Department of Computer Science
Final Year Project Report

Automatic Optical Music Recognition

Lee Sau Dan
University Number: 9210876

Supervisor: Dr. A. K. O. Choi
Second Examiner: Dr. K. P. Chan

Abstract

In this project, the topic of automatic optical music recognition was studied. It is the conversion of an optically sampled image of a musical score into a representation that can be conveniently stored in computer storage and retrieved for various purpose. It is analogous to optical character recognition. Optical character recognition recognizes text characters in the input images and output the text in a machine-readable format. Similarly, an optical music recognition system recognizes the symbols on a musical score and output the results in a binary format. Subsequent processing on this output can provide a wide variety of applications, such as reprinting and archiving.

In section 1, I will give an introduction to this topic. This section includes general discussion on the topic, areas of application of optical music recognition technology, as well as explanation on some technical terms for some common musical symbols.

Focus was put on offline optical music recognition systems in this project. Detailed elaboration of the design issues, difficulties and techniques will be given in section 2 of this report.

Based on the design considerations discussed in section 2, an offline optical music recognition system was built. Implementation details of the system is covered in section 3.

This report closes with conclusions in section 4.

Contents

List of Figures

1 Introduction

Automatic optical music recognition is the automatic analysis of images of musical notations, either written or printed. Research on automatic optical music recognition started early in the 1960s, when the speed and capabilities hardware were still very limited. The earliest researchers on automatic optical music recognition include Pruslin (1967) and Prerau (1970,1971,1975).

1.1 On-line vs. Off-line

Automatic optical music recognition can be roughly classified into two categories: on-line and off-line. In an on-line system, the machine analyses the musical score and generates the result almost instantaneously. Such a system can be attached to devices such as robotic arms adjacent to a piano to perform the piece of musical work in real time. In such a system, the machine must be able to carry out the analysis in a short time. This implies that the system may not have enough time to analyze the whole score before generating its output. Rather, small regions of the score are processed locally and the output is generated immediately. Previous errors may not be corrected at a later stage.

In an offline system, the score is first digitized as an image file and stored. Usually, optical scanners are used and cameras provide an alternative. The stored image is then analyzed by the computer and converted into a binary form using a coding that should be designed to be suitable for both performing the piece of musical work, and re-printing of the score. Since an offline system can analyze the whole score before generating its output, global information of the score can be used to improve the accuracy of recognition. For example, a sophisticated semantic checker can be developed to correct suspected mistakes made in an earlier stage of recognition.

In this project, off-line automatic optical music recognition systems for printed musical scores are studied. The remainder of this report will be discussions on the techniques, difficulties encountered in developing off-line optical music recognition systems for printed music.

1.2 A brief comparison with Optical Character Recognition

Many researchers have made comparison between optical music recognition and optical character recognition. There are many similarities between them. This page of this document is an illustration of a printed document. Figure 1 on page 4 gives an example of a piano score.

Text is printed by placing characters from several fonts onto a blank sheet, while a musical score lays musical symbols, which can be treated as characters in some special fonts, onto a sheet containing a set of of staff lines.[1] A musical score can be divided into groups, with symbols attached to the same staff being considered as a group. These groups

[1]See section 1.5 for explanation of some terms that frequently occur in this report.

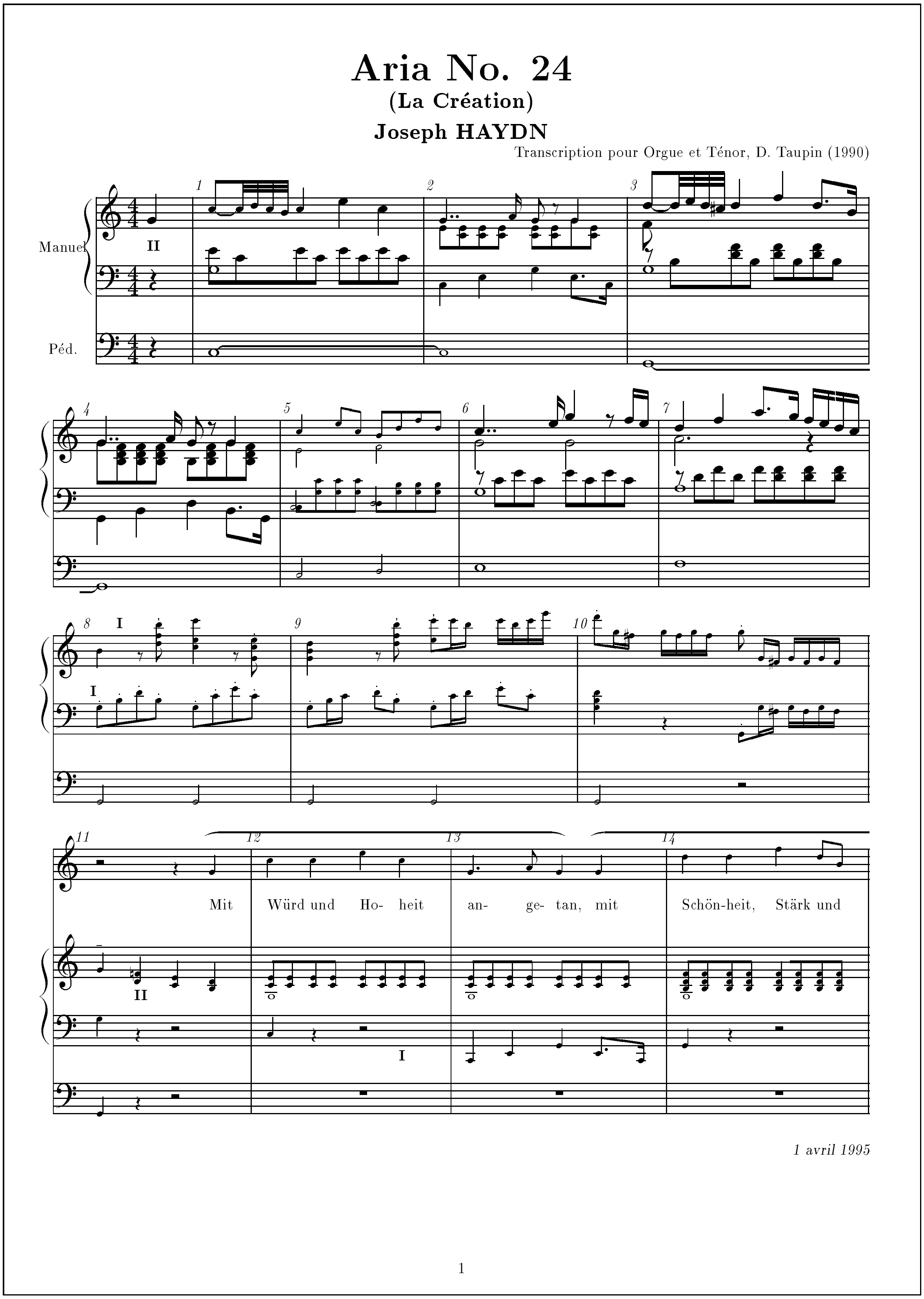
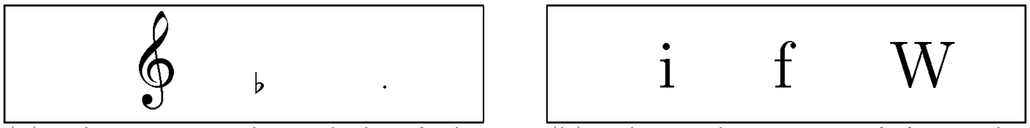



Figure 1: A piano score

are then analogous to the rows of characters in optical character recognition. These observations suggest that current optical character recognition systems may be adapted (say, with introduction of fonts for musical symbols) to perform optical music recognition. This would be desirable, since optical music recognition has been under development for many years, with commercial version currently available. Being able to modify existing optical character recognitionprogram to handle musical scores, we need not start to develop an optical music recognition system from scratch.

Unfortunately, there are some major differences between text and musical scores that make the adaptation of current optical character recognition system to recognize music difficult. The most obvious difference is that most musical symbols in a musical score are connected together by the staff lines. As most optical character recognition systems assume (reasonably) that the characters on a piece of text are disjoint from one another, they would fail the recognition when presented with an unprocessed image of a musical score. Another difference is that musical symbols on the same score can have great variation in relative sizes. As shown in figure 2, a *treble clef* symbol can be much taller than a *flat* symbol, which in turn is much larger than a *duration dot*. For text, the sizes of the characters do not vary much. Thirdly, characters in printed text are arranged regularly so as to please the human eye. For musical scores, there is no standard rules governing the location of the symbols. The spacing between horizontally *adjacent* symbols can be arbitrary, and the density of symbols on a score can differ from publisher to publisher.



