



Recognizing Musical Symbols Using Optical Music Recognition

By

Suha Karkhy Mohammed

University of Kurdistan Hewlêr

Erbil, Kurdistan

BSc

June 2022



Recognizing Musical Symbols Using Optical Music Recognition

By

Suha Karkhy Mohammed

Computer Science, Department of Computer Science and Engineering

Student Number: 01-18-00036

A thesis submitted in partial fulfilment of the requirements for the degree of

Bachelors of Science

Department of Computer Science and Engineering

School of Science and Engineering

University of Kurdistan Hewlêr

BSc

June 2022

Erbil, Kurdistan

Declaration

I hereby declare that this dissertation/thesis entitled: "Recognizing Musical Symbols Using Optical Music Recognition" is my own original work and hereby certify that unless stated, all work contained within this is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgement is made in the text.

Signature:

Name: Suha Karkhy Mohammed

Date: June 26th, 2022

Supervisor's Certificate

This dissertation/thesis has been written under my supervision and has been submitted for the award of the degree of Bachelor's in Computer Science with my approval as supervisor.

Signature

Name

Date

I confirm that all the requirements have been fulfilled.

Signature

Name

Head of Department of Computer
Science and Engineering

Date

I confirm that all the requirements have been fulfilled.

Signature

Name

Dean of School of Science and Engineering

Date

Examining Committee Certification

We certify that we have read this dissertation/thesis:

.....

and as a committee have examined the student:

.....

in its content and what is related to it. We approve that it meets the standards of a dissertation/thesis for the degree of BSc in Computer Science.

Signature
Name:
Member
Date

Signature
Name:
Member
Date

Signature
Name:
Supervisor
Date

Signature
Name:
Chairman
Date

Signature
Name:
Dean of the School of
Date

Dedication

I dedicate this to my family, friends, and my professors who helped along the way to come to this point in my life.

Suha Karkhy Mohammed

June 26th, 2022

Acknowledgements

Firstly, I would like to express my utmost gratitude to my supervisor, Dr. Fattah Ibrahim Aziz, for being one of the most helpful and understanding supervisor I could have. It was a privilege to do this research under his supervision. Secondly, I would like to thank my parents for always being there for me when I needed them. Thirdly, I would like to thank my siblings, Sara, Marwa, and Ahmed for always having my back. Fourthly, I would like to thank my close relatives, Sanaria Rafid, Hala Mazin, and Tara Zuhair. Lastly, I would like to thank my friends for always helping and being there for me when I needed them, Parinaz Ahmadifar, Glar Sabah, Sanar Sabah, Sazan Salar, Omar Dler, Haroon Dizayee, Sara Bozan, Abdulla Mohammed, Shana Barwari, Asma Mahmood, and last but not least Yehya Zebari.

Abstract

For the past 50 years, there have been multiple researchers who tried to ease the process of musical compositions for composers. The researchers specifically try to develop systems that aid in detecting scanned images of musical compositions and converting them into machine-readable code called Optical Music Recognition (OMR). Therefore, this research builds an OMR system that recognizes monophonic musical compositions and convert them into a MIDI file, which is a file format for music. The data was extracted from the PrIMuS (Printed Images of Music Staves) dataset, which contains music incipits. This research discusses the steps that undertook which were the preprocessing stage, staff line detection and removal, music symbol detection and recognition, and conversion into a MIDI file. The last step gathers everything and creates a MIDI file as an output. The results show that using traditional image processing does not deal with all cases of the variability. The accuracy of Template Matching for each symbol was obtained by generating 50 random test cases from the PrIMuS dataset, and the same method was used for the pitch accuracy. The system showed an 83.8% accuracy in recognizing with each individual symbol detected, and an 74.4% pitch accuracy on the detected symbols.

Table of Contents

Declaration	II
Supervisor's Certificate.....	III
Examining Committee Certification	IV
Dedication	V
Acknowledgements	VI
Abstract	VII
Table of Contents	VIII
List of Tables	X
List of figures	XI
List of Abbreviations	XIII
Chapter 1: Introduction.....	1
1.1 Problem Statement.....	2
1.2 Objective.....	3
1.3 Thesis Organization	3
Chapter 2: Musical Notation	5
Chapter 3: Literature Review.....	11
3.1 Traditional Approaches	13
3.2 Modern Approaches.....	16
3.3 Binarization for OMR.....	19
3.4 Staff Line Detection and Removal	20
3.5 Conclusion	20
Chapter 4: Methodology	22

4.1 PrIMuS Dataset	22
4.2 Python as the Implementation Environment	24
4.3 The Preprocessing Stage	24
4.4 Staff Line Detection and Removal	25
4.4.1 Staff Line Detection	26
4.4.2 Staff Line Removal	28
4.5 Symbol Detection and Recognition	29
4.5.1 Clefs and Time Signatures	32
4.5.2 Rhythmic Symbols.....	33
4.6 Musical Reconstruction and Output	35
4.7 Summary	35
Chapter 5: Results and Discussion	36
5.1 Application to Showcase the OMR System.....	38
Chapter 6: Conclusion	40
6.1 Future Work	40
References	R1

List of Tables

Table 4.1 The symbols for the Template Matching function	30
Table 5.1 The accuracy rate of each symbol detected.	36

List of figures

Figure 2.1 Staff lines and ledger lines.	5
Figure 2.2 The symbolic representation of the Treble and Bass clefs.	6
Figure 2.3 Chart of notes corresponding with each position in the staff.	6
Figure 2.4 The symbolic representation of accidentals.	7
Figure 2.5 The time and key signatures.	7
Figure 2.6 The features of a musical symbol.	8
Figure 2.7 The features that define a note (PhamoxMusic n.d.).	8
Figure 2.8 The rhythmic symbols.	9
Figure 2.9 A beam connecting two semiquavers.	9
Figure 2.10 A beam connecting two semiquavers and a quaver	9
Figure 2.11 Contour in beams.	10
Figure 2.12 A monophonic musical composition.	10
Figure 3.1 Bainbridge and Bell's framework (2001).	12
Figure 3.2 The refined framework by Rebelo et al. (2012).	13
Figure 3.3 The OMR architecture by Wen et al. (2015)	16
Figure 3.4 The results of the binarization methods of Brink and Pendock (1996), Gatos et al (2004), and Otsu (1979) from Burgoyne et al. (2007).	20
Figure 4.1 OMR flowchart.	22
Figure 4.2 Samples of the monophonic incipits from the PrIMuS dataset (A-F).	23
Figure 4.3 Input image from the PrIMuS dataset.	24
Figure 4.4 After the preprocessing step.	25
Figure 4.5 Staff line thickness and space distance.	25
Figure 4.6 The steps to detect the staff lines.	26

Figure 4.7 Eroding the image to extract the staff lines.	27
Figure 4.8 The final result after extending the lines to their original length.....	27
Figure 4.9 The distances between the staff lines.	27
Figure 4.10 The symbols are isolated from the staff lines.	28
Figure 4.11 The steps to recognizing the symbols using Template matching.	29
Figure 4.12 function for recognizing the clefs using Template Matching.	32
Figure 4.13 The dictionary for the clefs and their pitch.	32
Figure 4.14 Finding the pitch for the rhythmic symbol	33
Figure 4.15 The horizontal coordination of the origin of a quarter note's head that is pointing upwards.	34
Figure 4.16 The recognized symbols with their associated pitch.....	35
Figure 5.1 The results of the OMR system on a distorted image.....	38
Figure 5.2 The results of the OMR system with the same image.	38
Figure 5.3 The first route (OMR page).	39
Figure 5.4 The second route (Test Cases Page).....	39

List of Abbreviations

CNN	Convolutional Neural Network
MIDI	Musical Instrument Digital Interface
MLP	Multi-Layer Perception
NN	Nearest Neighbor
OCR	Optical Character Recognition
OMR	Optical Music Recognition
PrIMuS	Printed Images of Music Staves
R-CNN	Region-based Convolutional Neural Networks
RISM	Répertoire International des Sources Musicales

Chapter 1: Introduction

Music composition has been around for centuries. Countless compositions are produced as copies or original papers. However, papers would eventually be debased and the quality of the musical writings would also decrease with it. Writings would start to become incomprehensible. As computers were starting to popularize, there were multiple musical notation software applications that were designed to aid composers in writing and reading music, with the creation of file formats such as MIDI (Musical Instrument Digital Interface) and MusicXML. These musical notation software applications include MuseScore, Notion 6, ScoreCloud, and many others. With the rapid advancement of technology, there are systems built that can store musical compositions by reading scanned images using OMR (Optical Music Recognition). There are many advantages to OMR in addition to storing musical compositions including replaying the audio, modifying the musical scores, musicological analysis, and conversions to other formats. Furthermore, digitizing musical scores from scanned papers will ease the process of editing for composers, and will make searching for musical sheets easier.

OMR is usually confused with OCR (Optical Character Recognition), but OCR is slightly different from OMR for the specific reason that an OMR system must look at the symbols that are within the five staff lines; the way the symbols are placed within the lines determines the pitch. So, recognizing a musical symbol out of context does not give it much semantical meaning alone, which is why it needs the staff lines. Consequently, an OMR system recovers the semantics of the musical composition, but OCR does not.

The purpose of this study is to develop a system that will recognize and convert musical compositions into a machine-readable code. There is a specific framework all the OMR systems follow in order to accomplish their tasks which are preprocessing, symbol recognition, musical notation reconstruction, and the output. These steps are known as the de facto standard for OMR systems. over the years, other frameworks were done and were set as the standard depending on how evolved the technology was (more on frameworks in Chapter 3). The next section

will discuss the challenges in OMR, and what must be done in order to fix these challenges for this proposed system.

1.1 Problem Statement

Musical compositions were written or printed on paper. Papers decrease in their quality in a physical storage area over time due to dust, temperature, and humidity which causes the compositions to have noise. In addition, compositions could completely be lost or dilapidated due to human error. Furthermore, musical composers often create music by playing the tune in their instruments then writing them down on paper. In the case of editing or creating a final draft, composers must write down everything all over again, which causes a great loss of time. The purpose of OMR is for composers to be able to scan their compositions and give it to the OMR system so that it copies all of the contents in the paper into a file format that they are able to use in a musical software. This will help composers to not only save their work in a safe environment so that it is not lost, but also be able to edit, copy, add, or remove anything they would like to change in their music.

The available challenges in the OMR can be seen as a quadruple. Most of the musical scripts might be old and of low-quality. In such documents, a variety of undesired elements such as spots, scratches, lines, symbol distortions, contrast variation, and background distortions can be seen in the image. Removing such artifacts need a stage called pre-processing, a preprocessing method must be done to be able to clear out as much noise of the image as possible and allow the system to work on the data and ignore the white background.

The second challenge is the staff lines. Staff lines are crucial for identifying the pitch, so there must be a specific method to identify the staff lines without letting the symbols tamper with the detection process. Moreover, if a ledger line exists within a certain bar, it must also be detected and considered as an additional line indicating a higher or lower pitch from the original staff lines. In other words, once a ledger line is identified, an extra line must be created and added to the staff lines.

