Optical Music Recognition

by Linn Saxrud Johansen

Thesis for the Degree of

Master of Science in Modelling and Data Analysis

Department of Mathematics

Faculty of Mathematics and Natural Sciences

University of Oslo

July 2009

Acknowledgments

This thesis completes my Master's degree at the Department of Mathematics, University of Oslo. The work on this thesis took place in the period from February 2008 to July 2009. During this period, many people have been helpful and supportive, for which I am truly grateful.

First of all, I would like to thank my supervisor, Anne H. Schistad Solberg, for invaluable guidance and feedback, and for introducing me to an interesting area of image analysis.

In addition, I would like to thank my fellow students and friends at *Parameterrommet* (B800) for their help and encouragement, as well as many enjoyable moments and great memories. A special thanks to Linn Cecilie and Ida who helped pointing out typos.

I would also like to thank my family, in particular my parents, for always being supportive and encouraging during my studies.

Last, but not least, a special thanks to Thomas and Pelo, for always making me feel welcome when I come home, and for always being able to make me laugh.

Linn Saxrud Johansen
Oslo, July 2009

Contents

1 Introduction

Every known civilization has had music of some form. Human beings most probably started to create music wanting to imitate the sounds of the nature – the sound of animals communicating, the sound of waterfalls, and the sound of rain, thunder and wind. Even in later times, composers and musicians got and still get inspired by the sounds surrounding them. But prior to the invention of music notation and the printing press, songs and melodies were carried on from generation to generation orally, and relied heavily on memory and improvisation (Bonds, 2006; Hanning, 1998).

Around the year 1025, an Italian monk, Guido d'Arezzo, invented staff line notation, for which he is considered the father of modern music notation. Nevertheless, music scores were still copied by hand, which was both expensive and time consuming. This resulted often in not correctly transmitted music; no two manuscripts were identical, no matter how careful the priests and monks who copied them were (Bonds, 2006).

Gutenberg's invention of the printing press in the middle of the 15th century made it possible to produce more precise copies of music scores. About 50 years later, the first music printed entirely with this new technology was published in Venice by a man called Ottaviano Petrucci. Notes could be joined in any order, rearranged and reused. Music could now spread faster, more efficiently and cheaper, to more people than it could through the precious handwritten manuscripts, which only the Church and noble people could afford. Moreover, the existence of printed copies meant that more works would become known widely and be preserved for later generations (Bonds, 2006; Hanning, 1998).

Nowadays records, radio, television and the internet spread music more widely than ever before, and an overwhelming number of musical works are available to us (Bonds, 2006). During the last decades, a great interest in converting music scores into a computer-readable format has arisen, and with this the field of Optical Music Recognition.

Optical Music Recognition (OMR) is the name of systems for music score recognition, and is similar to Optical Character Recognition (OCR) except that it is used to recognize musical

symbols instead of letters. OMR systems try to automatically recognize the main musical objects of a scanned music score and convert them into a suitable electronic format, such as a MIDI file, an audio waveform or ABC Notation [1](Rossant and Bloch, 2007; Bainbridge and Bell, 2003).

The advantage of such a digital format, compared to retaining the whole image of a music score, is that only the semantics of music are stored, that is notes, pitches and durations, contextual information and other relevant information. This way much computer space is saved, and at the same time scores can be printed over and over again, without loss of quality, and they can be edited and played on a computer (Vieira and Pinto, 2001). OMR may also be used for educational purposes - to convert scores into Braille code for blind people, to generate customized version of music exercises etc. In addition, this technology can be used to index and collect scores in databases. Today, there are a number of on-line databases containing digital sheet music, making music easily available for everyone, free of charge (Fornés et al., 2006; Szwoch, 2007; Bellini et al., 2001; Rossant and Bloch, 2007). One example is the *Mutopia Project*, which offers music free to download, print, copy etc. Currently, 1.577 pieces are available on this site. Another interesting project is the *Petrucci Music Library*, which is trying to create a virtual library containing all public domain music scores. In addition, composers willing to share their music may add their works. This library contains today 30.000 music sheets [2].

The earliest attempts at OMR were made by Pruslin in 1966 and Prerau in the early 1970's. During the last decades, OMR has been especially active, and there are currently a number of commercially available packages. The first commercial products came in the early 90's – *MidiScan* was released in 1991, and in the following years *SmartScore*, *SharpEye*, *PhotoScore* etc. were published. However, in most cases these systems operate properly only with well-scanned documents of high quality. When it comes to precision and reliability, none of the commercial OMR systems solve the problem in a satisfactory way (Bellini et al., 2001; Rossant and Bloch, 2007; Ng, 2002; Szwoch, 2007).

The aim of this thesis is to study various existing OMR approaches and suggest novel methods, or modifications/improvements of current algorithms. The first stages of the process is prioritized, and we limit to concentrate on identifying the main musical symbols, essential for playing the melody, while text, slurs, staff numbering etc. are ignored by our program. The last part of an OMR program usually consists of correcting classification errors by introducing musical rules. In this thesis, this is only applied to correct wrongly classified pitched for accidentals.

This thesis is organized as follows; basic music theory is introduced in Chapter 2. Here the rules and symbols, which will be used and referred to in the following chapters, are explained. In

[1] A language designed to notate tunes in plain text format, see: http://abcnotation.org.uk/.

[2] See `http://www.mutopiaproject.org/` and `http://imslp.org/` respectively.

Chapter 3, an overview of different OMR approaches is given. Here we see the differences and resemblances between various systems. Each step in the process is described. Literature on this topic is sizable and steadily growing and for a fuller and more historic review of OMR work, see Blostein and Baird (1992) and Selfridge-Field (1994). The main theory of the image analysis methods used in Chapter 5, is treated in Chapter 4. In Chapter 5, our approach to the OMR problem is explained in detail. Each single step is discussed, and examples, illustrations and algorithms are given. The main classification results are discussed in Chapter 6. In Chapter 7, a summary and some concluding remarks are given to sum up the work. Alternative solutions and ideas for improvements and further research are suggested. An overview of the dataset with composers and titles is found in Appendix A, together with some examples of input images. Some additional results are presented in Appendix B. Some of my program code is available on `http://folk.uio.no/linnsjo/`.

2 Music Notation Theory

In OMR systems it is very important to include prior knowledge. Music scores follow strict structural rules, and systems including these rules when classifying symbols obtain better results than systems which do not take this into consideration.

In this chapter we will explain the main structure of a music sheet, the most common symbols and some basic rules, which will be useful to know when reading the rest of the text.

The musical scores contain the following elements:

- Lines
  - Staff lines.
  - Bar lines, which separate each bar unit.
- Notes and rests. Notes are again composed of note heads, beams, stems and flags.
- Symbols at the beginning, such as clef, and time and key signature.
- Accidentals, note relations and dynamic markings.

2.1 Lines

Music scores consist mainly of straight lines; long horizontal lines called staff lines and two types of vertical lines:

- Note stems, which are attached to a note head (note heads are located at the bottom-left or the top-right of a stem depending on the height of the note head relative to the staff lines).
- Bar lines, which are longer than note stems and represent the physical separation between measures/time units.

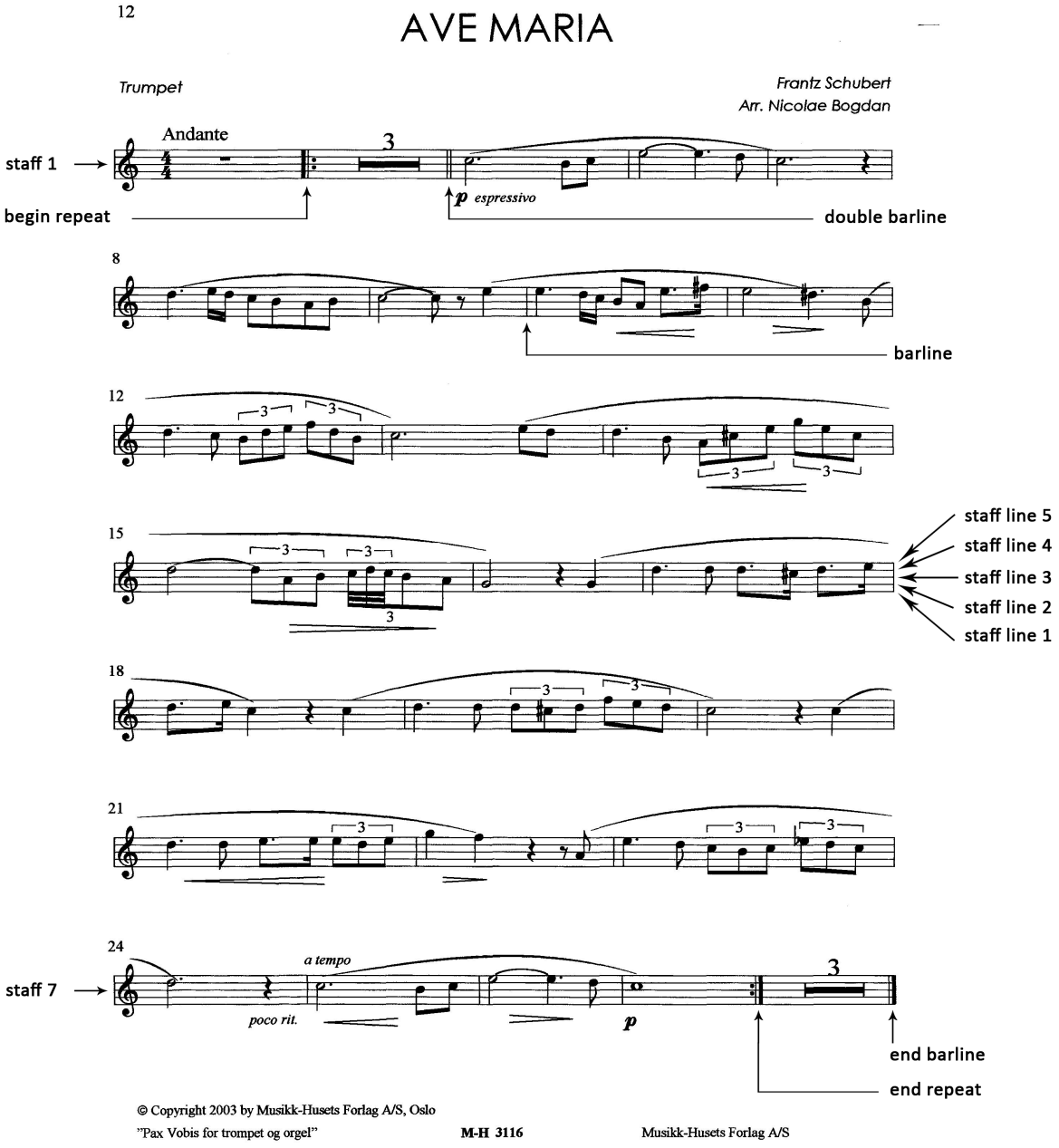


FIGURE 2.1: A typical note sheet. A score consists of several staffs, in this example seven. Each staff is again composed of five staff lines. These are numbered from the bottom, unlike the staffs which are numbered from the top.

2.1.1 Staff Lines

A staff is a group of five parallel, horizontal lines, called staff lines. Staff lines are the fundamental elements of a score, forming a grid system for musical symbols. Note heads can be placed on the staff lines or in the open space between two staff lines. The height of a note head must therefore be approximately the distance between two staff lines and the thickness of one or two staff lines, see Figure 3.3 (p. 20). A staff is again divided into smaller time units, divided by vertical bar lines (Ng, 2002; Luckner, 2006; Baumann, 1995).

In addition to the musical symbols found on the staff, music notation often has other symbols outside the staff area, such as bar numbers, dynamics, fingering information, lyrics, etc. (Bainbridge and Bell, 2001). It is therefore important to find out which objects are text in the form of lyrics, and which are text such as dynamic markings, that influence the playing of the melody. It is common to only concentrate on the most important symbols, and overlook the rest.

See Figure 2.1 for an example of a typical music sheet, showing the staffs and the staff lines, and how these are numerated together with various types of bar lines.

2.1.2 Bar Lines

Bar lines are the most important vertical lines in a music score, since they divide the score into time units. When every time unit is isolated, they can be processed independently, looking for musical symbols. Together with the time signature (found in the beginning of the first staff, immediately after the key signature, see Table 2.6), it is possible to classify the symbols using musical rules.

2.2 Notes and Rests

2.2.1 Notes

Notes are essential symbols in music notation. Note heads have a fixed shape, and may be hollow or filled. Further on they may appear one by one, or in relation to other notes. See Table 2.2 for the most common relationships between notes. Most notes have note stems attached to them, except for whole notes, which only appear as a hollow note head, see Table 2.1.

2.2.2 Rests

All rests are centered around the third staff line. The rests also have different durations, like notes, see Table 2.1.

2.2.3 Note Value

The length of a note is obtained from an analysis of the number of hooks (single note) or beams (group of notes), or duration dots. Hooks are found at the opposite end of the note stem compared to the note head, while duration dots may only appear after a note head or a rest (Rossant and Bloch, 2002). See Table 2.1 for an overview of notes and rests of different durations, and Table 2.2 for the most common relationships between notes.

2.2.4 Pitch

The pitch of a note is found from the vertical position of its head relative to the staff lines, together with the information from the clef (see Table 2.6), the key signature and accidentals (see Section 2.3.1). See Figure 2.2 for an illustration with the *names* of the pitches. The distance between two *C*'s (for instance a *C4* and a *C5*) or two *D*'s etc. is called an octave.

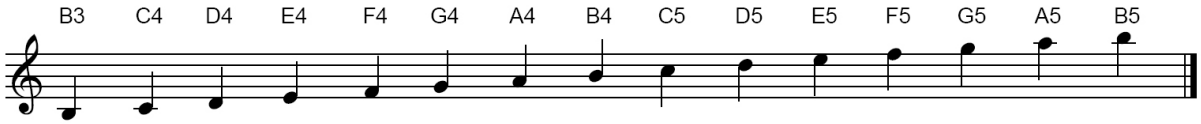


FIGURE 2.2: Pitches

Notes and Rests

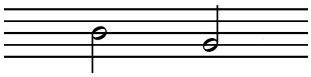
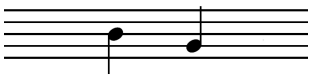
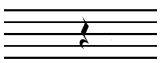
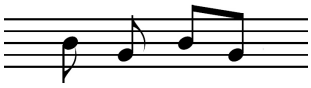
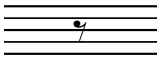
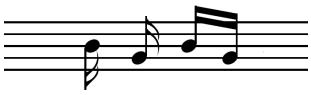
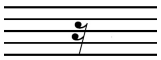
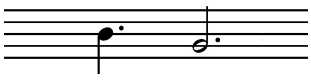
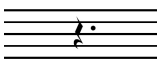
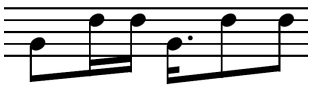
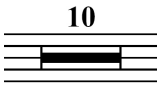
Notes	Rests	Note Value
		Whole
		Half
		Quarter
		Eighth
		Sixteenth
		Duration dot
		Beamed notes
		Measure rest

Table 2.1: An overview of the most important types of notes and rests, and their relative durations.

Note Relationships

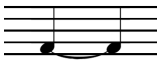
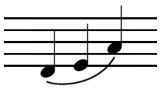
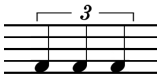
Symbol	Name	Description
	Tie	A tie is a curved line connecting the heads of two notes of the same pitch, indicating that they are to be played as a single note with a duration equal to the sum of the individual notes' duration.
	Legato	A legato indicates that the notes it embraces are to be played without separation.
	Tuplet	A number of notes of irregular duration are performed within the duration of a given number of notes of regular time value. Tuplets are named according to the number of irregular notes; e.g., duplets, triplets, quadruplets, etc., triplets are the most common ones.

Table 2.2: An overview of the most important types of relationships between notes (Wikipedia, May 2009).

Accidentals

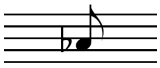
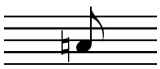
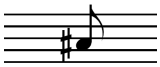
Symbol	Name	Description
	Flat	Lowers the pitch of a note by one semitone.
	Natural	Cancels a previous accidental, or modifies the pitch of a sharp or flat as defined by the key signature.
	Sharp	Raises the pitch of a note by one semitone.

Table 2.3: An overview of the most common accidentals (Wikipedia, May 2009).

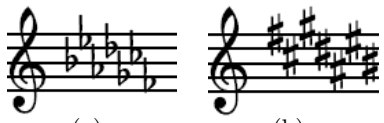
2.3 Other musical symbols

2.3.1 Accidentals and Key Signatures

Accidentals modify the pitch of the notes that follow them on the same staff position within a measure (between two bar lines), unless cancelled by an other accidental. The three most common types of accidentals are flats, sharps and naturals. These are described and shown in Table 2.3.

Key signatures are found in the beginning of each staff, right after the G clef, and are applied to the music that follows on the entire staff. This avoids the use of accidentals for multiple notes. An accidental in the key signature is valid for all octaves.

Different keys are defined by the number of sharps or flats in the key signature. The accidentals in the key signature follow a strict pattern, they are added in a certain order, from left to right in Figure 2.3(a) and 2.3(b) (see Table 2.4 for the same overview in form of a table). In other words, the number of accidentals in the key signature may vary between zero and seven, and either flats or sharps, never a combination.



(a) (b)

FIGURE 2.3: (a): Flat key signature, (b): Sharp key signature (Wikipedia, May 2009).

Number of accidentals in the key signature	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Flats are added in this order	B	E	A	D	G	C	F
Sharps are added in this order	F	C	G	D	A	E	B

TABLE 2.4: The order in which the accidentals are added in the key signature. Note that the order of flats is the opposite of the order of sharps.

2.3.2 Dynamics

Dynamics indicate the relative intensity or volume of a musical line, see Table 2.5 for the most common ones.

Dynamic Symbols

Symbol	Name	Description
pp	Pianissimo	Very soft
p	Piano	Soft
mp	Mezzo piano	Half as soft as piano
mf	Mezzo forte	Half as loud as forte
f	Forte	Loud
ff	Fortissimo	Extremely loud

Table 2.5: An overview of the most common dynamic symbols (Wikipedia, May 2009).

Clefs and Time Signatures

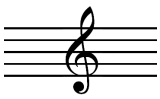
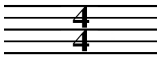
Symbol	Name	Description
	G clef	A clef defines the pitch range, and is usually the first symbol on a staff.
	Time signature	A time signatures defines the meter of the music by establishing the number of beats per time unit (between two bar lines).

Table 2.6: An overview of the most important types of symbols that appear at the beginning of a music score (Wikipedia, May 2009).

3 Overview of OMR Approaches

In this chapter we will give an overview of some OMR approaches. There is little variation when it comes to image processing strategies used in different OMR projects, while the differences between various systems mainly occur in the segmentation and recognition stages.

Rossant and Bloch (2007) refer to two main parts in the OMR process:

- In the first part *low-level* preprocessing is used. Isolated objects are detected and some hypotheses regarding these objects are stored.
- In the second part the final correct decision is made through *high-level* processing including contextual information and music writing rules. In Rossant and Bloch (2007), this *high-level* step relies on the fuzzy sets and possibility framework, since it allows dealing with symbol variability, flexibility and imprecision of music rules, and merging all these heterogeneous pieces of information.

The differences between various systems are typically found in the second part, while the *low-level* preprocessing is more or less similar, though there are some variations in names and terms.

A typical OMR system consists of a set of stages. The most common ones are:

1) Preprocessing

2) Segmentation

3) Classification

4) Post-processing

The terminology may vary, some articles talk about *Musical Object Location* or *Primitive Detection* when referring to the *Segmentation* stage. Similarly, Stage 3 is sometimes called *Musical Feature Classification* or *Musical Symbol Recognition*. In practice these stages are the same as those mentioned above (Anstice et al., 1996; Bainbridge and Wijaya, 1999; Bainbridge and Bell, 2001).

3.1 Common Problems and Difficulties

There are some problems that often occur in OMR processes. The most common ones are the problems of touching objects and broken objects. Because of these two main problems, which are unavoidable, it is important that the *segmentation* stage is tolerant to these defects (Anstice et al., 1996).

3.1.1 Touching Objects

Scores are often complex, with a very high density of symbols. As a result, connections are often found between musical objects that should not be touching, causing segmentation problems. Two or more objects that are connected appear as a single object. The problem does not only occur when symbols are connected to other symbols – the most significant form of this problem is that the symbols are superimposed on the staff lines. The staff forms an important *coordinate system*, but when it intersects with many symbols it may cause trouble. Removing the staff lines must be done with care, that is by trying to determine weather or not a pixel would have belonged to the background or to a musical symbol if the staff line had not been there.

3.1.2 Broken Objects

Fragmented objects may be a result of noise, caused by either bad scanning, thresholding, segmentation or poor quality of the original document. Objects affected in this way are small. The most frequent case of broken objects is caused by staff line removal. If we remove the lines without caring about the objects around, they will remain fragmented (Couasnon and Camillerapp, 1995).

Other problems that may occur are:

- Skew and bowing of staff lines.
- Discontinuity in staff lines and note stems.
- Lack of constraints related to musical rules in the recognition stage.
- The complexity of primitive arrangements (e.g grouped notes).
- Symbol variability and evolution (Rossant and Bloch, 2007; Baumann, 1995)

In the following sections we will go through each step of the OMR process, see p. 13. Different approaches for each single step are described.

3.2 Preprocessing

This section concerns mainly information about input data to the various systems, binarization and rotation methods, and staff line localization and removal.

3.2.1 Input data

Generally the input bitmap in OMR has 256 gray levels, and is of A4 size. Almost all OMR studies found in the literature use images scanned at a resolution of 300 dpi (dots per inch). A higher resolution will only consume more computer space and memory and make the system slower. An exception is the *Guido*-system, in Szwoch (2007), which operates on both binary, gray level and full-color images. In addition to scanned images, also images taken with a 5 Mpix digital camera with a resolution between 100-400 dpi are used. *Guido* also tries to deal with images of bad quality.