(a) Three musical symbols of the same scale. From left to right: treble clef, flat, duration dot. The relatives sizes between the symbols is great.

(b) Three characters of from the same font. Letter ‘i’ is usually smaller; an ‘f’ is tall while a ‘W’ is wide. Even so, their relative sizes are not large.

Figure 2: Relative sizes of musical symbols vs. text characters.

Owing to these major differences, ordinary optical character recognition techniques do not perform well for musical scores. Special techniques has been developed to handle musical scores efficiently and effectively. Some typical techniques will be discussed in section 2.

1.3 Applications

Automatic optical music recognition is essentially the process of converting an image of musical score into an equivalent electronic form that can be handled conveniently by machines. As a result, further processing of the piece of music can be performed by machines automatically. There are many possible applications of it. Some of them are listed below:

1.3.1 Applications concerning the editing of scores

Adaptation of existing works to other instrumentations. A full score may contain the voices for several instruments. The process of extracting the voice for a particular instrument (say piano) to produce piano score can be automated with optical music recognition.

Conversion into Braille code. Braille code is designed for the blind to read. Converting ordinary scores into Braille scores is a tedious jobs that is most suitably automated. With optical music recognition, existing scores can be converted into Braille code for blind musicians.

Transposing. Transposing a score is to translate the score to another key. It is a mechanical task. It involves shifting the symbols of an existing score, and the addition and removal of accidentals. Computers can replace human for such mechanical tasks.

Renewing. Old scores can be processed by an optical music recognition system, repaginated and re-printed.

1.3.2 Applications concerning the collection of databases

Archival. Thousands of existing musical scores can be processed by an optical music recognition system and stored effectively. While an image of a page of A4 sized score sampled at 300 dpi occupies about a megabyte of storage, the output of the recognition just occupies tens of kilobytes. This reduction in data size utilizes storage space more effectively. A 5.25" CD-ROM can store up to 600MB of data, that is thousands of sheets of scores. Compare this with the size of the same amount of physical scores (i.e. paper), and the time required to find a desired piece of score.

Analysis of musical structure and style. The binary form of a piece of music can be read by programs to analyze the structure and style of the music. This can help researches in musical theories. It is also possible to evaluate algorithms designed for automatic composition of music.

1.3.3 Other applications

Synthesis of existing musical works. Given a musical score, it can be analyzed by an optical music recognition system. The output of optical music recognition can be feed to machines specially designed to perform the piece of music. For example, mechanical devices can be installed on a piano to perform the music printed on a piano score.

Automatic harmonization. It is possible to design algorithms to automatically add chords to a given monophonic melody.

1.4 An implementation

An offline automatic optical music recognition system is implemented in this project. The system is a software that runs on SunOS or Linux. It requires around 2 megabytes of memory to run. 4 megabytes of memory should be sufficient, unless the image is severely skewed, in which case the program may need 8 megabytes of memory or more.

The input to the program is an image file scanned with a flatbed scanner, at a resolution of 300 dpi or higher. However, higher scanning resolution may drive the program to use more memory, to handle the extra information. At present, the program only handles bilevel images[2] and the input file must be in TIFF[3] format. At present, the system handles relatively simple piano scores. It can handle multiple, simultaneous staves (if they are connected by the bar lines). Polyphonic music may not be properly handled. Rests, accidentals, slurs, clefs and time-signature are ignored. Images skewed by up to 10 degrees can be handled by the program.

The program analyses a page of A4 in about 3 minutes on a Sun Sparc workstation running SunOS. It spends about the same amount of time on an i486 running Linux. For more details about the performance, please refer to section 3.11.

The program generates its output in printable ASCII characters. Since there is no international standard on binary representation of music, I have used my own format, preserving important information that has been retrieved from the image. Writing a simple program to convert the output of the program to other formats, such as DARMS[4], MUSTRAN[5] and ALMA[6] is a simple and trivial task.

1.5 Terminologies

Here is a list of some commonly occurring terms, with explanations. Their shapes and appearance can be found in section 1.6.

staff line On a musical score, a staff line is a long, thin, horizontal line on a musical score which defines a coordinate system. Along the line from left to right is the time axis. The scale on this axis is roughly linear. The direction perpendicular to staff lines is the dimension of pitch, with higher positions denoting higher pitches. Note that quantities in this dimension is quantized. Usually, a group of five staff lines are drawn as a group to form a stave. (See figure 3.)

[2] A bilevel image is one for which each pixel can have only 2 possible values, typically *black* and *white*.

[3] TIFF, standing for *Tagged Image File Format*, is an image format widely used for scanner output and storing fascimile documents

[4] DARMS stands for Digital Alternate Representation of Musical Scores. It uses ASCII characters to encode music. Numbers are used to represent pitches, and other non-numeric characters for other symbols. Details of DARMS can be found in three main papers (Bauer-Mengelberg 1970; Erikson 1975,1983).

[5] MUSTRAN stands for Music Translator. It uses more mnemonic symbols than DARMS does.

[6] ALMA stands for Alphanumeric Language for Music Analysis. It is another music format which features abbreviations and user-defined representational symbols.

staff space This is the distance between adjacent staff lines of the same stave. (See figure 3.)

stave (staff) A stave is also known as a staff. It is a group of five staff lines. (See figure 3.)

bar line A vertical line in a musical score to separate notes into groups called "bar units". (See figure 3.)

ledger line Ledger lines are additional horizontal lines added near a note symbol when a note lies too far above or below the staff. They help to clarify the positions of these notes. (See figure 3.)

bar unit It is a small unit of a piece of music. Each bar unit occupies the same length of time. (See figure 3.)

note symbol A note symbol (often abbreviated as "note" in this report when it is unambiguous) in a musical score is a symbol that represents a musical note and its duration. The pitch of a note is determined by the vertical position of the note symbol relative to the staff.

note head This is the elliptical portion of a note symbol. For whole notes and half notes, the note heads are hollow. For other notes, the note head is a solid ellipse. (See figure 3.)

note stem This is the vertical line segment of a note symbol. Besides a whole note, all other note symbols has a stem, with its end touching a note head. (See figure 3.)

note flag This is the 'tail' part of a note (as opposed to the term "note head") to determine the type of a note. The tail is on the end of the note stem other than that attached to the note head. A whole note, half note or quarter note does not have flags. An eighth note has one flag; a sixteenth note has two flags, and so on.

voice A voice is a musical line. A voice may correspond to a single instrument, though a piano part of a score is usually notated as two or more voices.

slur A slur is a thin, wide and curly line that spans across a group of note symbols. Slurs may span over several bar units.

pedal marking A pedal marking symbol tells a pianist how to control the foot pedals of a piano.

dynamic marking Dynamic markings are present in a musical score to indicate the loudness of subsequent notes.

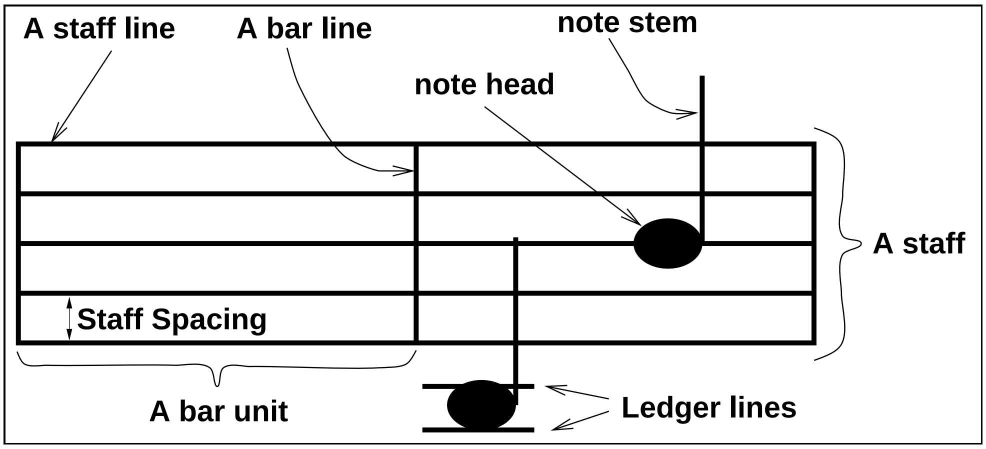


Figure 3: Names of various basic components on a musical score

1.6 Names of musical symbols

Here is a list of musical symbols, with the names given by their side.

