representations are suitable for representing brass symbols. MusicXML is a representation supported by all of the evaluated commercial OMR software applications, plus an additional 55 applications, and there is no evidence to suggest that GUIDO is supported by any of them. Each representation presents an easy method of encoding. GUIDO provides a shorter, more concise method where each section of music is presented as a string of symbol representations. This could prove complicated to interpret since it is difficult to see where each symbol falls within the bar and which symbol is represented by which code. MusicXML defines each symbol separately, thus producing more lines of code, but it is much easier to see exactly where the symbol falls in the bar and which symbol is presented by which piece of code. The flexibility of MusicXML outweighs GUIDO. GUIDO specifies that it is easy to represent new symbols, but on completion of thorough research, no evidence or example code could be found to support this claim.

On completion of this study the most appropriate symbolic music representation to represent brass symbols is MusicXML since it is suitable, interoperable, flexible and new symbols can be defined with ease.

Although MusicXML is currently the most appropriate symbolic music representation to represent brass symbols, looking to the future MPEG SMR will provide the most comprehensive representation. MPEG SMR has the capability of representing and linking SMR with other functions such as video, sound and gestural information. For example with the representational information stored in MPEG SMR a copy of the score can be shown. This score could have interactive capabilities, meaning that with the click of a mouse on a note or symbol a video clip, sound byte or gestural information could be shown.

Chapter 4

Digitisation Standards and Input Devices

4.1 Introduction

Digitisation standards and input devices are important to this project and the field of OMR because they determine how the creation of a ground truth data set should be attempted and managed. The input device choice determines the quality of the images which is key to the image recognition stage of the project.

The formal definition of digitisation is the sampling of analog data, in the context of OMR the sampling of a piece of sheet music, by some type of scanning device, resulting in digital data. The digital data can then be repeatedly interpreted and represented in analog form by the computer. An image digitisation, even of a high quality, is nearly always inferior to its analog source, since sampling provides finite representations of inherently infinite source data [43].

Image digitisation carries many advantages in the field of OMR. The use of computer stored image technology aides the recording and preservation of old musical scores. Various digital medium can be used to store data for archival purposes and it is possible, once the music sheets are scanned, to implement a graphical database providing researchers with a tool for retrieval and structured query of the images [43]. Such a database could form the foundation of a repository of digital music sheets which could be used to house a much needed standard ground truth data set. Since there is no standardised

ground truth data set, it is essential, in the context of this project, that the most efficient image acquisition techniques, image resolution, storage method, input device and graphics file format are selected.

4.1.1 Image Acquisition

The most efficient way of achieving maximally accurate digitisation requires direct contact between the original source and the digital device. In the situation of OMR the image quality is critical, therefore it must be ensured that the image is as close to the source as possible, as the further away an image from its source, the less accurate the representation becomes [43].

4.1.2 Image Resolution

The resolution of an image is measured in dots per inch (dpi). The resolution of an image acquisition technique affects the quality and size of the scan file: the higher the resolution, the better the quality and the larger the digital image file. This results in high resolution images occupying more disk space but producing clearer close ups and low resolution images occupying less disk space but close ups appearing blurry [43].

4.1.3 Storage Methods

In any size digitisation project storage should be a concern at all phases within the digitisation chain. File storage space is needed for work in progress, backing up, local archiving and third party archiving of digitised data. Storage can be considered in two media; portable and non portable. For further information on storage media, see Appendix D.

4.1.3.1 Project Storage Needs

This project would be considered as a small scale digitisation project. Thus, storage only needs to be considered for work in progress, backing up and possibly local archiving of the complete digitised resource. For work in progress it is most suitable to use the network disk to store data and save a backup copy to a flash drive. Once the project has been completed it may be appropriate to save an archive of the digitised data on to the network disk.

4.2 Overview of Ongoing Projects in the Area of Digitisation

It is beneficial to study digitisation projects out of the context of music, as this provides background knowledge to the whole area of digitisation, which may prove useful and adaptable to the area of digitising music scores. Full details of the Minerva, PULMAN and AHDS projects are discussed in Appendix D.

4.2.1 Summary

All of the projects have a single focus, which is to provide resources to their audience in a standardised format. In order to successfully implement a digitisation project certain limitations need to be acknowledged, including: image acquisition techniques, image resolution and storage; which are sufficiently covered within this chapter. If a large scale music score digitisation project was undertaken it would be important to set a modest aim, like that of the Minerva project, and then increase the scale of the project and encapsulate a wider audience. There would need to be a representative body from each type of music notation world wide for a standardised ground truth data set to be implemented. Before this type of project can be undertaken on a large scale, it is imperative that music notation of any kind is standardised, therefore leaving no room for ambiguity of notation understanding.

In the context of this project it is essential that the correct image resolution and image acquisition techniques are decided upon, and the aims and objectives of the digitisation project are not too advantageous, as they can always be refined at a later stage.

4.3 Input Devices

4.3.1 Scanners

Scanners are the most common digital capture device and their purpose is to convert analogue images to digital image files. There are three main types of scanner; flatbed, slide and drum.

The flatbed scanner is most apposite to the scanning of printed papers and graphs. It uses a linear Charge Coupled Device (CCD) array made up of a long line of CCD elements in a row. The optical resolution of the device is limited to the number of elements (pixels) in the detector array. CCDs can only detect the presence of light, so for scanning in colour, three passes of the image needs to be made using different colour (red, green and blue) filters in front of the CCD. The typical resolution of this

scanner ranges from 200dpi to 1000dpi.

The film/slide scanner is used for digitising photographic negatives and would be of no use in the digitisation of musical sheets.

The drum scanner would also be of no use in this situation as it has a typical resolution in excess of 4000dpi, they are slow and very expensive to purchase.

In this situation a flatbed scanner would be the optimal choice, with a resolution of 300dpi. This would optimise the storage space. In order to speed up the scanning time a monochrome bitmap format could be used, as only one pass would need to be made by the CCD over the image [35], [44], [66].

4.3.2 Digital Cameras

The principle use of the digital camera is to capture real time events. It is possible that the use of a digital camera could be adapted to capture images that for some reason could not be borrowed or scanned, therefore a portable device would be the most useful.

Currently there are two main types of detector used in digital cameras. CCD detectors are the most common and should represent the standard for image capture for information based resources such as sheet music, since they are capable of producing high quality images. They work by creating analogue signals that are digitised off the chip.

Complementary Metal Oxide Semi Conductors (CMOS) are a cheaper alternative to CCD. As their performance improves, their use is likely to increase as they are offering a comparable quality at a cheaper cost, although, in some circumstances, their output is subject to noise issues. They produce images by creating a digital signal on the chip.

Along with detectors there are three main types of arrangements: area array, scanning linear array and scanning area array. The area array is the most common. Within the arrangement there are a fixed number of vertical and horizontal pixels and during image capture the whole array is exposed. They are generally small, robust and portable and come with a built in storage for image files. They are also very easy to use since they are similar in operation to 35mm film cameras. When image quality is a factor, a variation of the area array, 3-chip, could be utilised. These use three individual area array detectors, each one receiving either red, green or blue light. These additional components increase both the cost and the size.

Scanning linear array cameras can be considered similar to flatbed scanners, with a lens replacing the flat glass plate. The array is moved across the imaging plane by a stepper motor. Most linear arrays use

a tri-linear array (three rows of detectors), for simultaneous detection of red, green and blue. Resolution is limited by the number of cells along the array and, in the other direction, by stepper motor resolution. They are considered to be high resolution devices which lead to large image files so it is infeasible to store the images on the device and instead they are sent straight to the computer. This means that they are not portable devices and it is possible that the operator would need to be trained and therefore no use in the capture of sheet music.

Scanning area array uses a combination of the previous two cameras. An area array is moved in the horizontal and vertical planes to build up a potentially very high-resolution image. This enables very high resolution outputs which are not limited to the number of pixels that the linear or area arrays can be manufactured with. In comparison to the other cameras they have limited portability, storage and speed and require a trained, skilled operator to gain the best results [35], [44], [65].

If a digital camera is needed in the context of this project, a CCD with an area array detector would be the most appropriate and suitable for the project, for example a Kodak C340.

4.4 Image File Formats

After analysis of digitisation standards and input devices it is important to discuss file formats to represent the input image. The input image will be analysed by both the software prototype and the commercially available OMR software. Bitmap (BMP), Tag Image File Format (TIFF) and Graphics Interchange Format (GIF) will be discussed to determine the most appropriate file format for the prototype. For detailed information on each format see Appendix D.

4.4.1 File Format Selection

BMP, TIFF and GIF would all be valid file formats for an image of a musical score. At this stage, the most appropriate image file format would be BMP saved in a monochrome format as the image will be saved in its most pure form. The other two formats would need extra coding in order to extract the image information from the format, therefore resulting in extra programming time, which would not be an efficient use of time and resources.

Chapter 5

Evaluation of Current OMR Software

It is important to the success of the project to assess the recognition capabilities of currently available OMR software applications. This chapter will present methods of evaluating OMR software and an evaluation of four OMR software applications and their recognition performance of specified symbols.

5.1 Methods of Evaluating Commercial OMR Software

According to George [32] very few attempts have been made to review commercial music recognition software and most attempts to date have drawn upon informal opinion. As yet no comprehensive study has been carried out into the evaluation tools needed to perform such a study. It is important to acknowledge that in order for such a study to be successful the terminology for all music primitives has to be standardised. This problem is further hindered by the fact that there is no universally accepted standard for music notation and symbols. Also, for a non bias survey of OMR software, a representative and sufficiently large ground truth data set of music sheets has to be collected. Presently, there is no such standard ground truth data set in existence. Therefore, in order to sufficiently test and evaluate OMR software it is imperative that the same test data is used for each software application that is tested.

Due to the complexity of OMR applications a thorough assessment/evaluation matrix is required. A proposal by the Interactive Music Network includes a three level approach which assesses the primitive,

note and interpretation level of the software. It uses a set of rules and metrics to define what aspects have been considered in the evaluation [8] (see Table 5.1). This matrix incorporates an evaluation of all aspects of the software but it is possible to adapt it. Also incorporated into the proposal was a weighting to determine the importance of each criterion.

Categories	**Weight**	**Aim**
Note with pitch and duration	10	Evaluate note reconstruction correctness in terms of pitch and duration.
Rests	10	Evaluate recognition of rests.
Note with accidentals	7	Evaluate association of accidentals [...] with a note.
Groups of beamed notes	10	Evaluate capability in reconstructing beamed notes.
Time signature and time change	10	Evaluate capability in identifying and reconstructing the time indication [...].
Key signature and key signature change	10	Evaluate capability in identifying and reconstructing the key signature [...].
Symbols above or below the notes	5	Evaluate capability in identifying and linking ornaments, symbols and accents [...].
Slurs and bends	7	Evaluate reconstruction of horizontal symbols: slurs and bends.
Augmentation dots	10	Evaluate augmentation dots linking to notes.
Clefs	10	Evaluate capability in recognising clefs.
Irregular note groups	10	Evaluate capability in recognising tuplets.
Number of measures	10	Evaluate capability in recognising the bar line and the number of measures.
Number of staves	10	Evaluate capability in recognising staves.

Table 5.1: “List of symbols and relationship considered in performance” [8]

.

5.2 Evaluation Criteria and Conditions

For this study, the focus is based on the OMR software’s ability to identify symbols above or below the stave. Due to the nature of this project there is the possibility of extending the range of brass symbols

to include ones which are located on the stave. Therefore the software's ability to recognise accidentals associated with a particular note are also evaluated in anticipation of further work and research into this area. These two criteria will form the basis of the evaluation. In this study the criteria are of equal importance and therefore the incorporation of a weighting value is not applicable. As there is not a standard ground truth data set which can be used for testing, it is essential that an adequately large data set is determined and used on each software application, which can be viewed on the accompanying disk.

Each music sheet, which makes up the data set, needs to be scanned using the same scanner and at the same resolution. For the purpose of this study a HP ScanJet5470C scanner, available in the School of Computing labs, was used to scan the music sheets at a resolution of 300 dpi. The scanned images were saved using a monochrome bitmap format, as this provides a two depth format which is easy to manipulate using pixel processing techniques.

Each software application will be tested using the bitmap as opposed to using the automatic scanning feature supported by some of the software applications.

5.3 Software Chosen for Evaluation

There are a wide range of available OMR software applications. Within this study, Photoscore, Smartscore, SharpEye and Capella-Scan have been chosen for evaluation.

5.3.1 PhotoScore

PhotoScore is available in two versions; PhotoScore Professional 4 and PhotoScore MIDI Lite 4. For this study, PhotoScore Professional 4 is the chosen version to be evaluated, since PhotoScore MIDI Lite 4 does not print or save in different file formats and can only recognise a limited range of symbols. Photoscore Professional 4 boasts a recognition rate of over 99% accuracy on most originals. It supports automatic scanning and chooses its own scanner settings. It is possible to save directly to Sibelius or G7 enabling further score processing. It is also possible to save in MusicXML or NIFF format to enable exports to other score processing applications or to save in MIDI, WAV or AIFF formats. The prices range from £24 for the MIDI Lite version to £199 for the Professional version [53].