The third challenge is to identify the symbols and classifying them into their correct category (whether it is a note symbol or a symbol that defines a note). For the

recognition step, the detection of the musical notes within the staff lines must be done without allowing the staff lines to interfere with the detection process. The recognition step is the most crucial part of the OMR system. Because once the musical symbols are identified accurately, the classification step would also be done accurately. Speaking of the classification step, the symbols must be classified into their correct groups, whether they are musical notes, clefs, bar lines, accidentals, etc. (further explanations and meanings on musical symbols are explained in Chapter 2). Additionally, the combinations of the musical notes are flexible, meaning that different rhythmic notes can be combined with each other (see Figure 2.6 and Figure 2.7), so there must be an algorithm that will dynamically recognize and classify these variations.

The last challenge is extracting the semantics of the identified symbols. In other words, once all the symbols and staves are identified and classified, the last step should combine everything together and store it in a file such as MusicXML or MIDI, which is a file format for music, as an output.

1.2 Objective

The main purpose of this research is to:

- Build a system that is able to read music sheets and convert them to a file format.
- Composers being able to scan their compositions and give it to the OMR system so that it copies all of the contents in the paper into a file format that they are able to use in a musical software.
- To create a system for composers to not only save their work in a safe environment so that it is not lost, but also be able to edit, copy, add, or remove anything they would like to change in their music.

1.3 Thesis Organization

The next chapter will explain the musical concepts and the meaning behind each musical symbol and terminology. Chapter 3 is the literature review, where other available articles and research papers about OMR will be reviewed and discussed.

Chapter 4 will discuss the methods that is used for the OMR system. Chapter 5 discusses the results of the OMR system. Finally, Chapter 6 is the conclusion and the future work.

Chapter 2: Musical Notation

In order to avoid any confusion when some terms and concepts are used for the system, this chapter introduces the meanings behind musical notation. In addition, by going into the foundation of music and its symbolism, one may better understand the more complex concepts that are to come in the upcoming chapters. Moving onward, musical notation is used to visually represent music by either using printed or handwritten sheets. Since there are different formats to musical notation, this chapter will specifically elaborate on the Modern Western Musical Notation (MasterClass staff, 2020).

A music sheet consists of a staff, treble clef and bass clef, and bar Lines. **Staff lines** are five lines that can identify the pitch. Music sheets do not only consist of five staff lines. There can be extra lines – called **ledger lines** – that determine pitches that are higher or lower than the original lines (see Figure 2.1).

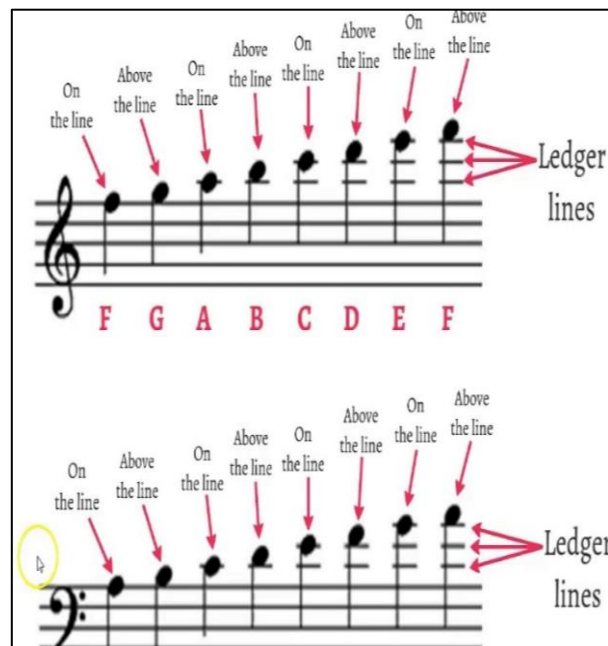


Figure 2.1 Staff lines and ledger lines.

There is a specific pitch, called middle C, that is used to determine whether a note is played before or after it. The **Bass clef** is considered as notes below the middle C, while the **Treble clef** is above it (see Figure 2.2 and Figure 2.3). **Accidentals** are

symbols that are located next to the notes. They indicate that that specific note is played on a different pitch. It can be a flat, natural, or sharp symbol (see Figure 2.4).

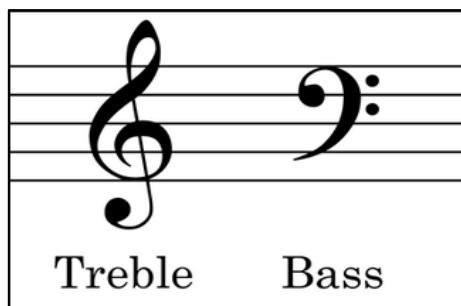


Figure 2.2 The symbolic representation of the Treble and Bass clefs.

The middle C is located in the same key location as demonstrated in the figure above. As the notes move before middle C, then the notes are considered to be in the Bass clef. While as the note move after middle C, then the notes are considered to be in the Treble clef. So, the pitch is different if the corresponding clef is either a Bass or a Treble.

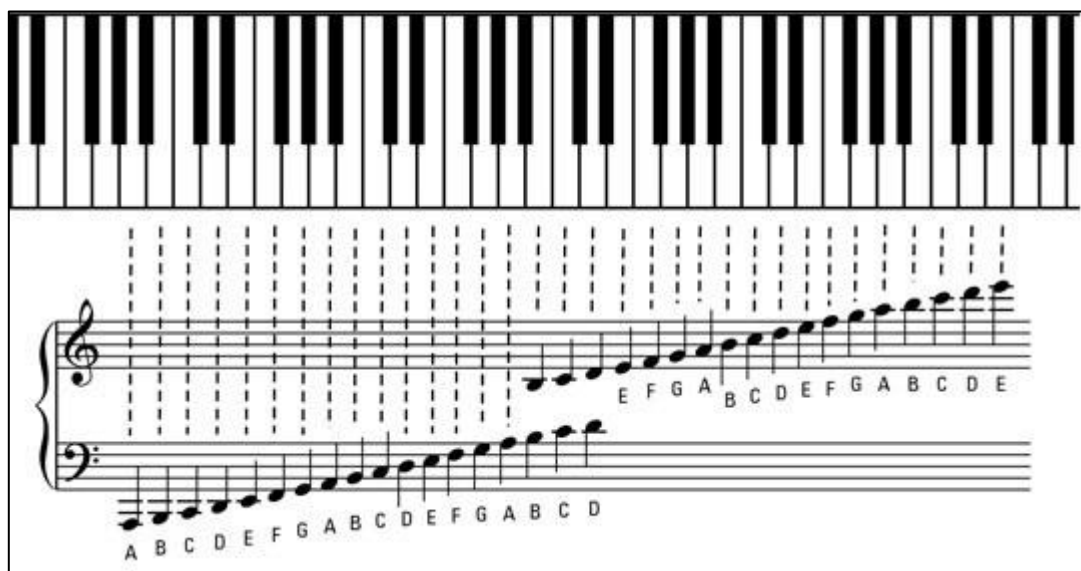


Figure 2.3 Chart of notes corresponding with each position in the staff.

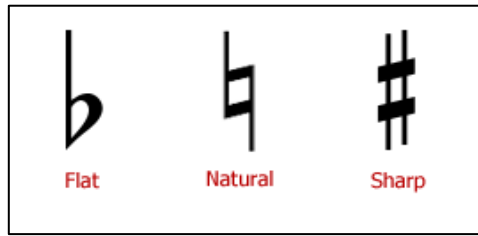


Figure 2.4 The symbolic representation of accidentals.

Time signatures and **key signatures** are located at the top alongside the clefs. The key signature determines if the pitch is sharp or flat, while the time signature determines the number of beats in each measure. In addition, there are **bar lines** within the staff that divides the music into sections (see Figure 2.5).

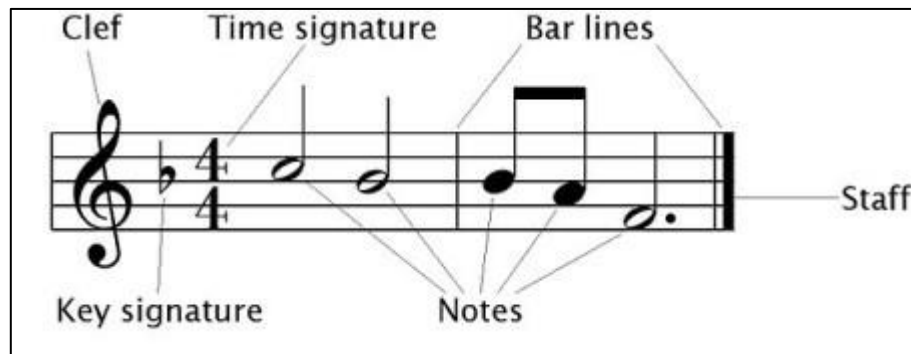


Figure 2.5 The time and key signatures.

Another equally important aspect of music is rhythm. **rhythmic symbols** determine a specific amount of time that a note is played. The symbols can be semibreve, minim, crotchet, quaver, and semiquaver. Each have a specific shape that determines its rhythm. Figure 2.8 shows the rhythmic symbols and their number of beats. **The rest symbols** indicate no notes being played for a specific amount of time. For example, a whole note (semibreve) is four seconds long, a crotchet is one second, and a quaver is a quarter of a second.

There are specific features in a musical symbol that it identifies with. The features of a musical note are its head, stem, and flag. A head can either be a hollow or solid note head, a stem is a vertical line that is connected to the head, and a flag is a hook attached to the stem (see Figure 2.6). These features determine how long the note is to be played depending on the time signature.

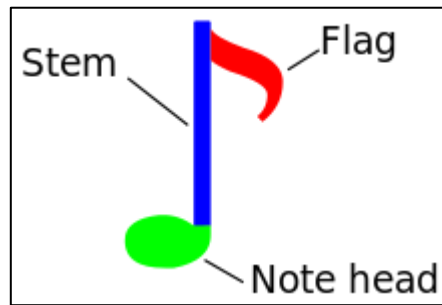


Figure 2.6 The features of a musical symbol.

One feature can differ from another and make up a specific musical note. For example, a hollow note head without a stem or a flag is known as a whole note (semibreve), a hollow head with a stem is known as a minim (half note), a solid head with a stem is known as a crotchet, a solid head with a stem and a flag is known as a quaver, and so on. Beamed notes have all the features of the single notes as well as a horizontal line connecting multiple single notes together. The demonstration of all the features in each note is shown in Figure 2.7.