Notes

- 𝅝 whole note
- 𝅗𝅥 half note
- ♩ quarter note
- ♪ eighth note
- 𝅘𝅥𝅯 sixteenth note

Rests

- 𝄻 whole rest
- 𝄼 half rest
- 𝄽 quarter rest
- 𝄾 eighth rest
- 𝄿 sixteenth rest

Accidentals

- 𝄫 double flat
- ♭ flat
- ♮ natural
- ♯ sharp
- 𝄪 double sharp

Note heads

- note head for a whole note
- note head for a half note
- note head for other notes

Flags

- flag for a stem-up eighth note
- flag for a stem-up sixteenth note
- flag for a stem-down eighth note
- flag for a stem-down sixteenth note

Clefs

- 𝄞 Treble clef
- 𝄢 Base clef
- 𝄡 Alto clef

Pedal markings

Ped. *

2 Offline Automatic Optical Music Recognition

2.1 Stages of the recognition process

Figure 4 shows the processing flow of a typical automatic optical music recognition system. This is not the only possible one. Other processing flows are possible. However, this project is based on this model.

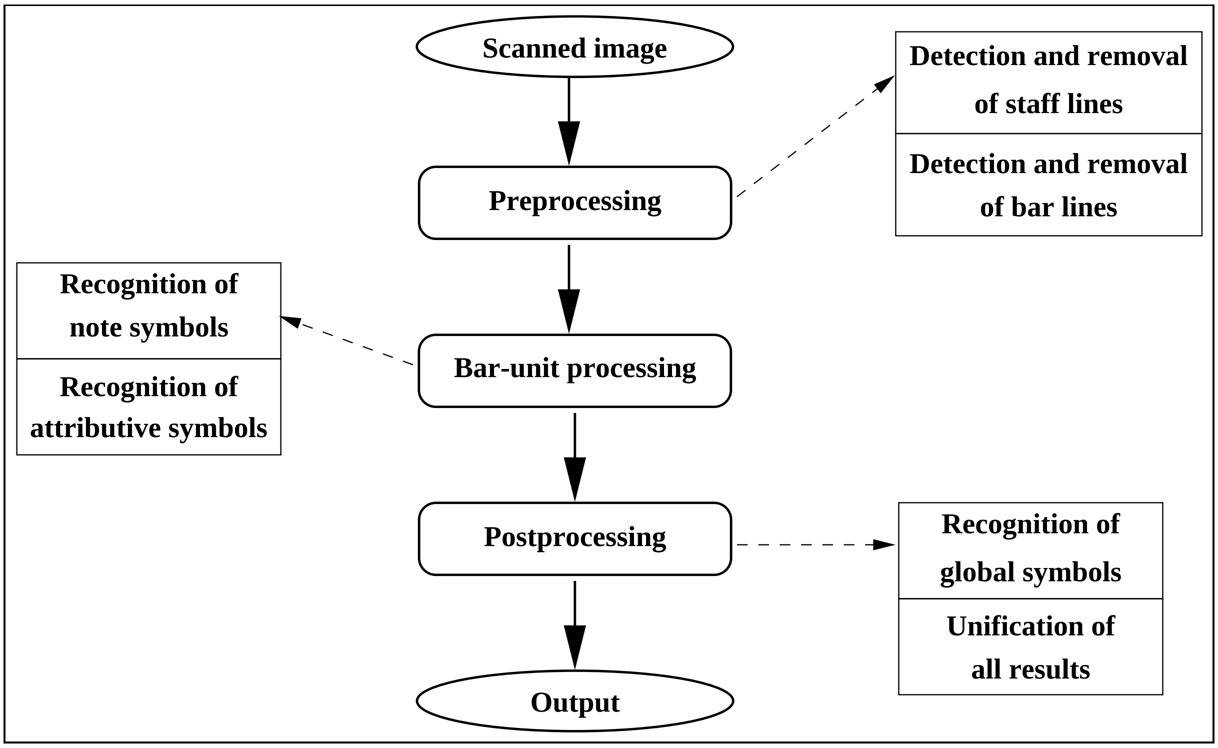


Figure 4: The processing flow

In an offline optical music recognition system, the musical score is first scanned with an optical scanner. The output of this process is a bitmap, which is the input to the optical music recognition.

The bitmap is analyzed to detect the staff lines. Details of this will be discussed in section 2.2. After this, there are two branches to go. One is to remove all the staff line segments that do not overlap with other musical symbols. This isolates the musical symbols that have been connected by the staff lines. Another option is to keep the staff lines. With this option, recognition in the later stages must use techniques that can work with the presence of staff lines. For example, template matching methods can be used to perform recognition in the presence of staff lines.

Having the staff lines detected, the next step is to detect the bar lines. Detecting the bar lines helps separating a musical score into smaller bar units. Being able to partition an image into smaller units, the optical music recognition can work more quickly and require less memory. This will be elaborated in section 2.3.

Done with the preprocessing stage, we come to the recognition of symbols. The image is processed bar unit by bar unit. First, the note symbols are recognized. Next comes the recognition of attributive symbols. Sections 2.4 and 2.5 will explain these in more detail.

The next stage is the post-processing of the image. In this stage, symbols that are not handled in the bar-unit processing stage are recognized. These are called global symbols. Such symbols include dynamic markings, pedal markings and slurs. They are common in that either they are too far away from the staff lines, or they have a scope of effect that spans over more than one bar unit. This makes them unsuitable to be picked up in the bar unit processing stage. So, they are left to the postprocessing. This will be discussed in section 2.6

Having the global symbols recognized, we have to unify the results of all previous stages. This gives a digitized version of the original musical score. This is the output of the automatic optical music recognition system. For details, refer to section 2.7.

2.2 Detection of staff lines

Staff lines play an important role in optical music recognition. Most musical symbols of a musical score are laid around the staff lines in a two dimensional manner. The horizontal axis is the time axis while the vertical axis tells the pitch of note symbols.

To a human reader, staff lines are important because they help the reader find out precisely the vertical position of note. From this information, the human reader can know the pitch of a note.

Interestingly, the importance of the staff lines to an optical music recognition system is quite different. Computers can often accurately find out the position of a note symbol on the vertical axis without the aid of auxiliary lines. However, the staff lines embed some other information that are very important for the optical music recognition.

These information, which is not obvious at first glance, is listed below:

1. The thickness of staff lines. The thickness of the staff lines in pixel units tells the optical music recognition system about the quality of printing of the original musical score and the resolution of scanning process used convert the score to a bitmap. Hence, it is used to set up many thresholds and acts as the tolerance value for many measurements and comparisons.

2. Staff spacing. The amount of space between adjacent staff lines gives the optical music recognition system a very important hint about the resolution of the scanned bitmap, as well as the size of the score printing. The staff spacing gives a size normalization that is useful for the subsequent recognition stages. Sizes and distances can be measured in units that are normalized to the staff spacing. This can avoid the inflexibility of absolute measures and static thresholds values.

3. The inclination of staff lines. Most of the time, the bitmap of the original musical score does not have the staff lines horizontal. Firstly, it is difficult to align the musical score with the scanner so that staff lines are exactly horizontal. Secondly, the

printing quality of the musical score may not be good enough so that the inclination of the lines varies without the score sheet. So, the image is skewed. The inclination of staff lines lets the recognition system know the image skew. This can help to improve the accuracy of the recognition system. For example, when the skew is too large, the system may rotate the image before further recognition.

Although the staff lines contain such useful information, their presence make the optical music recognition difficult:

1. The staff lines graphically connect most musical symbols, thus interfering with the recognition of the symbols.
2. Staff lines disturbs the contour of the musical symbols.
3. Musical symbols that have 'hollow' regions, such as sharp (♯) and flat (♭) symbols and a half note (𝅗𝅥), may have their hollow regions intersected with a staff line. The staff line runs through the hollow region, dividing it into two separate regions. This makes recognition of such symbols difficult.

So, the staff lines presents, to some extent, noise to the recognition of musical symbols. Without identifying and locating them, it is impossible to recognize the musical score. At least, determining the pitch of a note becomes impossible if staff lines are not identified.