5.3.2 SmartScore

SmartScore is available in Pro, SongBook, Piano and Guitar editions. SmartScore Pro will be evaluated in this study. It claims to exceed a recognition rate of 99% on well printed scores. It supports automatic scanning and resolution settings. It exports to MusicXML, Finale 200x (ENF), WAV and NIFF. The SongBook edition is limited to 3 stave maximum, other than this constraint it operates in the same way as the Pro edition. The MIDI edition is limited to 4 staves maximum and only allows MIDI editing and recording and exports to MIDI or ENF. The Piano edition is limited to a maximum of 2 staves and cannot perform any MIDI editing or recording although it can export to all file formats as outlined in the Pro edition. The Guitar edition has the same capabilities as the Piano edition with the constraint of a maximum of one stave. The cost varies from £225 for SmartScore 5 Pro edition to £170 for the SongBook edition to £60 for the other editions [51].

5.3.3 SharpEye

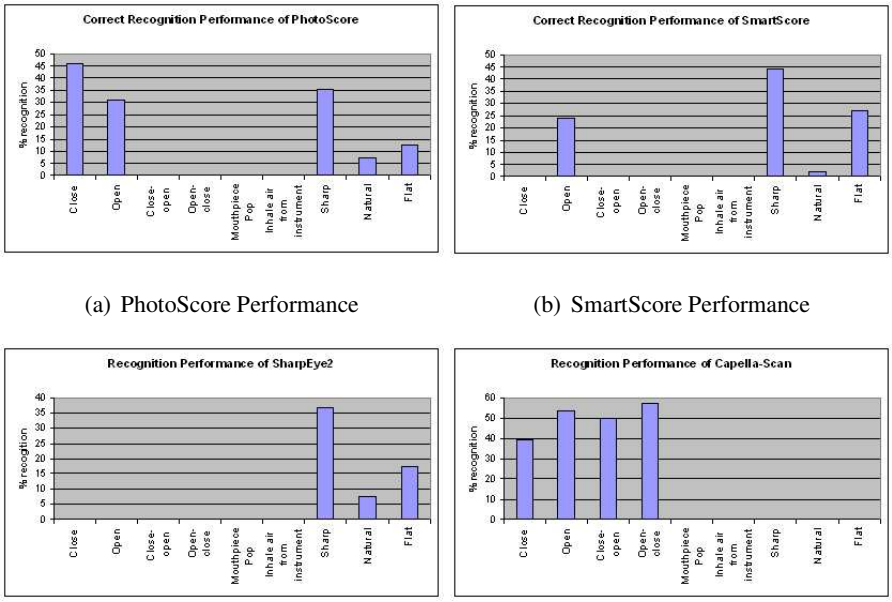
SharpEye2 is capable of reading BMP and TIFF input files and also direct scanning. It is capable of exporting MusicXML, MIDI or NIFF. It boasts a high recognition rate but does not give an exact figure. It explicitly states that it cannot cope with handwritten music. SharpEye2 supercedes SharpEye which is only capable of exporting NIFF and MIDI files and is not capable of direct scanning. For this study SharpEye2 was chosen to be evaluated. SharpEye2 retails at £85 and SharpEye retails at £30 [69].

5.3.4 Capella-Scan

Capella-Scan claims a high recognition rate but does not give a precise figure. It is capable of direct scanning and it exports to MIDI and Capella. Therefore, this means that further score processing can only be completed in Capella software as MIDI is incapable of representing many symbolic features of music. This software retails at £110 [19].

5.4 Comparative Study of Commercial OMR Software

The full scores for each software application and their ability to interpret the specific symbols can be viewed in the Evaluation Tables presented in Appendix E. The musical sheets which made up the data set were a combination of computer generated and professionally hand written scores and can be viewed on the attached disk.



(a) PhotoScore Performance

(b) SmartScore Performance

(c) SharpEye Performance

(d) Capella-Scan Performance

Figure 5.1: Recognition Performance of Evaluated OMR Software

Figure 5.1(a) shows the recognition performance of Photoscore. This software failed to recognise the close-open, open-close, mouthpiece pop and inhaling air from the instrument symbols. It appeared to be pretty poor at recognising accidentals but this was due to its poor recognition of the Berio Sequenza which contained many of these symbols. There were some instances of incorrect symbols being added in place of the symbol which was meant to be identified. On two occasions the sharp symbol was read as a natural symbol and a rest, the open-close symbol was read as an accent, the flat symbol was interpreted as a sharp and a natural on two occasions and the open symbol was interpreted as an accent twice.

The recognition performance of SmartScore is shown in Figure 5.1(b). SmartScore failed to recognise the close, close-open, open-close, mouthpiece pop and inhaling air from instrument symbols. It failed to recognise any of the Berio Sequenza, thus resulting in a poor recognition of accidentals. There were also some instances of incorrect symbols being added in place of the symbol which was meant to be identified. 25% of the close-open symbol were replaced for accents and the other 75% were missed. 14% of the open-close were identified as simply the open symbol the other 86% were missed. The close symbol was interpreted as a tenuto and an accent symbol.

Figure 5.1(c) shows the recognition performance of SharpEye2. It failed to recognise any of the brass symbols and yielded a poor recognition of accidentals due to difficulties interpreting the Berio

Sequenza score. On a few occasion the open-close symbol was incorrectly substituted for the open symbol. Also, on three occasions, i- and t- were substituted in place of the close symbol.

The recognition performance of Capella-Scan is shown in Figure 5.1(d). This software obtained a higher recognition rate of the brass symbols compared to the other software applications. Although it was better than average at the recognition of brass symbols, it failed to recognise any accidentals and its general recognition of notes and other symbols was somewhat poor. There were also instances of symbols being substituted for incorrect symbols and letters. These included: the close symbol substituted by h, i and H three times and a diagonal arrow twice, the open symbol was replaced by the number 0 and an accent was substituted for the mouthpiece pop eight times.

5.4.1 Summary

It is difficult to definitively state which software performed the best out of the applications evaluated because they all displayed poor recognition rates for the evaluation criteria used. None of the evaluated software was capable of recognising the mouthpiece pop or the inhaling air from instrument symbols.

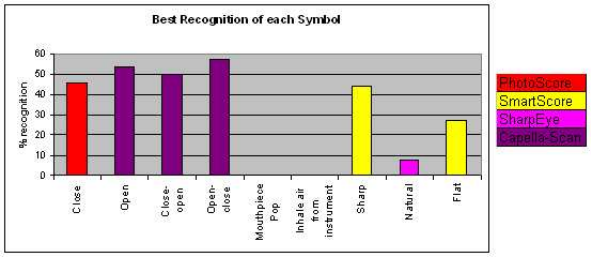


Figure 5.2: Best Recognition Performance of each Symbol

Figure 5.2 shows the best recognition performance of each symbol. Although Capella-Scan achieved above average recognition rates of the brass symbols, it failed to recognise any of the accidentals and its general recognition of notes and other symbols was poor.

From this comparative study it is evident that the commercially available software does not confidently support the recognition of brass symbols and that further development of a software prototype focusing on such symbol recognition is needed.

Chapter 6

Design and Implementation

6.1 Design Methodology

It is important for the implementation of the prototype that a suitable design methodology is adopted. By choosing the correct methodology the project can be split into smaller, more manageable stages/tasks to ensure the user's requirements are fulfilled, the programming of the code can be planned effectively and the project can be sufficiently organised.

The project's minimum requirements and objectives have been set and it is therefore unlikely that the requirements will change. With this consideration in mind, a methodology needs to be adopted which will allow several iterations within the design stage of the implementation thus ensuring that additional functionality can be incorporated into the software prototype with ease.

6.1.1 Traditional Life Cycle (Waterfall Approach)

The waterfall model is discussed in various texts [14], [18], [57] and consists of 6 key stages (requirements, analysis, design, coding, testing, maintenance). Each stage has to be completed before the next stage can be started. This approach forms the basis of most other well known methodologies.

It provides several advantages including: its ease to split tasks between individuals in a team, its ease in evaluating the progress of the project at each stage to decide whether the project should proceed, and

its effectiveness at helping to manage risk. In reality, real projects rarely follow such a simple sequential life cycle and it is plagued by fundamental disadvantages. It is likely that the requirements of the user will change and such a rigid design approach makes the project unresponsive to change. Iterations may be needed in order to review the design and analysis stage and it is possible that a long period of time will elapse between the systems engineering, and the system installation, due to all of the requirements being engineered in one go.

6.1.2 Phased Release Model

The phased release model, discussed in [45], is built upon the waterfall model but "introduces the notion of incremental development." It is in between the waterfall approach and the spiral model. It means that all of the requirements are finalised at the start of the development, which can, in most situations be considered a weakness. The design, implementation and testing stages are iterated over until the user's requirements are fulfilled.

6.1.3 Spiral Development

The main focus of this approach is risk, which is evaluated at the end of each loop. "Each loop in the spiral represents a phase of the software process" [64] and there are 4 processes within each (objective setting, risk assessment and reduction, development and validation, planning). At the end of each loop the phase is evaluated against the objectives and the outcome of the evaluation determines the next objectives to be set. According to [45], the spiral model encompasses an "incremental development that explicitly embraces prototyping and an iterative approach to software development."

6.1.4 Prototyping

As discussed in [14], prototyping is a design approach based on performing an initial analysis, which is used to define prototype objectives with an assigned priority. The priority determines the order in which the objectives should be implemented. The highest priority objectives are used to specify the prototype which is constructed and evaluated. The process is iterated again adding further objectives to be implemented and incorporating changes suggested at the evaluation stage. This approach offers early demonstrations of the system functionality ensuring that the user's requirements have been understood fully by the developer. Also, it gives the user a chance to suggest any further requirements, identify any difficulties with the interface early on, and test the feasibility and the usefulness of the overall system.

The a negative note, the prototype could cause user confusion, as it may be perceived as the final system, thus diverting attention from the functionality issues to design/interface issues. Also, managing the prototype involves careful decision making and significant user involvement.

6.1.5 Extreme Programming (XP) and Dynamic System Development Method (DSDM)

Through study of SE20 (Object Oriented Programming) and SE24 (Practical Software Development), it was discussed that the core principle of Extreme Programming, is pair programming, which reduces the number of mistakes, pulls on the whole teams programming knowledge and enforces thought and evaluation of many possible solutions [29]. Extreme Programming is better suited and designed for teams. DSDM, which controls the framework for Rapid Application Development (RAD), is of no use if there is "no representative sample of users" [27].

6.2 Methodology used for Software Prototype

As discussed in SE24 [29], it was identified that there are "hard" (engineering) and "soft" (people centric) approaches to design methodology. This project falls somewhere in the middle of these two approaches. After research into different methodologies it was decided to utilise the phased release model.

6.2.1 Reason for Chosen Methodology

The phased release model iterates over the design, implementation and integration and deployment stages resulting in the delivery of software in phases, which is suitably apt to the design of the software prototype. Since the minimum requirements and objectives have been identified, there is little chance that they will change and therefore no need to iterate over the requirements stage.

The waterfall model does not offer any iterations over any of the stages. Although the project requirements have been decided, and there is no need to iterate over the requirements stage, it means that the prototype has to be developed in a single iteration, which is not suited to its design. The spiral development methodology focuses mainly on risk. As risk is not the main focus of the project it is not appropriate to adopt a methodology with such a focus. The principle of the prototyping approach is quite suited to the project but it relies heavily on user evaluation for feedback into the next stage of the prototyping and it would have been suitable if there was a large base of users willing to evaluate the

system at each stage of the implementation. XP is more suited for teams as it utilises pair programming and DSDM, like prototyping, requires a representative sample of users to evaluate and test the system at each stage of implementation.

6.2.2 Phases in the Prototype Development

Phase 1 Open bitmap image file, perform raster scan, output representation of the bitmap into a text file.

Phase 2 Perform horizontal and vertical projections on the image and output the values to the text file, enabling the analysis of the image through representation of the projections in a histogram.

Phase 3 Identification of boundary pixels and implementation of the labelling algorithm, thus, enabling a whole piece of music to be scanned and the symbols to be isolated.

Phase 4 Identification of feature vectors to be used to classify the symbol.

Phase 5 Implementation of the k-NN classifier to enable the classification of a symbol.

6.3 Programming Language Selection

The choice of the programming language and development environment is of key importance for the development of the prototype. Since acquaintance with a new programming language was not a specified aim of the project it was decided to use an already familiar language. This narrowed the choice to Java, C and C++. It was essential that the programming language provided large libraries of existing functions, debugging capabilities, useful online help facilities and graphics analysis capabilities.

In comparison to C++, Java is relatively simple. This simplicity is overshadowed by its poor performance, limited expressive power and its use of large amounts of processing memory [50]. As the prototype will be dealing with the processing of many images it would be sensible to choose a language which can cope with such processing without slowing down the compilation of the program unnecessarily.

From research into existing projects undertaken in the area of OMR, it was discovered that a large number of them made use of the Visual C++ .NET language, using Visual Studio as the development environment. Visual C++ .NET offers the following advantages: it is portable, enabling easy distribution of the prototype; large number of libraries are provided; there are various sources of online help and it is a powerful and expressive language apt for image processing.

With these considerations in mind it was decided upon to utilise Visual C++ .NET. This would make it possible to integrate the recognition of brass features into existing prototypes with ease, if required.

6.4 Project Management

In order to make effective use of the available time and resources, an overall time plan was devised, as shown in Table 8.5 in Appendix H. At each stage a more detailed plan was devised (see Table 8.6, Table 8.7 Appendix H) to ensure that no time was wasted and that all sub-tasks that needed to be completed were planned for. Due to unforeseen circumstances, the project fell behind schedule and a revised time plan was needed to effectively plan the tasks leading up to the completion of the project (as shown in Table 8.8 Appendix H).