Most of the systems only operate on monophonic music (one voice per staff not containing chords) with five staff lines and written in *western common music notation* (also called CMN). There are a few exceptions; some programs are specialized in the area of ancient music or handwritten music, like Fornés et al. (2006); Pinto et al. (2000) and Vieira and Pinto (2001); while the *Guido*-system, in Szwoch (2007) handles printed music, with possible polyphony and one or two voices per staff.

In Rossant and Bloch (2005), some global information, such as the clef, the key signature and the time signature, are given as input to the system in addition to the image.

After the scanning, the image is prepared for segmentation. In this step the image is binarized and rotated (Vieira and Pinto, 2001). Some noise may also be removed by deleting small, isolated, black pixels (Ng, 2002).

3.2.2 Binarization

The input image is converted into a binary image. A binary image is an image I where $I(x,y)$ can take the value 0 or 1 (white or black). The binarization methods may vary from article to

article. A few of the methods mentioned are Otsu's global thresholding method and Niblack's local thresholding (Szwoch, 2007; Fornés et al., 2006).

3.2.3 Rotation

In OMR literature several different techniques for rotating the input image are described. Fornés et al. (2006) detect staff lines using the Hough Transform, while Rebelo et al. (2007) study horizontal projections made with different rotation angles, and keep the image with the highest local maxima.

3.2.4 Staff Line Localization

All reported OMR systems detect staff lines as the initial stage, after completing the rest of the necessary preprocessing described above. This is one of the fundamental stages in the OMR process. The staff lines create a two dimensional coordinate system or grid. By finding these lines, you get an idea both on the size of the music notation used in the current score, and on where to search for the remaining musical symbols.

From the staff lines it is possible to find an estimate of the staff thickness and the average distance between two staff lines, and consequently the size of the music notation. You also get information about rotation or deformation. It is therefore very important to correctly detect and process the staff lines (Bainbridge and Bell, 2001; Wijaya and Bainbridge, 1999).

3.2.4.1 Horizontal Projections

Different methods for staff line detection have been researched, but the most widely used method is based on *horizontal projections*. See Section 4.2 for a brief description of horizontal projections.

The staff lines appear as distinct peaks in the horizontal projection histogram. The peak information can also be used to find the rotation angle of the image (Bainbridge and Bell, 2001). Some choose instead to calculate the horizontal projection of smaller parts of the image, such as a part on the left side and another part on the right side. It is then possible to compare these two pieces, and compute the correlation between them. If the image is relatively straight, the correlation will be high (Bainbridge and Bell, 2001; Rossant and Bloch, 2002).

Other possible solutions to the staff line detection problem are the use of the Hough Transform, global projections or mathematical morphology. The *Guido*-system uses local projections, which

results in a more efficient detection of skew (see Szwoch (2007) for more details), while Rebelo et al. (2007) present a method based on a shortest path approach. Here also lines with some curvature, discontinuities or inclination are robustly detected.

3.2.5 Removing or Ignoring the Staff Lines

After completing the staff line identification, it is desirable to locate the individual musical objects. To do this there are two different approaches; removing or ignoring the staff lines (Bainbridge and Bell, 2001). In most cases the staff lines are removed, but this may vary from author to author. Keeping the lines in the image is mainly done to avoid that objects become fragmented, as we saw in Section 3.1.2. Therefore if you wish to remove the staff lines, it is important to do this very carefully, so that none of the musical objects superimposed on the staff lines get damaged.

3.2.5.1 Ignoring the Staff Lines

Bellini et al. (2001) choose not to remove the staff lines. The reason for this is that the staff lines give important information about the features of the music sheet, such as the staff line thickness, the distance between staff lines etc. Also the knowledge of their position makes it possible to detect the right pitch of the notes, and thus the staff lines are used in the segmentation process.

By ignoring the staff lines, it is possible to search between the staff lines for musical objects. When using this approach, it is important to decide when one object stops and the next one starts.

3.2.5.2 Removing the Staff Lines

A more popular method is to remove the staff lines, since the lines do not depend on the complexity of the music notation (Bainbridge and Bell, 2001). Besides, the information Bellini et al. (2001) mention is also possible to collect before removing the staff lines. An important argument for removing the staff lines is that the lines connect and intersect almost all other symbols, so their presence disturbs the following recognition of musical objects. It is therefore desirable to remove the staff line segments, which do not overlap any symbols to prevent that intersecting symbols become distorted (Reed and Parker, 1996; Rossant and Bloch, 2002). When removing the staff lines it is important to decide which line segments that can be deleted, and which cannot. Deleting staff lines uncritically may fragment most symbols (Fornés et al., 2006). There are mainly two different methods for doing this; descriptions of these follow below.

Method 1

The first method consists in traversing along each staff line, deleting black pixels except for where the thickness exceeds a certain threshold (depending on the average staff line thickness). This technique works well, except that some symbols, such as whole and half notes, may become fragmented. Problems with note fragmentation are completely avoided by working with the original image when detecting note heads (Reed and Parker, 1996). Also Rossant and Bloch (2002) write about the same method; each staff line, located at the y_0 position, is horizontally tracked. For each x-coordinate, the length of the vertical segment above and under y_0 is computed. If both of these measures are less than the average staff line thickness, the segment can be removed.

Method 2

The second method consists also of removing the staff lines piecewise. As for the first method, this algorithm follows along the line, replacing it with background pixels unless there is evidence of activity on either side of the line. However, instead of using a thickness-threshold to check if there are any musical objects interfering with the staff line, you search a region no more than two pixels away from the top or bottom of that part of the staff line. If the pixels above and below the staff line belong to the background, the staff line in the current y-coordinate can be deleted. Despite measures taken to keep objects intact, this algorithm may cause some fragmentation, especially if objects blend tangentially with staff lines, such as a bass clef or a hollow note head (Bainbridge and Bell, 2001).
Alternatively, other approaches simply remove the staff lines without bothering about the objects lying around the staff lines. For instance Bainbridge and Wijaya (1999) simply presume in the next stages of the system that fragmented objects will occur, and base the rest of the stages on this.

3.3 Segmentation

The symbol recognition cannot be done successfully if the symbol localization was not performed accurately. Finding the correct location of the symbols is therefore essential for the result of the final symbol recognition.

Now that the staff lines have either been removed or ignored, the remaining black pixels should be a part of musical objects. Segmentation means dividing the image into interesting parts, hence the next stage in the OMR process is to locate the musical symbols. In previous projects, different pattern recognition techniques have been used to solve this problem, such as projection methods and morphology.

3.3.1 Musical Primitives

In successful OMR systems, it is common to work on a sub-symbol level, that is to detect the musical primitives.

Musical symbols may be split into primitives, which sometimes do not correspond to musical symbols. For example, a stem is a primitive, to be classified independently of its note head, and a beam is recognized independently of its connected stems, see Figure 3.1. Each primitive can be used to build several musical symbols. Since musical symbols may be grouped to form compound symbols or symbols may overlap or touch other symbols, looking at the primitives instead of the musical symbols, makes the recognition process less complicated (Ng, 2002; Bainbridge and Bell, 2001; Bellini et al., 2001).

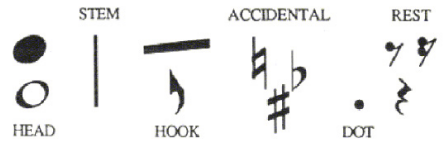


FIGURE 3.1: Example of musical primitives (Bellini et al., 2001).

The following sections describe various approaches to see the differences and resemblances between them.

3.3.2 Rossant and Bloch (2002, 2005)

After the staff line removal, Rossant and Bloch detect all *vertical objects* [1]. These symbols may be grouped in four main classes (see Figure 3.2):

- Notes (except whole notes)
- Accidentals: flats, sharps, naturals, grace notes
- Bar lines
- Rests: crotchet rests

In each column of pixels of the staff area, the longest of these vertical segments is detected. These segments may belong to a vertical line or be part of a thick horizontal line (e.g. noisy interconnected beams of note groups). A window is moved from left to right in order to analyze

[1] Musical symbols featured by a vertical segment longer than 1.5 staff spacing (that is $1.5d$, see Figure 3.3).

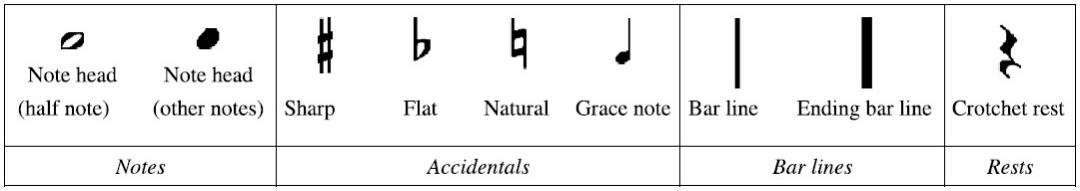


FIGURE 3.2: A set of *vertical objects* (Rossant and Bloch, 2002).

the neighboring segments and store only the main segment of each vertical line. It is then used as input in a region growing and image-labeling algorithm, such that the relevant musical symbol is isolated and precisely located by a bounding box (Rossant and Bloch, 2002).

3.3.3 Bellini, Bruno, and Nesi (2001)

Bellini, Bruno, and Nesi are among the few authors that chose not to remove the staff lines in the preprocessing stage. They use projection profiles to extract the basic symbols and their positions. Here, the basic symbols are considered the musical primitives.

The music sheet is analyzed and recursively split into smaller boxes by defining a set of horizontal and vertical cut lines that isolate the primitives. This procedure can be divided into three levels:

1) First, the image is segmented to extract sub-images including only one staff. In addition, a few parameters that are needed later, are estimated, e.g. staff line thickness, n, and the distance between staff lines, d, see Figure 3.3.

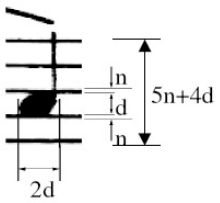


FIGURE 3.3: Illustration showing the staff line thickness, the distance between staff lines and the note head width (Bellini et al., 2001).

2) Secondly, the sub-image of each staff is processed to extract the symbols having a width close to that of a note head, that is $2d$. This is performed in three steps:

   - Detection of groups of symbols (e.g. beamed notes) and isolated symbols (e.g. clefs, rest, bar line).
   - Detection and labeling of note heads.
   - Detection of other musical symbols or parts of them. This typically includes groups of symbols located very close to each other. This step is not always needed.

3) Finally, the musical symbols are decomposed into primitives. Two different segmentation methods are used, one for image segments containing note heads, and one for image segments not containing note heads (these segments may contain other symbols). Both segmentation procedures are based on the horizontal and vertical projection profiles (see Section 4.2).

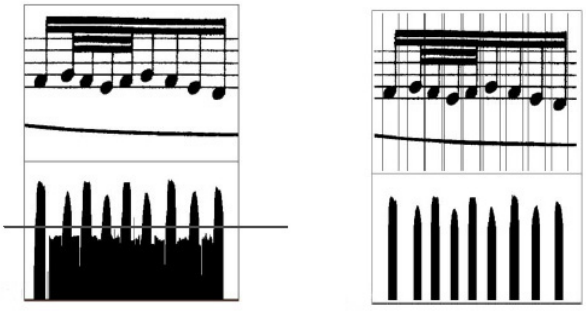
3.3.3.1 Detection of Groups and Isolated Symbols

The detection of groups and isolated symbols uses a moving window and a binary function F. The window is moved from left to right, and sets the values of F to 0 if the staff is empty, and 1 if the staff contains symbols. Here the isolated symbols, such as clefs, bar lines, time signatures and whole notes, are detected while groups of symbols are processed in the next phase in order to also decompose these symbols into smaller image segments.

3.3.3.2 Detection and Labeling of Note Heads

In this step, the goal is to slice the image segments where $F = 1$. In *western notation*, note heads have a diameter greater than or equal to the distance between two staff lines, and thus a reasonable approximation of the note head width is $2d$. Hence, image segments from the previous detection phase are processed based on their width, and only segments larger than $2d$ are considered, see Figure 3.3.

When it comes to isolating the note heads, a double threshold mechanism is used to obtain the results shown in Figure 3.4(b). Each sub-image containing a note head is then labeled.



(a) (b)

FIGURE 3.4: (a): Vertical projection before thresholding. (b): Vertical projection after thresholding; ready for extracting image segments with note heads (Bellini et al., 2001).

3.3.4 Fornés, Lladós, and Sánchez (2006)

Fornés, Lladós, and Sánchez work with ancient, handwritten scores, but the techniques for locating musical symbols are more or less the same as for modern scores. Here, vertical lines and note heads are the first graphical primitives to be recognized.

Median filters with a vertical structuring element are used to detect the vertical lines, so that only symbols with a vertical shape will remain. The size of the structuring element depends on the staff line spacing. It is also possible to detect the lines using Hough Transform.

A morphological opening with an elliptical structuring element is used to detect the filed note heads. The size of the structuring element depends on the staff spacing, and is oriented 30 degrees. Then the objects with adequate circularity and area are chosen. This approach gets all the filled note heads but also false positives (which are discarded in the next stages). Chen and Xia (2003) do something quite similar, but instead of a morphological opening, they use a morphological *hit and miss transform*.

3.4 Classification

Having found the location of the musical objects, it is time to recognize the symbols and classify them. Template matching is the most popular method used for this purpose, but some authors have also used projection methods, morphology, neural networks, geometrical features and moments. This stage is traditionally the weakest one in the OMR process, and contributes strongly to increasing the error rate (Rossant and Bloch, 2002; Bainbridge and Wijaya, 1999).

The first symbols to be recognized are often isolated primitives and symbols that normally are located at certain positions with respect to the staff (e.g. clefs, time signatures and key signatures). After this initial recognition of primitives, the symbols are reconstructed using basic musical syntax. Musical rules are often used to decrease the classification error. A set of syntactic, geometrical and logical rules are defined for each musical symbol. The rules describe primitives, their allowed spatial relations and location on the staff. These rules may be applied to reconfirm the recognition; for example, a dot may only appear above, below or after a note head, or after a rest. An accidental may only occur before a note head or another accidental, or as an isolated symbol at the start of a staff after the G clef (Ng, 2002; Szwoch, 2007; Ng et al., 1995).

3.4.1 Bar Lines and Note Heads

If a vertical line covers all five staff lines and has no note head attached to its extremes, it is most probably a bar line. Vertical lines can be found either by using vertical projections (see Figure 3.5) or by morphology using a line-shaped structuring element with size equal to the staff height ($5n + 4d$, see Figure 3.3).

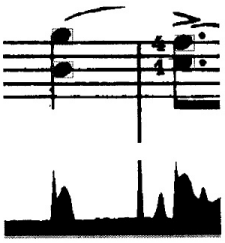


FIGURE 3.5: Vertical line detection based on vertical projection analysis (Su et al., 2002).

Su et al. (2002) use horizontal and vertical projections as features. Pattern recognition is then used to test every *head candidate*. Similarity measures between the features of the test image and the two types of note heads (filled and hollow) are computed, and the candidate is classified as a filled note head, a hollow note head, or neither one of them.

The following sections describe various approaches used by different authors, to see the differences and resemblances between their approaches.

3.4.2 Rossant and Bloch (2002, 2005, 2007)

Symbol analysis is mainly based on template matching; a normalized correlation score $C_s^k(x, y)$ between the segmented object s and the models M^k of the reference base (see Figure 3.6) is computed in a small area around s. This is done for all the segmented objects (Rossant and Bloch, 2005).

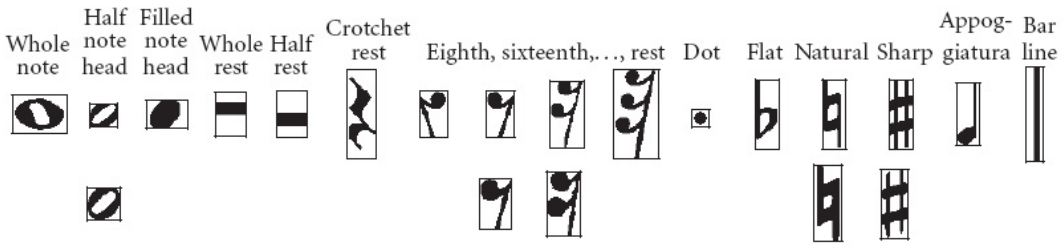


FIGURE 3.6: Templates M^k. Up to two models/templates are used for each class k (Rossant and Bloch, 2007).

The maximum correlation score is 1.0. So if the model and the segmented object are identical, that is if they match perfectly, the correlation score will be equal to 1.0. The score decreases with the number of pixels that differs from the model.

Next, at most three recognition hypotheses are selected. A recognition hypothesis classifies the object s to class k, if the associated correlation score is larger than a minimum threshold t_m. When the highest correlation score obtained for class k is less than a decision threshold, $t_d(k)$, it is also possible that there is no symbol. By using these thresholds, it is possible to take into consideration that some classes are more sensitive to typewriting variations (small value for $t_d(k)$) or have a higher probability of false detection (large $t_d(k)$) than others. This procedure has some limitations; the correlation scores may be ambiguous, and the highest score does not always correspond to the right hypothesis.

No universal music font exists, meaning that symbols of one class may have different shapes. Also the size of the symbols may vary, even within the same score. Template matching is sensitive to both typewriting variations and scaling. A way to deal with this is to store both different sized bases of models and variations of the same symbols. By using the staff spacing as a measurement scale, you find which base is to be used for analyzing the current music sheet. To deal with different fonts, the program tests at the beginning of the analysis process, several models for each class of musical symbols, and determines which model to use based on the highest correlation score.

Graphical knowledge is used to improve the computation of the correlation scores. That is why the analysis of symbols featured by a vertical segment is distinguished from the analysis of all the other symbols. The extracted geometrical features increase the reliability and speed of the analysis, by determining a small search area and eliminating impossible classes.

3.4.2.1 Analysis of *Vertical Objects*

For the *vertical objects* [2], it is interesting to examine the geometrical features extracted in the segmentation step. For example, a symbol with a narrow bounding box located between the first and the fifth staff lines may be a bar line.

For each of the symbols in Figure 3.2, five criteria are computed from geometrical features of the bounding box and of the vertical segment, and also from their position relative to the staff lines. When three of these are satisfied, the correlation with the models of the group is computed in an area derived from the bounding box and from the position of the vertical segment. For each tested model k, the correlation score, $C^k(x_k, y_k)$, and the corresponding location, (x, y),

[2] Musical symbols featured by a vertical segment longer than 1.5 staff spacing (that is $1.5d$, see Figure 3.3).

are stored.

Three threshold values have been defined:

1) Decision threshold t_d, validating the recognition of a template;

2) Minimum threshold t_m, which always has to be reached to store a template as a possible match;

3) Ambiguity threshold t_a, to deal with a second highest score.

The model with the highest score $C^{k1}(x_{k1}, y_{k1})$ is stored if it is greater than the minimum threshold, t_m, but also the hypothesis of absence of a musical symbol if the score is under the decision threshold, t_d. In case of ambiguity, the second highest score $C^{k2}(x_{k2}, y_{k2})$ is also stored.

3.4.2.2 Analysis of Remaining Symbols

This step mainly concerns rests and whole notes, see Figure 3.7. Rests have a fixed location around the third staff line, so the search for these symbols may be restricted to computing the correlation around this line, in the free spaces between the bounding boxes of the symbols from the previous section.

For whole notes, on the other side, the correlation score has to be computed for a larger area of the staff. A good match between a template and a symbol will give a correlation score larger than the minimum threshold, t_m, and the result is stored. As before, there may be some ambiguity when several models obtain a high correlation score at almost the same position. Hence, an iterative process looks for the model with the highest score and examines its neighborhood; when a second match overlaps the first one, it is also stored as the second hypothesis. The process runs until every hypothesis is put in the list of possible symbols, as first or second hypothesis.

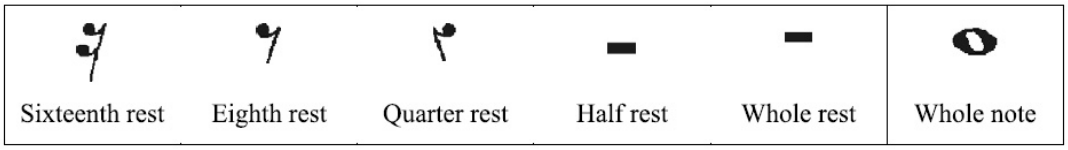


FIGURE 3.7: A set of models used in the analysis of remaining symbols (Rossant and Bloch, 2002).

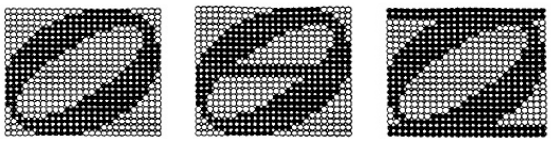
3.4.3 Reed and Parker (1996)

Reed and Parker have divided the symbol recognition into the detection of lines and curves, character-symbol recognition and note head recognition. Character-symbols are symbols which are isolated after staff line removal, such as accidentals, rests and clefs. Line adjacency graphs are used to recognize both horizontal and vertical lines, as well as curves (e.g. ties or slurs).

To recognize the character-symbols, character profiles are used. Character profiles are functions that measure the perpendicular distance of the character's outer contour from some reference axis, usually the left or bottom edge of the character's bounding box.

A rule based classifier, using measurements obtained from the profiles, can be used to recognize the symbols. The rules are constructed by building a database of features from known symbols and identifying common and unique features for each symbol class. These rules are then used to determine what class an unknown object belongs to. This technique requires that the symbols are isolated, but unfortunately, some symbols, especially accidentals, sometimes touch other symbols.