While staff lines make the recognition of musical symbols difficult, the musical symbols also makes the identification of staff lines difficult. The presence of musical symbols acts as noise in the staff line identification process. This is especially true for symbols that are long and thin, e.g. slurs. Unfortunately, such "noise" is substantial. The signal to "noise" ratio is just too high. Many general image processing techniques fails. Methods specially tailored for optical music recognition are needed.

2.2.1 Projection on the vertical axis

Perhaps, the most straightforward method to locate the staff lines is to project the whole image onto the vertical axis of the image. This is illustrated in figure 5. A group of five equally spaced peaks in the projection reflects the presence of a staff group. The staff line thickness can be found from the width of a peak and the staff spacing is the distance between successive peaks of the group of five peaks. Then, what about the staff line inclination?

The inclination is *zero*! Indeed, with this method, we have presumed (unrealistically) that the image skew is so small that the staff lines are almost horizontal.

Consequently, this method is sensitive to image skew. Figure 6 shows an example for a slightly skewed image. With a skewed image, the peaks in the vertical project becomes blurred. Adjacent peaks may then overlap with one another. When the skew is larger, the overlapping regions may merge to form false peaks (figure 7). The false peaks would be mis-identified as staff lines. Thus, this simple is not tolerant against image rotation.

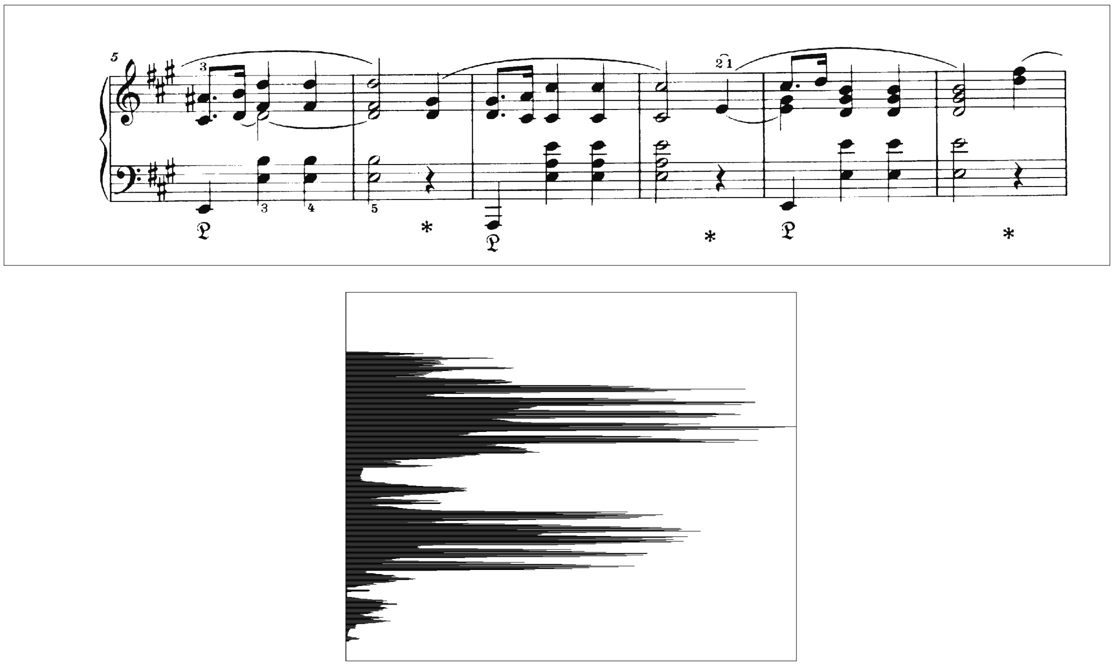


Figure 5: A fragment of a musical score and its vertical projection

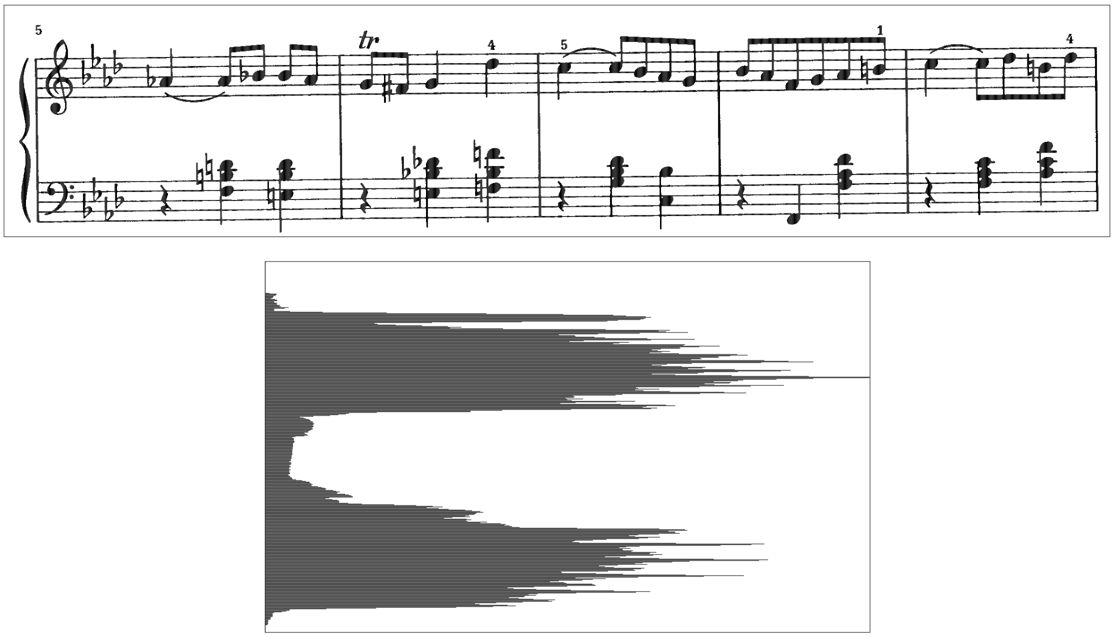
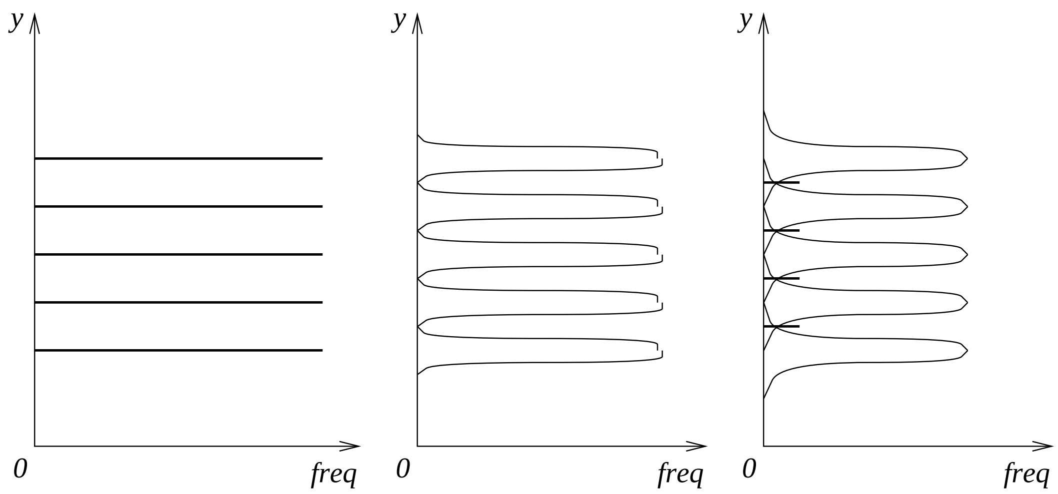


Figure 6: A fragment of a skewed musical score and its vertical projection



(a) (b) (c)

(a) five ideal peaks; (b) blurred peaks due to slight image skew; (c) false peaks resulted from overlapping of blurred peaks.

Figure 7: False peaks due to image skew

2.2.2 Hough transform

So, we need some methods to detect the staff lines even when the image is skewed. Thus, Hough transform was considered. Hough transform, patented by Hough (1962), is a method commonly used in image processing for locating straight lines from in an image. It can find out lines in all orientations and positions. In short, the Hough transform is a voting process in which each pixel of the image votes for the candidate lines that it belongs to. Candidate lines that get higher vote counts correspond to lines in the image. More information about Hough transform can be found in [6].

Although staff lines are long, thin straight lines in the musical score and Hough transform can detect straight lines in an image, empirical results showed that Hough transform is not a good method for staff line identification. Many false staff lines were found. The following are some suggested reasons for the failure:

1. The large amount of musical symbols present on the musical score act as noise to the identification, making the Hough transform to report more lines than desired.