6.5 Implementation Stages

In order to efficiently implement the prototype the phased release model methodology was used. The implementation was defined in five stages as outlined below. As per the minimum requirements, the prototype should identify a minimum of three symbols. At this stage six symbols were identified for recognition including: the close symbol (Figure 6.1a), the open symbol (Figure 6.1b), the close-open symbol(Figure 6.1c), the open-close symbol(Figure 6.1d), the mouthpiece pop(Figure 6.1e) and the inhaling air from the instrument symbol(Figure 6.1f).

(a) Close Symbol (b) Open Symbol (c) Close-open Symbol (d) Open-close Symbol (e) Mouthpiece Pop Symbol (f) Inhale Air from Instrument Symbol

Figure 6.1: Symbols identified for recognition

6.5.1 Phase 1

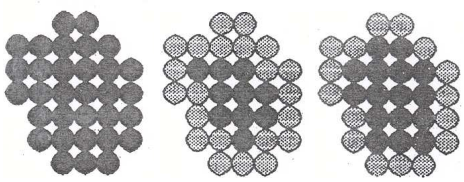
The first stage of the implementation process enabled a monochrome bitmap image to be opened by the program and a raster scan of the image to be performed. In order to ensure that these operations were completed as intended, a representation of the bitmap image was outputted to a text file.

6.5.2 Phase 2

The second phase of the implementation process performed horizontal and vertical projections of the image. This enabled analysis of the symbols through representations of the projections in a histogram. The histograms, although not identical for each symbol, have a similar characteristic shape (see Appendix F for histogram examples) and it is possible that the information extracted from them can be used as a feature vector for the classifier, implemented in phase 5. The projections were also used to determine the height and width of the symbol which was later used in the feature vector phase of the implementation.

6.5.3 Phase 3

The third phase firstly identified the boundary pixels of each symbol. As presented by James, in [40], there are two methods of finding boundary points 4-connected (Figure 6.2(b)) and 8-connected (Figure 6.2(c)). By identifying the boundary pixels it enabled isolation of the symbol within a piece of music. Also, the number of boundary pixels for both 4-connected and 8-connected were used as a feature vector, implemented in phase five. At this stage, to ensure that the boundary pixel identification was performed correctly, the boundary pixels were labelled with the letter "B" and outputted to the text file.



(a) Original Object (b) 4-Connected Pixel Neighbourhood (c) 8-Connected Pixel Neighbourhood

Figure 6.2: Pixel Neighbourhoods [40]

Secondly a labelling algorithm, as outlined on slide 13 of [55], was implemented. To determine the correctness of the labelling algorithm, the pixel labels were outputted to the text file and verified by eye.

With the correct implementation of the labelling algorithm and the boundary pixel location, the bounding box surrounding each symbol could be determined by calculating the minimum and maximum height and width coordinates of each symbol. This enabled the isolation of symbols within a music

score.

6.5.4 Phase 4

In order to classify a symbol, feature vectors of the symbol needed to be determined. James [40] provides a comprehensive set of feature vector implementations. From phase 2 and 3, respectively, the width and height of the symbol and the number of boundary pixels (both 4-connected and 8-connected) had been determined. From the height and width, the aspect ratio of the symbol was determined using the equation: **aspect ratio = width/height** and the bounding box area by using: **area = width*height**. The number of foreground pixels (area of the symbol) was determined by utilising a counter to count the number of horizontal projections. This could have also been implemented by counting the number of vertical projections.

Since "classification of binary objects is mostly a matter of shape discrimination [] the type of features which are looked for are indices of shape" [40]. According to [40], the best known shape index is the generalised PI index. The PI index = Perimeter^2/Area. The PI index has no dependency on scale and is computed with ease, also providing discrimination between classes of simple regular geometric shapes such as squares, ellipses and triangles. It was suggested by James that an improved estimate of the perimeter squared can be determined by the product of the number of 4-connected boundary pixels and the number of 8-connected boundary pixels [40]. This suggestion was implemented in the calculation of the PI index of each symbol.

6.5.5 Phase 5

In the fifth phase of the implementation the symbol was identified through use of a k-NN classifier. Such an algorithm/classification method determines the distance between clusters of data [27], in this case the data collected from the feature vectors. In total there were eight sets of feature vector data for each symbol. In determination of the clusters, a graph was plotted for each feature vector, graphically demonstrating the data for each symbol upon it. Since the k-NN classifier was used, this meant that the value of k needed to be determined for each symbol. It was deemed inappropriate to use all eight feature vectors for each symbol, as some of the feature vectors were indeterminate in its identification (see Appendix G for a copy of the feature vector clustering graphs).

A combination of the number of 8-connected boundary pixels and 4-connected boundary pixels provided a solid classification vector for all symbols. Their values were determined through examination

of the training data set.

The area feature vector could only be used to determine the mouthpiece pop and inhaling air from instrument symbols, as the other four symbols have a similar area there are no distinct clusters, even with the removal of anomalous results.

The PI Index was used to determine all symbols accept the open symbol, since the removal of the open symbol feature vector data enabled the generation of five distinct clusters to be determined. After the initial tests were completed the recognition of the open symbol proved poor compared to the other symbol recognition rates. Additional tests were undertaken to improve the classification of the open symbol, for example adding the PI index feature vector to its classification parameters, but they had no effect on the symbol identification that was made.

Width and bounding box area were also used as a firm feature vector in the classification process, as six distinct clusters were determined after examination of the training data set and removal of anomalous values.

Height was an indeterminate feature vector in this instance, as all of the symbols had a similar height, no distinct clusters could be concluded.

The aspect ratio was used solely to classify the inhaling air from instrument symbol. The other five symbol feature vector values resided in a single cluster.

6.6 Summary

The implementation was developed in 5 phases. At the end of each phase thorough testing and verification checks were performed to ensure that the phase was correctly and efficiently implemented before proceeding to the next phase of the implementation process.

The prototype was implemented using horizontal and vertical projections which were used to determine the height and width of the symbol, and thus the size of the bounding box to isolate it. The symbols were isolated further through implementation of the labelling algorithm. Once the symbols were isolated, a set of feature vectors were gathered in order to implement a k-NN classifier for symbol recognition. The value of k for each symbol was determined through analysis of the feature vector clustering for each symbol.

Chapter 7

Results Analysis and Evaluation

7.1 Results Analysis of Software Prototype Recognition Performance

In Chapter 5 a review of the currently available commercial software was undertaken. In order to make a comparison between the performance of the software prototype and that of the commercial software, it was necessary to use the same evaluation criteria and test data which were used to evaluate the commercial software. Before such a full comparison and evaluation was undertaken, it was necessary to evaluate the performance of the software prototype in terms of its recognition of individual symbols. This allowed adjustments to be made to the classification process, thus optimising its performance.

7.1.1 Individual Symbol Recognition Performance Discussion

Figure 7.1 shows the individual symbol recognition performance of the software prototype.

The close symbol yielded a recognition rate of 89.55%. Nearly 9% of the symbols were incorrectly identified, 67% of which were identified as the close-open symbol and 33% were identified as the open symbol. 1.5% of the close symbols were missed completely. This symbol was also substituted 4 times in place of another symbol.

The open symbol yielded the worst recognition performance out of the six symbols, with an overall recognition rate of just over 58%. The other 42% were identified incorrectly, 85% of which were

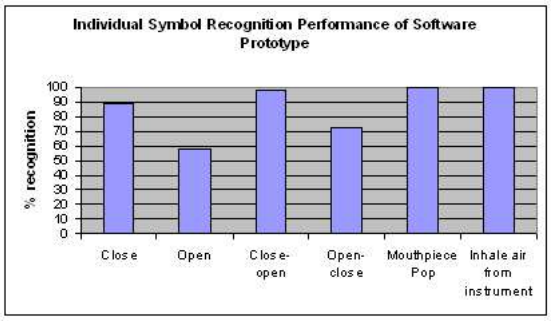


Figure 7.1: Individual Symbol Recognition Performance of Software Prototype

identified as the close-open symbol and the remaining 15% were identified as the close symbol. This symbol was also added twice in place of another symbol.

The close-open symbol supported a recognition rate of 98.36%. Only 1.64% of the symbols were incorrectly identified and 100% of these were the open-close symbol. Due to the poor recognition rate of the open symbol this meant that the close-open symbol had the highest number of instances of a symbol being added in place of another symbol.

The open-close symbol was correctly identified with a rate of just over 72%. 28% of the symbols were incorrectly recognised and the close-open symbol accounted for all of these instances. There was a single instance of this symbol being substituted in the place of another.

Both the mouthpiece pop and inhale air from instrument symbol supported 100% recognition with no instances of the symbol being added in place of another symbol.

7.1.2 Sheet Music Recognition Performance Discussion

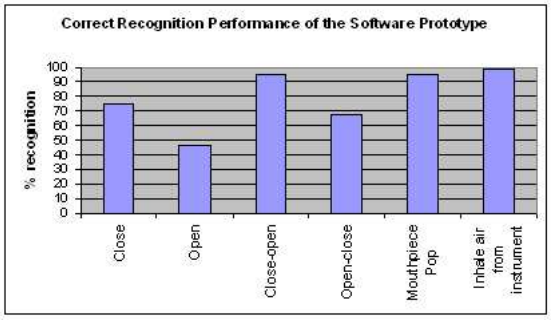


Figure 7.2: Sheet Music Recognition Performance of Software Prototype

Figure 7.2 shows the software prototype's recognition performance when identifying brass symbols from a piece of sheet music.

The correct recognition rate of the close symbol was 75.3%. 20.1% of the symbols were missed and an incorrect symbol was added in place 4.6%. This symbol was substituted for the open symbol, which was the case when the prototype was tested with individual symbols. This symbol was substituted for another symbol on 10 occasions.

The open symbol yielded a correct recognition rate of 46.89%. 14.35% of the symbols were missed and 38.76% of the symbols were incorrectly identified, 83% of which were identified as the open-close symbol and 17% were identified as the close symbol. This symbol was added in place of another on 8 occasions.

The correct recognition of the close-open symbol was 95.71% and only 1.95% of the symbols were completely missed. 2.34% of the symbols were incorrectly identified as the open-close symbol. On 54 occasions this symbol was added in place of another, nearly all of these were due to the incorrect recognition of the open symbol.

The open-close symbol yielded a recognition rate of 67.35%, 3.01% were completely missed and 29.64% of the symbols were incorrectly recognised. On all occasions they were substituted for the close-open symbol.

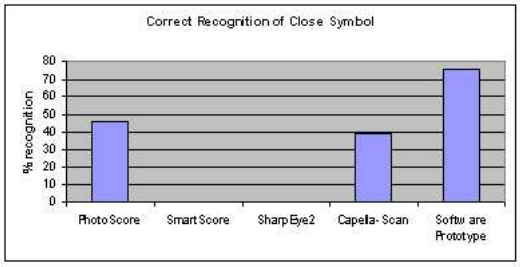
The mouthpiece pop supported a recognition rate of 94.36%, with only 5.64% of the symbols being completely missed. This symbol was not substituted for another symbol and there were no instances of incorrect recognition.

The inhale air from instrument symbol supported a recognition rate of 98.45%, with only 1.55% of the symbols being completely missed. Like the mouthpiece pop symbol, this symbol was not substituted for another symbol and there were no instances of incorrect recognition.

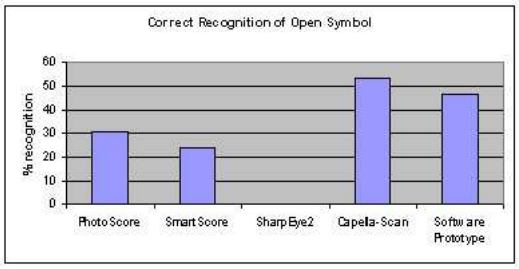
7.2 Results Discussion

In order to make an informed judgement on the recognition performance of the prototype, it is necessary to compare its recognition performance to the evaluated OMR software discussed in Chapter 5.

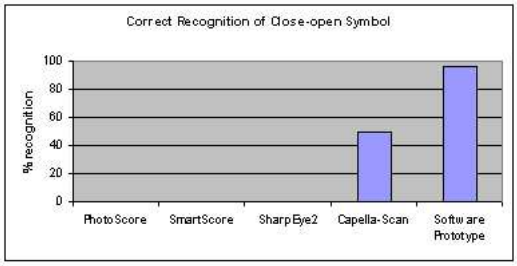
As shown by Figure 7.3(a), the commercially available software applications were out-performed by the software prototype on recognition of the close symbol, with the software prototype yielding a recognition rate of 75.3%.



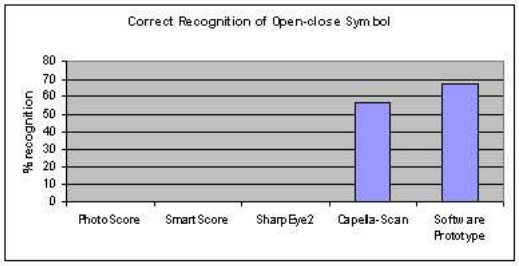
(a) Recognition Performance of Close Symbol



(b) Recognition Performance of Open Symbol



(c) Recognition Performance of Close-open Symbol



(d) Recognition Performance of Open-close Symbol

Figure 7.3: Individual Symbol Recognition Performance of Software Prototype

Figure 7.3(b) shows that the software prototype yielded an open symbol recognition rate that was just slightly less than Capella-Scan.