Reed and Parker also use template matching, but only to recognize the note heads. Since they usually are attached to stems, or even other note heads (in the case of chords), it is not possible to isolate note heads by connected component analysis. Template matching is therefore used. Template matching is performed on the original image, to avoid problems of fragmentation caused by staff line removal. Staff lines must therefore be included in the template, thus it is necessary to have several templates for each type of note head to deal with the different relative vertical positions a note can occur on a staff, see Figure 3.8.



(a) (b) (c)

FIGURE 3.8: Templates used for detecting half notes; (a): the original template, (b): on a staff line, and (c): between staff lines (Reed and Parker, 1996).

Further on, Reed and Parker write that ordinary template matching is very limited because it only considers the foreground pixels. Background pixels cannot contribute to a positive match. So if the interior background pixels (such as the holes in a half note or a whole note) of the symbol are the only pixels that differentiate from the template, the correlation score may give a

misleading answer. To improve the accuracy of template matching in such cases, the template pixels are labeled as:

- foreground pixel,
- background pixel, or
- interior background pixel.

The interior background pixels are matched against background pixels in the test image, and contribute to a positive match only when sufficiently many foreground pixels match.

Template matching has a very poor run-time performance. A way to deal with this is to use partial templates; the original template is divided into e.g. four templates such that each partition samples the original. When testing for a match, each partition participates in one pass over the test image. A temporary match index is computed after each pass, and this approximates the actual match index obtained by comparing each pixel. A sufficiently low match index aborts the template match, avoiding the cost of comparing every pixel when a mismatch is obvious.

3.5 Post-processing

In the previous sections, we looked at each object individually. In this section we will look at musical symbols in relation to each other. We will integrate musical rules, in order to control the classification, and possibly correct some mistakes from the previous stage.

3.5.1 Rossant and Bloch (2007)

Rossant and Bloch (2007) list some musical rules in their articles. The rules may be either graphical or syntactic rules. Most of these rules are flexible or contain imprecise parameters, so it is important not be too strict, but apply the rules with a certain degree of flexibility.

The syntactic rules are related to tonality, accidentals and metric. These rules are very flexible, and involve many symbols that may be situated far from each other.

The meter is generally considered at the end of the recognition process, in order to detect and correct errors. The number of beats per bar is checked, since it is a strict rule that always has to be satisfied.

3.6 Datasets and Results

Most of the OMR projects are meant to deal with monophonic music sheets, and compare results with commercially available systems. Most of the commercial systems can also deal with polyphonic scores. It is most common to scan the scores at a resolution of 300 dpi, which is a sufficiently large resolution, without occupying too much unnecessary space.

3.6.1 Reed and Parker (1996)

The program *Lemon*, in Reed and Parker (1996) was tested on a collection of ten scores, where only the first page of each score was used. The results were then compared with the results of two commercial systems, *MIDISCAN* and *NoteScan*. *Lemon* was slower than both the other systems, but on average it got higher recognition rates and significantly fewer false identifications. These results are summarized in Table 3.1, where also the names of the scores used are included.

Score	**Symbols**	**Recognition Rate**			**False Identifications**		
		Lemon	MidiScan	NoteScan	Lemon	MidiScan	NoteScan
Magnificat	239	99%	84%	88%	4	39	24
Canon in D	206	99%	88%	73%	1	2	5
Concerto No. 1	247	95%	98%	94%	9	5	5
Polovetzian Dance	281	98%	93%	86%	6	9	6
The Entertainer	233	94%	99%	64%	3	3	7
Für Elise	534	99%	96%	83%	5	30	79
Moonlight Sonata	179	96%	96%	74%	0	3	23
Sonata in C Major	256	99%	95%	88%	2	5	16
The Four Seasons (Winter)	264	81%	84%	79%	24	18	25
Antiphonae Marianae	214	80%	87%	90%	11	16	10

TABLE 3.1: Names of the music scores used in Reed and Parker (1996), with the associated recognition rates and number of false identifications, compared to two commercial programs. The best result within each category in each row is marked with gray background.

3.6.2 Rossant and Bloch (2002, 2004, 2005, 2007)

Rossant and Bloch have increased the number of music sheets used in their experiments from the oldest article (Rossant and Bloch, 2002) to the most recent one (Rossant and Bloch, 2007). In Rossant and Bloch (2002), experiments were performed on about 50 music sheets; in Rossant and Bloch (2004) they used 65 music sheets, and in Rossant and Bloch (2005, 2007) they had increased the number to 100 music sheets. If we look at the average recognition rate referred to the total number of symbols, it has increased from around 97% (Rossant and Bloch, 2002) using 50 sheets to 99.2% using 100 sheets, containing around 48 000 symbols. For an overview

Bar lines					Notes				
1:Initially right	2:Right corrected	3:Non corrected	4:False corrected	5:Missing or added	1:Initially right	2:Right corrected	3:Non corrected	4:False corrected	5:Missing or added
r1 = 99.5	r2=0.0	r3=0.0	r4=0.0	r5=0.5	r1 = 94.1	r2=3.1	r3=1.6	r4=1.1	r5=0.1
r = 99.5%					**r = 97.2**				
Accidentals					**Rests**				
1:Initially right	2:Right corrected	3:Non corrected	4:False corrected	5:Missing or added	1:Initially right	2:Right corrected	3:Non corrected	4:False corrected	5:Missing or added
r1 = 90.7	r2=2.0	r3=4.2	r4=0.9	r5=2.2	r1 = 75.2	r2=9.5	r3=1.7	r4=2.1	r5=11.5
r = 92.7					**r = 84.7**				

TABLE 3.2: Recognition rates from Rossant and Bloch (2002).

of the recognition rates from Rossant and Bloch (2002) divided into various symbol classes, see Table 3.2. The recognition rates are high for both symbol recognition (99.2%) and note length interpretation (99.3%), using a large dataset (Rossant and Bloch, 2007). The symbol error rate of 0.8% is divided into:

- Confusion: 0.2 %
- Symbol missing: 0.6%

For clearly printed music scores, the recognition can be perfect and the first hypothesis is generally the right decision (see Section 3.4.2 for a thorough explanation). However, for more difficult scores, strong ambiguities appear more frequently (Rossant and Bloch, 2002). Rossant and Bloch ignore text and annotations below and above the staff, and concentrate only on recognizing the symbols essential for reproducing the melody.

The algorithms are implemented such that it is possible to evaluate them by computing recognition rates, and to compare the program with commercial softwares, like *SmartScore*. Rossant and Bloch's approach leads to very good results, and is able to correct errors made by *SmartScore*. Examples of some of these errors are confusions between staccato dots and duration dots and interconnected symbols may be ignored. Rossant and Bloch do not have these problems because they include contextual knowledge by using musical rules.

In order not to train on specific test cases, the test set came from various composers and publishers. The music scores were also of various levels of difficulty, some with higher symbol density than others, more rhythmical complex, and thick printing causing connected objects. The program is given clef, tonality and metric as input parameters, so the program does not yet handle key change within a music score.

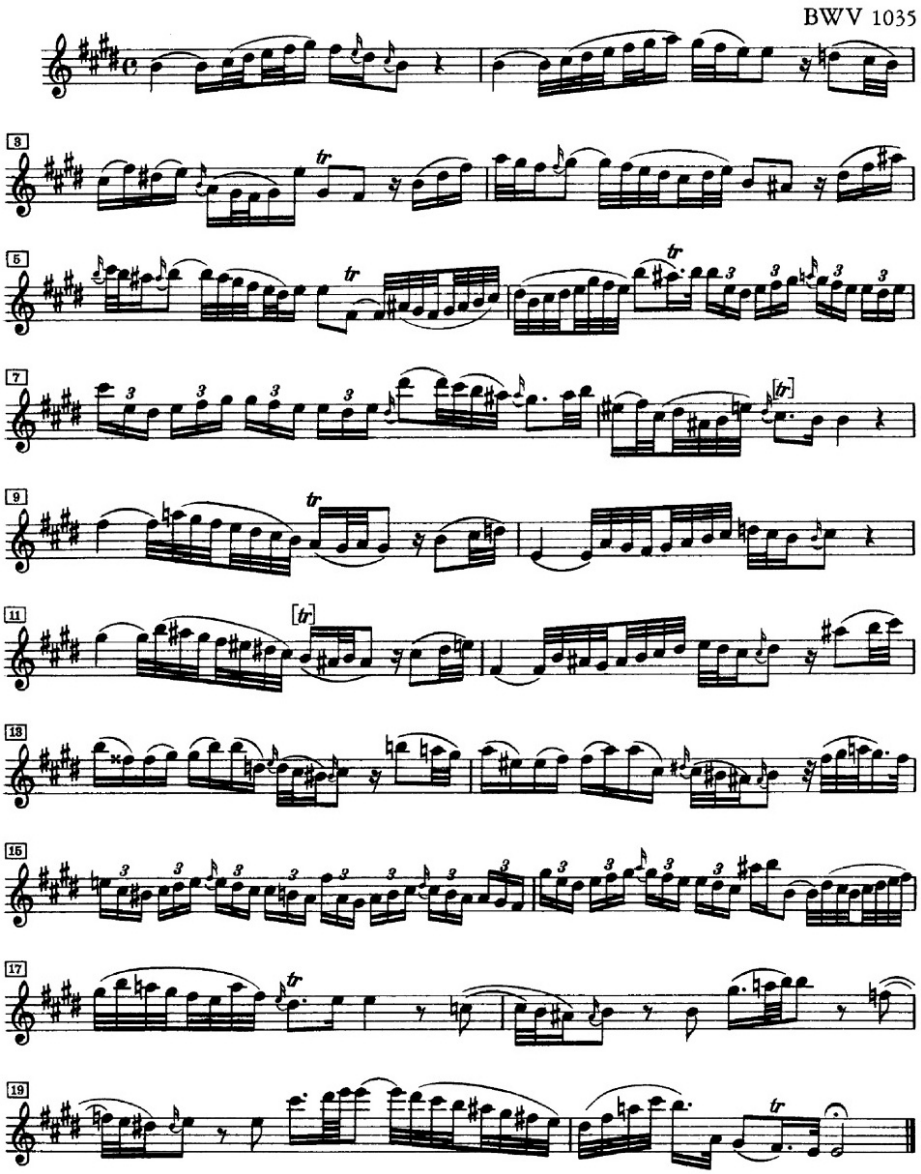


FIGURE 3.9: An example of an original music sheet used in Rossant and Bloch (2004).

3.6.3 Szwoch (2007)

To validate the recognition quality of the *Guido*-system in Szwoch (2007), 100 music sheets have been used in the experiments. The test set consists of images from various publishers and of different quality, mainly in A4 format. Some of the scores may contain polyphony (chords) and two voices per staff. The images were acquired by scanning at a resolution of 200-400 dpi or by photographing at about 200 dpi with a 5 Mpix camera.

4 Basic Concepts and Methods of Image Analysis

In this chapter we will review the image analysis methods which will be used and referred to in Chapter 5. This involves morphological operations, horizontal and vertical projections, correlation and template matching.

4.1 Morphology

In an OMR process it is common to binarize the input images. Therefore we limit to binary image morphology, excluding gray scale image morphology.

Morphological techniques are logical operations. A morphological operation on a binary image, I, creates a new binary image in which a pixel has a non-zero value only if the test is successful at the location in the input image. A structuring element is positioned at all possible locations in the input image and compared with the corresponding pixels (Efford, 2000, Ch. 11).

4.1.1 Structuring Elements

A structuring element, s, can be represented as a matrix of pixels with values 0 or 1. The dimension of the matrix is determined by the size of the matrix, while its shape is determined by the pattern of zeros and ones. The result of the morphological operation is influenced by the size and shape of s, however the main influence is given by its size, which determines the strength of the operation.

Placing a structuring element in a binary image, I, each of its pixels is associated with the corresponding pixel of the neighborhood under s. If a $pixel = 0$ in s, the corresponding pixel value in the image is irrelevant.

A structuring element has an origin, which usually is defined as the center pixel of s, but can also be defined as any other pixel in s, and even a pixel outside s. The origin should be indicated,

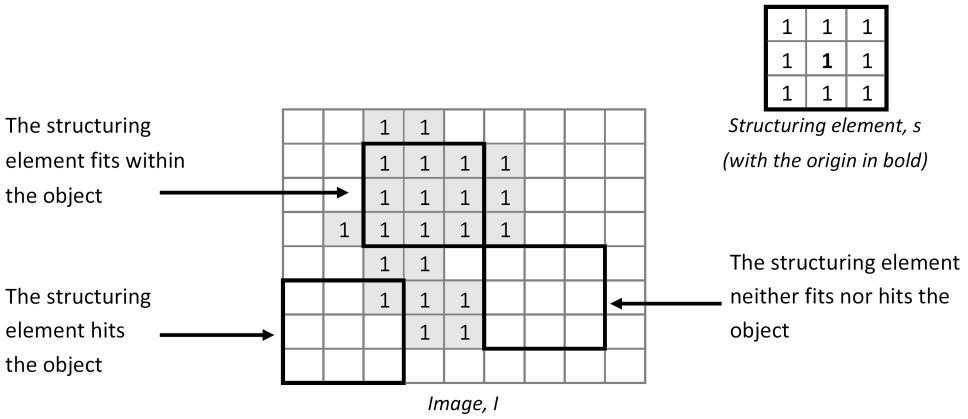


FIGURE 4.1: An illustration showing the various possibilities for a structuring element to fit, hit or miss an object.

for instance in bold as in Figure 4.1 (Efford, 2000, Ch. 11).

4.1.2 Erosion

The erosion of an image I by a structuring element s is defined as

$$I \ominus s.$$

To compute the erosion, position s such that its origin is at image pixel $I(x, y)$ and apply the rule

$$g(x, y) = \begin{cases} 1 & \text{if } s \text{ fits within } I \\ 0 & \text{otherwise,} \end{cases}$$

and repeat for all x, y. The structuring element s is said to *fit* in the image I in position (x, y), if each $pixel = 1$ in s corresponds to a pixel value $= 1$ in the image.

An erosion removes all objects which cannot contain s, and shrinks the rest by removing pixels from both the inner and outer boundaries of regions. Hence, an erosion will enlarge the holes enclosed by a single region as well as making the gap between different regions larger. It will also tend to eliminate small extrusions on a region's boundaries, besides, erosion is often used to remove small, unwanted features (Efford, 2000, Ch. 11).

4.1.3 Dilation

The dilation of an image I by a structuring element s is defined as

$$I \oplus s.$$

To compute the dilation, position s such that its origin is at image pixel $I(x, y)$ and apply the rule

$$g(x, y) = \begin{cases} 1 & \text{if } s \text{ hits } I \\ 0 & \text{otherwise,} \end{cases}$$

repeating for all x, y. The structuring element s is said to *hit* or intersect the image I in position (x, y), if a $pixel = 1$ in s corresponds to a pixel value $= 1$ in the image.

A dilation enlarges the objects by adding a layer of pixels to an object, both to its inner and outer boundaries. Hence, a dilation will shrink the holes enclosed by a single region and make the gaps between different regions smaller (Efford, 2000, Ch. 11).

4.1.4 Opening

The opening of an image I by a structuring element s is defined as

$$I \circ s = (I \ominus s) \oplus s,$$

that is, an erosion followed by a dilation, using the same s.

The advantage of this operation is that it does not shrink the regions, because the erosion is followed by a dilation. Any objects that survived the erosion will be reconstructed to their original size by the dilation. An opening can therefore create a gap between objects which are only connected via a thin bridge of pixels without shrinking the objects (Efford, 2000, Ch. 11).

4.1.5 Closing

The closing of an image I by a structuring element s is defined as:

$$I \bullet s = (I \oplus \hat{s}) \ominus \hat{s},$$

that is, a dilation followed by an erosion. Here, $\hat{s}$ is a rotated version of s, but this is often not important because structuring elements usually are symmetric.

The advantage of this operation is that it does not enlarge the regions, because the dilation is followed by an erosion. By eroding the result of the dilation the object will generally retain their shape and size. A closing can therefore fill an opening between two objects only separated by a small gap, without expanding the two regions (Efford, 2000, Ch. 11).

4.2 Horizontal and Vertical Projections

Projections are defined as operations which map a binary image onto a one-dimensional histogram or projection profile. The values of the histogram are the sums of the foreground pixels along a certain direction.

A horizontal projection of $I(x, y)$, projects a two-dimensional image onto the vertical axis, y;

$$h_H(y) = \sum_{k=1}^{w} I(k, y),$$

for all y, where h_H is the histogram and w is the image width. A a vertical projection, on the other hand, maps a two-dimensional image onto the horizontal axis, x;

$$h_V(x) = \sum_{k=1}^{h} I(x, k),$$

for all x, where h_V is the histogram and h is the image height. See Figure 4.2 for an illustration of these two projection directions.

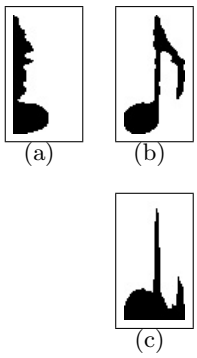


FIGURE 4.2: Illustration showing the image of an eighth note (b), together with its horizontal projection, (a), and its vertical projection, (c).

4.3 Correlation and Template Matching

A correlation between two objects, f and h, where the dimensions of h are $m \times n$, is defined as

$$g(x, y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j, k) f(x + j, y + k), \tag{4.1}$$

where $m_2 = m/2$ and $n_2 = n/2$.

It is common to normalize this, so that we obtain correlation values between 0 and 1:

$$g'(x,y) = \frac{g(x,y)}{m \times n}.$$

Correlation is often used to measure the similarity between images or parts of images. This can be done by making a small image selection which acts as a template. The template is then placed over the image and moved around until the position of maximum similarity or correlation is found. When we talk about template matching, h in Equation 4.1 is a template.

Correlation and template matching work well only if we know the size and orientation of the object we are interested in locating, and can create a suitable template. If the size and orientation of the objects might vary, different sets of templates are required to obtain good results (Efford, 2000, Ch. 7).

5 Algorithms and Analysis

In this chapter we will describe in detail each step in our proposed OMR approach. As presented in Chapter 3, an OMR system consists typically of four main steps;

1) Preprocessing

2) Segmentation

3) Classification

4) Post-processing

We will now look into each of these step in detail, describing the algorithms used, problems during the process and some discussion around the choice of methods. Step 4 is in this thesis limited to correct mistakes made in the pitch recognition for accidentals.

5.1 Preprocessing

In this section we will introduce the different preprocessing tasks; input data, binarization, rotation and staff line removal. By performing these tasks we prepare the input data for the following segmentation.

5.1.1 Input data

Our program takes a gray scale image as input data. The dataset consists of 33 monophonic music sheets, see Table A.1 for an overview of the titles and composers of these scores, and Figure A.1 for four examples from the dataset. The dataset is divided into a training set (14 sheets) and a test set containing the remaining 19 sheets. All of the sheets used can be found in *Pax Vobis for trompet og orgel* (Bogdan, 2003). The music sheets are all of A4 format, and they are scanned at a resolution of 300 dpi, and moreover the images contain 2496×3488 pixels. The size of typical symbols will be considered in Section 5.3.1.1.

We have only used the first page of scores that span several pages, and not used a score if it starts on the middle of a page. This way the format is equal for all of the sheets, with the title on top, followed by the staffs.

5.1.2 Binarization

None of the images in the dataset contain much variation, such as different light conditions in parts of the image etc. Therefore we tried to use a global threshold value, $t = 200$, which gave good results. However, manually finding a threshold value which works well for the entire dataset is hard. Thus we tried to use Otsu's global threshold method, and the results of this method turned out to be even better than the previous one. Using Otsu's method, the threshold is calculated automatically, which makes it easier to get a good result for all images, and even new sheets.

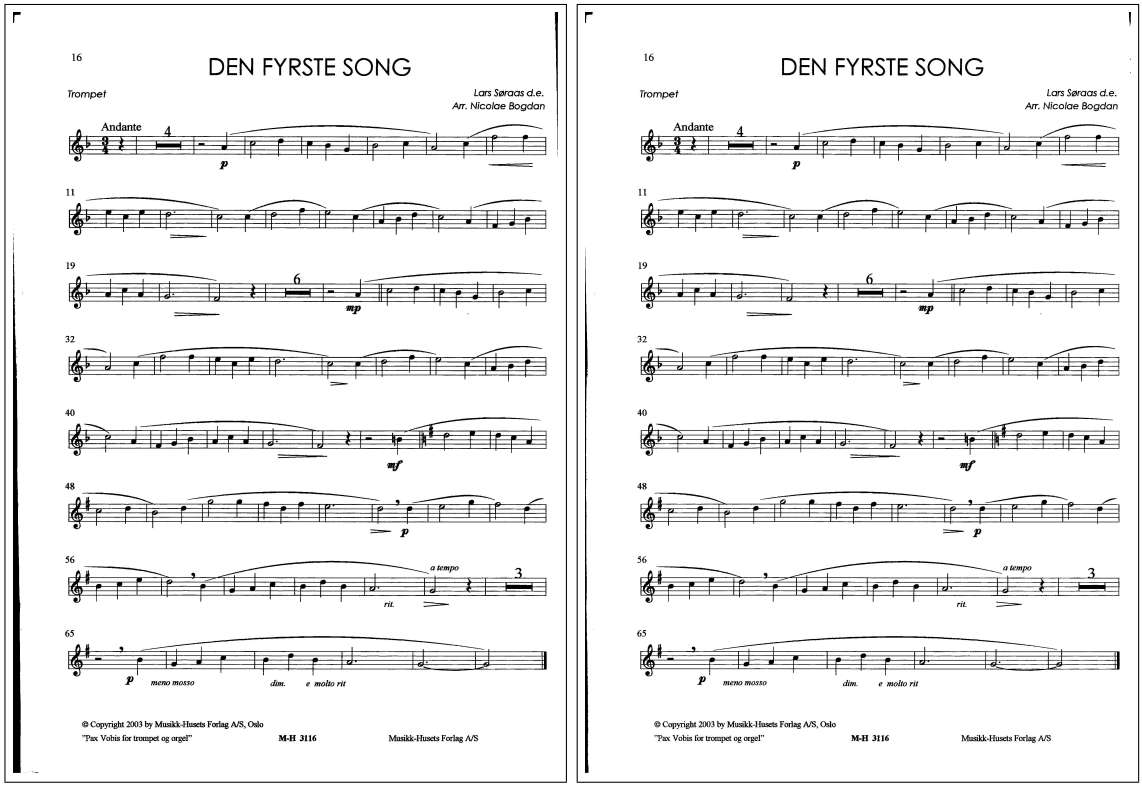
Otsu's Method *Otsu's method calculates the optimal threshold, by looking at where the two classes' (here foreground and background) within-class variance is minimized, and the separation between the classes is large (Albregsten, February 2008).*

Example 5.1 *In this example, an image, I, is binarized using the two different methods mentioned above. Figure 5.1(a) shows the result of the binarization using a global threshold value, $t = 200$, while Figure 5.1(b) shows the result of the binarization using Otsu's global threshold method. We can see that in Figure 5.1(a), some rather large black regions remain in the outer edges of the image. These regions are noise due to the scanning and are emphasized using this binarization method. Otsu's method, on the other hand, depreciates these regions, and reduces them.* □

After the binarization, we have a black and white image where the objects of interest, i.e. staff lines and musical symbols, are white, and the background is black. Having the image inverted and binarized simplifies the following processes, such as mathematical morphological operations.