2. The staff lines does not appear strictly straight in the image. This may be caused by the printing error of the score, or errors introduced by the scanner. As the Hough transform is for straight lines, it may not give good results to slightly curved lines.

3. The staff lines in the image has a substantial thickness. As a result, a "*thick*" staff line may contain one or more thin lines. These thin lines are voted for by the staff line, causing false lines to be reported. (See figure 8.)

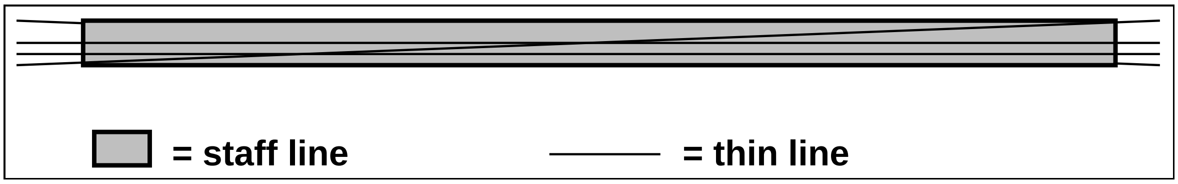


Figure 8: A thick line can contain several thin lines

Some modifications had been tried to reduce the effect of (1) above. One of which is to restrict the slope of the candidate lines to the range $[-1, 1]$ (for a skew of ± 45 degrees). This avoids having bar lines and note stems from being captured. Furthermore, long vertical runs of black pixels are not considered by the transform, because they are most probably not part of a staff line (remember that staff lines are thin). So, only short vertical runs of black pixels were allowed to vote for the candidate lines.

Despite these modifications, the Hough transform still reported too many staff lines, though slight improvements were observed. The effects of (2) and (3) above could not be reduced easily. Besides, the Hough transform is a rather computationally expensive operation, with time complexity of about $O(n^3)$ where n is the maximum of the image height and image width, in pixel unit.

2.2.3 Linear Adjacency Graph

So, in order to identify the staff lines accurately, we need a method that is tolerant to image skew and capable of of finding the staff lines when they are "thick" and not strictly a straight line. The following method from N.P. Carter appears suitable. This project is mainly based on this method.

In Carter's method, the input image is first scanned vertically and vertical runs of black pixels are run-length encoded. An individual vertical run of black pixels is called a *segment*. A transformed Line Adjacency Graph (LAG) is formed by linking together horizontally adjacent segments which overlap vertically. A group of segments linked together in this way is called a *section*.

In the construction of the LAG, the vertical run-length encoding is scanned from the left of the image to its right. Sections are gradually accumulated by appending qualifying segments to existing sections as the scan proceeds. A new section begins with a segment that does not qualify to join any existing sections. Sections form the *nodes* of the LAG.

A *junction* occurs when a segment overlaps several segments in an adjacent column. Sections are terminated at junctions. Associated with each junction are two adjacent sections that caused the junction. So, a junction is represented in the LAG by an *edge* whose end-points are these two adjacent sections.

To facilitate the break down of staff lines and ledger lines into separation sections, the following rule is used in forming sections. When a segment is to be appended to an existing sections, the height of the segment is compared to the average height of the section. If they differ by a factor of more than 2.5, then are not joined. Instead, the segment starts a new section, as if a junction has been encountered at that point. This rule also separates note heads from note stems, and note flags from stems. For a beam group of notes, the note heads, stems, and beam lines are separated.

A merit of the LAG is that it is invariant to moderate rotations of the image. The construction of the LAG is efficient and subsequent processing can operate directly on the LAG, rather than raw data from the image. This also reduces the amount of information that have to be handled in the subsequent processes. Consequently, the amount of working memory can be reduced. The structure of a LAG is suitable for many other processing algorithms which are applied subsequently.

Followed by the formation of the LAG is a noise removal pass. In this pass, the sections (i.e. nodes of the LAG) are examined. Any isolated or singly connected sections whose area is smaller than a certain threshold is considered noise and hence removed from the LAG. If the removal of a noise section turns a multi-way junction into a two-way junction, the two remaining sections of the junction are merged, provided that their average heights differ by a factor of no more than 2.5.

Next, the LAG is searched for sections that are thin and long (measured by the aspect ratio). They are called *filaments*. The set of filaments is a superset of the set of sections that contains staff lines not overlapping with musical symbols. A thickness threshold is imposed so that only thin sections are accepted. This excludes the beam lines from the set of filaments. To avoid slurs from being classified as filaments, the search of filament rejects sections that are non-linear (indicated by the variance figure of the least-squares fit line).

From the set of filaments, the search of staff line starts from the leftmost filament. This is a potential staff line. Since the edges of the LAG essentially links sections that are horizontally adjacent to one another, we can trace from the leftmost filament into the symbol that intercepts that filament, and find another filament that is a continuation of the staff line. This way, filaments are traced across the LAG from left to right, until it cannot go further. Then, the remaining filaments are traced. Consequently, we obtain chains of filaments, with each chain containing roughly colinear filaments.

Note that not all these chains are real staff lines. Some may come from the horizontal strokes of the characters “5” and “tr” that may occur in the score. To eliminate these chains, we exploit the observation that staff lines are very very long and spans over most space of the musical score. Thus, the chains are examined one by one and the difference between the leftmost and rightmost points reached by the chains are found. If this difference is large, then it is a staff line. If it is small, it is a false staff line.

Thus, false staff lines are eliminated. The remaining chains represent staff lines. From them, groups of five are recognized and each group corresponds to a staff. A this point, the staff line thickness, staff spacing and image skew (determined from the inclination of staff lines) have be found.

2.3 Detection of bar lines

The next processing is the detection of bar lines. Bar lines are thin, vertical lines in a musical score. Unlike staff lines, bar lines are seldom intercepted by other musical symbols.

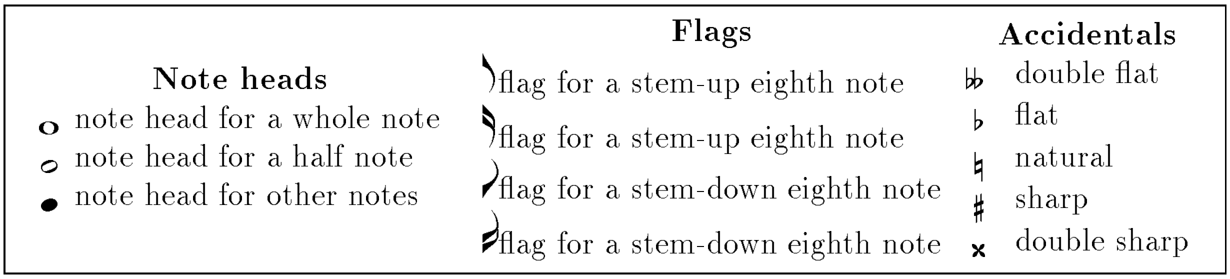


Figure 9: Primitive components of some musical symbols

This makes the detection of bar lines much easier.

However, care have to be taken, since some symbols, like slurs, may cross over bar lines. This problem can be solved by projecting the score in the skew-corrected horizontal axis. Bar lines are revealed as sharp peaks in the projection.

With bar lines identified, the LAG can be divided into regions, each representing a bar unit in the musical score. Then, processing can proceed bar unit by bar unit. A bar unit is a convenient unit for subsequent recognition. Bar units are much smaller than a whole picture and hence can be stored and processed with less memory. It also limits the search space for symbol features, and hence brings about speed improvements. The bar unit processing stage processes musical symbols that have a meaning local to the bar unit. Recognition of symbols that have a global effect are deferred to the post-processing.

2.4 Recognition of note symbols

Note symbols within each bar are recognized. With the LAG, the symbols are broken into primitive components as shown in figure 9. The different components are separated into different sections in the LAG.

First, the skew-corrected horizontal project of each connected component is examined to search for potential note stems, which corresponds to peaks in the projection. Having found a potential stem, the projection near the stem is examined to search for potential note heads. A note head has the shape of an inclined ellipse and may be hollow or solid.

Then, the component sections are examined to check if the sections that corresponds to potential note heads are true note heads. To determine the type of the note among quarter note, eighth note, sixteenth note, etc., note flags are searched around the stem. Since some notes may be a component of a beam group, its flags may be replaced by thick beam lines. They are handled like note flags.