As shown by Figure 7.3(c), the commercially available software was out-performed by the software prototype on recognition of the close-open symbol, with the software prototype yielding a recognition rate of 95.71%.

As shown by Figure 7.3(d), the commercially available software was out-performed by the software prototype on the recognition of the open-close symbol, with the software prototype yielding a recognition rate of 67.35%.

There was no need to represent the recognition rate of the mouthpiece pop and the inhale air from instrument symbols in a graph since the commercially available software applications failed to recognise the symbols. The software prototype yielded a recognition rate of 94.36% for the mouthpiece pop symbol and 98.45% for the inhale air from instrument symbol.

7.3 Evaluation of Criteria

In order to determine the success of the project, the evaluation criteria used was a comparison of the recognition performance of the software prototype with the recognition performance of the commercially available software, using a test data set (can be viewed on the accompanying disk).

7.3.1 Accuracy of Results

The aim of the project was to produce a prototype with the capability of recognising brass symbols. The prototype is limited to such a recognition and therefore only compares the symbols found in the sheet of music to a limited symbol set and returns any other symbol as unclassified. Within the whole context of music, there are thousands of different symbols and if this software prototype was integrated into an existing software application then more feature vectors would be needed and a more accurate implementation of the k-NN classifier could be used to improve the accuracy. Therefore, the software prototype produced accurate results within the environment it was created for.

7.3.2 Comparison to Previous Works

Since this project has been limited to the recognition of brass symbols it is difficult to compare to other presentations of software comparisons. Firstly due to its specific nature and secondly, because there is no set standard of performing software evaluations and comparisons.

Within the "Coding Images of Music Sheets" [8], a comparison of SmartScore and SharpEye2 was made but it does not specifically measure the recognition rate of symbols above the stave line, therefore no comparison can be made between the results.

7.3.3 Evaluation of Methodology

The phased release model was the chosen design methodology for this project. Since the minimum requirements and objectives were finalised at the beginning, this fitted perfectly into the structure of the model. The development and implementation were delivered incrementally, meaning that the stages were iterated over until the requirements had been fulfilled. This allowed the software prototype to be delivered in five phases and ensured it was sufficiently tested and verified at the end of each phase before proceeding to the next.

Chapter 8

Conclusion and Future Directions

In order to achieve the objectives, as set out in Chapter 1, it was important to undertake focused research into the area of OMR. It was immediately discovered that the process could be split up into four distinct phases and individual research was carried out into each phase to ascertain the most appropriate methods for the recognition of brass symbols.

The field of OMR is a widely researched area. Although, as yet, there are no available OMR software applications that yield 100% recognition of a piece of sheet music. It was important to survey currently available commercial OMR software applications and to evaluate their performance at recognising brass symbols. Before the evaluation took place research was completed into methods of evaluating commercial software. It was discovered that very few attempts had been made to review commercial OMR software and most attempts drew on informal opinion. This was due to the fact that there are no universal standards for the representation of music notation and no sufficiently large standard ground truth data set of music sheets. Before the evaluation took place, further research into digitisation standards and input devices was needed to ensure that a sufficient ground truth data set was established and that the correct input devices and settings were utilised. A ground truth data set (available on the attached disk) was created and for each image in the data set the horizontal and vertical projections were calculated along with its feature vector data. The data set was split into two groups; the training data set which was used to train the k-NN classifier and the test data set which was used to test the performance of the

commercial OMR software and the performance of the software prototype.

From investigation into symbolic music representations it was concluded that there was no standard representation due to commercial software applications implementing their own unpublished representation formats. A report was compiled discussing symbolic music representation and an evaluation was completed to determine the most suitable symbolic music representation for the representation of brass symbols. It was discovered that currently MusicXML was the most appropriate symbolic music representation for representing brass symbols, although looking to the future, MPEG SMR will have the ability to provide a comprehensive representation of all music information integrated with other media, such as musical scores with video capabilities.

Upon surveying four currently available OMR software applications it was discovered that their performance at recognising brass symbols on a first attempt, without any training, was pretty poor, as shown in Figure 8.1.

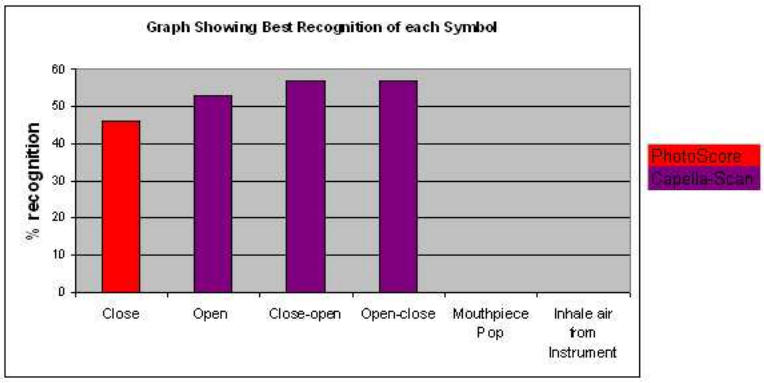


Figure 8.1: Best Recognition of Each Symbol

A data set of individual symbols was built in order to perform initial tests on the software prototype to enable changes to be made to the k-NN classifier to enhance the recognition performance. The initial test produced positive results. 90% of the close symbols were correctly identified, 98% of the close-open symbols were recognised, 72% of the open-close symbols were correctly recognised and both the mouthpiece pop and inhale air from instruments supported a 100% recognition rate. The only disappointing result was the open symbol that had a recognition rate of 58%, which was still better than the commercial software. Small changes to the k-NN classifier were made to optimise the recognition, although this did not improve the recognition of the open symbol due to the feature vectors that had been used to recognise the symbols.

From the previous chapter it can be seen that the software prototype out-performs the recognition performance of the commercially available software for close, close-open, open-close, mouthpiece pop and inhale air from instrument symbols when recognised from a piece of sheet music. The open symbol is the only symbol with a recognition rate that is slightly lower than the performance of Capella-Scan.

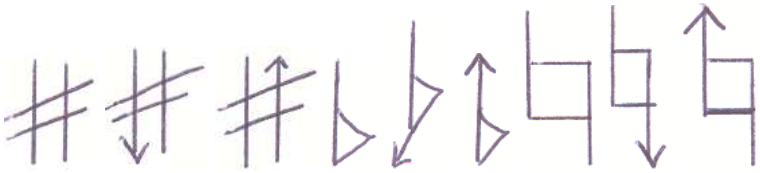
The project's resources, including time, were well managed, enabling the successful completion of both the project's minimum requirements and possible enhancements, thus satisfying the overall project aims and objectives.

8.1 Future Directions

Based on the results analysis and evaluation, there are many aspects of this project which could be improved, if further work was undertaken.

- A method of noise removal could be implemented in order to eliminate any black pixels which are not related to any symbols.
- If the prototype was integrated into a complete OMR software application, a method of stave line removal could be implemented in order to help the recognition of the symbols. It was recognised in [60] that stave lines interfere with the recognition process, this could be used in conjunction with a segmentation algorithm to isolate the symbols.
- As presented by Rossant [60], it is possible to utilise the stave line spacing, in pixel units, to express the scale of the score and normalise distances and sizes of objects within the score.
- In order to improve the classifier, a k-Mean classifier could be implemented. This method defines centroids around the clustering of the data based on a mean distance [27].
- In order to aid the recognition process more feature vectors could be used to classify the symbol. For example moments as presented in [40].
- As section 4.4 presents, there are a wide range of image formats which a music sheet can be saved in. A possible enhancement of the prototype would be to increase the file formats which can be interpreted by the prototype.

- The prototype could be further enhanced by implementing a method of saving the output in a symbolic music representation format. As per section 3.2, MusicXML was identified as the most appropriate symbolic music representation for representing brass symbols.
- If this project was completed on a larger scale it may be more useful to use the prototyping methodology as opposed to the phased release model used in this project, providing there was a sufficiently large group of users able to test and provide feedback on each prototype. This would take into consideration any requirements changes and provide useful feedback at the end of each prototype phase.
- From the research completed in section 2.1.3 and the commercial software evaluation in Chapter 5, it is possible to increase the number of symbols which can be interpreted by the prototype. Other symbols specific to brass are the quarter tone lower and quarter tone higher symbols (see Figure 8.2). These are an adaptation of the sharp, flat and natural symbols. Since the commercial software has already been evaluated for its interpretation of the sharp, flat and natural symbol and research has been completed into possible methods of its recognition, an extension could be made with ease.



(a) Sharp (b) Sharp quarter tone lower (c) Sharp quarter tone higher (d) Flat (e) Flat quarter tone lower (f) Flat quarter tone higher (g) Natural (h) Natural quarter tone lower (i) Natural quarter tone higher

Figure 8.2: Symbols

Bibliography

[1] A. Andronico and A. Ciampa. Automatic pattern recognition and acquisition of printed music. In *Proceedings of the International Computer Music Conference, Venice, Italy*, pages 245 – 278, 1982.

[2] D. Bainbridge. A complete optical music recognition system: Looking to the future. www.cs.waikato.ac.nz/ davidb/publications/complete/, 1994.

[3] D. Bainbridge. Optical music recognition: A generalised approach. In *Proceedings of Second New Zealand Computer Science Conference*, 1996.

[4] D. Bainbridge and T. Bell. An extensible OMR system. In *Proceedings of the 19th Australasian Computer Science Conference*, 1996.

[5] D. Bainbridge and T. Bell. Dealing with super imposed objects in OMR. In *Proceedings of Sixth International Conference on Image Processing and its Applications*, volume 2, pages 756 – 760, 1996.

[6] D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and Humanities*, 35:95 – 121, 2001.

[7] D. Bainbridge and S. Inglis. Musical image compression. In *Proceedings of DCC '98 Data Compression Conference*, pages 209 – 218, 1998.

[8] J. Barthelemy, I. Bruno, P. Nesi, K.C. Ng, and B. Ong. DE4.7.1 Coding images of sheet music. `www.interactivemusicnetwork.org/documenti/view_document.php?file_id=765`, 2004. Last accessed: 20/02/06.

[9] J. Barthelemy and G. Carpentier. DE4.3.2 Multimedia standards for music coding. www.interactivemusicnetwork.org/documenti/view_document.php?file_id=1197, 2005. Last accessed: 20/02/06.

[10] A. Belkin. NIFF. http://www.musique.umontreal.ca/personnel/Belkin/NIFF.doc.html, 2006. Last accessed: 20/03/06.

[11] P. Bellini, I. Bruno, and P. Nesi. *An offline optical music sheet recognition, in Visual perception of music notation: Online and offline recognition*, pages 40 – 77. IRM Press, 2005.

[12] P. Bellini, P. Nesi, and G. Zoia. Symbolic music representation in MPEG. *IEEE Multimedia*, 12:pages 42 – 49, 2005.

[13] C. Bellissant and P. Martin. Low-level analysis of music drawing images. In *Proceedings of the First International Conference on Document Analysis*, 1991.

[14] S. Bennett, S. McRobb, and R. Farmer. *Object Oriented Systems Analysis and Design using UML*. McGraw Hill, 2002.

[15] D. Blostein and H. Baird. *A critical survey of music image techniques, in Structured Document Analysis*, pages 405 – 434. Springer-Verlag, 1992.

[16] Minerva Editors Board. Minerva Knowledge Base. http://www.minervaeurope.org/, 2006. Last accessed: 08/04/06.

[17] R.D. Boyle and R.C. Thomas. *Computer Vision: A First Course*. Blackwell Scientific Publications, 1988.

[18] B. Bruegge and A. Dutoit. *Object Oriented Software Engineering Conquering Complex and Changing Systems*. Prentice Hall International, international edition, 2000.

[19] Capella. Capella-Scan. www.capella-software.com/capscan.htm, 2006. Last accessed: 07/04/06.

[20] N. Carter and R. Bacon. *Automatic recognition of printed music, in Structured Document Analysis*, pages 456 – 466. Springer-Verlag, 1992.

[21] N.P. Carter. *Automatic recognition of printed music in context electronic publishing*. PhD thesis, University of Surrey, 1989.

[22] G. Castan. NIFFML: An XML implementation of notation interchange file format. *Computing in Musicology*, pages 103 – 112, 1999.

[23] G. Castan, M. Good, and P. Roland. Extensible markup language (XML) for music applications: An introduction. *Computing in Musicology*, pages 95 – 102, 1999.

[24] A.T. Clarke, B.M. Brown, and M.P. Thorne. Inexpensive optical character recognition of music notation: A new alternative for publishers. In *Proceedings of the Computers in Music Research Conference*, pages 84 – 87, 1988.

[25] A.T. Clarke, B.M. Brown, and M.P. Thorne. Coping with some really rotten problems in automatic music recognition. *Microprocessing and Microprogramming*, 29:547 – 550, 1989.

[26] B. Couasnon and J. Camillerapp. A way to separate knowledge from program in structured document analysis: Application to optical music recognition. In *Proceedings of the International conference on Document Analysis and Recognition*, pages 1092 – 1097, 1988.

[27] R. Duda and D. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition edition, 2001.

[28] N. Efford. *Digital Image Processing: A Practical Introduction using Java*. Pearson Education, 2000.

[29] N. Efford and O. Johnson. SE20 and SE24 Notes. Lecture Slides and Notes, 2004.