5.1.3 Rotation and Staff Line Localization

The input images are seldom perfectly scanned when it comes to orientation. This may lead to slanted staff lines, which may be difficult to locate and remove. Besides, since we later in the OMR process will apply rotation variant methods, such as template matching, we wish to have all the input images on the same form, with horizontal staff lines.



(a) (b)

FIGURE 5.1: (a): Image binarized using a global threshold value; $t = 200$. (b): Image binarized using Otsu's global threshold method.

The challenge here is to find the rotation angle. For a description of the algorithm used to find the angle, see Algorithm 5.1. When rotating the image, we use Nearest-Neighbor Interpolation. Since we are dealing with binary images, this is the only interpolation method that will give an output image which also is binary. This is because the values of the interpolated pixels are taken directly from the pixels in the input image instead of computing a weighted average of neighboring pixels. Nearest-Neighbor Interpolation sets the values of pixels in the output image B that are outside the rotated image to zero. Rotation of the image is done using the `imrotate`-method in *MatLab*.

The coordinates found in Algorithm 5.1 (2)-(3) are plotted together with the image to see that the program detects the actual points of interest, see Figure 5.3(b). Figure 5.3(a) illustrates how the rule of Pythagoras (see Figure 5.2) is applied in practice to find the rotation angle. In addition to determining the rotation angle by means of H_1 and H_2, these pixels also inform us in which interval to look for all the staffs, and hence all musical symbols.

Once the image has been rotated, it is reasonable to use horizontal projections to find the staff lines. This is the most widely used method for this purpose in OMR projects, moreover it

Finding the Angle of Rotation

To rotate the image, I, we wish to find the coordinates of the upper left corner of the first staff, H_1, and the lower left corner of the last staff on the page, H_2.
Define $H_1 = (x_1, y_1)$ and $H_2 = (x_2, y_2)$. This will be used to compute the rotation angle, θ.

(1) Perform an erosion using a structuring element with a rotated L-shape.

(2) To find H_1: choose the point with the lowest x and y-coordinates
(the smallest sum of x and y-coordinates).

(3) To find H_2: choose the point with the lowest x and largest y-coordinate
(the largest difference between the y-coordinate and the x-coordinates).

(4) Apply the rule of Pythagoras to find the rotation angle (see Figure 5.2 for an illustration):

- define a as the change in the y-value between the two coordinates; $a = |y_1 - y_2|$,
- define b as the change in the x-value between the two coordinates; $b = |x_1 - x_2|$,
- the hypotenuse, h, is then $h = \sqrt{a^2 + b^2}$.

(5) Find the rotation angle θ:

$$\sin\theta = \frac{a}{h}$$
$$\Rightarrow \theta = \sin^{-1}\left(\frac{a}{h}\right).$$

(6) Rotate the image θ degrees using Nearest-Neighbor Interpolation.

ALGORITHM 5.1: Finding the Angle of Rotation

is a simple method, which is computationally fast (see Section 3.2.4.1 for more on horizontal projections). Horizontal projections may also be used to study images before and after rotation, to see that the lines become clearer and more distinct. This is illustrated in Example 5.2.

EXAMPLE 5.2 *In this example, we study the horizontal projections of two different music scores, "Den fyrste song" and "Ave Verum Corpus", both before and after rotation. As we can see in Figure 5.4(a), the projection of "Den fyrste song" has very clear, distinct peaks, even before rotation. The rotation angle for this sheet is found to be only 0.094 degrees. "Ave Verum Corpus", on the other hand is not scanned perfectly, as seen in Figure 5.4(c). Here we may see where the staffs lie, but not detect each single staff line, and the peaks are much shorter than in Figure 5.4(a). The rotation angle of this score is found to be 1.266.*

After the rotation, the lines are longer and more separated than in the unrotated images, see

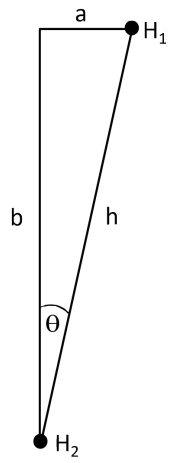
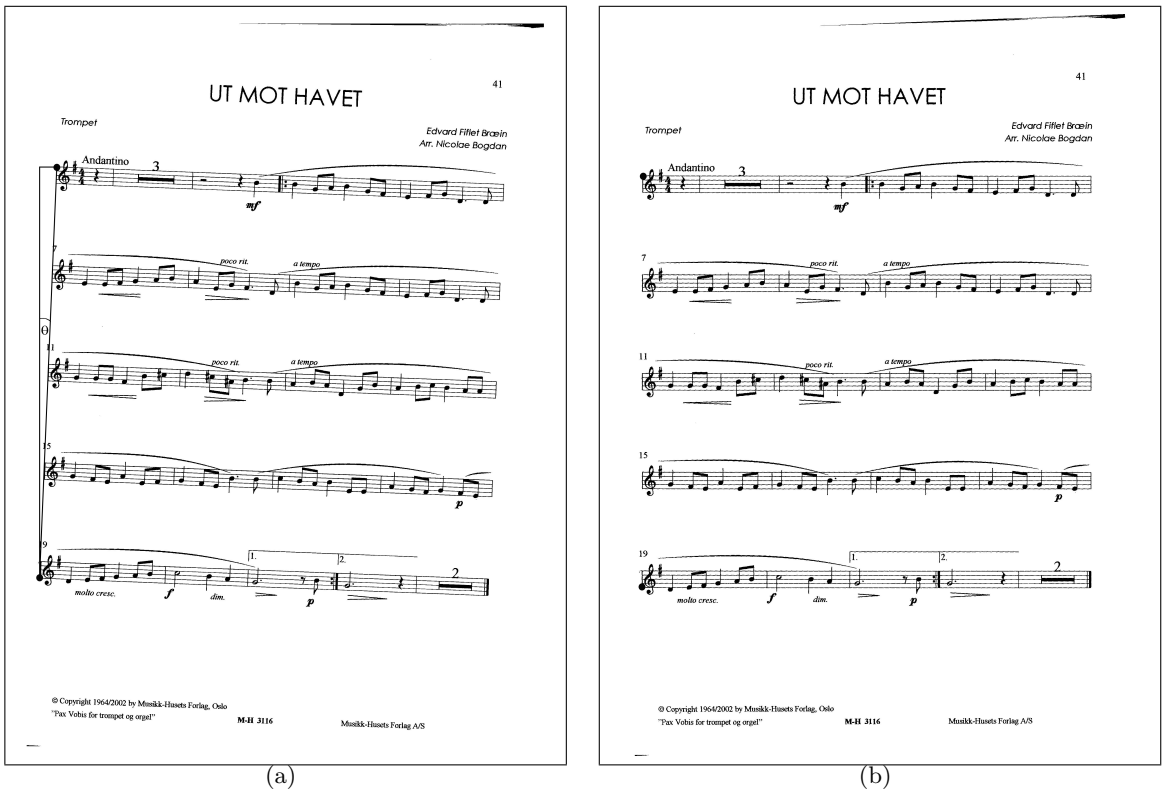


FIGURE 5.2: Illustration of the rule of Pythagoras. The angle θ is the rotation angle we wish to find. If H_1 is far more to the right than H_2, rotation is required.



(a) (b)

FIGURE 5.3: (a): *Ut mot havet* before rotation. The two circles indicate H_1 and H_2 respectively; and θ is the rotation angle, which in this example is 2.188 degrees. The marked lines are used to calculate the rotation angle, see Figure 5.2. (b): *Ut mot havet* after rotation. The two circles indicate H_1 and H_2.

Staff Line Localization

(1) Make a small image selection, H', of the horizontal projection, see Figure 5.5(b).

(2) Locate all the foreground pixels in H'.

(3) Create an array containing the y-coordinates of the pixels found in (2).

(4) Attach labels to the y-coordinates from (3), so that each row in the image belonging to the same staff line has the same label. In addition, *staff-labels* are attached to the y-coordinates, giving the first five staff lines the same label, the following five staff lines another label and so forth.

(5) Find and store the following measures for each score:

- average staff line thickness,
- average distance between the staff lines, defined as *staff spacing*,
- average distance between the staffs,
- number of staffs on the music sheet, found by: $\frac{\text{total number of staff lines}}{5}$.

(6) Open the image, I, using a structuring element; a 100 pixels long horizontal line.

(7) Create an array containing the y-coordinates of the pixels found in (6).

(8) Create a new array containing only the y-coordinates present in both the array from (3) and the array from (7).

(9) Recalculate the labels in (4) and the measures in (5).

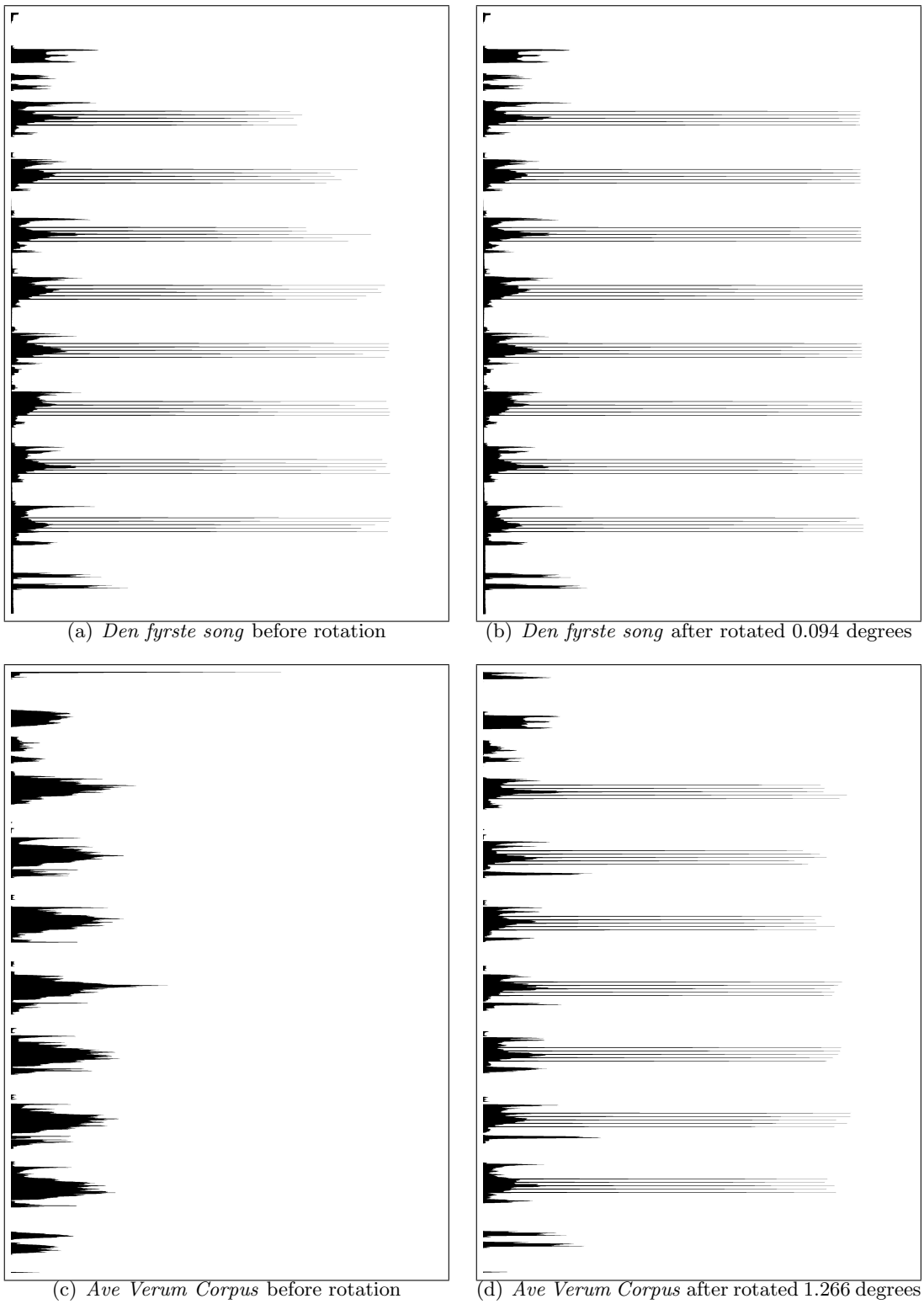
ALGORITHM 5.2: Staff Line Localization

Figure 5.4(b) and 5.4(d). This means that the rotation was successful. □

As we saw in Example 5.2, the aim is to rotate the image in order to see perfectly distinct peaks in the horizontal projection of the rotated image.

After having binarized and rotated the image, we want to localize the staff lines. This is one of the fundamental stages in the OMR process. The staff line localization method used, is outlined in Algorithm 5.2, and exemplified in Example 5.3.

EXAMPLE 5.3 *We cut the image of the horizontal projection of "Den fyrste song" at* 1/3 *of the image width, and make a small selection of the horizontal projection. This narrow image will only show clear lines where the staff lines are located, see Figure 5.5(b).* □



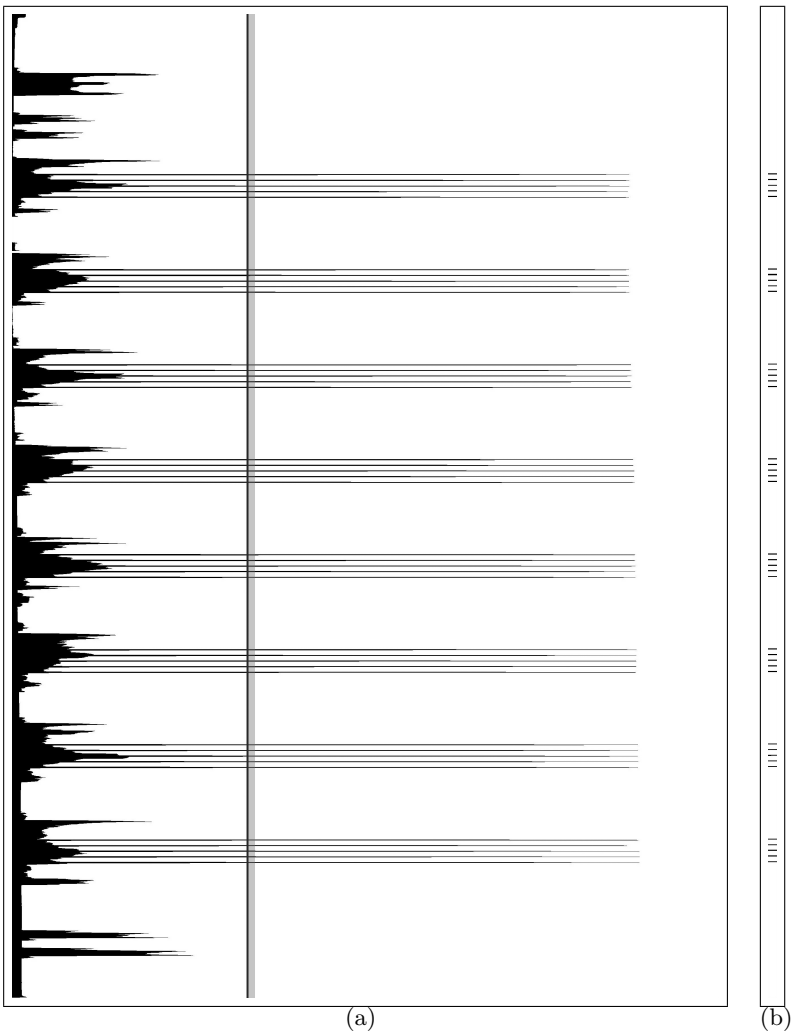
(a) *Den fyrste song* before rotation

(b) *Den fyrste song* after rotated 0.094 degrees

(c) *Ave Verum Corpus* before rotation

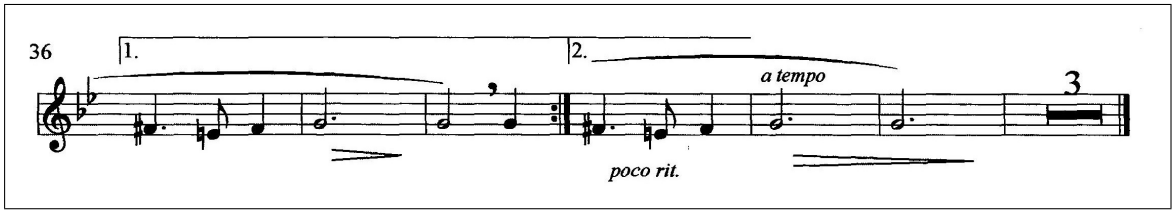
(d) *Ave Verum Corpus* after rotated 1.266 degrees

FIGURE 5.4: Horizontal projections of two different music scores, before and after rotation.

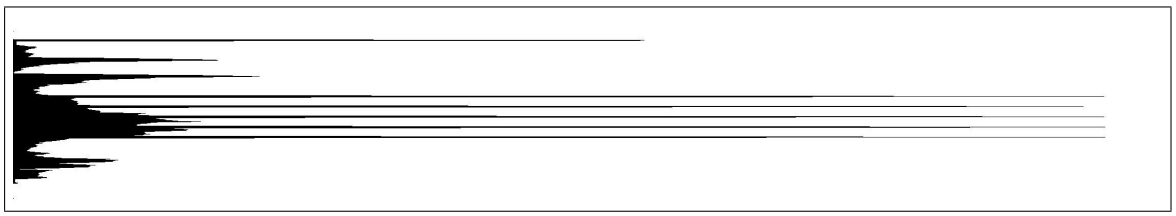


(a) (b)

FIGURE 5.5: (a): The horizontal projection of *Den fyrste song*, together with the cutoff line at 1/3. (b): The image selection, H' of the horizontal projection around 1/3 of the image width. The gray selection in (a) indicates (b).



(a)



(b)

FIGURE 5.6: (a): The sixth staff of *Greensleeves*, with a long line above the staff which may cause problems, as we can see in (b).

EXAMPLE 5.4 *We study the horizontal projection of the sixth staff of "Greensleeves" (Figure 5.6(b)), and observe that the long line ("volta bracket") above the staff may cause trouble if we choose the same procedure as in Example 5.3. However, if we instead cut at* 2/3 *of the image width, or look for the five longest lines within a certain interval, we will be able to locate the staff lines.* □

In addition to finding the y-coordinates of the lines using horizontal projection, Algorithm 5.2 performs a morphological opening of the image using a line shaped structuring element. The y-coordinates, which are common for both methods are the only ones used. This gives a better and more robust result, compared to using only horizontal projections. The y-coordinates found in Algorithm 5.2 (3) may be a little imprecise, and depend on where we cut the image. The lines in the image selection may be somewhat thinner than in the image, I. Another fact, which we will see in the next section, is that some lines are not complete, meaning that even though the image has been rotated, some lines may be uneven. See Figure 5.7 for an illustration of this. However, in some cases this is caused by rotation; if the rotation angle is rather big, rotating using Nearest-Neighbor Interpolation will give a jagged appearance.

0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1

FIGURE 5.7: An illustration showing how a staff line might be a uneven.

5.1.4 Staff Line Removal

We have located the staff lines, and it is now prudent to remove the staff lines before locating the musical symbols. When removing the lines, it is very important to do this cautiously, so that no musical objects get damaged in the process.

We will here explain the three different *staff line removal methods* we have tried out, followed by a discussion and conclusion on the choice of method.

5.1.4.1 Method A

We create a new array of labels, *lab*,

$$lab(y) = \begin{cases} 1 & \text{if } y \text{ is the first pixel row in a line,} \\ 0 & \text{if } y \text{ is a center pixel row in a line,} \\ 2 & \text{if } y \text{ is the last pixel row in a line.} \end{cases}$$

We traverse along each line, and the rows where $lab = 1$ have to be checked upward, to see if there are objects connected with the line on the upper side of the line, while the rows where $lab = 2$ have to be checked downward. The center rows can be deleted if the rows above and under *agree*. There also might be bigger objects, which interconnect with several lines, and where special care has to be taken.