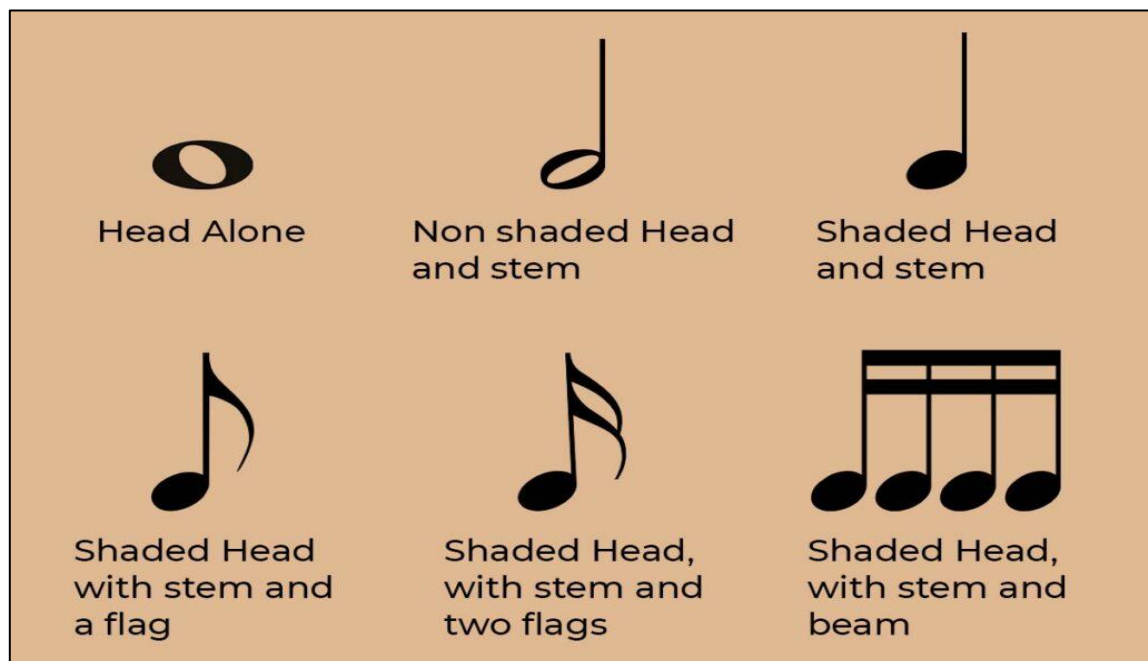


Figure 2.7 The features that define a note (PhamoxMusic n.d.).











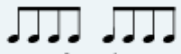




Symbol	Name	Number per bar (4/4)	Rest
	Semibreve	 1 per bar	
	Minim	 2 per bar	
	Crotchet	 4 per bar	
	Quaver	 8 per bar	
	Semiquavers	 16 per bar	

Figure 2.8 The rhythmic symbols.

Beams are horizontal lines that connect multiple notes together. Beams can be a single combination of one rhythmic note, or it can be a combination of different rhythmic notes. For example, Figure 2.9 shows that two semiquavers can be combined with a beam, and Figure 2.10 shows that a quaver can be combined with two semiquavers.



Figure 2.9 A beam connecting two semiquavers.



Figure 2.10 A beam connecting two semiquavers and a quaver

Another important factor to mention for beams is that the lines are not always necessarily straight. **Contour** looks at the first note and the last note in a beam and identifies whether a horizontal line should be ascending, descending, or a straight line (Figure 2.11).

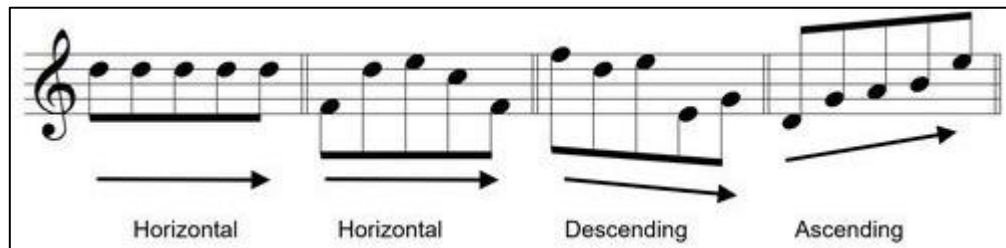


Figure 2.11 Contour in beams.

Combining all these symbols and terminologies together makes up a simple musical composition (Figure 2.12).



Figure 2.12 A monophonic musical composition.

To sum up, there are various aspects to musical notation, and each have a unique meaning and representation. These aspects include staff lines, ledger lines, time signatures, key signatures, clefs, rhythm, rests, beams, and contour. The main goal for the proposed OMR system is to recognize the musical characteristics that were explained in this chapter. The next chapter will discuss the previous works done by other researchers for OMR.

Chapter 3: Literature Review

Recognizing and digitizing musical scores is a step forward towards evolving the methods of writing and reading musical notations. This chapter debates the similarities and differences of the former research works conducted by the leading OMR researchers.

OMR research began in the early 1960s, where scanners were newly affordable. The first digitized musical notation was released by David Prerau in 1971. The more technology progressed, the more innovative designs and frameworks were developed for OMR. The steps for OMR have changed over the years, as there are modern approaches that use deep learning methods and traditional approaches use hand-crafted feature detection/extraction methods.

In the early years of the OMR development, a framework used to be four simple stages that focused mostly on the detection process, specifically the detection of the objects. This framework which was published by Bainbridge and Bell (2001) is shown in Figure 3.1. Many researchers used the general framework including Pugin (2006) who used this specific framework for early typographic prints. While the general framework was a positive start, Rebelo et al. (2012) refined and evolved the framework (see Figure 3.2). The refined framework by Rebelo et al. (2012) is the most used framework by many researchers to this day.

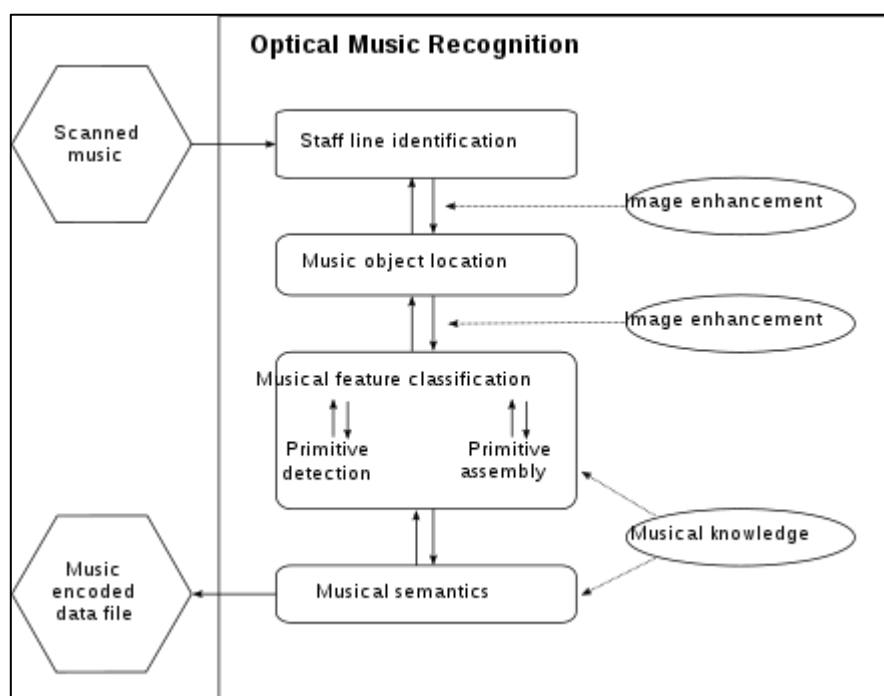


Figure 3.1 Bainbridge and Bell's framework (2001).

The two frameworks have similarities and differences from each other. Figure 3.2 contains the same objectives as the previous framework such as preprocessing, staff detection and removal, symbol detection and classification, and musical reconstruction. Most of the terminologies used in both frameworks for OMR are different but they have the same meaning. However, the steps for the refined framework show the sub-stages within the stages.

This chapter is divided into four sections. The first two sections discuss the specific methods used for the traditional and modern approaches. Sections 3.3 discuss the different types of binarization methods used in OMR. Section 3.4 discusses the staff line detection and removal approach. Sections 3.3 and 3.4 are approaches that are considered for the methodology (Chapter 4). Finally, the last section is the conclusion.

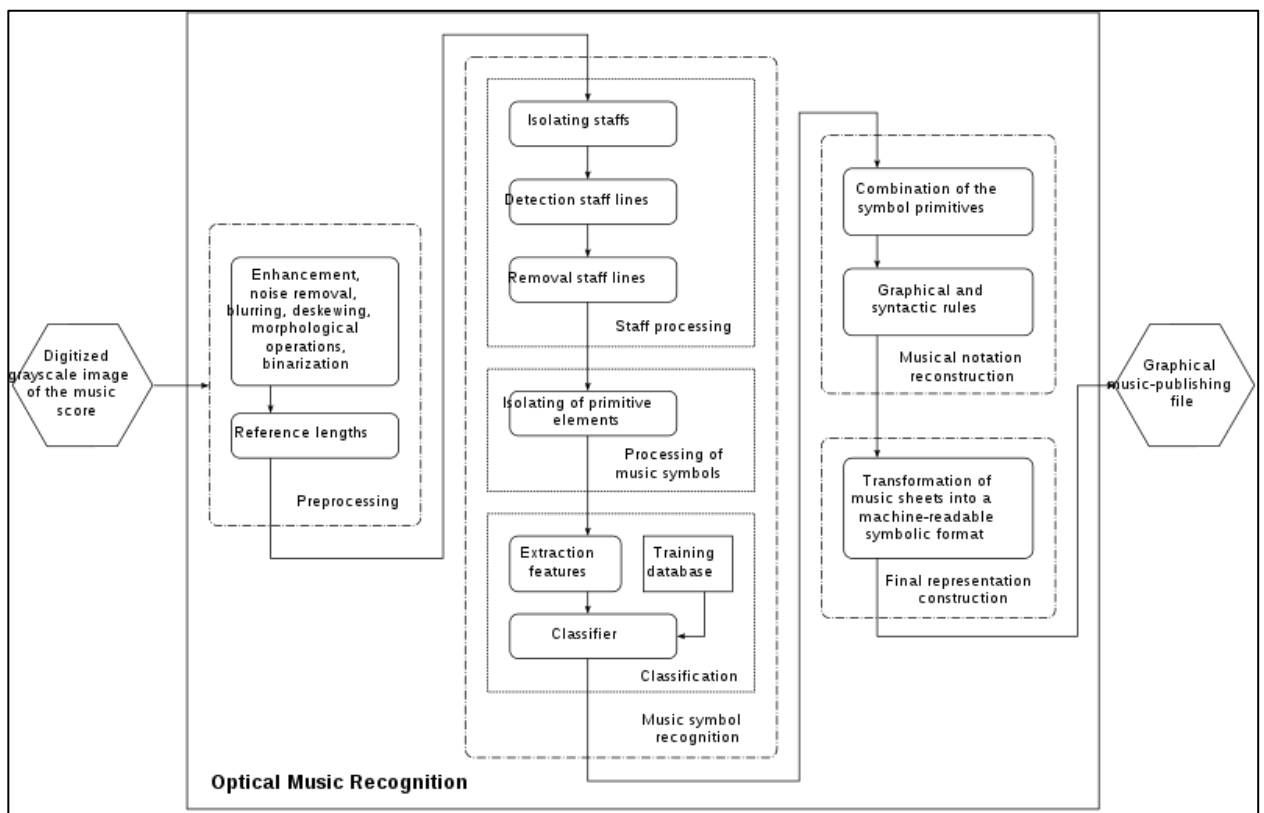


Figure 3.2 The refined framework by Rebelo et al. (2012).