[30] I. Fujinaga. *Optical music recognition using projections*. PhD thesis, McGill University, 1988.

[31] I. Fujinaga. *Adaptive optical music recognition*. PhD thesis, McGill University, 1997.

[32] S. George. *Evaluation in the visual perception of music, in Visual Perception of Music Notation: Online and Offline Recognition*, pages 304 – 349. IRM Press, 2005.

[33] M. Good. MusicXML for notation and analysis. *Computing in Musicology*, pages 113 – 124, 1999.

[34] C. Grande. *The Notation Interchange File Format: A Windows compliant approach, in Beyond MIDI*, chapter 31, pages 491–512. MIT Press, 1997.

[35] C. Grout and J. Rymer. Creating Digital Resources for the Visual Arts: Standards and Good Practice. http://vads.ahds.ac.uk/guides/creating_guide/contents.html, 2006. Last accessed: 28/03/06.

[36] Guido. Guido Music Notation Format. http://www.informatik.tu-darmstadt.de/AFS/CM/GUIDO/, 2006. Last accessed: 28/03/06.

[37] W. Hewlett and E. Selfridge-Field. *MIDI, in Beyond MIDI*, chapter 2, pages 41 – 72. MIT Press, 1997.

[38] H. Hoos, K. Hamel, K. Renz, and J. Kilian. Representing scorelevel music using the GUIDO musicnotation format. *Computing in Musicology*, pages 75 – 95, 1999.

[39] W. Icking. *MuTeX, MusicTeX, MusiTex, in Beyond MIDI*, chapter 17, pages 222–231. MIT Press, 1997.

[40] M. James. *Pattern Recognition*, chapter 6, pages 102 – 134. BSP Professional Books, 1987.

[41] I. Kato and S. Inokuchi. *A recognition system of printed piano music using musical knowledge and constraints, in Structured Document Analysis*, pages 435 – 455. Springer-Verlag, 1992.

[42] D. Kay and J. Levine. *Graphics File Formats*. Windcrest/ McGraw-Hill, 2nd edition edition, 1995.

[43] W. Koseluk. Digitisation of musical sources: An overview. *Computing in Musicology*, pages 219 – 226, 1999.

[44] J. Lacey. *Complete Guide to Digital Imaging*. Thames and Hudson, 2002.

[45] R. Laganiere and T. Lethbridge. *Object Oriented Software Engineering Practical Software Development using UML and Java*. McGraw Hill, 2nd edition edition, 2005.

[46] D. Langoff, N. Bapiste-Jessel, and D. Levy. NIFF transcription and generation of braille musical scores. *Computing in Musicology*, pages 34 – 44, 1999.

[47] M.W. Lee and J.S. Choi. The Recognition of printed music score and performance using computer vision system. *Journal of Korean Institute of Electronic Engineers*, 22(5):429 – 435, 1985. Translated by Blostein and Baird.

[48] A. Low. *Introductory Computer Vision and Image Processing*. McGraw Hill, 1991.

[49] J.V. Mahoney. *Automatic analysis of musical score images*. PhD thesis, Massachusetts Institute of Technology, 1982.

[50] B. Milewski. Battle of the Languages - Java vs C++. `http://www.relisoft.com/web/c_java.html`, 2004. Last accessed: 08/04/06.

[51] Musitek. Musitek - SmartScore Music Scanning Software. `www.musitek.com`, 2006. Last accessed: 07/04/06.

[52] Pulman Network. PULMANweb Public Libraries Mobilising Advanced Networks. `www.pulmanweb.org`, 2006. Last accessed: 08/04/06.

[53] Neuratron. PhotoScore Music Scanning Software. `www.neuratron.com/photoscore.htm`, 2006. Last accessed: 07/04/06.

[54] K.C. Ng. *Automated computer recognition of music scores*. PhD thesis, Leeds University, 1995.

[55] K.C. Ng. Multimedia Imaging Lecture Slides. `http://icrism.leeds.ac.uk/kia/musi2614/`, 2006. Last accessed: 08/04/06, Lecture 6 Musical Score Information and Score Following and Lecture 7 OMR and Pixel Processing.

[56] K.C. Ng and R.D. Boyle. Reconstruction of music scores from primitive sub-segmentation. Technical report, School of Computer Studies, University of Leeds, 1994.

[57] M. Priestly. *Practical Object Oriented Design with UML*. McGraw Hill, 2nd edition, 2003.

[58] Recordare. MusicXML Definition. `http://www.recordare.com/xml.html`, 2006. Last accessed: 28/03/06.

[59] C. Roads. *Grammars as representations for music, in Foundations of Computer Music*, chapter 23, pages 403 – 442. MIT Press, 1987.

[60] F. Rossant. A global method for music symbol recognition in typeset music sheet. *Pattern Recognition Letters*, 23:1129 – 1141, 2002.

[61] M. Roth. An approach to recognition of printed music. Technical report, ETH Zurich, Switzerland: Swiss Federal Institute of Technology, 1994.

[62] E. Selfridge-Field and T. Hall. *DARMS, in Beyond MIDI*, chapter 11 - 13, pages 163 – 200. MIT Press, 1997.

[63] A. Smith. MusicXML: Music Notation Interchange for the Internet. http://brainstormandraves.com/archives/2002/06/30/musicXML, 2002. Last accessed: 01/03/06.

[64] I. Sommerville. *Software Engineering*. Pearson Education, 7th edition, 2004.

[65] TASI. Technical Advisory Service for Images - Cameras. www.tasi.ac.uk/advice/creating/scanners.html, 2006. Last accessed: 08/04/06.

[66] TASI. Technical Advisory Service for Images - Scanners. www.tasi.ac.uk/advice/creating/scanners.html, 2006. Last accessed: 08/04/06.

[67] ThinkQuest. Music Dictionary. http://library.thinkquest.org/2791/MDOPNSCR.htm, 2006. Last accessed: 20/04/06.

[68] B. Victor. NIFF 6a.3 Notation Interchange File Format. http://neume.sourceforge.net/niff1998, 1998. Last accessed: 02/03/06.

[69] Visiv. SharpEye Music Reader. www.visiv.co.uk, 2006. Last accessed: 07/04/06.

[70] J. Watkinson. *The MPEG Handbook*. Focal Press, 2nd edition edition, 2005.

[71] WEDELMusic. WEDELMUSIC XML Notation. www.wedelmusic.org/lang/xmlnot.html, 2006. Last accessed: 08/03/06.

Appendix A

Before entering the final year I thought in depth about the type of final year project I wanted to undertake. Since I major in Computer Science and minor in Music I thought it would be an interesting idea to encompass a project which covered both parts of my degree programme, because until now they have always been completely separate. I also thought about my strengths and weaknesses, which would strongly influence my choice of project. Programming is not my main strength so I wanted a project which would not solely rely on my programming skills with the possibility of incorporating research and evaluation into it. With these ideas in mind I looked through the list of possible projects available and discovered that none of them were really suited to my initial thoughts. It was at this point I discovered optical music recognition (OMR) and decided to seek further advice to the viability of this project.

With no prior knowledge of imaging, computer vision or artificial intelligence this project was an extremely ambitious choice. Before beginning my initial research I spent two weeks gaining a basic knowledge of computer vision using [17], [28] and [48] as a starting point. Once I had started my initial research I realised the enormity of the project which I had chosen to undertake, so I decided to focus on the recognition of brass symbols above the stave, firstly evaluating the recognition performance of commercial software to see whether the project was viable. This also provided a basis for my evaluation.

The progress of the prototype implementation was moderately slow to begin with due to familiarising myself with the Visual C++ syntax. Although I had a prior knowledge of C, there is slightly different syntax and also I had had to learn to use Visual Studio effectively and efficiently. In order to do this I spent the Christmas holidays working through tutorials, which helped both my understanding of Visual C++ .NET and my ability to use Visual Studio effectively.

My original project plan, see Appendix H, outlines the milestones and timings of different tasks that needed to be undertaken. At each stage I produced a more detailed plan to ensure that no time was wasted and that all the sub tasks that needed to be completed were planned for. My estimations of time up to Christmas were very accurate and I adhered to the plan. After the examination period in January I fell really ill meaning that the development of the prototype was seriously delayed. This meant that subsequent tasks relying on the completion of the prototype could not be completed on time. I produced a revised time plan, Table 8.8 in Appendix H, to get me back on track. This could have caused my project to not be completed on time. This was avoided because I kept up with the writing of the report throughout the project meaning that the Easter holiday was not needed solely for the writing of the report as originally planned. This meant that the prototype could be completed and tested during the holidays alongside the redrafting and proof reading of the report, and thus allowing the project to be delivered on time.

In retrospect, I am glad that I kept up with the writing of the report, because even if I had not fallen ill, it would have been exceptionally difficult to complete it all during the Easter holidays. Firstly, to remember everything that had been done and secondly, there would not have been enough time to go through the vast pages of notes I had made from my research throughout the project. If I was to complete a project on this scale again I would definitely ensure that I write draft chapters throughout the progress of the project.

I have learnt many lessons from completing this project. Firstly, project planning is key and unforeseen circumstances affecting the plan are impossible to anticipate. In the situation I was faced with it was important to keep calm and focused and plan each day in advance. I found that setting goals to be completed before the end of the day played an important part in the success of the project. Also I found reviewing the work I had completed at the end of the week ensured that I had included all of the necessary details in my plan and ensured that my time estimations had been accurate.

The second lesson I learnt was the importance of thinking seriously about the type of project that I wanted to undertake and to being open to new concepts and ideas. This was of key importance because I had very little knowledge of computer vision and image processing apart from the computer graphics modules I had completed in the second and third year. This project was extremely overwhelming at first and I found some the concepts and theories hard to grasp, particularly because a lot of the literature was quite technical and I only had a basic knowledge of the subject. I soon learnt that the only way to overcome this problem was to not be afraid to ask for help and advice however trivial the questions may

sound.

I found it difficult, when writing the report, to judge the amount of detail to go into when discussing concepts. As this project incorporates both computer science and music, I had to think about the concepts which might be alien to computer scientists and vice versa for musicians. To overcome this challenge I decided to produce a glossary of musical terms to help computer scientists reading the project with unfamiliar words and concepts, see Appendix B. As supplementary information for musicians, I provided additional information on symbolic music representations (see Appendix C) and additional information on digitisation standards and file formats (see Appendix D).

I found my project extremely interesting and enjoyable and there is a huge scope for further projects in this area. My one piece of advice to future students is to seize the opportunity of undertaking such an exciting project. OMR is a rapidly evolving research topic but there are still vast areas which need to be researched and standardised, such as the symbolic music representations, as discussed in Chapter 3 and the digitisation of music scores with the possibility of creating some form of repository to store them, as discussed in Chapter 4.

Appendix B

Glossary of Musical Terms using [67] and [54] as references.

Accents Placed above a note to indicate stress or emphasis.

Accidentals Sharp, flat or natural symbol attached to the note which is not represented in the key signature.

Bar A metrical unit made up of a set of beats enclosed by two bar-lines.

Bar-line The vertical line placed on the stave to divide the music into measures.

Beam Joins notes to help make the metre clear, by showing where the beats begin and end.

Bend When playing a brass instrument, movement of lips to change the air flows "bending" the note in and out of tune.

Chord A group of notes to be played simultaneously.

Clef A symbol located at the beginning of a stave to indicate the pitches of notes on the stave.

Close Refers to either covering the bell of a brass instrument with the hand or covering the mute with the hand.

Close-open Refers to either the covering the bell of a brass instrument with the hand and gradually opening it or covering the mute with the hand and gradually opening it.

CMN Common Music Notation.

Crescendo Gradually getting louder in dynamic level.

Diminuendo Gradually getting softer in dynamic level.

Dotted note A way of extending the note value by placing a dot after the note.

Flat An accidental sign that lowers the pitch of a note by a semitone.

Glissando The rapid scale achieved by moving the sliding in and out on a trombone.

Grace notes An ornamentation symbol which usually occupies half the duration of the main note.

Inhale air from instrument Cover the mouthpiece of a brass instrument and breathe in.

Key The major or minor scale in which the piece of music is based.

Key Signature A set of accidentals placed at the beginning of a stave after the clef sign.

Mouthpiece Pop Tapping the mouthpiece of a brass instrument with the palm of the hand.

Mute Can be metal or plastic and they are inserted into the bell of a brass instrument to create different sounds.

Natural An accidental sign that cancels an sharp or flat, restoring the pitch to its unaltered state.

Note A symbol that carries relative duration information and pitch.

Open Refers to either uncovering the bell of a brass instrument which was previously covered with the hand or uncovering the mute which was previously covered with the hand.

Open-close Refers to either the gradual covering the bell of a brass instrument with the hand and or the gradual covering the mute with the hand.

Ornamentation Shorthand of symbols to indicate certain conventions of performance.

Pitch One of the twelve different pitches that divide and octave.

Quarter tone higher Specific to brass instruments. Using the lips to change the pitch of the note.

Quarter tone lower Specific to brass instruments. Using the lips to change the pitch of the note.

Rest Has an associated length and represents silent notes within the bar.

Score Collections of sheets of music.

Sharp An accidental sign that raises the pitch of a note by a semitone.

Slurs A curved line over a group of notes indicating that they are to be performed as a smooth unbroken phrase.

Staccato Articulation marking to indicate a short or brief note. It is notated as a dot above or below the note.

Staff See stave.