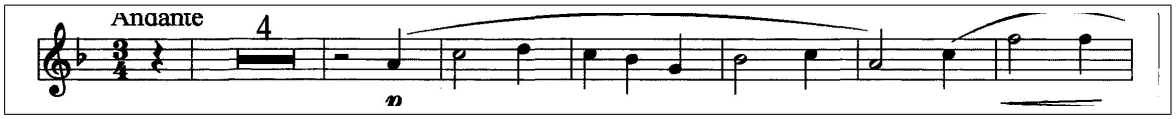
Some of the lines remain even though no symbol is connected to them. The reason for this is that some of the lines are not complete, i.e. the lines may be a bit uneven if you follow the line from left to right, despite the rotation. It is therefore not enough to check the pixel just above and below the line, but e.g. two pixels above and below. This leads to a far better result. See Figure 5.8(b).

5.1.4.2 Method B: *Width-based removal*

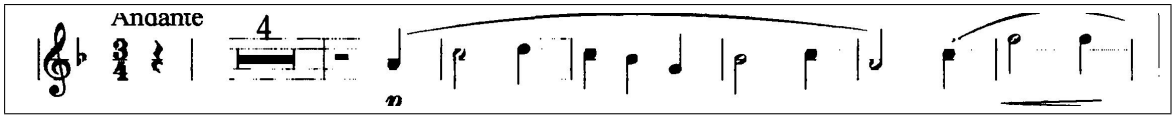
An alternative method involves looking at the average thickness of the lines, and go through each single line from left to right, checking if the line is thicker than a certain threshold (typically the average staff line thickness, found in Algorithm 5.2, or a bit smaller). The line, at position x, can then be deleted safely if the width of the line is smaller than this threshold. If the line is much thicker it indicates that there is an object going through the line, from above or below, or even an object attached to the line on both sides. The result of this method is shown in Figure 5.8(c).

5.1.4.3 Method C: *Combination of the two above mentioned methods*

Studying the image results of the two previous methods (see Figure 5.8, and B.1-B.4), *Method B* seems to work better than *Method A*, but fragments of some of the staff lines remain, while these fragments are being removed by *Method A*. By looking at the common features between the images, we may obtain a better result, but some of the symbols may also be lost. See Figure 5.8(d). Since we do not want to loose any symbols, we conclude that *Method B* gives the most satisfactory result, even though fragments of some staff lines remain, but this can be dealt with later.



(a)



(b) Method A

(c) Method B

(d) Method C

FIGURE 5.8: (a): The first staff of *Den fyrste song*; before removing the lines, and (b)-(d): after removing the lines using the three different methods.

EXAMPLE 5.5 *In this example we look at all the three methods, A-C, applied on the same staff, see the original image in Figure 5.8(a). In Figure 5.8(b) the lines have been removed using Method A; checking if there are pixels above or below the line belonging to an object, while in Figure 5.8(c), the lines have been removed by thresholding with respect to the average staff line thickness. Figure 5.8(d), on the other hand, is a combination of the other two figures.*

We can see that by using Method A we get an image containing large amounts of staff line fragments. Method B is able to remove these line segments, but may add some others, see for instance on the stem of one of the quarter notes. This is indeed a much smaller problem than

the lines remaining after using Method A. The combination of these methods does not improve the results much, in fact some symbols may even be lost or damaged, like the half notes in this example. □

5.2 Segmentation

In the previous section, the staff lines were successfully removed from the images, and the foreground pixels which are left should belong to musical symbols, but there might be additional noise, text or other symbols we would like to ignore. Now we want to locate the symbols of interest, that is the symbols essential for playing the melody.

Many OMR programs detect musical primitives, as described in Section 3.3.1. We choose to follow Rossant and Bloch's approach for detecting objects. We divide the objects into two main groups; *vertical objects* [1]and the remaining symbols. This grouping is done in order to perform analysis on these two types of symbols separately in the following *classification* stage.

The symbols in the first group are mainly: notes heads with stem (all notes except whole notes), accidentals, bar lines, crotchet and measure rests, and dynamic markings such as *forte*, *piano* etc. The symbols in the other group are whole notes, the remaining pauses, and some symbols, which are of no interest to us, and hence will be overlooked.

To be able to find out which objects belong to which group, we perform a morphological opening with a suitable structuring element, see Algorithm 5.3 for details.

An object is here defined as a group of connected foreground pixels. That means that if a note and an accidental are situated too close, they might appear as one object. The same is valid for a group of four beamed notes. This will be dealt with in the *recognition* stage, where these symbols will be separated.

Finding each group of connected foreground pixels is done using the `bwlabel`-function in *MatLab*. This method returns a matrix of same size as I, where the elements are integer values ≥ 0;

$$L(x,y) = \begin{cases} 0 & \text{if } I(x,y) \text{ belongs to the background,} \\ 1 & \text{if } I(x,y) \text{ belongs to the first foreground object,} \\ \vdots & \\ n & \text{if } I(x,y) \text{ belongs to the } nth \text{ foreground object,} \end{cases}$$

[1]Musical symbols featured by a vertical segment longer than 1.5 staff spacing.

Locating *Vertical Objects*

(1) Define a line shaped structuring element, s, of size 1.5 staff spacing (found in Algorithm 5.2) rotated 90 degrees.

(2) Perform a morphological opening:

$$I \circ s = (I \ominus s) \oplus s.$$

(3) The result of this opening gives us the objects of interest, i.e. the *vertical objects*.

ALGORITHM 5.3: Segmentation

where n is the total number of foreground objects. The method may use 4- or 8-connectivity. Here the default number of connected objects is used, which is 8.

5.3 Classification by Template Matching

Rossant and Bloch use template matching to recognize the different musical symbols in their projects. They divide this stage into two main parts;

- analysis of *vertical objects*, and
- analysis of the remaining symbols.

We will also divide the template matching task in the same way, but vary the approach, and combine the use of template matching and morphological operations.

5.3.1 Musical Symbol Recognition

This section is divided into the recognition of *vertical objects* [2], and the recognition of remaining objects.

5.3.1.1 Making templates

The templates are made from some of the images in our dataset. Since the music scores are more or less of the same form, we have only one set of templates. However, on the same music sheet, differences in size and shape of musical objects belonging to the same class might occur. The

												
3x83	25x22	26x22	17x57	15x61	18x64	22x67	101x43	54x148	46x51	41x27	39x39
Bar line	Filled note	Half note	Flat	Natural	Sharp	Crotchet	Measure	G clef	Forte	Mezzo	Piano
Lines	Notes		Accidentals			Rests		Clefs	Dynamics		

(a)

									
23x39	34x18	28x12	34x23	21x59	20x59	19x15			
Eighth	Half	Whole	Whole	Eighth	Sixteenth	Duration			
Rests			Notes	Hooks		dots			

(b) (c)

TABLE 5.1: The various groups of templates; (a): The template used in the *vertical object* recognition; (b): The templates used in the remaining symbol recognition; (c): The templates used in the note value recognition.

templates are divided into different groups; the *vertical objects* in Table 5.1(a), the remaining objects in Table 5.1(b) and templates regarding the note values in Table 5.1(c). In these tables the names, symbols and dimensions of the templates (in pixels) are listed.

5.3.1.2 *Vertical Objects* Recognition

The first task is to analyze the *vertical objects*. This excludes whole notes and most of the pauses (except for crotchet rests and measure rests), which we will look into later, in Section 5.3.1.3.

We use template matching with fourteen templates to recognize these symbols. See Table 5.1(a) for an overview of these.

Template matching is performed by computing the correlation between a segmented object and each template. This process is described in Algorithm 5.4. The threshold value used in Step (5) is found experimentally. This value cannot be too high, since there always are some shape variations even within the same music sheet.

The resulting correlation matrix between a segmented object and a template may contain several clearly distinct areas with high values. High values indicate a possible good match, and more than one area with high values indicate that there might be multiple musical symbols within the segmented object. To limit the search it is possible to look at objects classified as note heads and look at the width of these objects. If they are considerably wider than the note

[2]Musical symbols featured by a vertical segment longer than 1.5 staff spacing.

Template Matching

We define the template, h, with dimensions: $m \times n$, where m is the width and n the height, and the image of the segmented object, f.

(1) Compare each coefficient in the template, h, with the image pixel that lies directly underneath it.

(2) Move the template around in the image.

(3) Compute the correlation

$$g(x,y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j,k) f(x+j, y+k),$$

where $m_2 = m/2$ and $n_2 = n/2$ (Efford, 2000).

(4) Divide the correlation matrix by the dimensions of h

$$g'(x,y) = \frac{g(x,y)}{m \times n},$$

obtaining values between 0 and 1.

(5) Threshold the matrix using a threshold value 0.85: $g''(x,y) = g'(x,y) \geq 0.85$.

ALGORITHM 5.4: Template Matching

head template, it is worth to check if the object contains several note heads. It is also possible that more than one template obtain a high correlation with the segmented object. This may be caused by ambiguities between similar symbols, or by two touching symbols of different kind. It is for instance common that accidentals are situated close, or even attached to notes.

When we search for more objects, we may concentrate on searching for filled note heads, half note heads and accidentals. However, if a symbol is classified as a dynamic symbol, we must search for other dynamics within the same segmented object. These dynamic expressions are often composed by two single dynamic letters, since we have chosen to define the templates like that.

Combining the Use of Morphology and Template Matching

In addition to template matching, we have performed morphological openings of the image. We have used two different structuring elements; s_1 and s_2. See Definition 5.1 for information regarding these elements.

Structuring Elements Used in the Classification Process

Three different structuring elements, s_1, s_2 and s_3:

- s_1: a vertical line, 3 staff spacings high.
- s_2: a disk of size 0.5 staff spacing.
- s_3: rectangle shaped structuring element, 40 pixels long and 5 pixels high.

DEFINITION 5.1: Structuring Elements

By opening the image using s_1, we will obtain an image containing the largest vertical lines, mostly bar lines and note stems, but also some lines that are part of accidentals etc. By performing a check on which objects these lines are parts of, the lines that most probably belong to for instance bar lines, G clefs etc. are removed from this image. The goal is to get an image containing only the note stems.

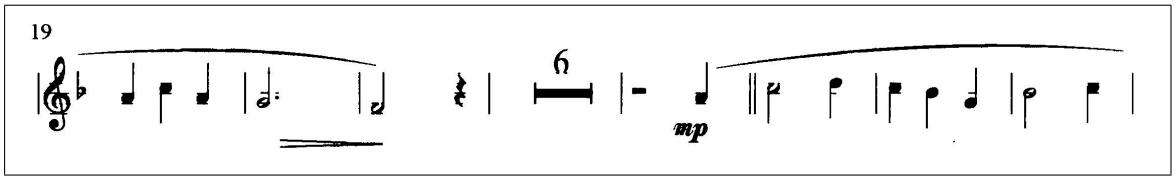
Opening the image using the other structuring element, s_2, all the filled note heads will be found. The note heads of half notes will not be found here, but by combining the results of these two morphological openings, one may be able to see which lines that are note stems belonging to the filled note heads we found, and which lines are not. The lines which does not belong to the filled note heads are most probably the note stems of half notes, but may also be bar lines that interconnect with other symbols etc. See Example 5.6 for illustrations of these results.

EXAMPLE 5.6 *In this example, we look at the image results of the morphological openings on a music sheet to get a clearer view of what these operations do. Figure 5.9(e) is opened using s_1. This image contains the longest vertical lines. We check which objects these lines belong to, trying to eliminate the lines that are part of other objects than notes. We then obtain the image in Figure 5.9(f).*

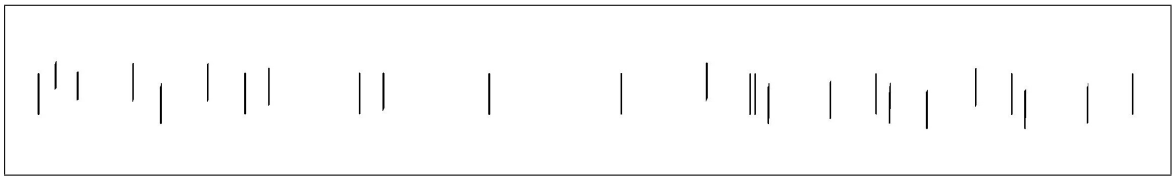
Figure 5.9(g) is the combination of Figure 5.9(f) and the morphological opening using s_2. By opening the image using s_2, all the filled note heads will be found, but the unfilled note heads will not be found here. However, by combining the results of these two morphological openings, we may be able to see which lines that are note stems belonging to the filled note heads, and which lines are note heads belonging to half notes.

See Figure 5.9(g) for the filled notes and Figure 5.9(h) for the lines most probably belonging to half notes. □

By combining the results of these morphological operations with the results of the template

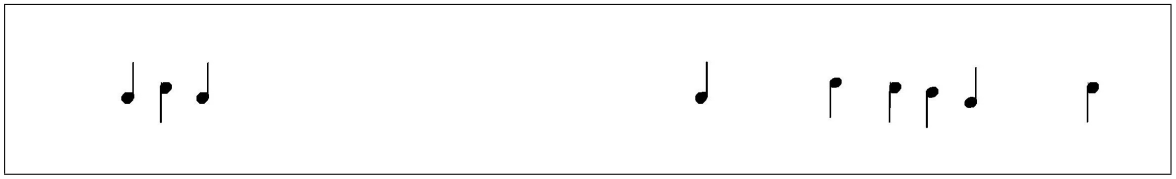


(d)

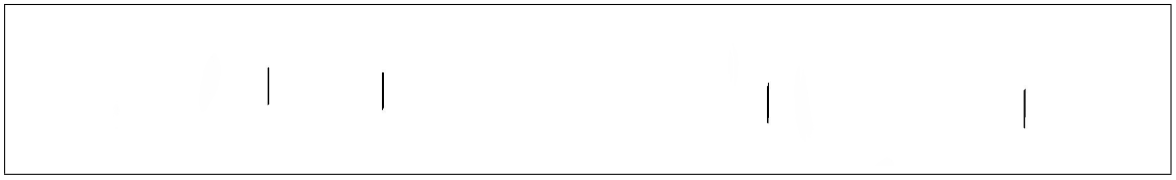


(e)

(f)



(g)



(h)

FIGURE 5.9: (d): The third staff of *Den fyrste song* after staff line removal. Results of the morphological openings of (d); (e): Opened using s_1. (f): Some lines from (e) have been removed by looking at which objects the lines belong to, their temporary label and the size of their bounding box. (g): Opened using s_2, and then combined with (f). (h): Contains the lines from (f) which do not appear in (g), and do not belong to bar lines.

matching, the recognition rate increases, especially for the note heads. This eases the localization of numerous note heads within one single segmented object, which is the case when we have beamed notes. This can also avoid that unknown objects (objects not assigned a template yet), like time signatures get classified as filled note heads. If a symbol is classified as a filled note head, but is not present in the morphological opened image (Figure 5.9(g) and 5.10(d)), then it cannot be a filled note. Further on, we can make the same restrictions when it comes to half notes; a half note has to be present in the image Figure 5.9(h).

EXAMPLE 5.7 *This example is similar to Example 5.6, only that this staff contains several beamed notes. Here we see from Figure 5.10(d) that we may have some problems finding the correct position of the note heads, because the structuring element, s_2 also fits inside some of the beams.*

One way to deal with this is to perform another morphological opening, using the rectangle shaped structuring element, s_3 (see Definition 5.1). Since some beams are horizontal, others tilted upward and others tilted downward, s_3 is rotated -15 *degrees and* 15 *degrees. We then perform three morphological openings, using the original structuring element, and using this structuring element in two rotated versions; rotated* -15 *and* 15 *degrees respectively. Finally we combine these three images.*

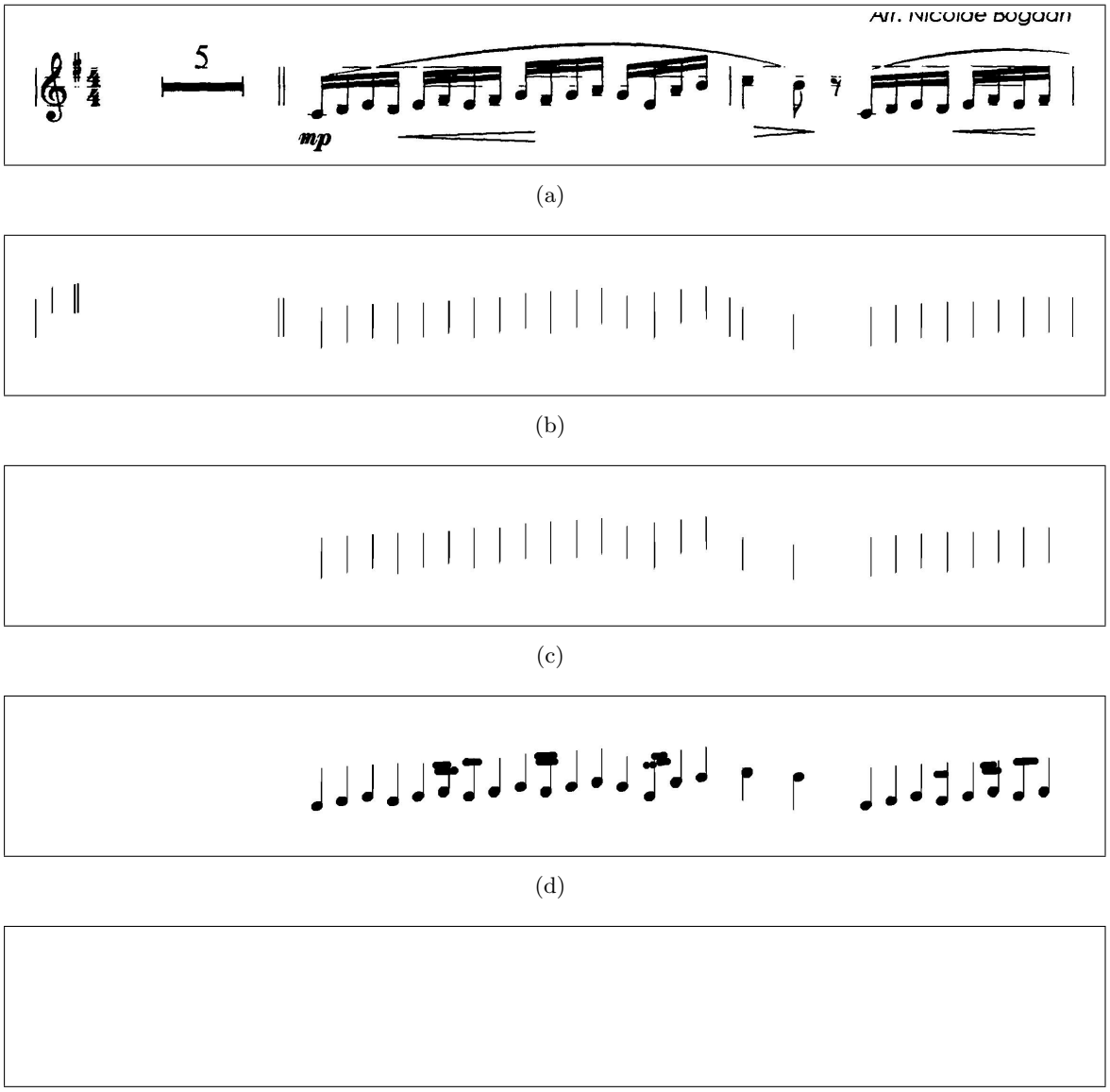
As we can see from Figure 5.11(c), by using this morphological opening, it is possible to reduce the ambiguities in Figure 5.11(b), and get a clearer image with less uncertainties, as seen in Figure 5.11(d). □

Discussions Around the Number of Templates

In our experiments we used fourteen templates together with morphological operations. However, this was not the initial idea. We started using only template matching, and fewer than fourteen templates, but landed on a better solution in the end.

During the first experiment we used nine templates; only filled note heads, half note heads, accidentals, quarter rests and G clefs.

Bar lines were then initially found using a restriction on the size of the objects' bounding box; objects having a very narrow shape, spanning five staff lines were classified as bar lines. This, however, was not very robust, since bar lines often interconnect with other symbols or slurs, and they would not be recognized using this method. A way to solve this was to include a bar line template, and not only consider the bounding box feature. The use of morphology together with template matching made the bar line classification even more robust. Morphology also improved the classification rate of note heads considerably, especially for beamed note heads.