3.1 Traditional Approaches

Traditional methods are considered as methods that did not use deep learning and artificial intelligence. Well-known approaches were based on feature extractions. For example, for line detection and removal, studies from Randriamahefa et al. (1993) and Fujinaga (2004) used Horizontal Projection. As for symbol extraction, methods such as Modified Hough Transform were used by Rossant and Bloch (2006). This section discusses the other traditional methods used that did not include deep learning.

Ng and Boyle et al. (1996) discuss the differences of characteristics of musical scores compared to a character, which includes the horizontal lines called staves, rhythm, the overlapping of symbols within the staff lines, quavers, and the appearance of non-consecutive musical notes within the staff line that make up a chord. They have found many inconsistencies in the scanning area where there was a great deal of noise. Therefore, two strategies are applied to recognize the musical

features: enhance the robustness of the recognition against poor musical elements and follow a phased method by interpreting the score than using an intelligent search for the unclear attributes. The research paper released by Ng and Boyle (1992) was used for the preprocessing phase, which includes the Iterative Thresholding Method of Ridler and Calvard (1978), and skews correction. The next step is the recognition and removal of the staff lines. Once the staff lines are removed, the musical symbols and other features, including bar lines, note heads, or complex musical objects, are left. Before the sub-segmentation phase is started, simple characters are identified. For example, the treble and the bass clef sign, the musical features are specified using a bounding box, the bar lines, and the quaver rests. After the segmentation stage comes to the sub-segmentation location, which is to recognize any musical features that are too large as a primitive feature (instead of the musical part being a single musical characteristic, it is a composition of primitive elements), a primitive feature can be notes, chords, or accidentals. After the sub-segmentation phase, an NN (Nearest Neighbor) classifier recognizes the musical primitives. In addition, the classifier only uses the width and height of the bounding box. The use of the classifier is found to have 90% reliability; it classifies each musical object correctly and only mistyping objects because of unpredicted inter-connection of more than a musical primitive that has no breakpoint. Finally, the reconstruction phase groups the sub-segmented musical construction and reforms its original musical semantics. In conclusion, the results of *vertical and horizontal segmentation* were very robust except when trying to identify the breakpoints if the musical symbols are tilted. However, these minor errors can be fixed in the reconstruction phase. Also, *reconstruction of notes and compound symbols* has few problems if it has the correct classification, and *recognition of accidentals and other characters* in the segmentation phase is also very efficient and does not have much of the misidentified musical features.

In a slightly different work, Randriamhefa et al. (1993) aimed to develop a system that decodes printed music into Braille music. As discussed before, the first step is detecting and removing the staff lines. Two methods were used to detect the staves. Therefore, two results were taken: using the Hough Transform method was

acceptable for some scores but not pleasing for curved and fragmented lines, while the second method utilized the fit line square method to combine crests in the same line. The second method was more robust because it connected fragmented or skewed lines. Once the staff lines are removed, Randriamhefa et al. used Zhang and Suen's (1984) algorithm for thinning digital patterns. Randriamhefa et al. also discussed the technique for recognizing blackheads (crotchet), half notes (minim or a hollow head), bar lines, and other musical symbols. The crotchets vary because they can be infinite combinations (eighth, sixteenth, 32nd note, etc.). Therefore, to recognize any crotchet, an identification of the basic structures will be the head, stem, and hook. In conclusion, the staff line algorithm was accurate, and the crotchet recognition rate was close to 100%.

Szwoch (2005) uses the same strategies as Ng and Boyle et al. (1996) that include preprocessing and segmentation, staff detection, recognition and interpretation, and edition and correction. However, some of the methods done within the steps differ from Ng and Boyle's. More specifically, Szwoch follows the Guido OMR system. For example, for the preprocessing stage, Szwoch uses Otsu's (1979) and Niblack's (2003) thresholding methods. For staff detection, an original algorithm is used by matching the histograms of the local projections. His original algorithm is very efficient and fast. Finally, Szwoch uses context-free grammar, MusBNF, as musical grammar. More specifically, the representation of musical notes can be revealed in a parse tree. In conclusion, they tested the system's quality with several A4-sized papers with different grades of melodic songs including musical notations that contain chords. The results of a PC with a 2 GHz processor had a processing time of 5–6 seconds compared to SmartScore, which had a processing time of 2.5 seconds. There were three experiments made to test the recognition and the efficiency rate: for the first experiment, good quality musical scores were provided and scanned at 150-400 dpi, which had an average recognition rate of 96%. For the second experiment, bad quality musical scores were supplied and scanned at 300 dpi, an average recognition rate of 85%. The decreasing recognition rate for the second experiment specifies that the thresholding method needs improvement. The third experiment provided a good quality of musical

scores with a digital camera at 200 dpi, which resulted in dark and patchy distortions around the corner of the images. Therefore, the average recognition rate decreased to 80%.

3.2 Modern Approaches

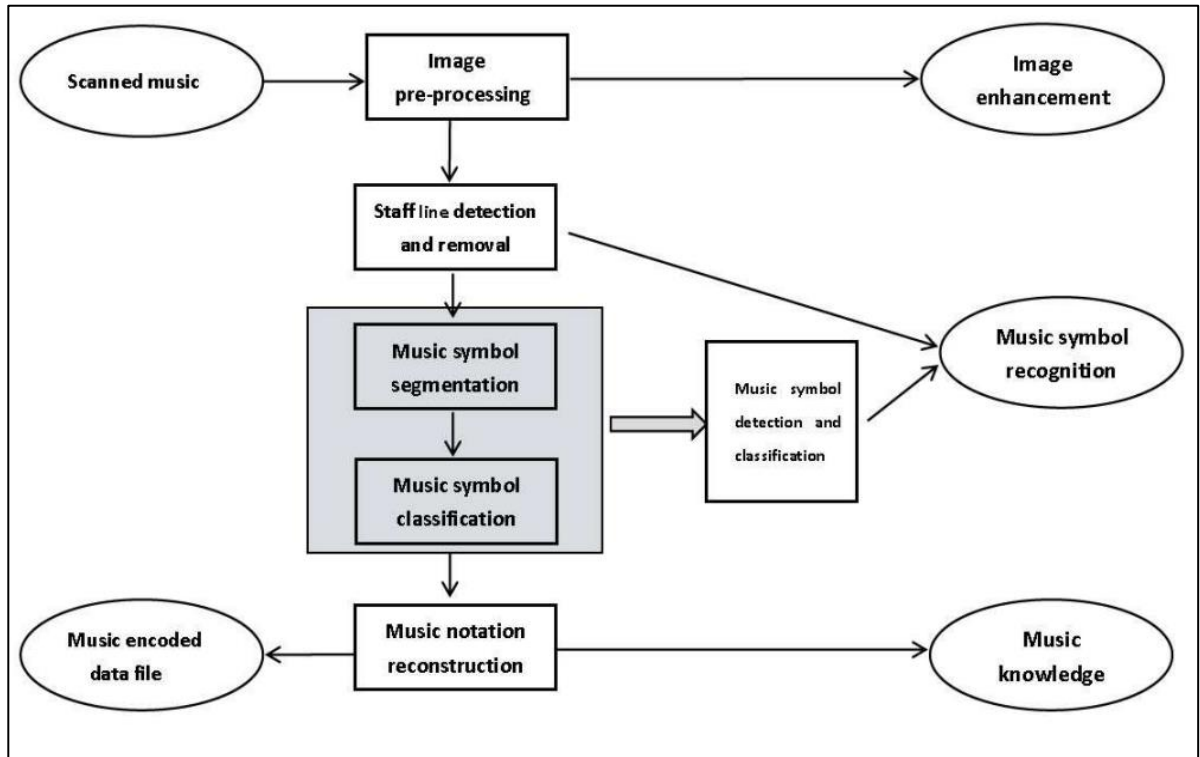


Figure 3.3 The OMR architecture by Wen et al. (2015)

A deep learning approach by Wen et al. (2015) who proposed a type of optical music recognition with the use of Combined Neural Network (CNN). In addition, using the CNN simplifies the Optical Music Recognition (OMR) steps into a lesser number of steps. The usual steps of OMR are image processing, staff line detection and removal, symbol segmentation, symbol classification, and symbol reconstruction. However, with this paper, the symbol segmentation and symbol classification steps are combined (see Figure 3.3). They use binarization and noise removal methods by Otsu's (1979) thresholding method, which is also used by Szwoch (2007). The CNN will classify the symbols based on Lee and Srihari's (1995) theory, which refers to the combination of results of a classifier to get a more enhanced classifier, using a Multi-Layer Perception (MLP). The CNN is trained by given sets of handwritten

and printed musical scores. They used MATLAB to pull out the symbols to teach and train the classifiers. Once the symbols are grouped, a positive level of symbol identification is achieved. An evaluation percentage is calculated of the recognition process, in which the data is divided into three parts which are: training, validation, and test sets. The divisions are repeated 4 times to have a more accurate result. In this case, 2 classifiers were tested: CNN_NETS_20, and CNN_NETS_5. CNN_NETS_20 grouped symbols based on their shapes, whereas CNN_NETS_5 saved objects automatically by dividing them into five classes including all their aspects (note heads, vertical lines, dots, and every other musical symbol). The accuracy rate for CNN_NETS_20 was 98%, while the accuracy rate for CNN_NETS_5 was 92%. Another different technique used is noise processing which takes the failed recognized symbols back for more processing. In conclusion, the average accuracy rate for this system resulted to be approximately 91%-96% for effectively identifying the crotchets, the treble clefs, the sharp symbols, and most of the other symbols. 64% precision rate with symbols that have a lot of noise due to symbols being arranged with additional symbols or are split by the bounding box. Additionally, the rate to recognize if an object is a noise or a symbol is 98%, with 92% precision to remove the noise successfully. However, the rate at which a symbol is correctly recognized decreased to 82% due to the increase of the precision rate.

A recent study made on handwritten musical scores, by Baró et al. (2019), using Convolutional Neural Network (CNN) to extract image features. In addition, they used Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN) to read each staff, in its order. They tested their algorithm on printed documents and handwritten documents. The best results for the printed documents were the Bi-directional LSTM (BLSTM) method, with the use of data augmentation and CNN. As for the handwritten documents, they chose pages from MUSCIMA++ dataset (2017). With the use of BLSTM, the Symbol Error Rate (SER) decreased comparing to the other methods. Also, the system was increasingly able to recognize the musical symbols using CNN.

In a similar context, Pacha and Calvo-Zaragoza (2018) presented an algorithm using deep neural networks with handwritten mensural musical scores. They proposed an algorithm that contains Region-based Convolutional Neural Networks (R-CNN). The stages of their work were object detection, classification of the position, and reconstruction and coding. The experiments included 60 pages of mensural representation from the 16th-18th century, in which all were taken by a camera. The precision rate for the dots, bar lines, and rests was 10%-40%, which means it incorrectly classified parts of the script. Using the InceptionResNet-v2, it accomplishes 98% accuracy for detecting the position from 1318 models and a 76% accuracy rate for detecting the symbols. In addition, the average time it takes to detect and classify on a GPU is 30 seconds, while 210 seconds on a CPU. Therefore, the approach needs further improvements in detecting small objects.