Staff line See stave line.

Stave A set of five parallel stave lines form a stave, which is generally used in CMN.

Stave line A stave line is a horizontal line of a stave which defines the coordinate system for the music notation.

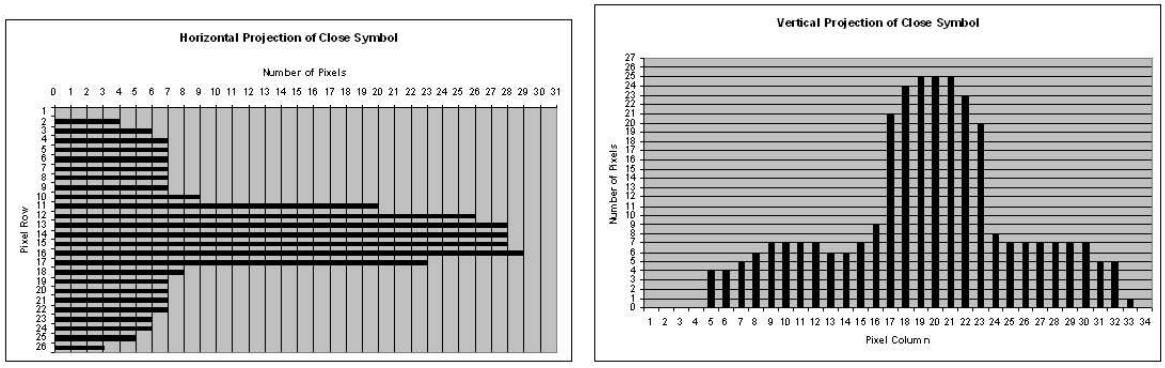
Tenuto Articulation marking to indicate a broad note.

Time Signature A fraction generally located at the beginning of a piece of music after the key signature, to indicate the nature of the organisation of pulse and the number of beats in a bar.

Tuplets A group of three notes required to be played in the duration allowed by the time signature for two of the same duration.

Appendix C - Horizontal and Vertical Projections

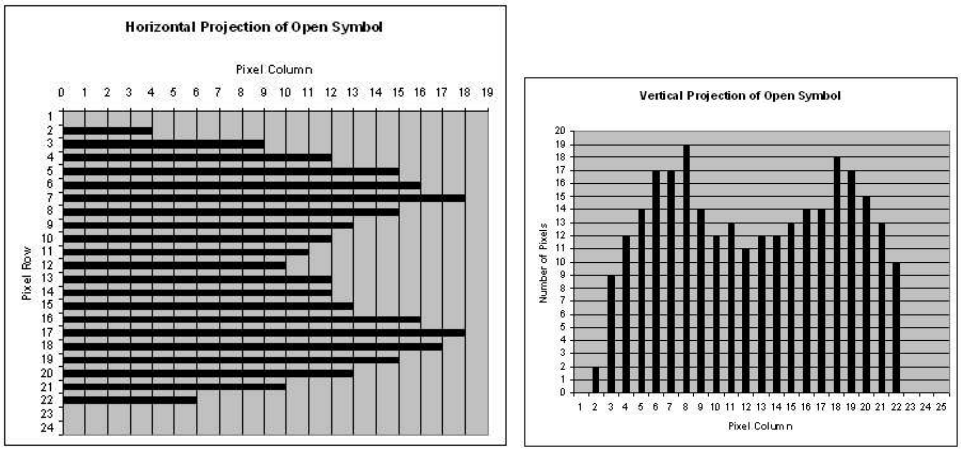
This appendix presents a typical horizontal and vertical projection histogram for each symbol.



(a) Horizontal Projection

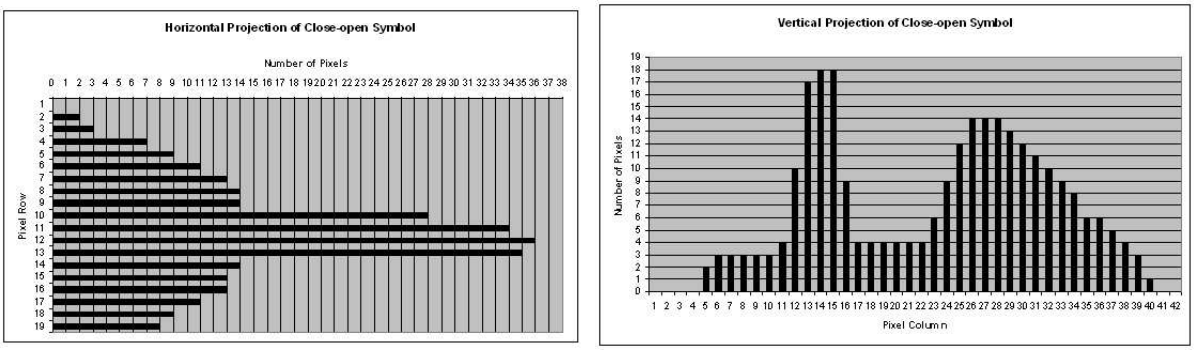
(b) Vertical Projection

Figure 8.3: Projections of Close Symbol



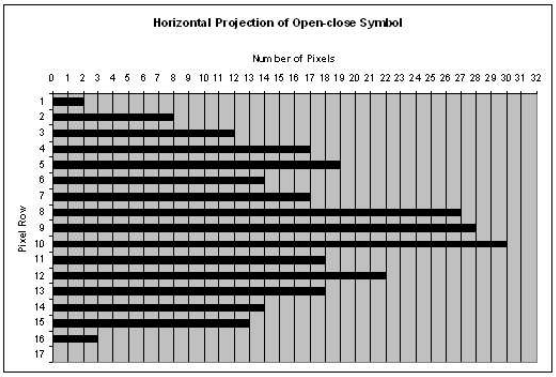
(a) Horizontal Projection (b) Vertical Projection

Figure 8.4: Projections of Open Symbol

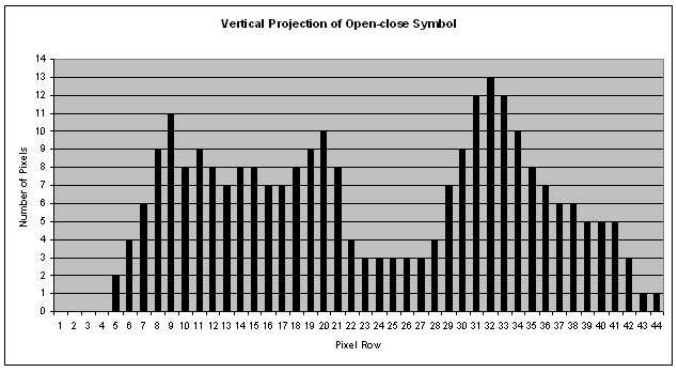


(a) Horizontal Projection (b) Vertical Projection

Figure 8.5: Projections of Close-open Symbol

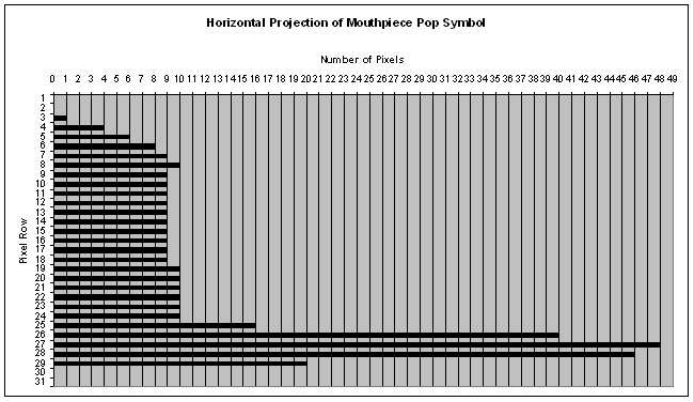


(a) Horizontal Projection

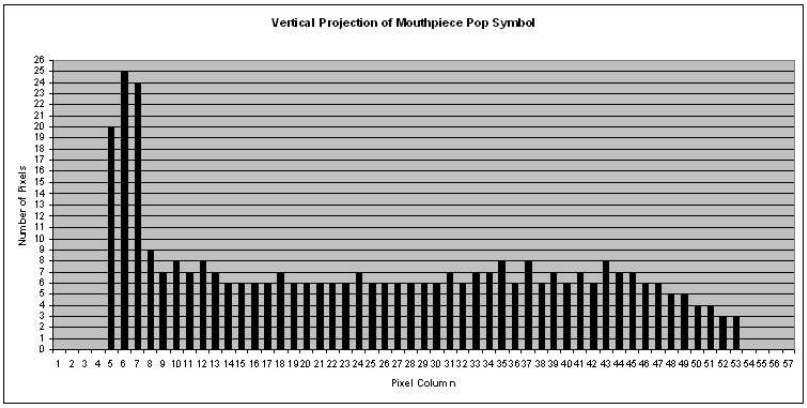


(b) Vertical Projection

Figure 8.6: Projections of Open-close Symbol

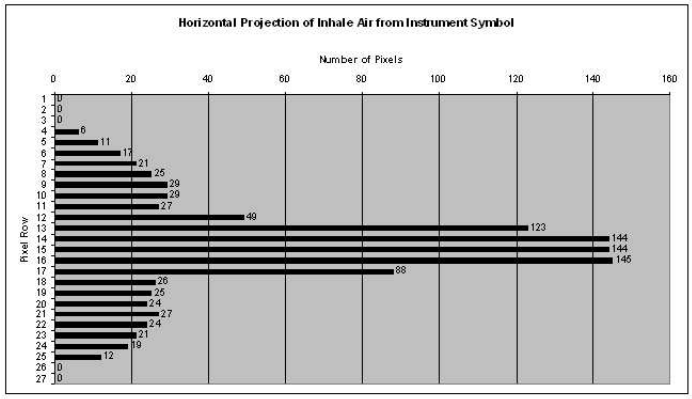


(a) Horizontal Projection

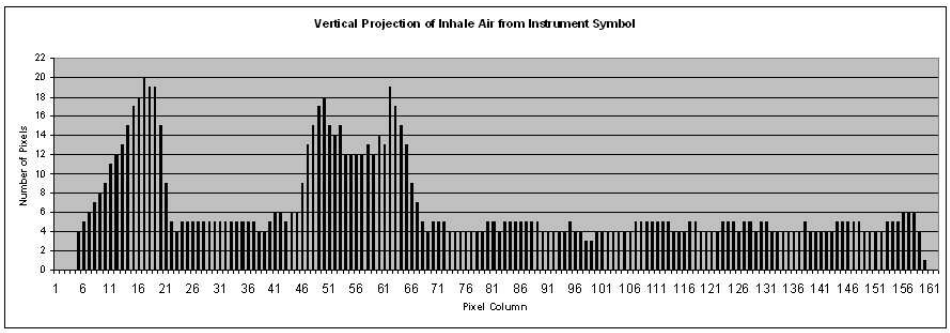


(b) Vertical Projection

Figure 8.7: Projections of Mouthpiece Pop Symbol

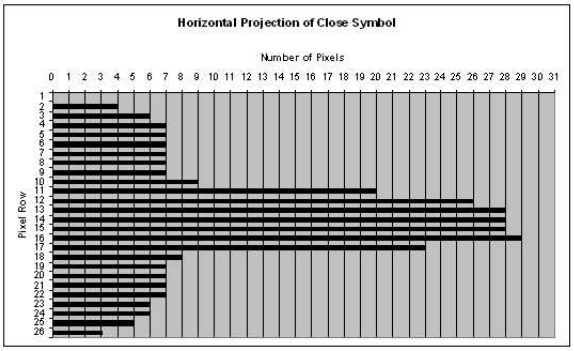


(a) Horizontal Projection

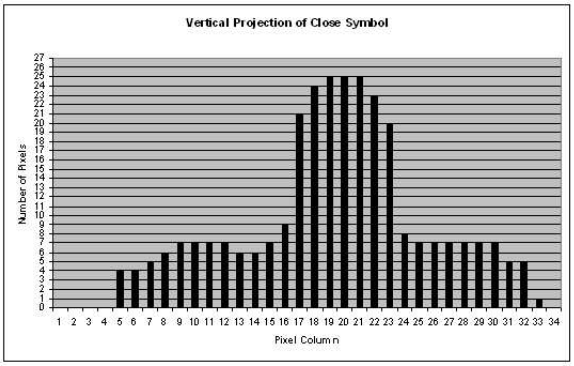


(b) Vertical Projection

Figure 8.8: Projections of Inhale Air from Instrument Symbol



(a) Horizontal Projection



(b) Vertical Projection

Figure 8.9: Projections of Close Symbol

Appendix D - Feature Vector Clustering

This appendix presents the graphs produced to determine feature vector clusters.