(a)

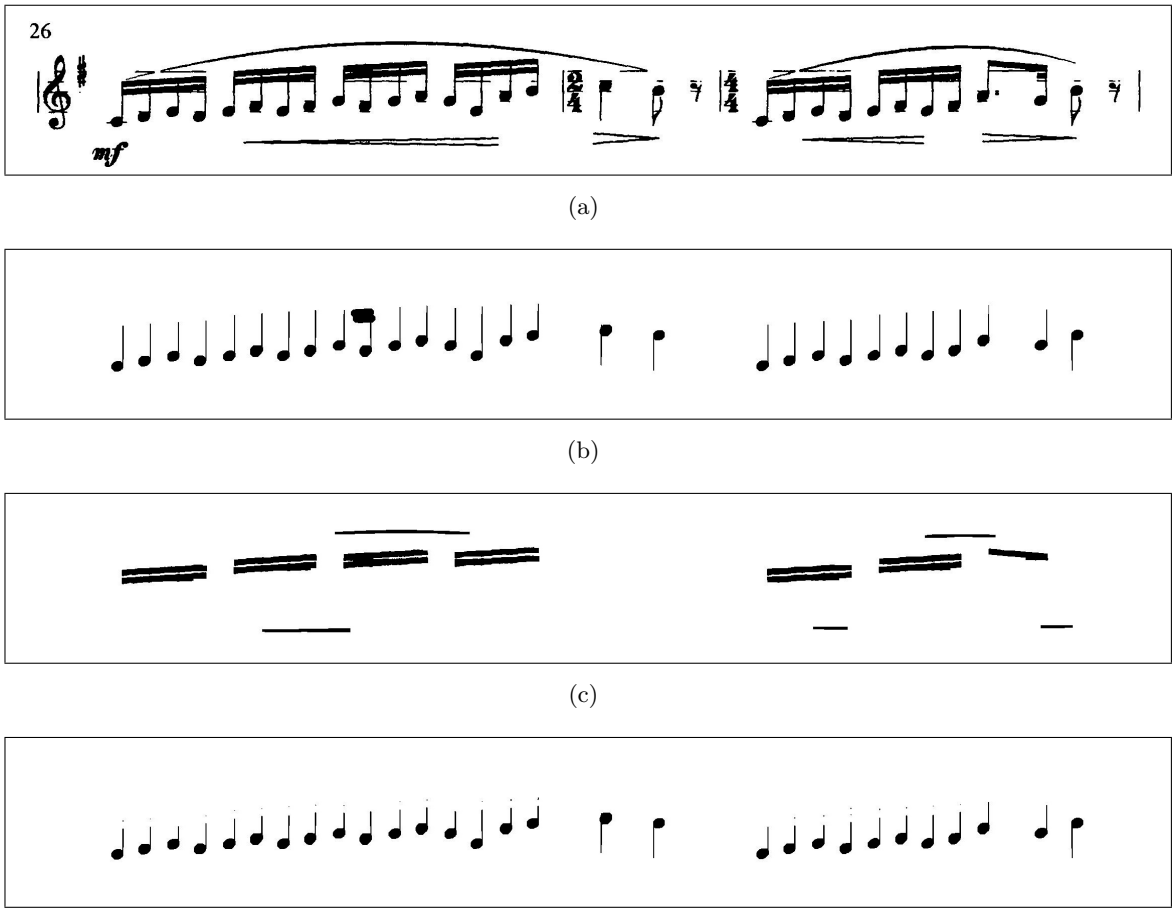
(b)

(c)

(d)

(e)

FIGURE 5.10: (a): The first staff of *Time to Say Goodbye* after staff line removal. Results of the morphological openings of (a); (b): Opened using s_1. (c): Some lines from the (b) have been removed by looking at which objects the lines belong to, their temporary label and the size of their bounding box. (d): Opened using s_2, and then combined with (c). (e): Contains the lines from (c) which do not appear in (d), and do not belong to bar lines.



(a)

(b)

(c)

(d)

FIGURE 5.11: (a): The sixth staff of *Time to Say Goodbye* after staff line removal. Results of the morphological openings of (a); (b): corresponds to 5.10(d). (c): A combination of three openings; using a 40 pixel long, and 5 pixel high rectangle-shaped structuring element tilted 15 degrees, -15 degrees and 0 degrees. (d): (b) minus (c), which makes it more difficult or impossible to find note heads in a beam.

Another problem using fewer templates, was that there were also other types of *vertical objects*, not mentioned by Rossant and Bloch in their articles. These symbols could obtain a high correlation score with e.g. a filled note head template. This could be large symbols like *measure rests*, or also the dynamics such as *piano*, *mezzo* and *forte*.

We experimented further, including some of these misclassified symbols as templates. The additional templates were dynamic symbols; *forte*, *mezzo* e *piano*, and the *measure rest*. The *measure rest* symbols may vary in width, so a cropped version of it was used temporarily, for testing. This way, also the shortest measure rests were recognized.

5.3.1.3 Remaining Symbol Recognition

We started classifying only the *vertical objects*, while the remaining symbols, see Table 5.1(b) were put aside for the moment. In this section we will look at these symbols, mainly concerning whole notes and some additional rests. There are several other pauses, but in our current dataset, the rests in Table 5.1(b) are the only one present, so we will concentrate on these. Half rests and whole rests look identical, the only thing that distinguishes them is their location. A half rest is always situated on top of the third staff line, while a whole rest is situated below the fourth staff line.

When searching for these four symbols, we can limit our search down to the spaces not already occupied by symbols found in the previous section, and for the rests it is enough to search in an area around the third staff line.

5.3.2 Pitch Recognition

It is essential to find the pitch for note heads and accidentals. The pitch is determined by the vertical position of note heads relative to the staff lines, together with the information from the key signature and accidentals.

The note height relative to the staff lines, is found by looking at the maximum correlation point of the note head. The pitch is found by observing if the maximum correlation point is closest to one of the lines or situated between two lines, or even above or below the staff.

It is also necessary to find the height of the accidentals, but the simple and straightforward method described above might not work as good for these symbols. Looking at the maximum correlation coordinates for these symbols, will most probably turn out wrong, at least for flats. Sharps and naturals are centered around the pitch they represent, but a flat is placed higher up,

so it is necessary to look at the lower part of the flat to find the right pitch. That is, the pitch is determined by a point, which is not center of the symbol.

A simpler way to solve this for the accidentals, is to introduce musical rules; the accidentals in the key signature, situated right after the G clef follow a strict *pattern*, see Table 2.4. For now we concentrate on finding the height of note heads, and combine this information with the key signature and improved accidental heights found in Section 5.4.1.

5.3.3 Note Value Recognition

To find the length of the notes, we look for duration dots, hooks or beams. Hooks and beams are found at the opposite end of the note stem, while duration dots need to be searched for in a small area behind the note head. Duration dots can appear after all kinds of note heads, both filled and not filled, single and beamed.

For grouped notes, we need to determine the number of beams. This might be done by looking at the horizontal projection of the note. On the other hand, for single filled notes, we use template matching to find possible hooks, with the templates from Table 5.1(c).

5.3.3.1 Duration Dots

A duration dot is a small dot placed right after a note head, at approximately the same height. If a note is situated on a line, the dot is placed on the space above the line. The dot lengthens the note's duration. One dot lengthens the note by half its value, two dots by three-quarters, and so on. Rests can be dotted in the same manner as notes (Wikipedia, May 2009).

Finding these dots is therefore essential for finding the right duration of the notes. A template is used to locate these dots, see Table 5.1(c). Some background is included around the small dot in the template, so we will find black dots on a white background.

5.3.3.2 Beamed Notes

Having a group of notes, we can cut out a slice of the image around each note head, see Figure 5.12 and 5.13(b). If the stem is going up, the image may be cut right over the note head, so the image only contains the note stem and a part of the beam(s), and cut right under the note head if the stem is going down. This way, the horizontal projection of the image will not produce a peak where the note head is, but only look at the stem and beam(s).

Finding the Rotation Angle of an Image Selection of a Beamed Note

We perform erosion on an image selection, I', of a beamed note, see Figure 5.12.

(1) Rotate a rectangle-shaped structuring element from -15 degrees to 15 degrees.

(2) Erode the image using the structuring element rotated -15 degrees.

(3) Store the number of *hits* for this degree.

(4) Repeat 2) and 3) for all of the degrees.

(5) Find the angle with most hits.

The angle found in 5) is the rotation angle of I', that is the degree which makes the beams lie horizontally.

ALGORITHM 5.5: Rotation of Beamed Notes

As illustrated in Figure 5.12 and 5.13(b), the duration of the notes are determined by the number of beams around the note stem. For the first and the last note, there are only beams on one side of the note stem, while a center note has beams on both sides of its note stem. The duration of a center note is therefore determined by the side with the maximum number of beams.

EXAMPLE 5.8 *In this example we look at some different kinds of beamed notes. As we can see, the beams in Figure 5.12(a) are as good as horizontal. The horizontal projections are therefore easy to interpret, see Figure 5.13(c)-5.13(d). However, in most cases the beams are rotated either upward or downward (see Figure 5.12(b)-5.12(c) and 5.13(b)), and this makes it much more difficult to analyze the horizontal projections (Figure 5.13(e)-(g)).*

To solve this the image selections have been rotated, using Algorithm 5.5. The horizontal projections of these rotated image selections show clear and distinct beams, see Figure 5.13(j)-(l). □

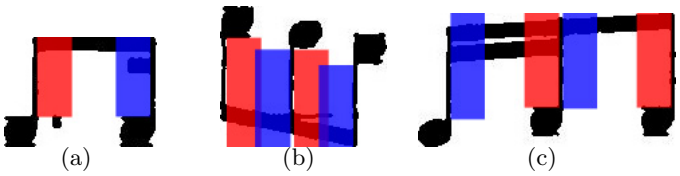


FIGURE 5.12: The red rectangles illustrate slices cut out on the left side of a note stem, and the blue rectangles slices on the right side of a note stem.

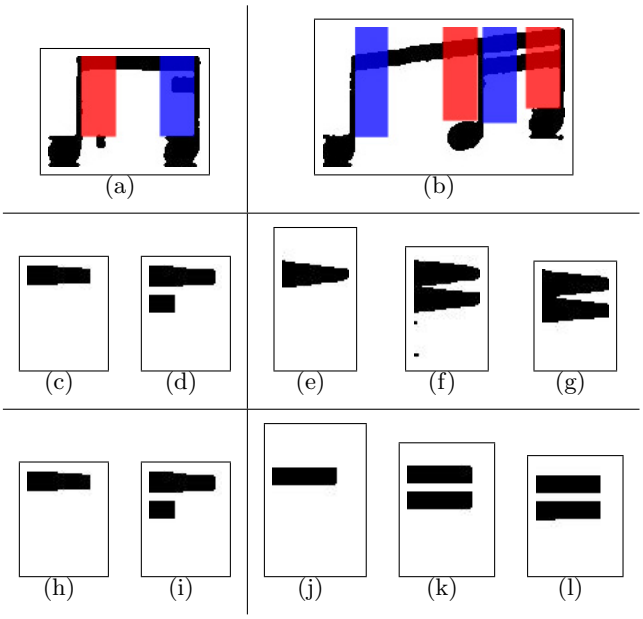


FIGURE 5.13: Two examples of different beamed notes, (a) and (b). (c)-(g) show the horizontal projection of each of the colored rectangles from the images above before rotation. (h)-(l) show the horizontal projection of each of the colored rectangles from the images above after rotation.

5.4 Post-processing Applying Musical Semantics

In the previous sections, we treated each musical symbol individually. In this section we will look at musical symbols in relation to each other, by introducing musical rules. This way we will probably correct some of the classification errors. In this thesis we will only test this on pitch recognition for accidentals.

5.4.1 Accidentals

In the Section 5.3.2 we saw that determining the height of accidentals may not be as easy as determining the height of note heads. This is especially a problem for flats, because the value of these are not determined by their middle y-coordinate, as it is for note heads.

By using musical rules, the classification rate will increase concerning the accidentals' height, but it may also correct some errors when it comes to classification of accidental type. We will, however, only look into the first case.

The accidentals in the key signature follow a strict pattern. Knowing that only one sharp is situated here, we also know that this sharp is a *F sharp*. The same goes for flats; if it is only one flat in the key signature, this has to be a *B flat*. When adding more sharps or flats in the key signature, this has to be done in a certain order, and there are either flats or sharps in the key signature, never a combination. See Table 2.4, and Figure 2.3 for an overview of the order. This can also be used to check if the classification is right; to control that a flat and a sharp do not appear in the same key signature. A natural can for instance not be situated in a staff without a flat or sharp at the same height in the key signature or within the same time measure as the natural.

For the accidentals, which are not placed in the key signature, these are situated before the note head it modifies, and has therefore the same height as the succeeding note head. This, however, depends on correctly classified note pitches.

6 Classification Results

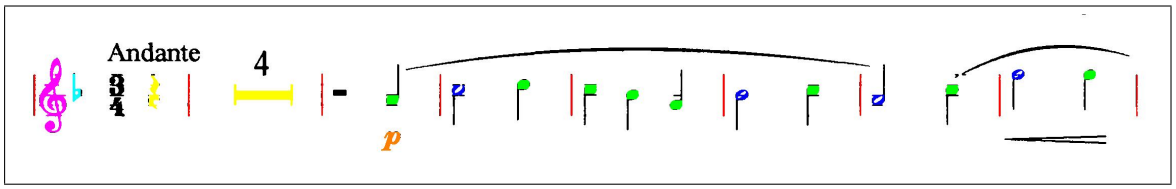
We have now gone through the algorithms in detail, and in this chapter we will study the main results of the classification thoroughly. As in Chapter 5, we look at the different parts of the recognition one by one; recognition of musical symbols, recognition of pitches and recognition of note values. In the first section, we consider all symbols, first the *vertical objects*, then the rest. When looking at the *vertical objects*, we also study the classification of accidentals separately. In the following section we review the results of the pitch recognition, for notes and accidentals, while in Section 6.3 we concentrate on the duration of notes. Duration dots, however, need to be searched for also in relation to rests, but this part is not prioritized here, so the results of the duration recognition regard only the duration of notes. In Section 6.4 we will examine the improved results regarding pitch recognition for accidentals when introducing musical rules. Finally, in Section 6.5, we study the performance of each of these classification *categories* on each single score, to see if there are huge differences in the results between the sheets.

The dataset used in our experiments consists of 33 monophonic note sheets; divided into a training set (14 sheets) and a test set with the remaining 19 sheets. See Table A.1 for an overview of the titles and composers of these scores, together with the division of the dataset, and Figure A.1 for four examples from this dataset. All 33 notes are available in Bogdan (2003), and all of the notes are trumpet notes. Some parts of the program have also been tested on other scores, and will be commented in Chapter 7. However, all the results in this chapter concern only the original 33 sheets. Further on, we contacted Rossant and Bloch with the aim of obtaining a common dataset. Florence Rossant responded us in April 2009, saying that it might be difficult to send a list of the music pages they used, but that she would try to provide a database of images. Unfortunately, we did not receive anything in time. However, in all the articles we have read, no one uses the same dataset, but many compare their own system to one or more commercially available systems.

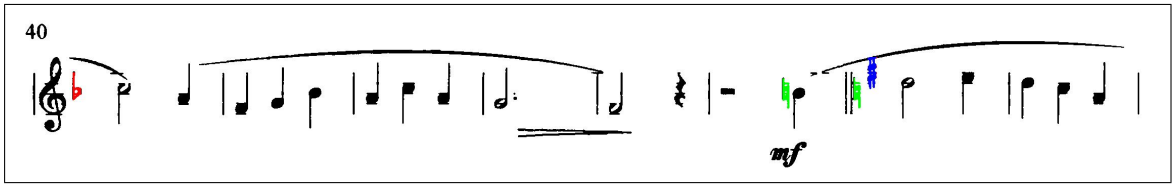
In all of these sections, color images have been made, in addition to confusion matrices, to better be able to control the results of the methods. Symbol classes, pitches and note values are illustrated in separate images, and for each of these illustrations different colors are assigned to the different sub-types. See Table 6.1 for an overview of the colors used in each section.

Category	Section	Type		Color
Musical Symbol	6.1.1 Vertical Objects	Bar lines	1	red
		Filled notes	2	green
		Half notes	3	blue
		Accidentals	4	cyan
		G clefs	5	magenta
		Crotchet rests	6	yellow
		Measure rests	7	yellow
		Dynamics	8	orange
	6.1.1.1 Accidentals	Flats	♭	red
		Naturals	♮	green
		Sharps	♯	blue
	6.1.2 Remaining Objects	Eight rests	9	red
		Half rests	10	green
		Whole rests	11	blue
		Whole notes	12	magenta
Pitch	6.2 Pitches		C	red
			D	green
			E	blue
			F	magenta
			G	yellow
			A	cyan
			B	orange
Note Value	6.3 Note Values	Dotted half notes	𝅗𝅥.	orange
		Half notes	𝅗𝅥	red
		Dotted quarter notes	♩.	yellow
		Quarter notes	♩	green
		Dotted eighth notes	♪.	cyan
		Eighth notes	♪	blue
		Dotted sixteenth notes	𝅘𝅥𝅯.	purple
		Sixteenth notes	𝅘𝅥𝅯	magenta
		Thirty-second notes	𝅘𝅥𝅰	gray
		Sixty-fourth notes	𝅘𝅥𝅱	brown

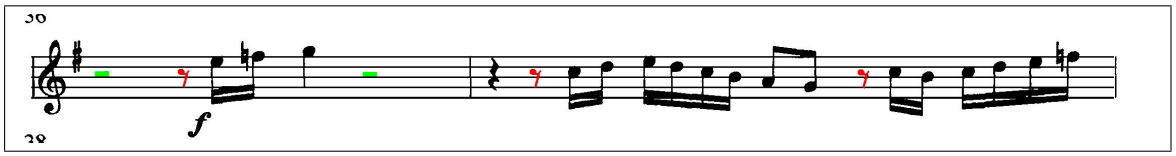
Table 6.1: The different recognition categories, together with the sub-types within each section and the colors used to emphasize them in the result images.



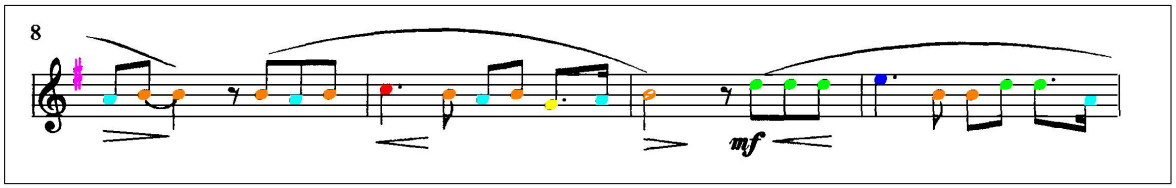
(a) Vertical Objects (Section 6.1.1)



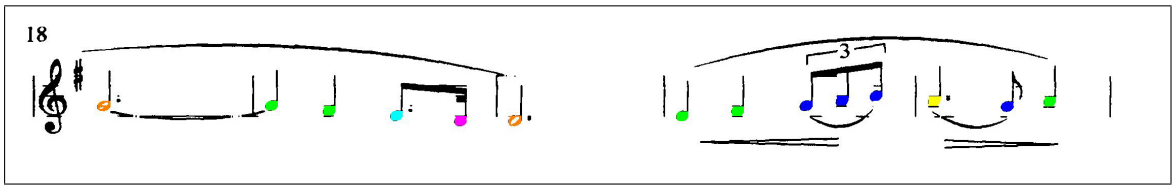
(b) Accidentals (Section 6.1.1.1)



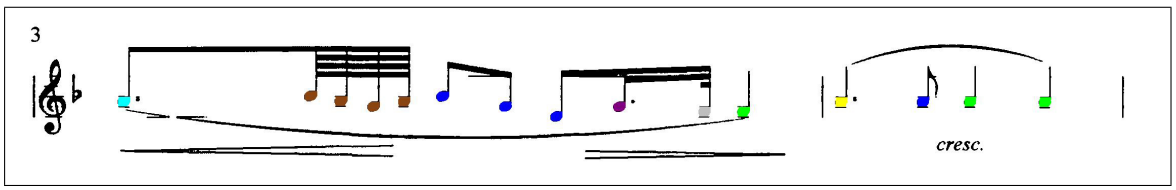
(c) Remaining Objects (Section 6.1.2)



(d) Pitch (Section 6.2)



(e) Note Value (Section 6.3)



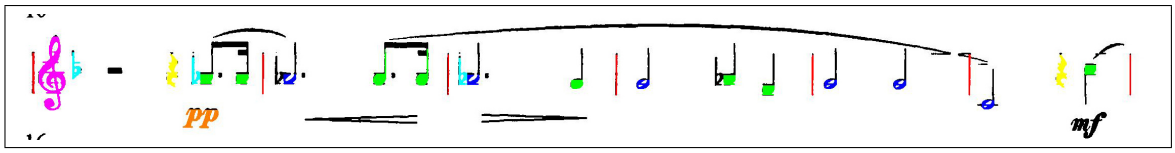
(f) Note Value (Section 6.3)

FIGURE 6.1: Images showing the results of the different recognition categories, illustrating how the colors in Table 6.1 are used in practice. The images where the height of the symbols is essential for analyzing the result, are plotted together with the staff lines.

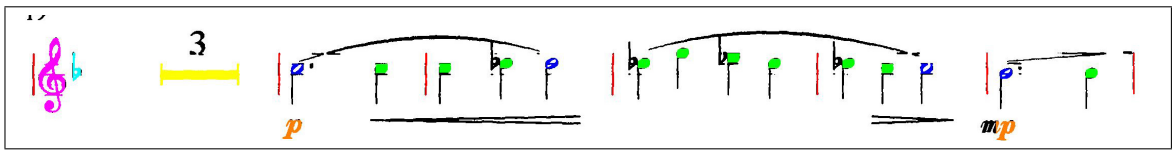
6.1 Musical Symbol Recognition

6.1.1 *Vertical Object* Recognition

The confusion matrices in Table 6.2 are found using morphological openings combined with template matching, see Section 5.3.1.2. See also Figure 6.1(a) for an example of a result image. First we study the performance on the training dataset in the upper part of Table 6.2. By analyzing the confusion matrix, we are able to see what goes wrong, and accordingly improve the method to decrease the error rate. The overall error rate on the training set is 0.059. As we can see, most of the errors are due to missing symbols, which in Table 6.2 correspond to the row *omitted*. These symbols are for some reason overlooked by our system. Examples of such cases might be accidentals attached to notes (Figure 6.2), dynamics which are cropped (Figure 6.3), or symbol variation.



(a)



(b)

FIGURE 6.2: Restults of the vertical object recognition showing that missing accidentals might occur.

We compare the classification rates on the training set and the test set, to see if the method overfits with respect to the training set. The error rate on the test set is 0.069, which is not particularly much higher than the error rate of the training set, and we may conclude that we have no overfitting.

The following discussion considers some of the symbol classes separately.