Another Deep learning-based approach has been presented by Calvo-Zaragoza and Rizo (2018). They studied CNN in an end-to-end method with monophonic musical symbols, using convolutional and recurrent NN. The Printed Images of Music Staves (PrIMuS) is a training-based dataset that contains 87678 music incipits. The evaluation is measured in Sequence Error Rate and Symbol Error Rate. According to their research, there are two types of representations: the agnostic, and the semantic. The agnostic representation concentrates on the graphical point of view, while the semantic representation reconstructs the symbols into a code. The sequence and symbol ER for the agnostic representation is higher than the semantic representation. The most common errors for the agnostic and the semantic representation are the bar lines. However, the semantic representation performs better. In conclusion, using the neural end-to-end method is successful however further studies must be done to be able to use the neural approach for polyphonic symbols.

Van Der Wel and Ullrich (2017) built a Convolutional Sequence model to have a trainable Optical Music Recognition (OMR) pipeline, and reduce the amount of data needed for training. In this case, the combination of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) was used. This method proposes that it would be possible to pull away from symbol segmentation and

detection. Multiple image augmentations were used and evaluated, to have a stronger model towards low-quality images and a variety of musical fonts. As a result, the model showed that it is effectively robust towards noisy images. However, the negative aspect of the model is that it needs further improvement in pitch classification. Finally, the experimentations of the pitch and duration were only tested on monophonic musical scores. Therefore, further study is required for polyphonic scores.

Pacha and Eidenberger (2017) worked on building a Self-Learning Optical Music Recognition. In other words, the model objectives are to understand musical scores automatically in images. The algorithm aims to solve challenges the other algorithms faced towards generalizing a set of musical scores, scores with noise, and scores with different fonts. Therefore, the system's aim was to learn to mimic the way a person reads and identifies musical scores by conceiving and extracting features and rules on its own. Two experiments were made, and two questions were asked (Q-I and Q-II). Q-I was to recognize and classify the musical scores into scores or others. Q-II was to classify secluded handwritten symbols into 32 classes. The first experiment correctly classified 98.5% of the images. The second experiment had conditions concerning the size of the image, how thick a stroke is, and covered staves. The second experiment had an accuracy rate of 98.02% from 1520 images when compared to previous results of 97.26% and 96.01%. In conclusion, the answer to Q-I is humans and machines can get similar results, and the answer to Q-II is machines are able to classify handwritten symbols into 32 classes.

3.3 Binarization for OMR

A Comparative Survey of Optical Music Recognition on Degraded Musical Sources by Burgoyne et al. (2007) and Pugin et al. (2006) Compared and evaluated image binarization algorithms to musical compositions from the 16th century. Both researchers used a software application, Aruspix, that measures the performance and the precision rate of the binarization algorithms used. Burgoyne et al. used 8,000 images and evaluated different approaches. The most effective system they found was Brink and Pendock's Minimum Cross-Entropy Threshold Selection method (1996). Figure 3.4 displays the results of their binarization methods. However, the

most used binarization method for printed music sheets is Otsu's (1979) thresholding method, which is used by Szwoch (2007), Dai et al. (2012), and Wen et al. (2015).



Figure 3.4 The results of the binarization methods of Brink and Pendock (1996), Gatos et al (2004), and Otsu (1979) from Burgoyne et al. (2007).

3.4 Staff Line Detection and Removal

Dos Santos Cardoso et al. (2009) created an algorithm that would automatically detect the staff lines, where the staves are measured as connected paths between the margins of the page. This method was proposed for the improvement of staff line detection and evading the symbols that are within the lines. As a result, they thought out of two possibilities of errors of staff detection, staff lines being detected incorrectly or missing the detection of staff lines. They compared their algorithm with another algorithm by Dalitz (2008). Their algorithm, the shortest path approach, had a better performance than Dalitz's algorithm. Moreover, the stable path approach improved the staff detection speed. In conclusion, the stable path approach is a much more adaptive algorithm towards a big number of images with different levels of noise.

3.5 Conclusion

These previous researchers have found multiple different ways to implement OMR. In the earlier years of OMR development, researchers such as Ng and Boyle (1996) were using Nearest Neighbor classifiers for symbol classification. As for the modern approaches, most of the researchers, like Baro et al. (2019) and Calvo-Zaragoza and Rizo (2018), used deep learning methods such as Convolutional Neural Network (CNN) to recognize and classify symbols.

This research will use image processing algorithms in order to implement the OMR system. The third and the fourth steps of OMR will follow a slightly different approach from the other researchers. Firstly, Otsu's (1979) thresholding method will be used in the preprocessing stage since it was the most used method by other researchers. Secondly, the staff line detection and removal step will be implemented using morphological operations. Thirdly, the symbol detection and recognition step will be implemented using Template Matching. Finally, the recognized symbols along with their pitch will be assembled together and converted into a MIDI file. The implementation of these steps will be discussed in a detailed manner in the methodology (Chapter four).

Chapter 4: Methodology

The purpose of the current study is to develop an OMR system that reads and converts monophonic musical compositions into machine-readable code. In addition, the proposed OMR system will take printed scores as input – Instead of handwritten scores – which contains single and sequential notes. The steps that will be carried out are the preprocessing stage, staff line detection and removal, symbol detection and recognition, and then producing an output MIDI file. Figure 4.1 shows the flowchart for the OMR system. Each of the specific steps will be explained in the next sections. But firstly, an introduction on the dataset will be elucidated.

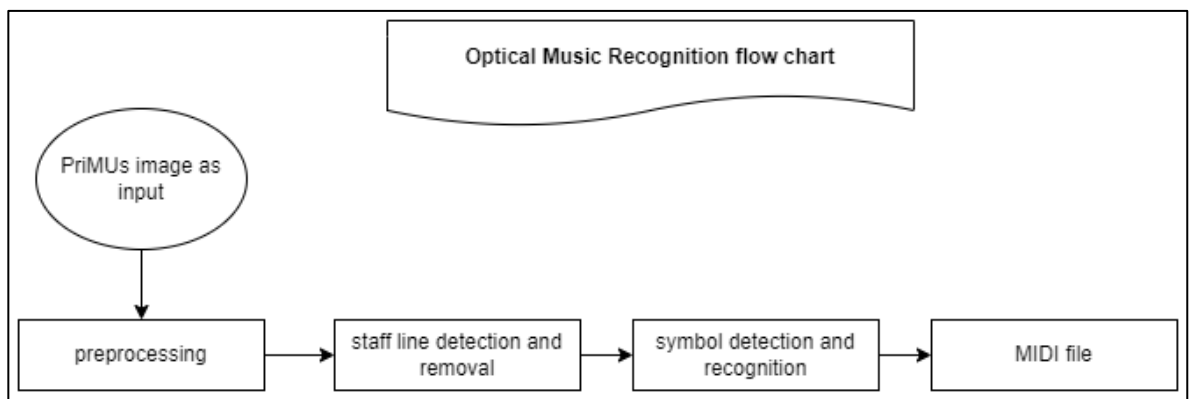


Figure 4.1 OMR flowchart.

4.1 PriMuS Dataset

The *Printed Images of Music Staves, PriMuS*, dataset contains 87678 music incipits. The definition of incipits is the very beginning or introduction of a musical sheet. The PriMuS dataset was made to have as much of the simple monophonic notations as possible. Each of the incipits include a clef, key signature, time signature, rhythmic symbols, rests, and accidentals. The file formats that are available for the PriMuS dataset are in PNG, MEI (Music Encoding Initiative) file format, MIDI (Musical Instrument Digital Interface) file, semantic file, and agnostic file. These incipits were taken from the Répertoire International des Sources Musicales (RISM) inventory, which is an international inventory for music. Figures 4.2 (A-F) shows samples of monophonic incipits from PriMuS.



Figure 4.2 Samples of the monophonic incipits from the PrIMuS dataset (A-F).

The images above are examples that were used as an input for the steps in the OMR system.

4.2 Python as the Implementation Environment

Python is an open-source programming language that contains many useful libraries to ease programming. There are many libraries that were used for the OMR system, which include:

- OpenCV (Open Computer Vision) is used for computer vision tasks.
 - Morphology operations.
 - Contours.
 - Template Matching.
- NumPy is used to work on arrays.
- matplotlib is a visualization library.
- matplotlib.pyplot uses matplotlib to work similar to MATLAB.
- OS is used to work with the operating system.
- Midiutil is used to write MIDI files.

Together, these libraries were used for image processing algorithms, visualization, and reading/writing files.

4.3 The Preprocessing Stage

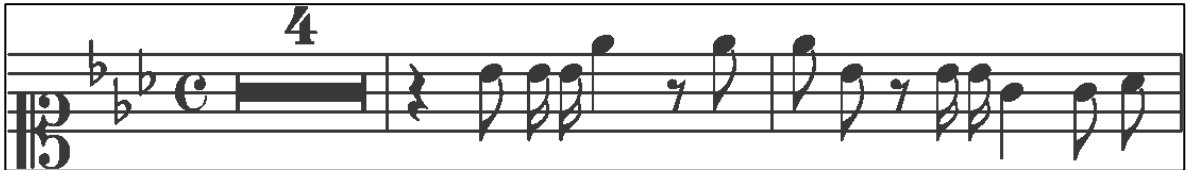


Figure 4.3 Input image from the PrIMuS dataset.

The preprocessing stage is one of the first steps in OMR. Because the PrIMuS dataset does not contain much noise, the only preprocessing that is used in this step is Otsu's (1979) thresholding algorithm for binarization of the images. The image's threshold is then inverted, meaning the black objects will be white and the white background will be black. Beforehand, the images are first converted into grayscale. The images may be in black and white but when they are loaded, they are in RGB channels. Therefore, when it is converted into grayscale, the three-dimensional channel becomes a single channel with gray colors. Figures 4.3 and 4.4 illustrate a sample of the results of the image before and after the preprocessing step.

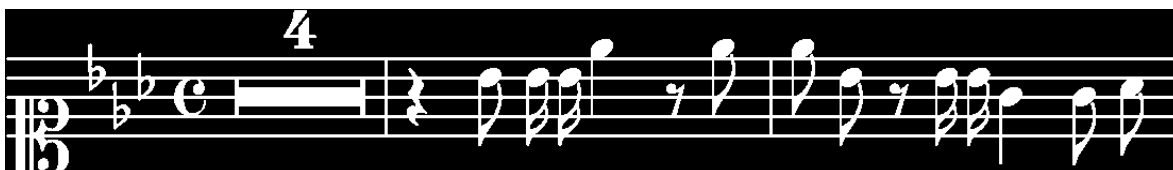


Figure 4.4 After the preprocessing step.

Once the images are preprocessed, it is then ready for the next step. The figures in the following sections are deliberately inverted back to their original color for demonstration purposes, but the background of the images is originally black and the shapes are white just as Figure 4.4.

4.4 Staff Line Detection and Removal

One of the most crucial steps of OMR is the staff line detection and removal. The staff line detection and removal influence two factors in the OMR system which are the pitch detection and the symbol recognition. In other words, if the staff lines are not accurately detected then the pitch will not be accurately detected as well, and it will also decrease its accuracy to recognize the symbols in the symbol recognition step. The staff lines are known to have two characteristics which are the line thickness and space distance. These two characteristics define the pitch when the musical notes are detected. In other words, detecting a note and finding whether it is in the line or in the space. Figure 4.5 shows the staff line thickness and space distance.