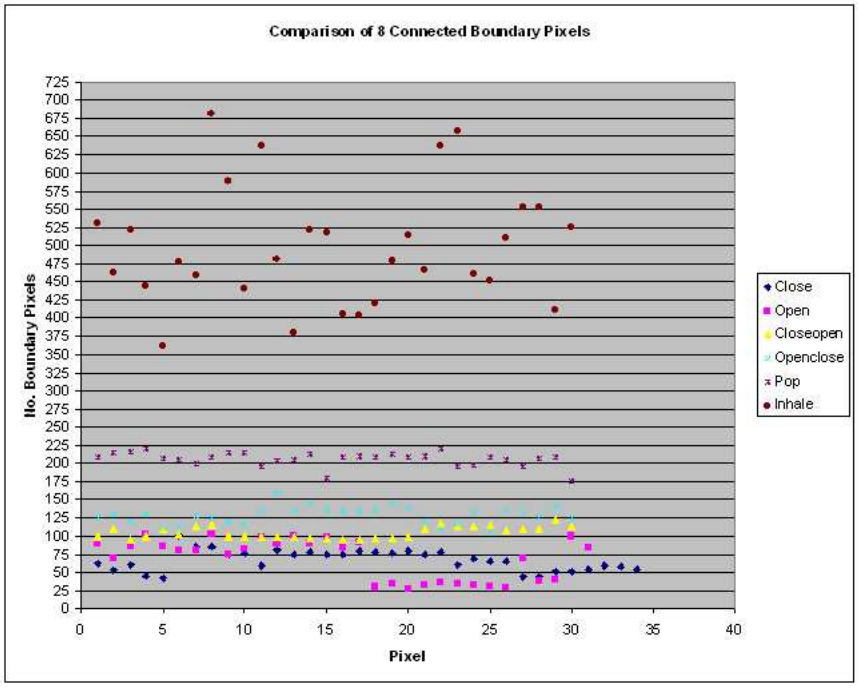


Figure 8.10: Feature Vector Clustering - 8 Connected Pixel Neighbourhood

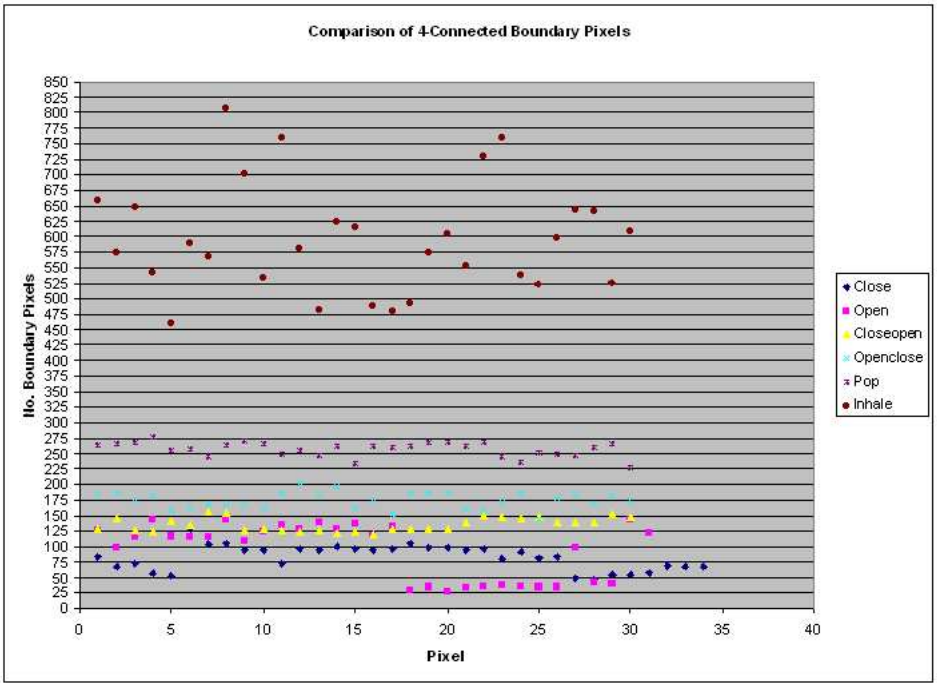


Figure 8.11: Feature Vector Clustering - 4 Connected Pixel Neighbourhood

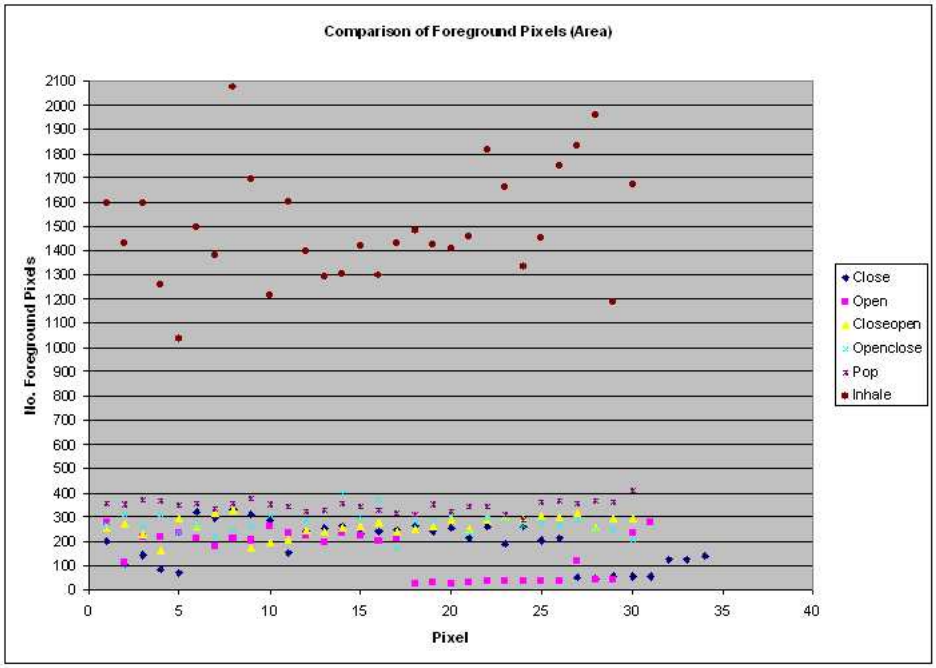


Figure 8.12: Feature Vector Clustering - Foreground Pixels(Area)

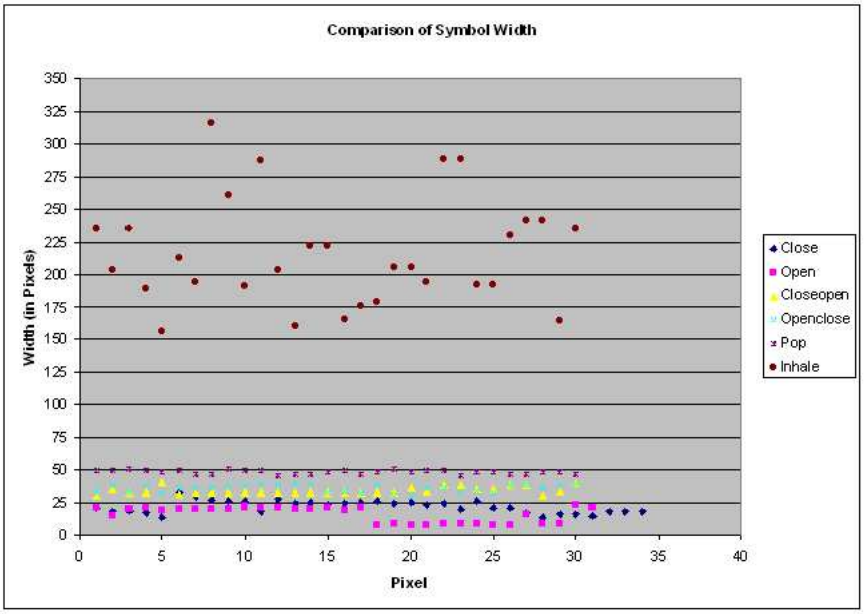


Figure 8.13: Feature Vector Clustering - Width of Symbol

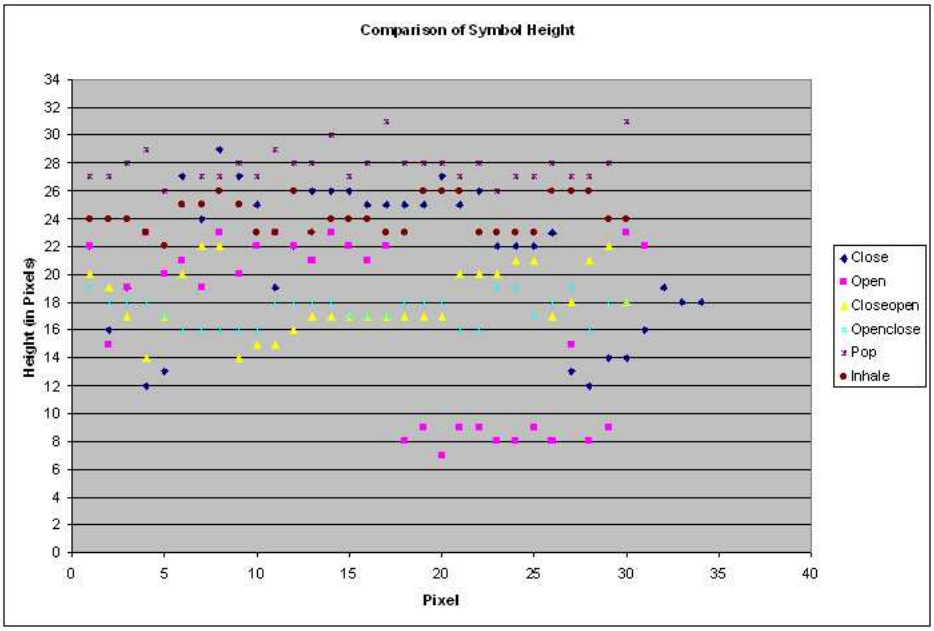


Figure 8.14: Feature Vector Clustering - Height of Symbol

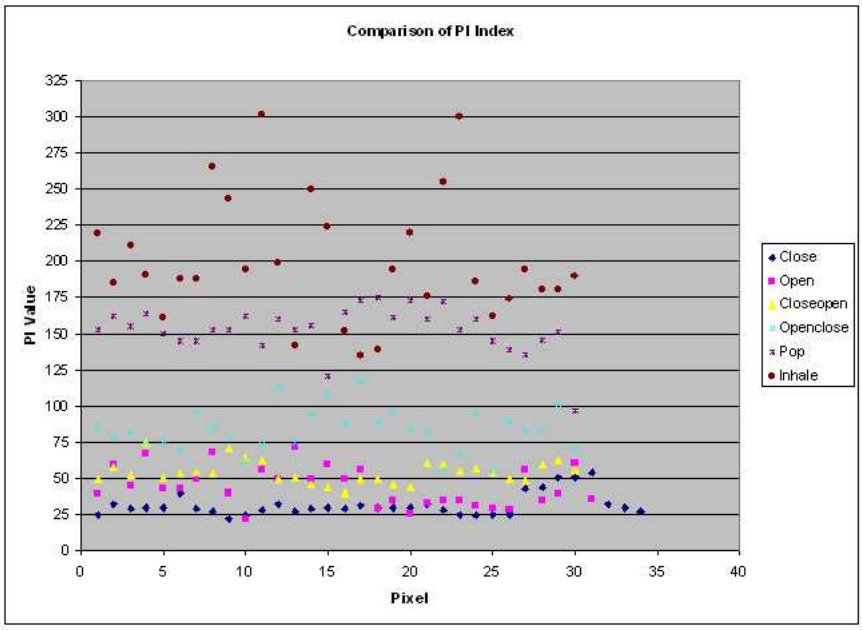


Figure 8.15: Feature Vector Clustering - Generalised PI Index

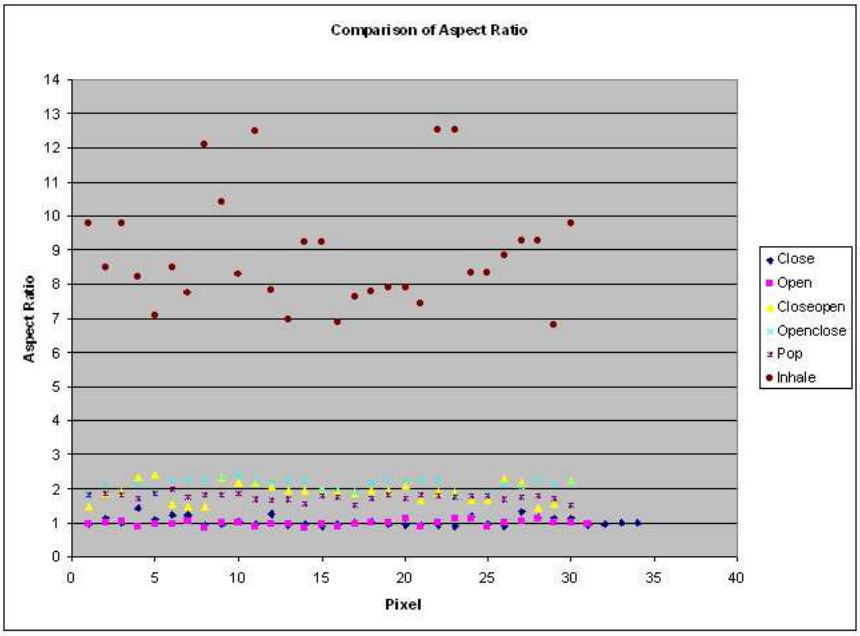


Figure 8.16: Feature Vector Clustering - Aspect Ratio

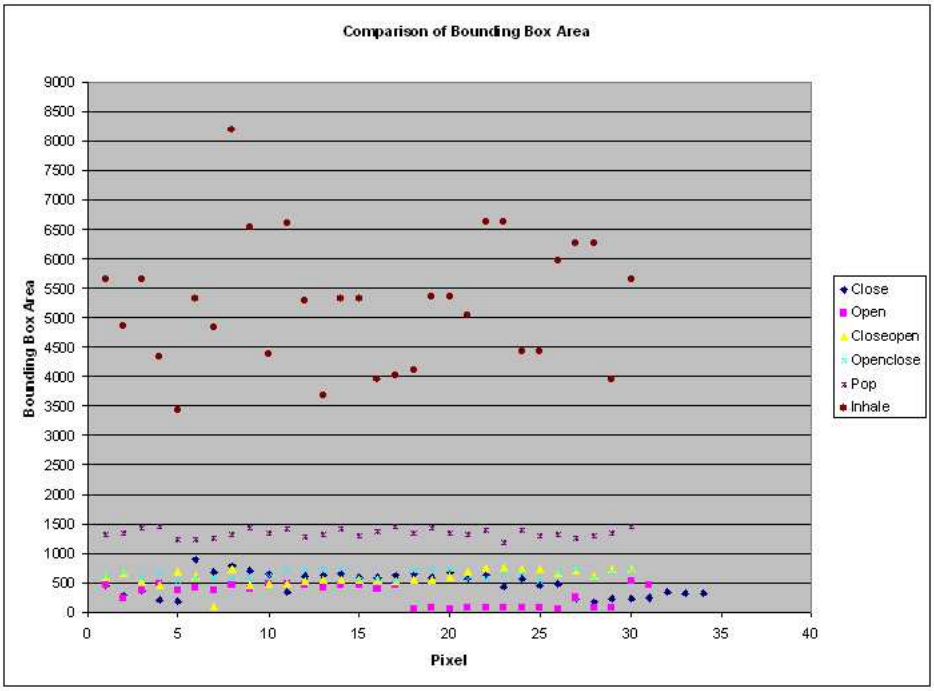


Figure 8.17: Feature Vector Clustering - Bounding Box Area

Appendix E

This appendix presents additional information on MIDI and other symbolic music representations.