Duration dots and accidentals attributed to a note is graphically isolated from the note symbol after staff line removal. They can be identified separately during the bar unit processing and combined with the results of recognition of note symbols. The duration dot is a tiny dot whose bounding rectangle is very small. Hence, duration dots can be identified as connected regions whose bounding rectangle is approximately a square with side length is just slightly greater than the staff line thickness. Accidentals can be identified by the aspect ratio of their bounding box, and an analysis of their vertical and horizontal projections.

2.5 Recognition of attributive symbols

Attributive symbols include the clefs, key-signature and time-signature. They are recognizable in bar unit processing and they affect the interpretation of other symbols.

Clefs have odd sizes and can be easily identified by comparing their normalized (with respect to staff spacing) width and height to standard values, allowing a certain amount of error below a threshold. Moreover, their position are quite restricted. Clefs can only appear at the left side of a bar unit. The vertical position of the clefs are fixed with respect to the staff.

The key-signature consists of a series of sharps or flats. The positions of the series are governed by some strict rules. These rules can be exploited to locate the series one by one. From this, the key-signature can be determined.

The time-signature comprises two digits. Their vertical positions are fixed relatively to the staff. Moreover, by musical knowledge, the combination of them is limited.

2.6 Recognition of global symbols

Global symbols include slurs, dynamic markings and pedal markings. They are different from other symbols in that their scope of effect can extend over several bar units. The slurs in the musical score can span over several bars. As they carry global information, they are unsuitable to be handled in bar unit processing.

The recognition of slurs is simple. As mentioned in section 2.2.3, slurs have similar thickness as staff lines. They are long and thin. The feature that makes them not misidentified as staff lines is that they are curved. This feature can be exploited to find out the slurs.

2.7 Unification of all results

The final stage of the recognition process is the gathering of all the results from the previous processes. Recognition results from different bar units are combined and sequenced according to their positions in the original score. Next, meanings of the the global symbols can now be applied to the information obtained during bar unit processing. A more sophisticated optical music recognition system should have a semantic checker to check the consistency of the results. For example, a bar with a time signature [4,4] should not contain five quarter notes. The potential errors in the recognition can then be reported or corrected.

3 An implementation

Having discussed about the techniques in optical music recognition in the previous section, let us discussion on the actual implementation. In this project, a program was written to implement the ideas as discussed in section 2.

3.1 System requirements

The program is written on UNIX platforms. It was developed simultaneously under SunOS on a Sparc station as well as Linux on an i486. It is written in C++ language and developed with the GNU C++ compiler (also known as g++) version 2.6.0. Depending on the size of the input image and the complexity of the score, the program dynamically allocates different amounts of memory from the system. With images scanned at 300 dpi, the program requires around 2MB of memory. Images scanned at a higher resolution would need more memory. 4MB should be sufficient in most cases. For a heavily skewed images, the program tends to use much more memory.

The GNU C++ compiler was used because of its high portability to machines of different architectures. The source code of the program compiled on both SunOS and Linux. It needs the TIFF library routines as well as `libg++` (the GNU C++ class library), though. The C++ language was chosen because of its modularity and ease of prototyping using existing libraries.

The whole program is broken down into several modules. Each module corresponds to a rectangular box in figure 4. Since the whole recognition process is a sequential flow, each module passes information to the next module using C++ objects. To minimize the dependencies among different modules, they do not exchange information directly. Instead, the main program calls the modules one after another, and data is passed between the modules and the main program. The main program get the results from a called module, and passes the relevant information to the next module. So, communication among the modules is done via the main program.

In order to implement the LAG, a template class `Graph` was written. It supports operations of adding and delete nodes or edges on the graph. Traversal on the graph is facilitated by having member function that return the adjacency list of a node. The `Graph` class is implemented by keeping an list adjacency nodes for each node. Another important template class is the `ilist` class. It is a replacement for singly linked lists where deletion of elements is not requried. It is a container that supports adding elements to it. It is written in order to reduce the overhead of frequently allocating and deallocating memory from the memory allocator. The overhead of storing pointers is also reduced. It does so by requesting for and releasing chunks of memory at a time.

3.2 Input Format

There are many image file formats available. Examples include GIF, PCX and TIFF. However, TIFF file is the only input format of the program. TIFF is chosen because most scanner programs can generate TIFF format. It is widely used for storing scanned images. The program uses the TIFF library routines for reading TIFF files. Moreover, the program only accepts a single page black and white TIFF image. While programs that convert other image files to the TIFF format are widely available, it would be unwise to generalize the program to accept other formats. Rather, time is mainly spent on writing the recognition part of the program.

3.3 Preprocessing

The program starts by reading in the TIFF file. Internally, the program re-encodes the TIFF file into vertical run-lengths of black and white pixels. This processing helps the construction of LAG as described in section 2.2.3. It also reduces the amount of data to be processed subsequently. The program reads the file only once, reducing overheads of unnecessary disk operations.

Unlike the original LAG approach taken by Carter, the program determines the staff line thickness and staff spacing before the construction of the LAG. The trick used here is that the vertical run-length of black pixels that occurred the most frequently is the height of the vertical runs of a staff line. The staff spacing is simply the vertical run-length of white pixels that occurred the most frequently.

Thus, this pre-processing can determine the staff line thickness and staff spacing earlily. This is important because many threshold values are scaled on the staff line thickness and staff spacing. The pre-processing also makes a preparation for the construction of LAG. The lengths of vertical runs of black pixels are stored in memory, sorted according to their position in the image file; while lengths of vertical runs of white pixels are forgotten after the staff spacing is determined.

3.4 Construction of LAG

The construction of LAG is aided by the preprocessing described in section 3.3. Since the vertical runs of black pixels of the whole image are already in memory, and are sorted, the LAG can be constructed rapidly.

The vertical runs are scanned from left to right. The program compares pairs of adjacent columns. The vertical runs of the same column are already sorted in a top to bottom order. So, the LAG program uses a scanline algorithm. The scanline advances from the top of the image to the bottom. As the scanline descends, the program checks to see if any vertical runs in the pair of columns in concern are intercepted by the scanline. When the two vertical runs (one from each column) are intercepted, the two vertical runs have vertical overlap, and a link is formed between these two vertical runs. Proceeding this way, the scanline descends to the bottom of the image. Then, the next pair of adjacent columns are considered and a scanline is run from top to bottom to detect vertically overlapping runs of black pixels from the column pair.

When the scanline reaches the bottom of the image, the program examines the links that have been newly established in that pass. If any vertical run is associated with two or more newly established links, it is a junction as defined in section 2.2.3. For each junction, a new node of the LAG is created for the vertical run on the right-hand column associated with the colliding links. An edge for the LAG is added to join this new node and the node on the LAG that the new node is linked to. For vertical runs on the right-hand column that are not linked to any runs on the left-hand column, it starts a new LAG node. A singly linked vertical run on the right-hand column does not start a new LAG node. Rather, it is appended to the section that contains the vertical run that it links to. Proceeding this way, the LAG is constructed when the left-to-right scan completes. In the program, the LAG is a directed graph, with edges pointing from sections on the right hand side to sections to their left. This directional information helps the program to determine which neighbour of a section is to its left.

Now, a LAG as described in section 2.2.3 is created. Its nodes are the sections, and its edges represent junctions. The LAG separates most musical symbols from the staff lines. They get broken into different sections.

After the construction of the LAG, the noise in the LAG is removed. Here noise is defined to be sections with a small area. The threshold is half of the square of the staff line thickness found in the preprocessing. Sections with an area smaller than this threshold are removed from the LAG. If the removal of a noise section turns a multi-way junction into a two-way junction, the sections associated with the junction are joined into one single section if the the ratio of their average heights differ by no more than a factor of 2.5.

3.5 Identification of staff lines

After the LAG is formed, it is examined to identify the sections that are part of the staff lines. To do this, *filaments* of the image are extracted. Filaments are sections with a thickness approximate equal to that staff line thickness as determined in the preprocessing. Filaments are wide and thin. To avoid including slurs into the set of filaments, the variance figure of the least-squares fit line through the centers of the constituent vertical runs of the section is calculated. Sections whose variance figure are large are not included as filaments.

From the set of filaments, staff lines are identified in the following way. A leftmost filament is taken out. Then, it is traced towards the right at an inclination suggested by its least-squares fit line. The section to the right of a filament is found from the LAG, since there is an edge on the LAG connecting the filament to its neighbouring sections.