6.1.1.1 Recognition of Accidentals

On the result images with all symbols, all accidentals are given the same color, even though the labels the program assign to each of them are different. The same goes for the accidentals (*class 4*) in the confusion matrices (Table 6.2). Therefore it might seem that all accidentals are

Vertical Object Recognition

Train		True									
		1	2	3	4	5	6	7	8	Others	Sum
Predicted	1 (bar lines)	724	0	1	0	0	0	0	0	1	726
	2 (filled notes)	1	1394	3	0	0	0	0	0	1	1399
	3 (half notes)	1	0	378	0	0	0	0	0	7	386
	4 (accidentals)	0	0	0	181	0	0	0	0	2	183
	5 (g clefs)	0	0	0	0	114	0	0	0	0	114
	6 (crotchet rests)	0	0	0	0	0	36	0	0	0	36
	7 (measure rests)	0	0	0	0	0	0	21	0	0	21
	8 (dynamics)	0	0	0	0	0	0	0	118	0	118
	Omitted	9	2	16	18	0	1	2	29	0	77
	Sum	735	1396	398	199	114	37	23	147	23	3060
	Rate	0.985	0.999	0.950	0.910	1.000	0.973	0.913	0.803	–	0.941

Test		True									
		1	2	3	4	5	6	7	8	Others	Sum
Predicted	1 (bar lines)	820	0	1	0	0	0	0	0	0	821
	2 (filled notes)	0	2207	0	0	0	0	0	0	1	2208
	3 (half notes)	0	0	247	0	0	0	0	1	0	248
	4 (accidentals)	0	0	0	262	0	0	0	0	3	265
	5 (g clefs)	0	0	0	0	158	0	0	0	0	158
	6 (crotchet rests)	0	0	0	0	0	95	0	0	0	95
	7 (measure rests)	0	0	0	0	0	0	18	0	0	18
	8 (dynamics)	0	0	0	0	0	0	0	170	0	170
	Omitted	13	12	10	20	0	0	6	34	0	90
	Sum	833	2219	258	282	158	95	24	205	4	4078
	Rate	0.984	0.995	0.957	0.929	1.000	1.000	0.750	0.829	–	0.931

Table 6.2: Confusion matrices regarding the classification of *vertical objects*; for the training set and the test set. True symbols horizontally, predicted vertically. The rate corresponds to the classification rate.

perfectly classified, except for the missing ones. However, an accidental might be classified as a wrong kind of accidental without it showing in these tables and images. Therefore, confusion matrices only regarding the accidentals (see Table 6.3) are made, which makes it easier to see what goes wrong, and how to deal with this. See also Figure 6.1(b) for an example of a colored result image.

We observe that the classification rate for accidentals is 0.854 for the training set, and 0.893 for the test set. We see that the biggest problem is accidentals which are not recognized at all by the system, and placed as *omitted* in the confusion matrices. These might be situated too close to the succeeding note head. When it comes to misclassification between the various types of accidentals, this is mostly a problem for sharps. This problem might be fixed completely by including musical rules, mentioned in Section 5.4.1. However, this is disregarded in this thesis.

Recognition of Accidentals

Train		True				
		♭	♯	♮	Others	Sum
Predicted	♭	57	7	0	1	65
	♯	0	90	0	1	91
	♮	0	1	23	0	24
	Omitted	10	11	0	0	21
	Sum	67	109	23	2	201
	Rate	0.851	0.826	1.000	–	0.854

Test		True				
		♭	♯	♮	Others	Sum
Predicted	♭	57	10	0	2	69
	♯	0	182	0	0	182
	♮	0	3	19	1	23
	Omitted	9	4	5	0	18
	Sum	66	199	24	3	292
	Rate	0.864	0.915	0.792	–	0.893

Table 6.3: Confusion matrices regarding the classification of the different types of accidentals before including musical rules; for the training set and the test set respectively. The true accidentals horizontally, predicted vertically. The rate corresponds to the classification rate.

6.1.1.2 Recognition of Dynamics

Dynamics (*class 8*) belong to the symbols class with the lowest classification rate. However, as we can see from the confusion matrices in Table 6.2, none of the dynamic symbols are misclassified as other musical symbols, but all the mistakes are caused by the program overlooking some of

these symbols. The reason for this is that not all of the dynamics belong to the group of *vertical objects* and therefore are missing. The majority of these are missing because we treat each staff separately, and cut the music score into image selections containing only one staff. When doing this, some of the dynamics may be cropped, because they are situated below the staff. See Figure 6.3 for an example illustrating this.

FIGURE 6.3: An example of cropped dynamics, causing classification problems.

This problem could be solved rather easily by cutting the image so that the staff images overlap each other, and this way not damage the dynamics. However, we have not prioritized dealing with this in this thesis, due to time constraints, since dynamic symbols are not among the most important symbols in a music score; they do not affect the actual melody, only the intensity in which the melody is played.

6.1.2 Remaining Symbol Recognition (Rests and Whole Notes)

The confusion matrices in Table 6.4 are found using template matching with the templates from Table 5.1(b). See also Figure 6.1(c) for an example of a result image. Rests are only searched for in a limited area around the third staff line. Common for all these remaining symbols is that we only search for them in areas which are not already *occupied* by *vertical objects* from the previous section.

As we can see, there are some confusions between whole rests and half rests. These look the same, and are only distinguished by the height, so this is understandable. This may, however, be reduced by looking at the original image, and include the staff line in these two templates, see Figure 6.4.

FIGURE 6.4: Example of how one could make the templates for half rests, (a), and whole rests, (b), including the staff line.

The most severe problem is the number of lost whole notes, which is relatively big (6 of 30 symbols in the training set and 8 of 23 in the test set). This is due to shape variations between the music sheets, and may be reduced by making templates of different size and shape for each musical symbols, or use size invariant features.

Remaining symbol Recognition

Train		True					
		9	10	11	12	Others	Sum
Predicted	9 (eighth rests)	23	0	0	0	0	23
	10 (half rests)	0	12	2	0	0	14
	11 (whole rests)	0	0	8	0	0	8
	12 (whole notes)	0	0	0	24	0	24
	Omitted	1	0	0	6	0	7
	Sum	24	12	10	30	0	76
	Rate	0.958	1.000	0.800	0.800	–	0.882

Test		True					
		9	10	11	12	Others	Sum
Predicted	9 (eighth rests)	69	0	0	0	0	69
	10 (half rests)	0	27	3	0	0	30
	11 (whole rests)	0	0	10	0	0	10
	12 (whole rests)	0	0	0	15	0	15
	Omitted	2	0	0	8	0	10
	Sum	71	27	13	23	0	134
	Rate	0.972	1.000	0.769	0.652	–	0.903

Table 6.4: Confusion matrices regarding the classification of the remaining symbols; for the training set and the test set. The true symbols horizontally, predicted vertically. The rate corresponds to the classification rate.

6.2 Pitch Recognition

6.2.1 Pitch Recognition for Notes

In this section we look at the pitch recognition for notes, and the confusion matrices with the results are found in Table 6.5. We see that we obtain a classification rate of 0.893 for the training set and 0.935 for the test set. See also Figure 6.1(d) for an example of a result image.

In the program two C's from different octaves (i.e. a *C4* and a *C5*, see Figure 2.2), are given different values, to be able to assign the correct pitch. However, here in the confusion matrices, a C is a C, no matter which octave it belongs to. This does not affect the confusion matrices in any way, since a *C4* would never be wrongly classified as a *C5*. Most mistakes are caused by confusions between neighboring pitches. However, some more distant confusions appear, and these must be dealt with. In particular, we observe that in the test set most of the wrongly classified pitches are classified as a pitch two pitches below, and not to the neighboring pitch.

Pitch Recognition for Notes

Train		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	169	18	4	1	0	0	1	193
	D	5	192	19	0	0	1	0	217
	E	4	3	141	10	1	0	0	159
	F	0	0	3	116	15	0	1	135
	G	4	1	0	2	201	14	0	222
	A	1	0	0	0	3	181	13	198
	B	13	0	0	0	0	2	191	206
	Omitted	1	1	2	0	0	0	0	4
	Sum	197	215	169	129	220	198	206	1334
	Rate	0.858	0.893	0.834	0.899	0.914	0.914	0.927	0.893

Test		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	238	1	4	0	0	1	1	245
	D	1	334	3	10	0	0	2	350
	E	0	0	250	3	19	0	0	272
	F	0	0	0	220	0	11	0	231
	G	0	0	0	0	314	0	28	342
	A	16	0	0	0	0	340	1	357
	B	1	24	0	0	0	2	433	460
	Omitted	3	4	3	2	4	2	1	19
	Sum	259	363	260	235	337	356	466	2276
	Rate	0.919	0.920	0.962	0.936	0.932	0.955	0.929	0.935

Table 6.5: Confusion matrices regarding the pitch of filled notes and half notes for the training set and the test set. The true pitches horizontally, predicted vertically. The rate corresponds to the classification rate.

For the training set this is not the case. These errors may be due to imprecisions in the *staff line removal method*, which is where the y-coordinates of each staff line are stored. This illustrates how important it is that the staff line removal is performed carefully.

6.2.2 Pitch Recognition for Accidentals

In this section we look at the pitch recognition for accidentals. In Table 6.6, we have the confusion matrices for all accidentals together. As we can see, the classification rate is not high neither for the training set nor the test set (0.543 and 0.685 respectively). However, we find it reasonable to split this problem, and look at each of the three types of accidentals separately.

See Table 6.7-6.9 for the confusion matrices regarding flats, naturals and sharps.

As we discussed in Section 5.3.2, finding the correct pitch of accidentals may be done using musical rules. The results in this section is, however, the results before introducing musical semantics, and illustrates the problems of leaving this out. In Section 6.4 we will study the improved results when applying musical rules.

Sharps and naturals obtain rather good classification results, 0.957 and 0.789 respectively for the training set, and 0.542 and 0.920 for the test set, compared to 0 correct classified flats. However, it is curious that the naturals have a higher error rate for the training set than for the test set, while for the sharps it is the opposite. This might simply be caused by shape and size variations, and illustrates again how important it is to create several sets of templates.

Since flats are not centered around the pitch they represent, it is logical that all the flats are assigned a pitch which is one or two pitches higher than its true pitch. This is seen in the confusion matrices in Table 6.7. See Figure 6.5 for an illustration of how flats are not centered around their pitch.

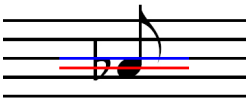
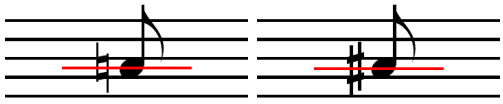


FIGURE 6.5: The blue line indicates the center of the flat with respect to its height, while the red line indicates the actual pitch of the flat.



(a) (b)

FIGURE 6.6: For naturals and sharps, the red line indicates both the center of the accidental with respect to its height, and its actual pitch.

Pitch Recognition for Accidentals (♭, ♮, ♯)

Train		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	21	0	0	0	0	0	33	54
	D	0	2	0	3	0	0	1	6
	E	0	2	7	4	0	0	0	13
	F	0	0	13	68	0	0	0	81
	G	0	0	0	0	1	0	0	1
	A	1	0	0	0	1	1	0	3
	B	2	0	0	0	0	8	8	18
	Omitted	8	4	2	3	2	2	2	23
	Sum	32	8	22	78	4	11	44	199
	Rate	0.656	0.250	0.318	0.872	0.250	0.091	0.182	0.543

Test		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	32	0	2	0	0	0	24	58
	D	0	10	0	6	0	0	3	19
	E	0	0	9	1	0	0	0	10
	F	0	0	12	119	0	0	0	131
	G	0	1	6	3	20	0	2	32
	A	1	0	0	0	0	3	4	8
	B	0	0	0	0	0	5	5	10
	Omitted	4	4	1	2	2	1	7	21
	Sum	37	15	30	131	22	9	45	289
	Rate	0.865	0.667	0.300	0.908	0.909	0.333	0.111	0.685

Table 6.6: Confusion matrices regarding the pitch of accidentals for the training set and the test set. The true pitches horizontally, predicted vertically. The rate corresponds to the classification rate.

Pitch Recognition for Flats (♭)

Train		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	0	0	0	0	0	0	33	33
	D	0	0	0	0	0	0	1	1
	E	0	2	0	0	0	0	0	2
	F	0	0	13	0	0	0	0	13
	G	0	0	0	0	0	0	0	0
	A	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	8	0	8
	Omitted	0	4	2	0	0	2	2	10
	Sum	0	6	15	0	0	10	36	67
	Rate	–	0.000	0.000	–	–	0.000	0.000	0.000

Test		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	0	0	0	0	0	0	25	25
	D	0	0	0	0	0	0	3	3
	E	0	0	1	0	0	0	0	1
	F	0	0	12	0	0	0	0	12
	G	0	0	6	0	0	0	0	6
	A	0	0	0	0	0	0	4	4
	B	0	0	0	0	0	0	0	5
	Omitted	0	2	1	0	0	0	7	10
	Sum	0	2	20	0	0	5	39	66
	Rate	–	0.000	0.050	–	–	0.000	0.000	0.015

Table 6.7: Confusion matrices regarding the pitch of flats for the training set and the test set. The true pitches horizontally, predicted vertically. The rate corresponds to the classification rate.

Pitch Recognition for Naturals (♮)

Train		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	0	0	0	0	0	0	0	0
	D	0	0	0	0	0	0	0	0
	E	0	0	7	1	0	0	0	8
	F	0	0	0	6	0	0	0	6
	G	0	0	0	0	0	0	0	0
	A	0	0	0	0	0	1	0	1
	B	0	0	0	0	0	0	8	8
	Omitted	0	0	0	0	0	0	0	0
	Sum	0	0	7	7	0	1	8	23
	Rate	–	–	1.000	0.857	–	1.000	1.000	0.957

Test		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	0	0	2	0	0	0	0	2
	D	0	0	0	0	0	0	0	0
	E	0	0	5	0	0	0	0	5
	F	0	0	0	3	0	0	0	3
	G	0	0	0	1	0	0	2	3
	A	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	5	5
	Omitted	3	0	0	2	1	0	0	6
	Sum	3	0	7	6	1	0	7	24
	Rate	0.000	–	0.714	0.500	0.000	–	0.714	0.542

Table 6.8: Confusion matrices regarding the pitch of naturals for the training set and the test set. The true pitches horizontally, predicted vertically. The rate corresponds to the classification rate.

Pitch Recognition for Sharps (♯)

Train		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	21	0	0	0	0	0	0	21
	D	0	2	0	3	0	0	0	5
	E	0	0	0	3	0	0	0	3
	F	0	0	0	62	0	0	0	62
	G	0	0	0	0	1	0	0	1
	A	1	0	0	0	1	0	0	2
	B	2	0	0	0	0	0	0	2
	Omitted	8	0	0	3	2	0	0	13
	Sum	32	2	0	71	4	0	0	109
	Rate	0.656	1.000	–	0.873	0.250	–	–	0.789

Test		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	32	0	0	0	0	0	0	32
	D	0	10	0	6	0	0	0	16
	E	0	0	3	1	0	0	0	4
	F	0	0	0	116	0	0	0	116
	G	0	0	0	1	20	0	0	21
	A	1	0	0	0	0	3	0	4
	B	0	0	0	0	0	0	0	0
	Omitted	1	3	0	0	2	1	0	7
	Sum	34	13	3	124	22	4	0	200
	Rate	0.941	0.769	1.000	0.935	0.909	0.750	–	0.920

Table 6.9: Confusion matrices regarding the pitch of sharps for the training set and the test set. The true pitches horizontally, predicted vertically. The rate corresponds to the classification rate.

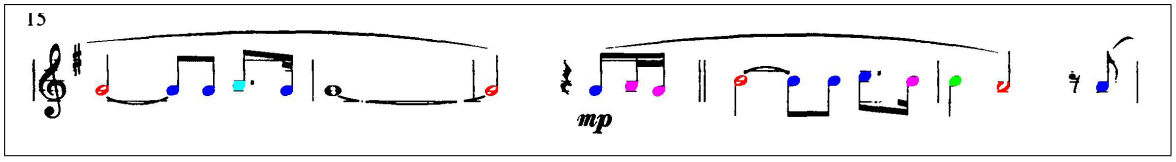
6.3 Note Value Recognition

Confusion matrices regarding the results of the note value recognition are found in Table 6.10, and two result images are presented in Figure 6.1(e)-6.1(f).

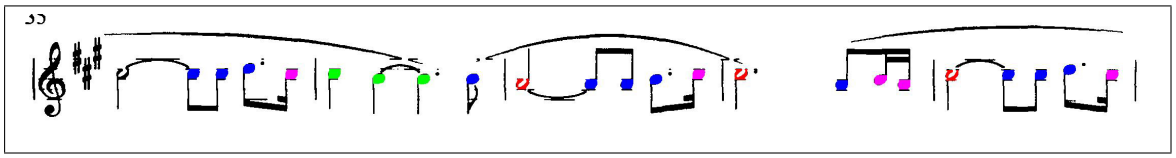
We can easily see that most of the errors in the current method are related to duration dots, which sometimes are overlooked, and in some few cases found even though they are not there. This may be caused by variations in size and shape between the dots, and noise may be interpreted as dots. It is especially a high percentage of half notes and quarter notes followed by a duration dot, which get classified as only a note, with no duration dot.

Another severe problem, is the number of eighth notes either classified as quarter notes or as no symbol. This happens only for single eighth notes, and not beamed notes; and indicates that also the size and shape of these hooks may vary.

Wrongly classified durations may be corrected by introducing musical rules, and checking whether the notes and rests within each time measure make up the right number of beats or not. This, however, depends on correctly classified notes, rests and bar lines, and that none of these symbols are missing. In addition, time signatures need to be recognized or be sent as input to the program by the user. This step is excluded from this thesis, but would be an interesting task to look into for future research.



(a)



(b)

FIGURE 6.7: (a): The result of the note value recognition of the fourth staff of *The Rose*. We see that one duration dot is overlooked, and one sixteenth note is classified as an eighth note, due to interconnecting beams (caused by staff line removal). (b): The result of the note value recognition of the eighth staff of *The Rose*. Here several duration dots are overlooked.

NOTE VALUE RECOGNITION

TRAIN		True										
		𝅗𝅥.	𝅗𝅥	♩.	♩	♪.	♪	𝅘𝅥𝅯.	𝅘𝅥𝅯	𝅘𝅥𝅰	𝅘𝅥𝅱	Sum
Predicted	𝅗𝅥.	58	3	0	0	0	0	0	0	0	0	61
	𝅗𝅥	25	269	0	0	0	0	0	0	0	0	248
	♩.	0	0	35	4	0	0	0	0	0	0	39
	♩	0	0	26	504	0	15	0	2	0	0	547
	♪.	0	0	0	0	24	2	0	0	0	0	26
	♪	0	0	0	0	8	580	0	2	0	0	590
	𝅘𝅥𝅯.	0	0	0	0	0	0	0	0	0	0	0
	𝅘𝅥𝅯	0	0	0	0	0	0	0	155	0	0	155
	𝅘𝅥𝅰	0	0	0	0	0	0	0	0	3	0	3
	𝅘𝅥𝅱	0	0	0	0	0	0	0	0	0	0	0
	Omitted	12	31	0	4	0	29	0	3	0	0	79
	Sum	95	303	61	512	32	626	0	162	3	0	1715
	Rate	0.611	0.888	0.574	0.984	0.750	0.927	–	0.957	1.000	–	0.907

TEST		True										
		𝅗𝅥.	𝅗𝅥	♩.	♩	♪.	♪	𝅘𝅥𝅯.	𝅘𝅥𝅯	𝅘𝅥𝅰	𝅘𝅥𝅱	Sum
Predicted	𝅗𝅥.	35	0	0	0	0	0	0	0	0	0	35
	𝅗𝅥	22	182	0	0	0	0	0	0	0	0	204
	♩.	0	0	66	0	0	0	0	0	0	0	66
	♩	0	0	68	596	0	87	0	0	0	0	751
	♪.	0	0	0	0	60	1	0	0	0	0	61
	♪	0	0	0	0	53	886	0	5	0	0	944
	𝅘𝅥𝅯.	0	0	0	0	0	0	1	0	0	0	1
	𝅘𝅥𝅯	0	0	0	0	0	0	0	323	0	0	323
	𝅘𝅥𝅰	0	0	0	0	0	0	0	0	1	0	1
	𝅘𝅥𝅱	0	0	0	0	0	0	0	0	0	8	8
	Omitted	4	15	6	16	1	27	1	12	1	0	83
	Sum	61	197	140	612	114	1001	2	340	2	8	2477
	Rate	0.574	0.924	0.471	0.974	0.526	0.885	0.500	0.950	0.500	1.000	0.871

TABLE 6.10: Confusion matrices regarding the note values; for the training set and the test set. The true note values horizontally, predicted vertically. The rate corresponds to the classification rate.

6.4 Error Correction Using Musical Semantics

In this section we will study the results of the pitch recognition regarding accidentals after introducing musical rules. We will compare these results to the results found earlier in this chapter.

6.4.1 Error Correction Regarding the Pitch of Accidentals

In this section we consider the results of the error correction. By applying musical rules, we were able to correct several pitch mistakes, especially concerning flats, which we observed in Table 6.6, turned out to be a severe problem. Before including any musical rules, we obtained a classification rate of 0.543 on the training set and 0.685 on the test set. However, after the error correction, these rates increased considerably, see Table 6.11. We now observe an error rate of 0.197 for the training set and 0.159 for the test set. This illustrates how important it is to include musical rules and restrictions in an OMR program.