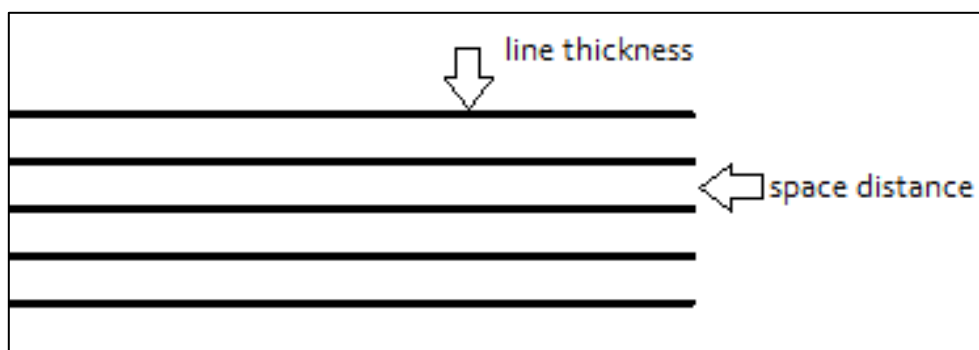


Figure 4.5 Staff line thickness and space distance

This stage is split into two sub-stages, the first sub-stage being the staff line detection step and the second sub-stage being the staff line removal step. The sub-sections elaborate on the image processing algorithms that were used in order to isolate the staff lines from the symbols.

4.4.1 Staff Line Detection

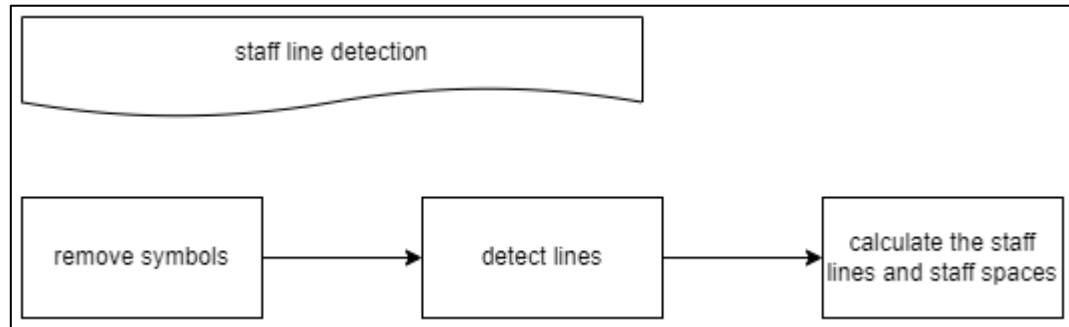


Figure 4.6 The steps to detect the staff lines.

The general steps for the staff line detection are shown in Figure 4.6. The first step to detect the staff lines is to remove the symbols in the image. Therefore, Morphological operations are used to separate the symbols from the staff lines. Morphology is a set of operations that process images based on the shapes in the image. The operations include erosion and dilation. **Dilation** adds pixels to the original image, while **erosion** removes pixels. Dilation is a method where it enhances the shapes in the image. While erosion reduces the shapes into smaller and thinner objects, and only will remove isolated noisy pixels thus only leaves out the important objects. Other than erosion and dilation, there is one more operation used in OpenCV called `getStructuringElement`. A structuring element is a matrix of a binary image, which contains an origin that always compares its values to the original image. Furthermore, a structuring element is the mask or shape that will be used to do the operations. It contains an origin, which is the usually located in the middle of shape. The pixels of the input image get compared to the structuring element and it will either remove or add pixels to the input image. The structuring element that was used for the staff lines is a one-dimensional array of pixels that have values of 1s. Also, the width of the image is divided by two, which gives us our structuring element's length.

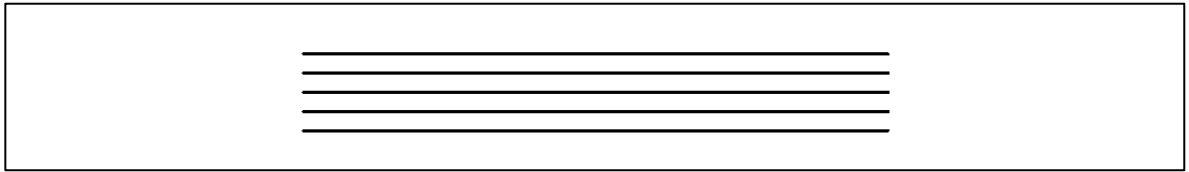


Figure 4.7 Eroding the image to extract the staff lines.

The first morphological operation that is performed on the preprocessed image is erosion, Figure 4.7 shows the result of erosion. The staff lines are much shorter due to the erosion operation; therefore, an extra dilation operation is used to extend the lines. Figure 4.8 demonstrates the final result of the isolated staff lines.

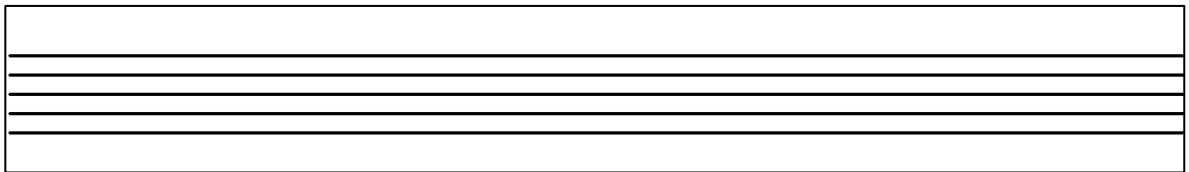


Figure 4.8 The final result after extending the lines to their original length.

Finding the staff distances is the next stage in the staff detection step. Contours are used to find the outline of an object. In this case, the outlines are 5 horizontal lines. Thus, Contours in python will return a NumPy array of the coordination for each of the lines. As a result of that, the horizontal coordination is extracted. For example, using the current sample image (Figure 4.3), the horizontal coordination that is extracted is [45, 63, 81, 99, 117] which is the five horizontal lines



Figure 4.9 The distances between the staff lines.

Using this information, the staff distance will be calculated with the lines representing all of the even index numbers, while the staff space represents all of the odd index numbers in the Staff Distance variable. Hence, the following formula is used:

$$Staff\ Distance[i * 2] = y..... (1)$$

As for the odd index numbers:

$$Staff\ Distance[i * 2 - 1] = \frac{current_y + previous_y}{2}..... (2)$$

Current_y is found using Equation 1, while previous_y represents the previous even index. Furthermore, Equation 2 finds the average distance between the current and the previous even indexes and uses that value to find the staff space distance. As a result, Figure 4.9 demonstrates the distances between the staff lines using Equations 1 and 2.

4.4.2 Staff Line Removal

After the staff lines are detected and the staff distance is calculated, morphological operations are used once again for staff line removal. However, the structuring element is changed from a horizontal array of 1s to a vertical array of 1s. Figure 4.10 shows the result of the isolated symbols. once the symbols are isolated, the next step is to detect and recognize the symbols.



Figure 4.10 The symbols are isolated from the staff lines.

4.5 Symbol Detection and Recognition

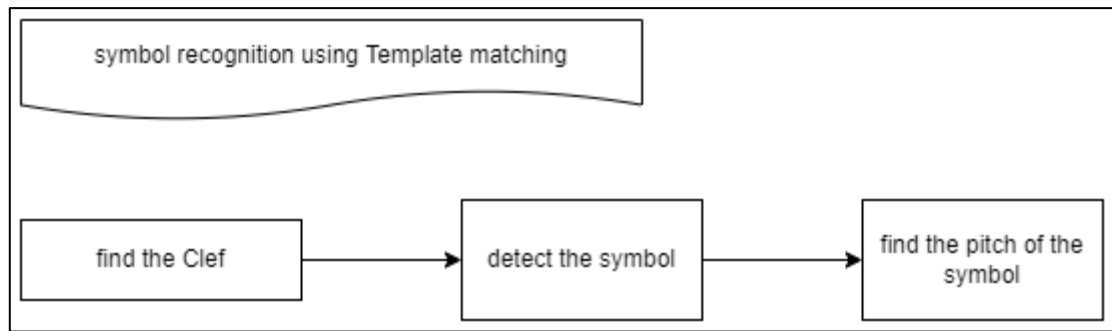


Figure 4.11 The steps to recognizing the symbols using Template matching.



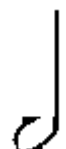
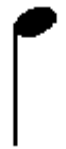
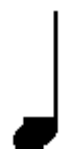





This stage will take the binarized and staff-less image as input and will extract the musical features. The features can be musical notes (eighth notes, whole notes, half notes, and quarter notes), and clefs (G or F or C). Symbols such as rests, clefs, accidentals, time signatures, and key signatures are consistently shaped symbols.








Template Matching is used for the recognition process, and each step is demonstrated in Figure 4.11. Template Matching searches for an image inside a larger image. In this step, the main notes that are targeted for recognition are the clefs, the time signatures, whole notes, half notes, quarter notes, eighth notes, and a sixteenth note. The Template Matching process can be seen as a two-dimensional convolution, meaning that the template images is put on top of an area of the base image. Then the template is compared with that area of the base image then slides to the next area of the image. The template will return an array of how accurate of a match there is of each area of the base image. The result of the array can be seen with the following formula:

$$(W - w + 1, H - h + 1) \dots\dots\dots (3)$$

W stands for the width of the base image and w stands for the width of the template image. Similarly, H stands for the height of the base image and h stands for the height of the template image. The output array will show a percentage of how close the base image's area to the template. In this step, the values with a specific success ratio in the resulting array are considered as a successful match. Template matching is used for recognizing all of the symbols in the dataset. Table 4.1 shows all of the templates that were used to find a match.

Table 4.1 The symbols for the Template Matching function

Music Templates		
Musical Notes	Head Up	Head Down
Whole note		
Half note		
Quarter note		
Eighth note		
Sixteenth note		
Clefs		
C Clef		

G clef	
bass clef	
Time signatures	
4/4	
3/4	
6/4	
2/4	
Common Time (4/4)	

Using Figure 4.10 as an example, each of these templates compare their pixel values in all of the areas of the image. The first combination of templates that are used are the clefs, which will be explained in the next section.

4.5.1 Clefs and Time Signatures

Clefs are the first symbols that are recognized. The templates that are compared are the G Clef, C clef, and the Bass clef. If in any case the none of the templates matched, then there will be a default clef value so that the system does not fail. Figure 4.12 is a code snippet that shows the template matching for the Clefs.

```
# default
CLEF = "g_clef"

for clef in clef_templates:
    clef_img = cv2.bitwise_not(clef_templates[clef])
    try:
        matches = cv2.matchTemplate(symbols, clef_img,
cv2.TM_CCOEFF_NORMED)
        coords = np.where(matches>=0.5) # if template matched to 50%
        if len(coords[0]):
            CLEF=clef
            break
    except:
        continue
```

Figure 4.12 function for recognizing the clefs using Template Matching.