The program traces black pixels towards the right in the direction of the least-square fit line of the filament, until either no more black pixels are encountered in that direction, or until another filament is encountered. In the former case, the tracing stops and the list of filament that have been traced is recorded as a *chain*. Then, the program restarts tracing from the remaining filaments, until no more filaments are left untraced. If another filament is encountered, then the program continues the tracing in the direction of the least-square fit line of the newly encountered filament. This strategy can handle staff lines with slight local variations in the inclination of staff lines.

After all filaments have been traced, the chains are collected. Each chain corresponds to a straight line which passes through all the filaments it contain. So, each staff line appears as a chain. However, there can be some false chains. For example, the horizontal stroke of the digit "5" can form a short chain. The work around is to remove the short chains from our consideration. For this, the *width* of a chain is defined to be the distance between the leftmost pixel of the leftmost filament of the chain and its rightmost pixel of the rightmost filament. The widths of all chains are averaged and the mean is taken as a threshold to remove the short chains.

As a result, the chains that remain correspond to staff lines. Using the staff spacing determined in the preprocessing, groups of five chains separated from one another by one staff spacing are taken out and identified as a staff. The positions of the staff and the inclination of the staff lines are recorded.

3.6 Removal of staff lines

After the staves are identified, the staff lines are removed from the LAG. This is the process of removing all filaments whose distant from a closest staff line is smaller than a threshold. The threshold is set to one-third of the staff spacing. Figure 10 shows the result of processing at this stage. The input is the score fragment in figure 6 on page 13. In figure 11, the positions of the staff lines detected by the program are also shown. Note that in these figures, the sections are outlined, to illustrate how the symbols are separated into sections.

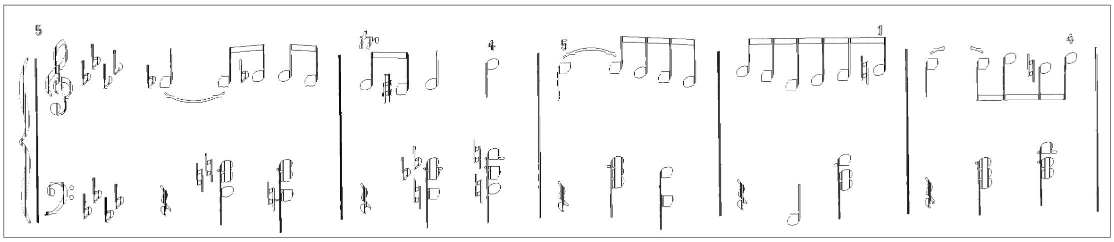


Figure 10: Staff lines are removed

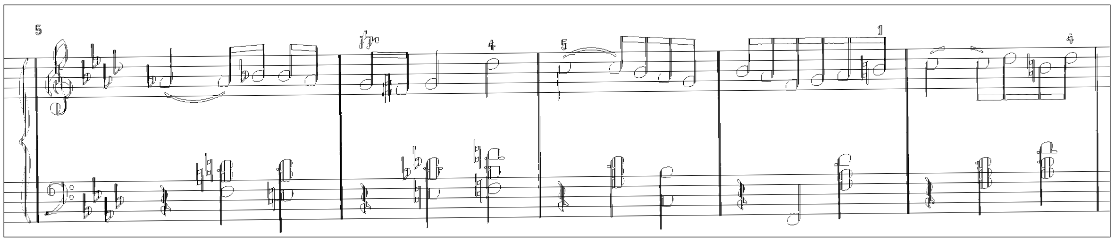


Figure 11: Location of the recognized staff lines

Followed by the removal of staff line sections is another pass of noise removal. This pass is identical to the pass mentioned in section 3.4.

The removal of staff line sections may turn multi-way junctions into two-way junctions. So, after removal of staff lines, the LAG is searched for singly connected nodes.

These nodes are joined with their neighbours provided that the neighbour has no other neighbours on that side of the section and that the average heights of these two sections do not differ more than a factor of 2.5.

3.7 Connecting the sections

At this point, the LAG is in fact a forest of sections. Each connected component of the LAG represents an object in the input image that is connected. To facilitate subsequent processing, the a breath-first search is performed on the LAG. Connected components of the LAG are collected. For each connected component, the constituent sections are examined and vertical and horizontal projections (in the skew-corrected direction) of the connected component are calculated and stored. The width and height of the bounding box, in the skew-corrected direction, is also kept.

3.8 Bar line detection

With the connected components, bar lines are found by examining the size of the bounding box of each component. A component with a bounding box narrower than 3 times the staff line thickness and taller than 4 times the staff spacing is classified as a bar line.

After the bar lines are detected, they are grouped according to the staves that they belong to. Then, the bar units are determined.

3.9 Note symbol extraction

The connected components within each bar unit are examined. The horizontal projection of the connected component are searched for peaks with a peak width no larger than half of the staff spacing. These peaks suggest the presence of note stems, because note stems are tall and thin. Then, the left side and the right side of the stem are consider in turn, with priority to the left side, since note heads are usually on the left of a stem. For each side, the horizontal projection is first checked to see if there is potentially a node head on that side. If so, the program examines the sections on that side and performs a test. The test checks to see if the bounding box of the section is approximately square, with side length approximately the same as the staff spacing. The area of the section is then checked to see if it exceeds three quarter of the bounding box. If all these conditions are satisfied, the section is considered a note head.

Having found the note head, the program proceeds to the other extreme of the note stem in order to find the note flags. The number of flags tells type of the node that is found. At present, whole notes and half notes are not recognized. They have a hollow note head and the note head is separated into two different sections.

3.10 Output

The program outputs the result of recognition in ASCII printable characters. Since there is no international standards on music file format, I have designed my own one. It is hoped to preserve as much important information recognized as possible. It is designed so that conversion program can be easily written to convert the data into other formats, or directly work on the data.

For the input image shown in figure 6 on page 13, the program produces the following output:

```
bar(0){
  staff(0){5 4 5 8 6 8 6 8 5 8 }
  staff(1){}
}
bar(1){
  staff(0){4 8 3 8 4 4 8 4 }
  staff(1){}
}
bar(2){
  staff(0){7 4 7 8 6 8 5 8 4 8 }
  staff(1){}
}
bar(3){
  staff(0){6 8 5 8 3 8 4 8 5 8 6 8 }
  staff(1){1 4 }
}
bar(4){
  staff(0){7 4 7 8 8 8 6 8 8 8 }
  staff(1){}
}
```

The output is in a specially designed grammar. The lines "`bar(`*n*`){`" and "}" bracket the information found in a bar unit, with n being the unique sequence number assigned to each bar. The sequence numbers are increasing with respect to the time scale of the music. Within each bar is a list of staff structures bracketed by "`staff(`*m*`){`" and "}". Here m is a sequence number for the each staff. This number is local to each bar unit. Within each staff structure are pairs of numbers. The first number represents the pitch of the note. "`0`" represents a middle C, "`1`" represents note D, and so on. The second number represents the note type. "`1`" denotes a whole note; "`2`" denotes a half note, etc. The pairs of numbers appear according to temporal order.

3.11 Performance

The program has been profiled with the `gprof` profiler to find out the most time-consuming parts. Optimization has been done on these parts. With the compiler option "`-O2`", the

GNU C++ compiler produces an executable of 416KB on `sunmp`. The program processes the image shown in figure 12 in 66.74 CPU seconds. It used 2.2MB or memory. The average data segment size of 975KB.

3.12 Improvements

At present, the program can only recognize simple score forms on which symbols are not printed too densely. I have been working with piano score, though other scores should be recognized without problems. Whole notes and half notes are note recognized yet. The program ignores clefs, time-signatures and key-signatures. It assumes a treble clef. Rests, accidentals and duration dots are not handled. Here is a list of the places where improvements to the program that are suggested:

Add the capability of recognizing whole notes and half notes. Unlike other notes which might be attached to beam groups, these two notes are always isolated from other notes. So, it is possible to find them out by comparing the size of the bounding box and examining the horizontal and vertical projections.

Recognizing bar lines even if they are intersected by other symbols. The current system uses a very simple method to locate bar lines. The method can fail when the bar line is intersected by other musical symbols, such as slurs. An improvement would be to examine the vertical projection of the connected components. A vertical line shows a peak in the horizontal projection. From the height of the peak, the height of the line can be estimated. If the height exceeds a certain threshold, it can be classified as a bar line.