MIDI

It is often a common misconception that MIDI is a symbolic music representation but it has no representational capabilities. A Standard MIDI File (SMF) contains time stamped data in a format specially designed to work with MIDI hardware devices. It is available in three file type formats. Format 0 is vertically one dimensional and is best used for monophonic music. Format 1 is intended for music with multiple voices and is essential for music with tracks that are melodically independent. Each voice occupies a single track and each track is vertically one dimensional. Format 2 is horizontally one dimensional and it accommodates music with tracks that are temporally independent.

As a potential format for data interchange, SMFs have the disadvantage that they represent relatively few musical attributes. They provide a poor basis for translation to formats that support notation programs. When MIDI data is used as input to commercial printing programs, it is not able to record non sounding information relevant to the printed page, for example rests, nor can it make any enharmonic distinction of pitch [37].

Other Symbolic Music Representations

It is necessary to also mention some other symbolic music representations that have formed the foundations of many of the representation used today.

SMDL was the first mark up language for music with an aim to define an interchange abstract model

for symbolic music representation. It was divided into four sections; logical (symbolic), gestural, visual and analytical. The logical aspect included the music content consisting of pitches, rhythms, dynamics and articulation markings. The visual domain described the musical typographic details of the score including information about the symbol, its placing and the font types used. It was discovered that SMDL could not be used as a standard interchange format for the visual aspect due to the lack of a formal model for the representation of visual aspects. This lack of formal model resulted in a lack of tools for editing and printing [71].

MNML was a simplified version of SMDL which was capable of representing a basic melody and lyrics of a music piece. It was incomplete in its description of the piece as it failed to accommodate dynamic markings and articulation. For this reason it was not widely adopted [71].

Digital Alternate Representation of Musical Scores (DARMS) was developed to provide a means for writing music with an ordinary keyboard thus representing all elements of musical notation alphanumerically. The efficiency of DARMS code is facilitated by the syntax that is highly sensitive to the order and function of individual elements. It represents a specific set of characters and symbols and codes that fail to appear in a prescribed position may be misinterpreted or ignored. Since it represented a specific data set its nature is static therefore it is difficult, if impossible, to add new symbols. Also the characters are represented by ASCII code, which is limited to the number of characters and symbols that can be represented to the number of available ASCII codes [62].

MuTeX, MusicTeX and MusicTeX, classified under the M*TeX family, were ASCII representation, public domain sets of macros for music typography that operated within the TeX typesetting environment. It was possible to use them for both music typesetting and for setting musical scores and parts for a wide range of repertories. MuTeX was developed in 1987 for the setting of monophonic music and had the ability to accommodate lyrics. MusiTeX supported scores of up to six instruments. Since it was a macro package operating on top of TeX it was possible to run on nearly any platform or base and it provided a device independent output format. MusicTeX was intended for the representation of orchestral scores and polyphonic music. A full range of graphical symbols as well as beams and slurs were available and maybe positioned at any place using a mark-up language which describes musical attributes such as pitch and duration. The demise of this representation was due to lack of funding and interest as it was never formally adopted as a symbolic music representation [39].

Appendix F

This appendix presents information on storage methods, digitisation projects and file formats.

Portable Media

Portable media is a media which is easily transferable from one place to another. Therefore, it has a limited capacity per unit and is extendible only by increasing the number of units. Portable media can be used for storing back up copies of data and local archiving. The medium ranges from a floppy disk which can hold 1.4MB to flash drives holding 512MB to DVDs which hold 17GB. Portable media are susceptible to technological changes and their degree of platform compatibility and driver requirements are variable [16], [52], [35].

Non Portable Media

Non portable media generally provide much larger storage capacities and transfer rates, which are dependant on the make and model of the media. This type of media is usually used to store work in progress, back ups and local archives of the digitised data. Non portable media includes hard drives, network drives and data warehousing [16], [52], [35].

Minerva

Minerva stands for Ministerial NETWORK for Valorising Activities in digitisation. The original aim of the project was to create a network of member states' ministries to discuss, correlate and harmonise activities carried out in digitisation of cultural and scientific content to create an agreed European

common platform and present recommendations and guidelines about digitisation, meta data, long-term accessibility and preservation. This was extended in February 2004 to MinveraPLUS which built upon the already achieved results to pave the ways to a full integration of new countries in the existing European Mainstream with the aims to coordinate national programmes in the area of national digitisation activities [16].

PULMAN

The aim of the PULMAN network is to stimulate and promote the sharing of policies, in public libraries and cultural organisations which operate at local and regional level. They plan to compile and publish digital guideline manuals covering all aspects of the innovative public library server provision, such as support for access to culture online and digital literacy [52].

AHDS

The mission of the AHDS (Arts and Humanities Data Service) is to support, research, learning and teaching by providing visual arts digital resources through robust systems for Internet access and long term preservation. They aim to; provide collections of visual arts digital resources and advice for their creation and use, to preserve visual arts digital resources to ensure their long term use and to promote good practice for the creation and use of visual arts digital resources [35].

BMP

According to Kay [42], bitmap representation is the most commonly used image file format because it is easy to implement and within limits, works for any images. Its original intended use is for the display and storage of images. A bitmap representation splits an image into a grid with each pixel recording its light value. Although bitmaps can record virtually any image high resolution bitmaps result in large file sizes. It is possible to save an image in colour or in monochrome format. Since music sheets are usually printed in black and white it is appropriate to save in monochrome format which saves files in their simplest form making interpretation simpler.

TIFF

TIFF was originally intended for data exchange for desktop publishing and related applications. A TIFF file is a bitmap image type stored in a three level hierarchy consisting of a file header, one or more

directories called Image File Directories (IFD) containing codes and their data and the actual image data. The IFD provides an indefinitely long series of 12-byte entries. Each beginning with a numeric code called a tag that specifies the type of data that follows [42]. In the context of implementing the prototype extra code would need to be written in order to extract the image data from the file format. This file format is supported by most of the commercial OMR software but further testing would need to be completed before deciding upon it.

GIF

This file format was originally intended for the online transmission of graphics data using a bitmap image type. Since its design was primarily as a transmission format for a data stream, rather than as a storage format for a file, it has a sequential file organisation. GIF has five principal components that appear in a fixed order. All of the components are made up of one or more blocks. Each block is distinguished by an identifying code, or tag, in the first byte. The sequence of the components are; the header block; the logical screen descriptor block; an optional "global" colour table block (palette); blocks of image data or special purpose blocks; and the trailer block; a terminating code [42]. Like TIFF, this would take additional code to decode the data in the image format and tests would need to be completed to ensure that the file format can be interpreted by commercial OMR software.

Appendix G

Evaluation table results for commercially available OMR software comparison.

Symbol	**Correct Recognition%**	**Missed Symbol%**	**Added in place of another symbol**	**Incorrect symbol added in place%**
Close	45.95	51.35	2	2.70
Open	31.11	64.44	0	4.44
Close-open	0	100	0	0
Open-close	0	71.43	0	28.57
Mouthpiece Pop	0	100	0	0
Inhale air from instrument	0	100	0	0
Sharp	35.71	62.50	3	1.79
Natural	7.41	88.89	5	3.70
Flat	12.65	84.94	1	2.41

Table 8.1: Evaluation of PhotoScore Professional 4

Symbol	**Correct Recognition%**	**Missed Symbol%**	**Added in place of another symbol**	**Incorrect symbol added in place%**
Close	0	97.30	0	2.70
Open	24	76	1	0
Close-open	0	75	0	25
Open-close	0	100	0	0
Mouthpiece Pop	0	100	0	0
Inhale air from instrument	0	100	0	0
Sharp	44	56	0	0
Natural	2	98	0	0
Flat	27	73	3	0

Table 8.2: Evaluation of SmartScore 5 Professional Edition

Symbol	**Correct Recognition%**	**Missed Symbol%**	**Added in place of another symbol**	**Incorrect symbol added in place%**
Close	0	95.95	0	4.05
Open	0	100	1	0
Close-open	0	100	0	0
Open-close	0	85.71	0	14.29
Mouthpiece Pop	0	100	0	0
Inhale air from instrument	0	100	0	0
Sharp	36.61	63.39	0	0
Natural	7.41	92.59	0	0
Flat	17.47	82.53	0	0

Table 8.3: Evaluation of SharpEye2

Symbol	**Correct Recognition%**	**Missed Symbol%**	**Added in place of another symbol**	**Incorrect symbol added in place%**
Close	39.19	51.35	0	9.46
Open	53.33	44.44	0	2.22
Close-open	50	50	0	0
Open-close	57.14	42.86	0	0
Mouthpiece Pop	0	66.67	0	33.33
Inhale air from instrument	0	100	0	0
Sharp	0	100	0	0
Natural	0	100	0	0
Flat	0	100	0	0

Table 8.4: Evaluation of Capella-Scan

Appendix H - Project Management

This appendix presents the various time plans which were produced in order to ensure the project was completed on time.

Date	Task
21/10/05	**Complete project aim and minimum requirements form**
26/10/05 to 18/11/05	Background Research
18/11/05 to 27/11/05	Write up background research for mid project report
28/11/05 to 09/12/05	Formulate mid-project report
09/12/05 9am	**Hand in mid-project report**
09/12/05 5pm	**Hand in electronic copy of mid-project report**
10/12/05 to 15/12/05	Evaluate commercial OMR software applications
10/12/05 to 23/12/05	Plan for programming prototype
23/12/05 to 23/02/06	Write software prototype
10/02/06	Prepare draft chapter
23/02/06 to 03/03/06	Test software prototype
04/03/06 to 19/03/06	Evaluate software prototype
10/03/06	**Submit contents table and draft chapter**
02/05/06	**Submit project report**
03/05/06	**Submit electronic copy of project report**

Table 8.5: Original Project Schedule

Date	Task
26/10/05 to 30/10/05	Overview of OMR
30/10/05 to 03/11/05	Different approaches to OMR
03/11/05 to 07/11/05	Stave line identification
07/11/05 to 11/11/05	Musical object location
11/11/05 to 16/11/05	Symbol identification
16/11/05 to 18/11/05	Musical semantics

Table 8.6: Detailed Plan of Background Research

Date	Task
10/12/05 to 12/12/05	Research evaluation techniques
12/12/05 to 13/12/05	Research commercial OMR Software
13/12/05 to 14/12/05	Perform evaluation
14/12/05 to 15/12/05	Write up results

Table 8.7: Detailed Plan of Evaluation of OMR Software

Date	Task
27/03/06	Implement boundary pixel identification
27/03/06	Finish slides for progress meeting
28/03/06	Implement boundary pixel labels
28/03/06	Progress meeting preparation
29/03/06 4pm	**Progress Meeting**
29/03/06	Implement labelling and bounding box algorithm
31/03/06	**Finish boundary pixel identification, pixel labels, aspect ratio**
02/04/06	**Redraft programming language selection**
03/03/06	Implement feature vectors
04/04/06	**Redraft symbolic music representation chapter**
05/03/06	Implement k-NN classifier
06/04/06	**Redraft input devices**
07/04/06	**Complete test plan**
10/04/06	**Complete implementation**
11/04/06	**Redraft comparison of commercial OMR systems**
13/04/06	**Redraft digitisation standards**
16/04/06	**Draft of implementation chapter**
21/04/06	**Draft of results analysis and conclusion chapter**
24/04/06	**Draft of evaluation chapter**
24/04/06 to 01/05/06	Proof reading
25/04/06	**Draft of Appendix A**
26/04/06	**Draft of Appendices**
02/05/06 5pm	**Hand in project report**
03/05/06 5pm	**Hand in electronic copy of project report**

Table 8.8: Detailed Plan to Finish Project