Once the clefs are recognized, then the pitch is identified on the staff. Each of the clefs identify a specific pitch order in the staff. For example, at the topmost line of the staff, if a G clef is recognized then the pitch order for that line is an F5, but if a C clef is identified then it is a G4. Figure 4.13 is a code snippet that shows a python dictionary which contains the clefs with their corresponding pitch in an array.

```
clef_table = {"bass_clef":["A3", "G3", "F3", "E3", "D3", "C3", "B2", "A2", "G2"],
             "c_clef":["G4", "F4", "E4", "D4", "C4", "B3", "A3", "G3", "F3"],
             "g_clef":["F5", "E5", "D5", "C5", "B4", "A4", "G4", "F4", "E4"]}
```

Figure 4.13 The dictionary for the clefs and their pitch.

The first index in the array is considered as the topmost line in the staff lines. As the index increases, the pitch values decrease indicating a much deeper sound. Afterwards the time signature is identified. The time signature's role in this step is only to be identified then it is later used for when the MIDI file is created.

4.5.2 Rhythmic Symbols

After the clefs and the time signatures are recognized, the last recognition step is to find and recognize the rhythmic symbols. There are specific matching percentages that are used in order to pick out the best match. If a match is greater or equal than a specific note, then it will be considered as that note. For example, a whole note's percentage is 65%, half note is 67%, quarter note is 68%, eighth note is 60%, and sixteenth note is 63%. Hence, any match that is below these percentages are not considered. Afterwards, the symbol that matched the most will be considered as that note.

All of the templates go through the base image to try and find a match. Even though there are specific matching percentages, there might still be multiple false matches due to some symbols' similarities to each other. Therefore, the number of matches for each template for the symbol will be counted. Then, the maximum number will be considered as that symbol. Next, the pitch is detected. Figure 4.14 shows the code snippet which is the process for detecting the pitch for the note.

```
def detect_pitch(symbol_y, symbol_h, note_direction, clef):
    note_y = symbol_y + 15 # skips the white space
    if note_direction=="up":
        note_y = symbol_y + 15

    if note_direction=="down":
        note_y = symbol_y + symbol_h - 15

    si = np.argmin(np.abs(staff_ds-note_y)) # staff index of the min value
    return clef_table[clef][si]
```

Figure 4.14 Finding the pitch for the rhythmic symbol

The rhythmic symbols differ from the other musical symbols because of their heads and their stems. For example, if a half note's head points upwards then its stem will point downwards, or the head can point downwards and its stems will point upwards.

Once a rhythmic symbol is recognized, the pitch is detected, which depends on whether the head is pointing upwards or downwards. Furthermore, if the head is pointing upwards, 15 pixels are added to the horizontal coordination of the detected symbol. As for the head pointing downwards, the very bottom of the detected symbol is subtracted by 15 pixels. This is due to the head being at the very bottom of the template image. The reason behind adding or subtracting 15 pixels from the detected image is because the position of the head within the staff lines determines the pitch of that specific musical symbol.

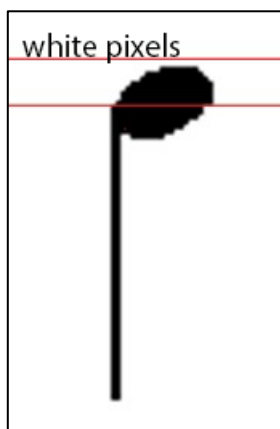


Figure 4.15 The horizontal coordination of the origin of a quarter note's head that is pointing upwards.

The best way to determine the pitch accurately is to find the origin of the note; In this case, the center of the head. Figure 4.15 shows an example of finding the horizontal coordination of a quarter note with its head pointing upwards. Once the horizontal coordinates for the center of the head is found, it is then compared with the horizontal coordination of the staff distances. The index of the minimum value will be the pitch, as it is considered the closest. Looking back at Figure 4.13, the index of the minimum value will correspond with the index of the array for the specific clef. This process will end with giving all of the symbols their name and their pitch. Figure 4.16 shows the recognized rhythmic symbols with their pitch values.

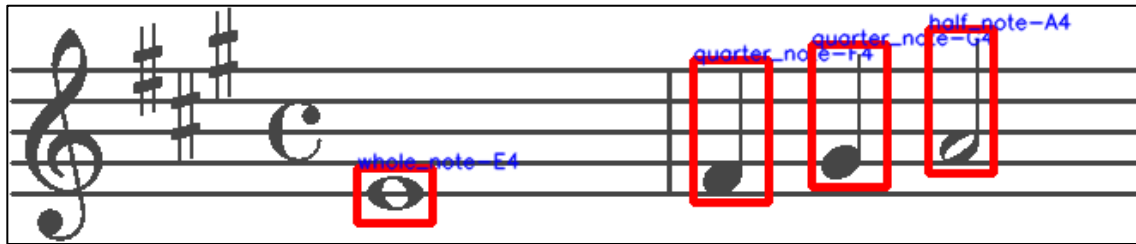


Figure 4.16 The recognized symbols with their associated pitch.

4.6 Musical Reconstruction and Output

The very last step is to assemble everything that is detected and recognized into a MIDI file. Once all the necessary symbols are recognized, it is then all put into a python dictionary with their corresponding bounding boxes.

Python's MIDIUtil library allows the creation of a MIDI file to be written from a python program. Once the assembled MIDI file is formed, the file can be played, or it can be used in a musical software such as MuseScore which allows one to play, edit, add, or delete the musical composition.

4.7 Summary

Firstly, the system first preprocesses the image using a thresholding method. Secondly, the staff lines are detected and isolated from the symbols using morphological operations. By then the staff lines are separated and the symbols are separated. Thirdly, the isolated staff lines are detected and numbered with the help of Contours, which will help in the next process for when the clef and the symbols are detected and recognized. Fourthly, the image will take the isolated symbols and perform Template Matching. The first symbol that it will recognize is the clef so the pitch values are defined, then the time signature. Afterwards, it will detect and recognize the symbols, depending whether the note's direction is upward or downward. Based on the note direction the pitch is identified. Lastly, the staff lines, the pitch, and the symbols will be organized and reconstructed into a MIDI file using python's MIDIUtil library.

Chapter 5: Results and Discussion

Each symbol was looked at individually until 50 units of that symbol was evaluated. The percent accuracy and errors/miss detection were annotated in Table 5.1. All 14 whole note errors were due to the system misreading the number 2 in time signature of 2/4, as a while note. All actual whole notes were detected with 100% accuracy if the misinterpretation of the system is ignored. As regards to the clefs, only the C clef presented errors and that was in the form of the system misrepresenting some C clefs as G clefs. The time signatures had two major errors that were detected and annotated for the cut time and for the 2/4 time. The system misinterpreted all cut time errors annotated as common time due the similarity in the symbol. The 2/4 time and 3/4 time look quite similar as well leading to the system to misinterpret certain 2/4 times as 3/4. The results above were found from randomly generating 50 images at a time until 50 units were marked as detected or not for each row. This method was used manually.

Table 5.1 The accuracy rate of each symbol detected.

Symbols	Number of Errors	Percent Accuracy
Whole notes	14	72%
Half notes	5	90%
Quarter notes	0	100%
Eighth notes	8	84%
Sixteenth notes	15	70%
G clef	0	100%
C clef	29	42%
F clef	0	100%
4/4	0	100%

Symbols	Number of Errors	Percent Accuracy
3/4	0	100%
6/8	0	100%
2/4	19	62%
Cut-time	15	70%

Using template matching gives the system 83.8% accuracy for detecting the main symbols. This is disregarding the other notes for detection. If the other notes were considered for this then the accuracy is estimated to decrease drastically. As for the pitch detection, this system only considers the first five lines and four spaces, the pitch detection accuracy is 74.4% on the symbols that are detected. This accuracy was extracted by getting the number of symbols detected along with the number of pitch values that were detected correctly for each test case.

The system is able to recognize the main symbols including the clefs, time signatures, whole notes, half notes, quarter notes, eighth notes, and sixteenth notes. The system is also able to identify the pitch according to the clef. However, this system has its setbacks. Firstly, this project takes in music incipits which are fairly similar sizes, if a full-sized image of a musical composition is given to the system, it will not consider the other staff lines. There is no dynamic factor that can identify the staff lines according to the image. Secondly, the system is able to recognize staff lines that are straight. In addition, if the lines are bent or the image is slanted, then it will not detect the staff lines accurately. As mentioned before, if the staff lines are not properly preprocessed, then none of the symbols' pitches will be recognized either. Figures 5.1 and 5.2 show the results of a distorted image going through the OMR system and an image with normal quality.

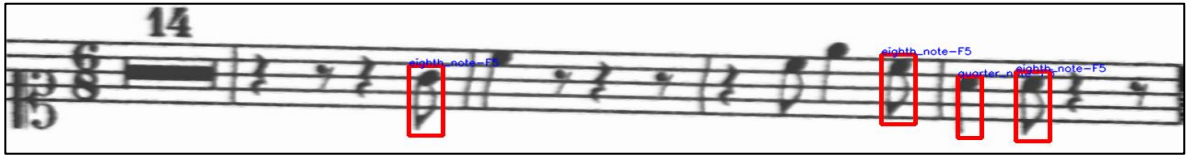


Figure 5.1 The results of the OMR system on a distorted image.

The image above is blurred and slanted. Due to that, the staff lines are not detected in an orderly fashion such as Figure 5.2, which is the same image but without any noise.

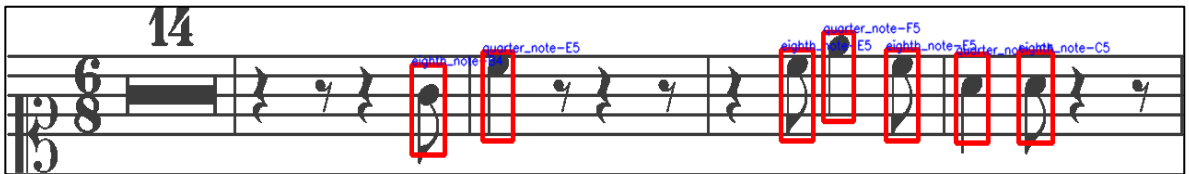


Figure 5.2 The results of the OMR system with the same image.

The detection process runs almost smoothly unless it contains beamed notes. Beamed notes are not considered within this scope of the project to keep the system simple. Thus, the results show that template matching might work on very specific cases for object recognition, but it is inconsistent and is not a dynamic method to detect other symbols such as beamed notes. The system as a whole has an 83.13% accuracy rate with the symbols mentioned in Table 5.1 and the pitch accuracy using images without distortions.

5.1 Application to Showcase the OMR System

A web application is created to showcase the OMR system using Flask, which is a web framework for Python. The website will contain two main routes. For the first route, the user will be able to browse through their own files and choose an image. Once an image is chosen the system will convert it into its MIDI file, and the user will be able play the MIDI file. Figure 5.3 shows the first route of the website. As for the second route, the results of the images from the dataset will be shown along with their original MIDI file and the MIDI file of the OMR system.