Add the capability of recognizing rests and duration dots. Duration dots can be easily identified because of its size. After staff line removal, the remaining symbols that have a square bounding and side length smaller than half the staff space should be a duration dot. Whole and half rests appears as short strokes whose height are approximately half the staff spacing. They are rectangular. So, they can be identified by examining the size of bounding box, as well as the symbol area. Whole note and half note differ by the position at which they appear on the staff. Other rest symbols have approximately the same size. A check on the size of the bounding box would be able to identify them.

Add the capability of recognizing clefs Clefs has strange sizes and a simple check on the size of the bound box may be able to identify them. Moreover, the position of clefs is very restricted. So, clefs can be found by paying attention to the region where they can appear.

Other improvements

- Add the capability of recognizing accidentals.

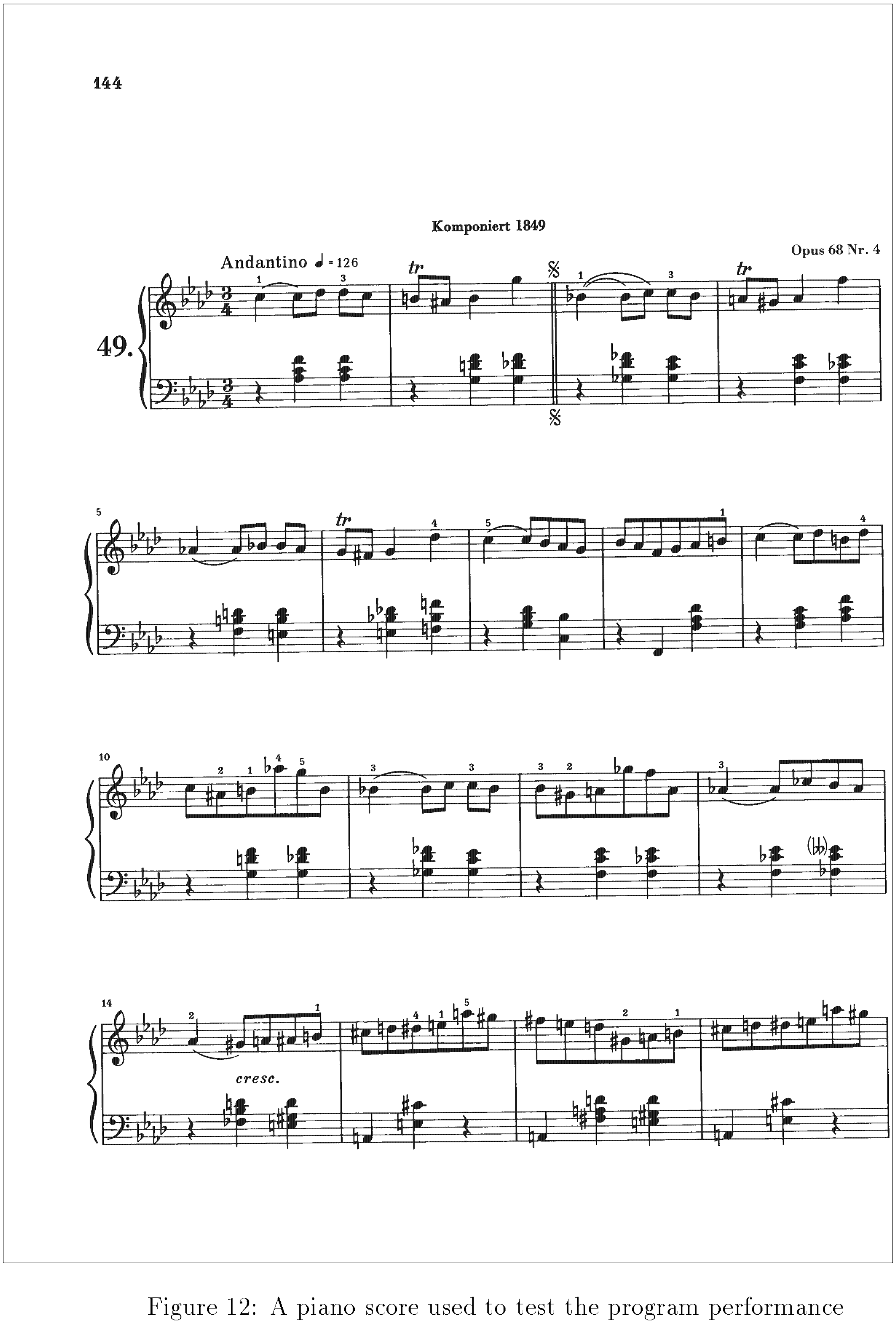



Figure 12: A piano score used to test the program performance

- Add the capability of recognizing key-signatures.
- Add the capability of recognizing time-signatures.
- Add a simple semantic checker.

4 Conclusions

In this project, optical music recognition was investigated. The existence of staff lines makes optical music recognition unique from the class of optical recognition topics. Since musical symbols are connected together by staff lines, to successfully recognize the symbol requires special techniques that are tailored for optical music recognition.

The staff lines are important since it they tell the size of the symbols, quality of scanning and the image skew. However, they are at the same time noise to the recognition of musical symbols on the score. So, intuitively, staff lines should be removed before further processing. However, there are some optical music recognition system that employs template matching techniques to recognize the symbols in the presence of staff lines.

Removed or not, the staff lines have to be identified. The identification of staff lines is non-trivial. This is because the musical symbols behave as noise to the staff line identification. Nevertheless, a good staff line detector is important to optical music recognition, since subsequent processing would rely on information obtained from the staff line identification process.

So, the greatest challenge in optical music recognition is the development of methods to identify the staff lines. Much time was devoted to developing and improving the staff line identifier in this project. The LAG methods invented by Carter was used. It is efficient and can cope with slightly skewed images. It can also cope with slight local variations of staff line inclination. A good staff line identifier should be able to locate the staff lines and find out their inclination accurately. Staff spacing and staff line thickness, and image skew are important information that the identifier should retrieve. Speed of the identifier is another criterion for a good identifier.

Once the staff lines are identified, the subsequent recognitions become easy. Many other general techniques may be applied. In this project, the approach of removing staff lines is taken. Having stave lines removed, the musical symbols isolate from one another. Then, groups of connected pixels are be examined and recognized one by one. Many techniques for pattern recognition can be employed. For example, template matching may be used to recognize the symbols. Artificial neural networks is an alternative. Tailored techniques are developed to makes use of the special features of musical symbols to improve accuracy and speed up.

References

[1] H.S. Baird,H. Bunke,K. Yamamoto, (Eds). *Structured document image analysis*, pp. 403–476. Springer-Verlag Berlin Heidelberg, 1992. ISBN 0-387-55141-7, 3-540-55141-7.

[2] Edward J. Wegman *Statistical Image Processing and Graphics* Marcel Dekker, Inc., New York and Basel, 1986. ISBN 0-8247-7600-3.

[3] N.P. Carter, R.A. Bacon, T. Messenger. *The Acquisition, Representation and Reconstruction of printed Music by Computer: A Review*, Computers and the Humanities **22** (1988) 117-136. Kluwer Academic Publishers.

[4] Bruce Pennycook. *Towards Advanced Optical Music Recognition*, Advanced Imaging (1990 April).

[5] A.K. Jain. *Cluster Analysis, Hand book of Pattern Recognition and Image Processing.* Academic Press, N.Y., 1985.

[6] Henri H. Arsenault, Tomasz Szoplik, Bohdan Macukow. *Optical Processing and Computing.* Academic Press, Inc., 1989. ISBN 0-12-06-4470-3.

[7] Theo Pavlidis. *Algorithms for Graphics and Image Processing.* Bell Laboratories, Computer Science Press, 1982.

[8] A.G. Arkadev, E.M. Braverman. Translated from Russian by W. Turski and J-D. Cowan. *Teaching Computers to Recognize Patterns* Academic Press, London and New York, 1967.

[9] Sankar K. Pal, Dwijesh K. Dutta Majumder. *Fuzzy Mathematical Approach to Pattern Recognition.* Wiley Eastern Limited, 1986. ISBN 0-85226-193-4.

[10] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981. ISBN 0-306-40671-3.

[11] John A. Hartigan. *Clustering Algorithms.* John Wiley and Sons, Inc., 1975. ISBN 0-471-35645-X.

[12] D.J. Hand. *Discrimination and Classification.* John Wiley and Sons Ltd., 1981. ISBN 0-471-28048-8.

[13] P.S. Moharir. *Pattern-Recognition Transforms.* John Wiley and Sons Inc., 1992. ISBN 0-86380-137-4, 0-471-93633-2.