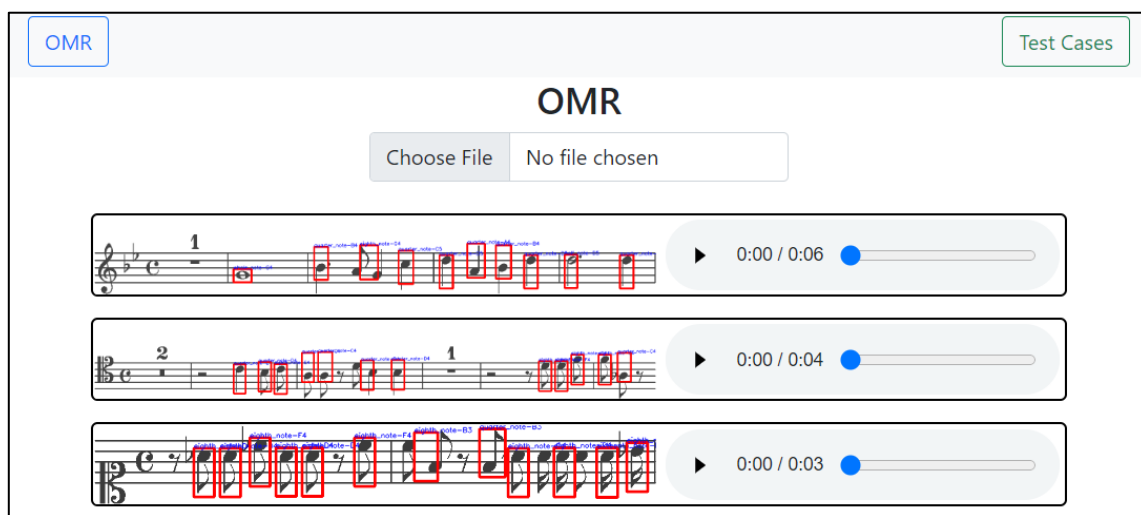


Figure 5.3 The first route (OMR page).

This page is called the OMR page, where the user is able to choose a file from their own computer, but the file has to be an image (PNG or JPG). The detected image will be added in the web page along with its MIDI file. As for the second route, it is called the Test Cases page where multiple random images from the dataset is added with their original MIDI file and the MIDI file that the OMR system generated. Figure 5.4 shows the Test Cases page.

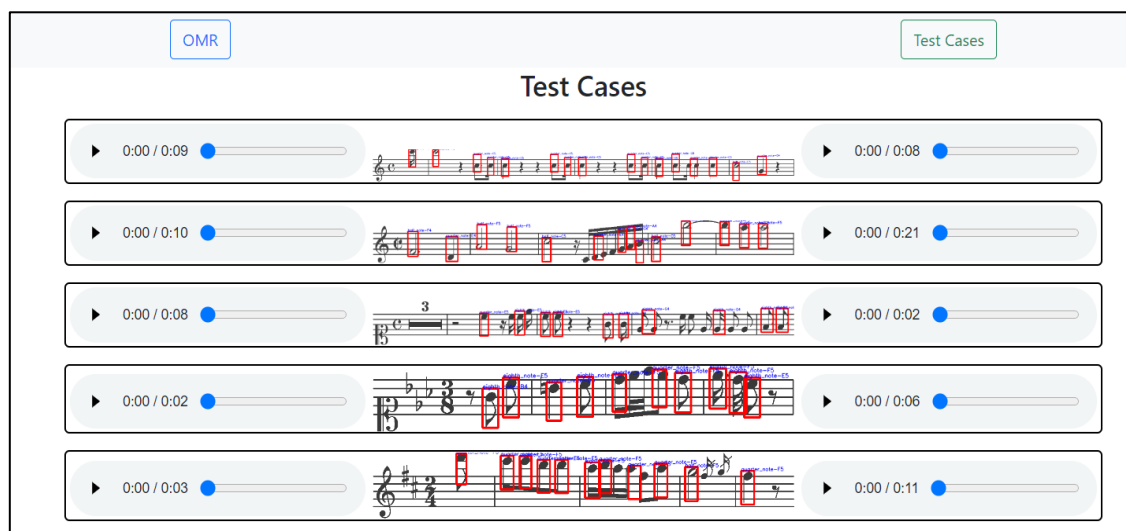


Figure 5.4 The second route (Test Cases Page).

The following link is the source code on GitHub:

<https://github.com/suskarkhy/OMR---Optical-Music-Recognition>

Chapter 6: Conclusion

Optical Musical Recognition is a system that takes a musical composition and converts it to a machine-readable code. The main idea behind this study is to create a program that takes in printed musical compositions and convert the results into a MIDI (Musical Instrument Digital Interface) file. Four steps are implemented for the OMR system, which are preprocessing, staff line detection and removal, symbol detection and recognition, and converting the detected image to a MIDI file. The preprocessing step includes converting the image into grayscale then using Otsu's (1979) thresholding method. The second step, where the staff lines are detected and removed, uses morphological operations which isolates the staff lines from the symbols and numbers the lines. Then, the next step uses Template Matching to recognize the symbols. The output will end up reconstructing all the recognized symbols and their pitch and create a MIDI file. OMR can be useful for musical composers, learners, and music editors. The most significant benefits include saving time, loss prevention, ease of editing/copying, and data storage.

Although using image processing algorithms are helpful but they can only go so far. On the other hand, using only image processing algorithms will not guarantee an accurate result, as it needs to focus on many different characteristics. As discussed, errors in image quality and staff line detection can greatly affect the efficiency of the OMR system. Furthermore, the best method that is recommend for this system is to use a modern approach including computer vision, which focuses on AI (Artificial Intelligence) to turn images into meaningful information.

6.1 Future Work

There are multiple different experiments, algorithms, and tests that were left for the future as a consequence of time. My future work concerns trying out a more modern approach for the OMR system including using computer vision and machine learning techniques. In addition to that, to create my own dataset which would include all of the different formats (handwritten and printed) and fonts for music.

There are some methods that I would have liked to add during the implementation phase, including adding more preprocessing algorithms that would remove noise, remove distortions, and fix the angled images to ease the next steps for OMR. as for the symbol recognition step, improving the recognition step by identifying and classifying the beamed notes, rests, accidentals, and key signatures. One aspect that is considered for the enhancement of this system is to consider the variability of the image.

Lastly, adding error prevention methods that allow users to re-enter the images, that they want to scan, depending on different variables including image quality, unrecognizable symbols, wrong number of staff lines, and other types of human error.

References

- Baró, A., Riba, P., Calvo-Zaragoza, J. and Fornés, A., 2019. From optical music recognition to handwritten music recognition: A baseline. *Pattern Recognition Letters*, 123, pp.1-8.
- Burgoyne, JA., Laurent, B., Greg, P. and Fujinaga, E.I., 2007. A comparative survey of image binarization algorithms for optical recognition on degraded musical sources.
- Brink, A.D., and Pendock, N.E., 1996. Minimum cross-entropy threshold selection. *Pattern recognition*, 29(1), pp.179-188.
- Calvo-Zaragoza, J. and Rizo, D., 2018. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8(4), p.606.
- Dalitz, C., Droettboom, M., Pranzas, B. and Fujinaga, I., 2008. A comparative study of staff removal algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 30(5), pp.753-766.
- Dai, S., Tian, Y. and Lee, C.W., 2012. Optical music recognition and playback on mobile devices. *Department of Electrical Engineering Stanford University, Stanford, CA, 94305*, pp.2012-12.
- dos Santos Cardoso, J., Capela, A., Rebelo, A., Guedes, C. and da Costa, J.P., 2009. Staff detection with stable paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6), pp.1134-1139.
- Fujinaga, I., 2004. Staff detection and removal. In *Visual Perception of Music Notation: On-Line and Off Line Recognition* (pp. 1-39). IGI Global.
- Good, M., 2004. *MusicXML 4.0*. [online] W3.org. Available at: <<https://www.w3.org/2021/06/musicxml40/>> [Accessed 10 February 2022].
- Hajič, J. and Pecina, P., 2017, November. The MUSCIMA++ dataset for handwritten optical music recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (Vol. 1, pp. 39-46). IEEE.

Lee, D.S. and Srihari, S.N., 1995. A theory of classifier combination: the neural network approach. In *Proceedings of 3rd International Conference on Document Analysis and Recognition* (Vol. 1, pp. 42-45). IEEE.

MasterClass staff, 2020. Music 101: *What Is Musical Notation? Learn About the Different Types of Musical Notes and Time Signatures*. Available at: <https://www.masterclass.com/articles/music-101-what-is-musical-notation-learn-about-the-different-types-of-musical-notes-and-time-signatures> (Accessed: 18 Feb 2022)

Ng, KC, and Boyle, R.D., 1996. Recognition and reconstruction of primitives in music scores. *Image and Vision Computing*, 14(1), pp.39-46.

Ng, KC, and Boyle, R.D., 1992. Segmentation of music primitives. In *BMVC92* (pp. 472-480). Springer, London.

Niblack, W., 2003. An introduction to digital image processing (1986) In Graham Leedham and Chen Yan and Kalyan Takru and Joie Hadi Nata Tan and Li Mian, Comparison of Some Thresholding Algorithms for Text/Background Segmentation in Difficult Document Images. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*.

Otsu, N., 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1), pp.62-66.

PhamoxMusic n.d., *The Musical Notes and Their Time Values in Staff Music*. Available at: <https://phamoxmusic.com/musical-notes/> (Accessed: 19 Feb 2022)

Pugin, L., Burgoyne, J.A. and Fujinaga, I., 2007, June. Goal-directed evaluation for the improvement of optical music recognition on early music prints. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries* (pp. 303-304).

Pacha, A. and Calvo-Zaragoza, J., 2018, September. Optical Music Recognition in Mensural Notation with Region-based Convolutional Neural Networks. In *ISMIR* (pp. 240-247).

- Pacha, A. and Eidenberger, H., 2017, December. Towards self-learning optical music recognition. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 795-800). IEEE.
- Randriamahefa, R., Cocquerez, J.P., Fluhr, C., Pepin, F. and Philipp, S., 1993, October. Printed music recognition. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)* (pp. 898-901). IEEE.
- Ridler, T.W. and Colvard, S., 1978. Picture thresholding using an iterative selection method. *IEEEtranss Syst Man Cybern*, 8(8), pp.630-632.
- Rossant, F. and Bloch, I., 2006. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP Journal on Advances in Signal Processing*, 2007, pp.1-25.
- Szwoch, M., 2007, September. Guido: A musical score recognition system. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (Vol. 2, pp. 809-813). IEEE.
- Szwoch, M., 2005, September. A robust detector for distorted music staves. In *International Conference on Computer Analysis of Images and Patterns* (pp. 701-708). Springer, Berlin, Heidelberg.
- van der Wel, E. and Ullrich, K., 2017. Optical music recognition with convolutional sequence-to-sequence models. *arXiv preprint arXiv:1707.04877*.
- Wen, C., Rebelo, A., Zhang, J. and Cardoso, J., 2015. A new optical music recognition system based on combined neural network. *Pattern Recognition Letters*, 58, pp.1-7.
- Zhang, T.Y. and Suen, C.Y., 1984. A fast-parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3), pp.236-239.