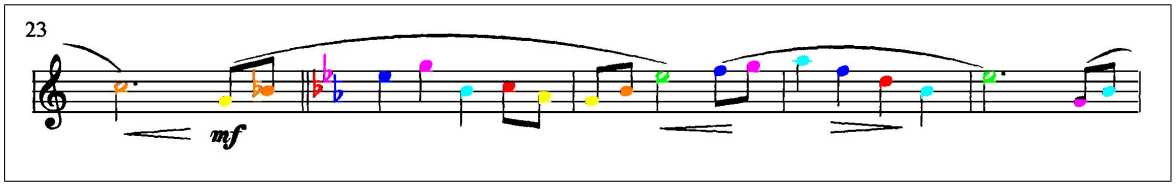
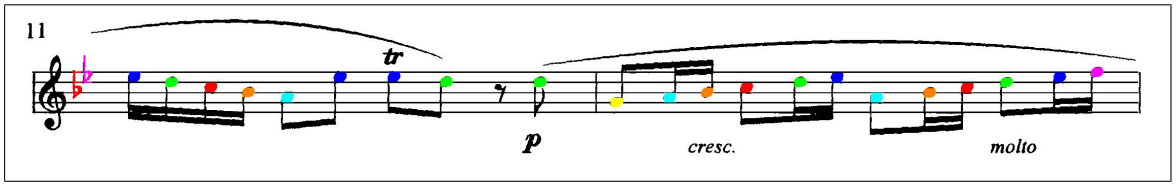
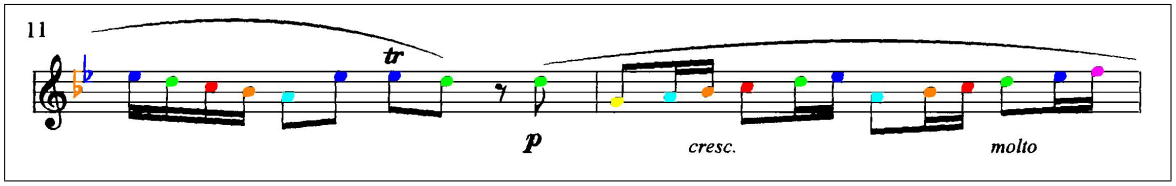


FIGURE 6.8: The result of the pitch recognition of the fourth staff of *Bred dina vida vingar* using musical rules. We can see that key signature changes in the middle of a staff may cause problems. However, the first flat has been corrected.



(a) Pitch recognition before error correction.



(b) Pitch recognition after error correction.

FIGURE 6.9: The result of the pitch recognition of the sixth staff of *Vinter*, before and after introducing musical rules to correct errors regarding accidentals.

Pitch Recognition for Accidentals (Musical Rules)

Train		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	21	0	0	0	0	0	2	23
	D	0	3	0	3	0	0	0	6
	E	0	1	19	4	0	1	0	25
	F	0	0	1	68	0	0	0	69
	G	0	0	0	0	1	0	0	1
	A	1	0	0	0	1	6	0	8
	B	2	0	0	0	0	2	41	45
	Omitted	8	4	1	3	2	2	4	24
	Sum	32	8	21	78	4	11	44	198
	Rate	0.656	0.375	0.905	0.872	0.250	0.545	0.932	0.803

Test		True							
		C	D	E	F	G	A	B	Sum
Predicted	C	32	0	2	0	0	0	5	39
	D	0	10	0	6	0	0	0	16
	E	0	0	27	1	0	0	0	28
	F	0	0	0	119	0	0	0	119
	G	0	0	0	2	20	0	2	24
	A	1	0	0	0	0	7	3	11
	B	0	0	0	0	0	1	29	30
	Omitted	4	5	1	2	3	1	7	23
	Sum	37	15	30	130	23	9	46	290
	Rate	0.865	0.667	0.900	0.915	0.870	0.778	0.630	0.841

Table 6.11: Confusion matrices regarding the pitch of accidentals for the training set and the test set after including musical rules. The true pitches horizontally, predicted vertically. The rate corresponds to the classification rate.

Figure 6.8 illustrates the problem with key signature changes in the middle of a staff. Many OMR programs do not handle this, and in this thesis we have not looked into solving this issue. Figure 6.9 shows the pitch recognition on one staff before and after using musical rules, and we can easily see the improvements.

6.5 Overall Results for Each Music Sheet

In this section we want to take a closer look at the performance of each of the recognition categories on each of the music sheets separately. This way we can find out if there are big differences between the sheets or if the differences are insignificant.

We can see from Table 6.12 that there are some variations between the performances of the different scores. Some, like *sheet 1* and *sheet 26* obtain very high classification rates in all categories, while for instance *sheet 8* gets many wrongly classified accidentals, both regarding accidental type and pitch. As we can, *sheet 9* increased its pitch recognition rate for accidentals from 0.000 to 0.761, using musical rules. This enormous improvement is due to that the score only contains flats, and no other type of accidentals.

It is desirable to minimize the between-sheet differences, since this means that we do not have a program which is robust enough when it comes to dealing with symbol variability. This could be a possible topic for future research.

		Symbol rate			Pitch Rate			Duration rate
	No.	Vertical objects	Accidentals	Remaining symbols	Notes	Accidentals before rules	Accidentals after rules	Notes
Training Set	1	1.000	0.991	1.000	0.991	0.546	1.000	0.974
	2	0.910	0.763	0.857	1.000	0.263	0.703	0.821
	3	0.969	0.941	0.778	0.795	0.882	0.882	0.846
	4	0.955	0.818	1.000	0.951	0.818	0.818	0.783
	5	0.985	0.778	0.714	0.991	0.889	0.889	1.000
	6	0.981	1.000	1.000	0.961	0.833	1.000	0.755
	7	0.987	0.714	–	0.843	0.571	0.571	0.945
	8	0.947	0.650	0.714	0.743	0.300	0.650	0.949
	9	0.991	1.000	0.750	0.863	0.000	0.769	0.971
	10	0.975	0.909	1.000	0.987	0.364	0.909	0.885
	11	0.989	1.000	–	0.881	1.000	1.000	0.958
	12	0.984	0.692	0.833	0.860	0.846	0.846	0.893
	13	0.981	0.889	1.000	0.887	0.778	0.778	1.000
	14	0.957	1.000	0.917	1.000	0.636	0.636	0.949
Test Set	15	0.833	0.995	1.000	0.932	0.917	0.917	0.805
	16	0.980	1.000	1.000	0.721	0.125	0.125	0.809
	17	0.973	0.909	0.250	0.992	0.727	0.727	0.902
	18	0.983	0.889	–	0.981	0.222	0.889	0.802
	19	0.974	0.600	1.000	0.917	1.000	1.000	0.936
	20	0.984	0.883	0.857	0.961	0.692	0.692	0.755
	21	0.979	0.967	0.857	0.936	1.000	1.000	0.832
	22	0.959	1.000	1.000	0.952	1.000	1.000	0.927
	23	1.000	1.000	–	0.992	0.909	0.909	0.805
	24	0.976	0.903	0.857	0.875	0.355	0.742	0.838
	25	0.979	0.895	1.000	0.981	0.105	0.737	0.981
	26	1.000	1.000	1.000	1.000	1.000	1.000	0.793
	27	0.953	0.852	1.000	0.869	0.815	0.815	0.917
	28	0.912	0.647	0.944	1.000	0.647	0.647	0.900
	29	0.981	1.000	0.950	0.815	0.833	0.833	0.864
	30	0.993	0.875	0.500	1.000	0.875	0.875	0.935
	31	1.000	0.889	1.000	0.994	1.000	1.000	0.957
	32	0.974	1.000	1.000	0.890	0.238	1.000	0.903
	33	0.964	0.846	0.833	0.918	0.923	0.923	0.866

Table 6.12: An overview of rates of the different recognition groups for each music sheet separately.

7 Concluding Remarks

Optical Music Recognition has been an active research field during the last decades. The main objective of this thesis was to study existing OMR systems, and experiment to improve current methods and come up with new ideas and suggestions on how to solve the OMR problem. We limited to concentrate on the first stages of the process, and to identifying the most important musical symbols, and ignoring other objects.

In Chapter 2, we reviewed some basic music theory, explained how a typical music sheet is constructed, including the most common musical symbols, and some musical rules which in Section 5.4.1 were used to correct some classification errors. This *correction step* is a very important part of OMR, and systems which do not include musical knowledge generally obtain much lower classification rates. Several existing OMR systems have been studied, and these were summed up in Chapter 3. This chapter gave an overview of the most important differences and similarities between the programs, and described them step by step. During the learning-process when we studied various articles, we started to get ideas on how we wanted to approach the OMR problem. The main image analysis methods used were explained in Chapter 4. This involved morphological operations, projection profiles, template matching and correlation. Our suggested approach followed in Chapter 5. Here, all stages were treated, and we studied examples and algorithms thoroughly. The main results of the classification were presented in Chapter 6. Here we looked at several parts of the recognition separately, and discussed the results.

As mentioned in Section 3.1, there are some problems that commonly occur in OMR programs. Among the issues treated in this thesis, the most problematic ones for us were symbol variability and to little use musical rules. We applied musical rules in the error correction of pitches for accidentals (particularly for flats), and we saw in Section 6.4 that this decreased the error rate considerably. However, further testing and improvements would involve applying musical rules in more than only this branch of the recognition, but also for obtaining better classification rates for e.g. note values.

Also earlier in the process, much prior knowledge about the construction of music scores was used. We were then able to narrow down the search areas when looking for certain types

of symbols. All groups of musical symbol have some sort of restrictions regarding where it is allowed to place them etc. For instance, dynamic symbols may only appear below the staff, while accidentals have to be placed either in the key signature or right in front of a note. Duration dots and staccato dots look the same but are distinguished from each other by their position relative to a note head. Therefore, by including a lot of contextual knowledge, we were able to both prevent some misclassification and make the program run faster.

The algorithms are implemented such that it is possible to evaluate them by computing confusion matrices with classification rates. However, this presumes labeling, which may be extremely time consuming if we are dealing with a large dataset. Therefore, we chose also to create colored output images to easily illustrate the results and in an even clearer way, see exactly where the algorithms fail, and possibly be able to deal with this.

A great challenge in the OMR field, is that there exist no universal music font. However, the intent is to be able to create a program which can deal with music sheets of different form and layout. In future studies, it would be interesting to try to generalize the methods so that they can be used on a completely new dataset. In this thesis, we have only tested the complete system on our 33 note sheets, which all are collected from the same book. However, as we saw in Chapter 6, there exist size and shape variations even within one score. The preprocessing stage was also tested on some new sheets, with a rather different layout and size on the musical objects. The main difference between these scores and the ones in our dataset was that the new sheets did not have a bar line at the beginning of each staff. Therefore, our algorithm that find the angle of rotation (Algorithm 5.1) did not work properly. This was solved by searching for the right corners instead of the left corners. Since we eventually only used the original 33 notes, we held on to the initial method, without changing anything.

In the classification stage, several changes have to be made to generalize this to be valid for notes from different publishers. First of all, template matching is not scale invariant, which means that if a music sheet is more compact than another, we should adopt another set of templates. Rossant and Bloch mention that they have a set of different templates, made from sheets of divergent form. In the preprocessing stage, when we found the staff line spacing, this can be used to find out which of the set of templates to use. If we had had more time, this would be an interesting path to follow.

A Dataset

	No.	Title	Composer
Training Set	1	Adagio	Ludwig van Beethoven
	2	Air	Johann Sebastian Bach
	3	Ave Maria	Johann Sebastian Bach
	4	Ave Maria	Hugo Gyldmark
	5	Ave Maria	Franz Schubert
	6	Amazing Grace	Virginia Harmony
	7	Ave Verum Corpus	Wolfgang Amadeus Mozart
	8	Bred dina vida vingar	Swedish folk tune
	9	Den fyrste song	Lars Søraas the elder
	10	Greensleeves	Irish folk tune
	11	Hymne	Nils Larsen
	12	Intermezzo Sinfonico	Pietro Mascagni
	13	Jesu Joy of Man's Desiring	Johann Sebastian Bach
	14	Time to Say Goodbye	F. Sartori & L. Quarantotto
Test Set	15	Largo	George Frideric Händel
	16	Litanei	Franz Schubert
	17	Mitt hjerte alltid vanker	Swedish folk tune
	18	Mot kveld	Agathe Backer Grøndahl
	19	Nocturne	Rolf Løvland
	20	On Wings of Song	Felix Mendelssohn
	21	Panis Angelicus	César Franck
	22	Pie Jesu	Andrew Lloyd Webber
	23	Sæterjentens søndag	Ole Bull
	24	Solveigs sang	Edvard Grieg
	25	Sommernatt ved fjorden	Kjetil Bjørnstad
	26	Stjernesangen	Jean Sibelius
	27	Svanen	Camille Saint-Saëns
	28	Tears in Heaven	Eric Clapton
	29	The Rose	Amanda McBroom
	30	Ut mot havet	Edvard Fiflet Bræin
	31	Veslemøys sang	Johan Halvorsen
	32	Vinter	Antonio Vivaldi
	33	Vitae Lux	Frode Alnæs

Table A.1: An overview of the training set and the test set, together with the titles and composers of the music scores.

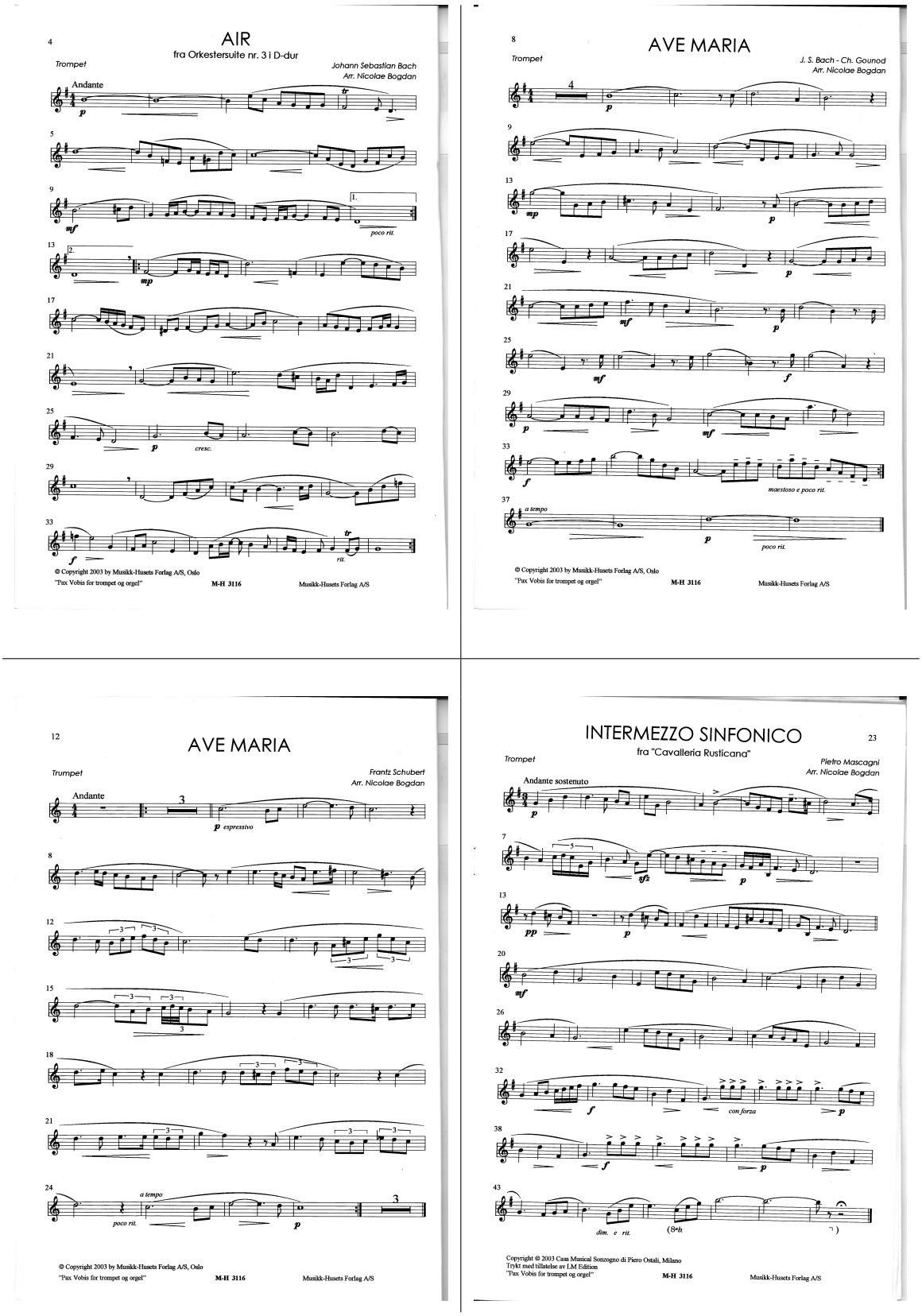


FIGURE A.1: Four of the input images.

B Additional Results

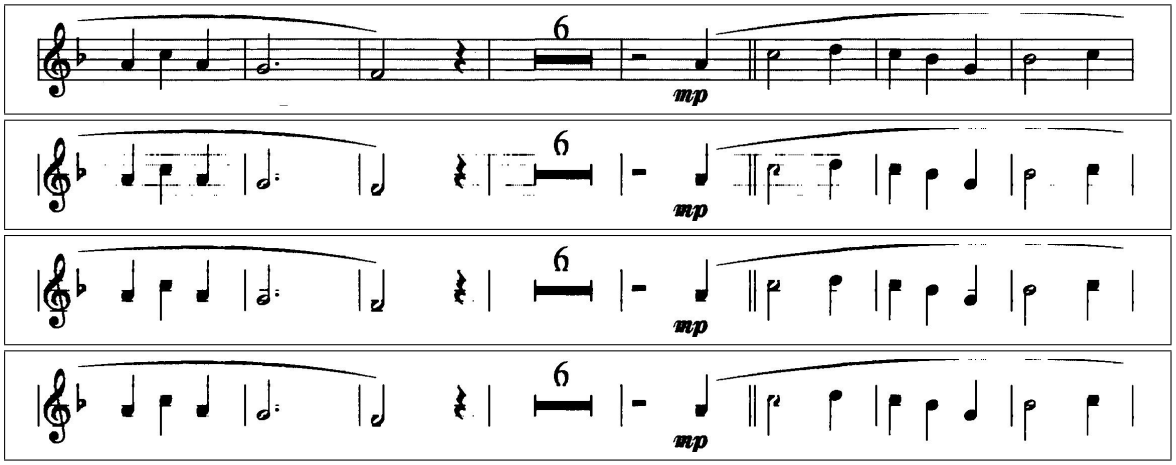


FIGURE B.1: The third staff of *Den fyrste song*; before removing the lines, and after removing the lines using Method A-C described in Section 5.1.4.

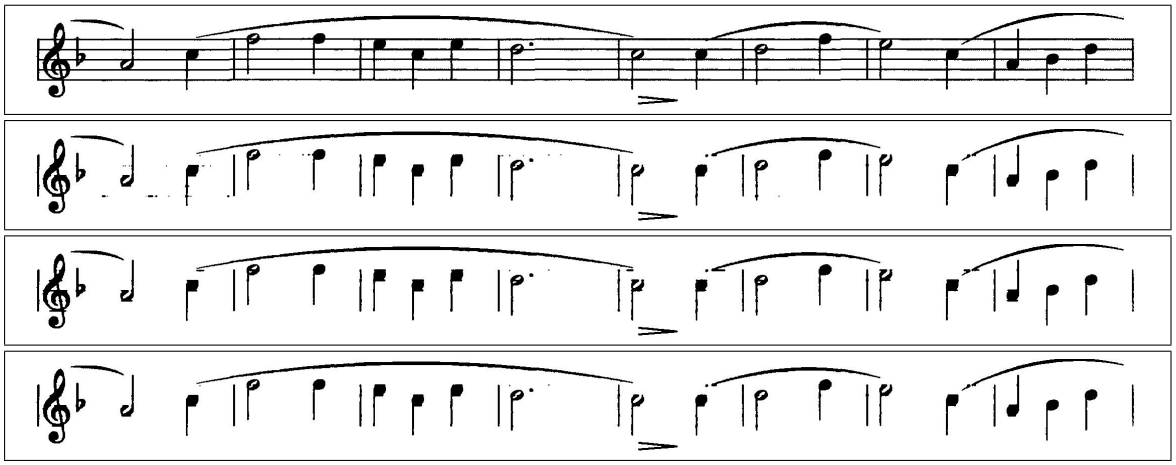


FIGURE B.2: The fourth staff of *Den fyrste song*; before removing the lines, and after removing the lines using Method A-C described in Section 5.1.4.

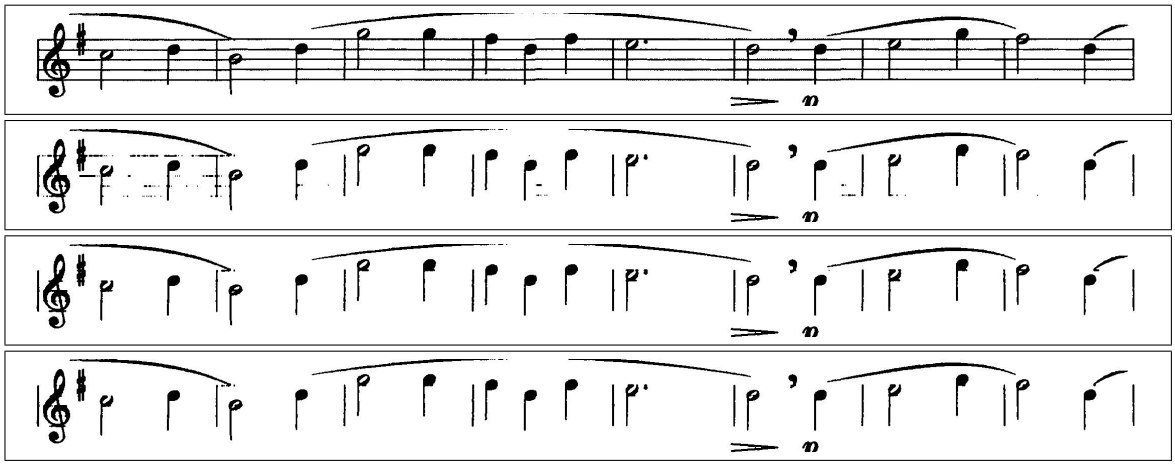


FIGURE B.3: The sixth staff of *Den fyrste song*; before removing the lines, and after removing the lines using Method A-C described in Section 5.1.4.

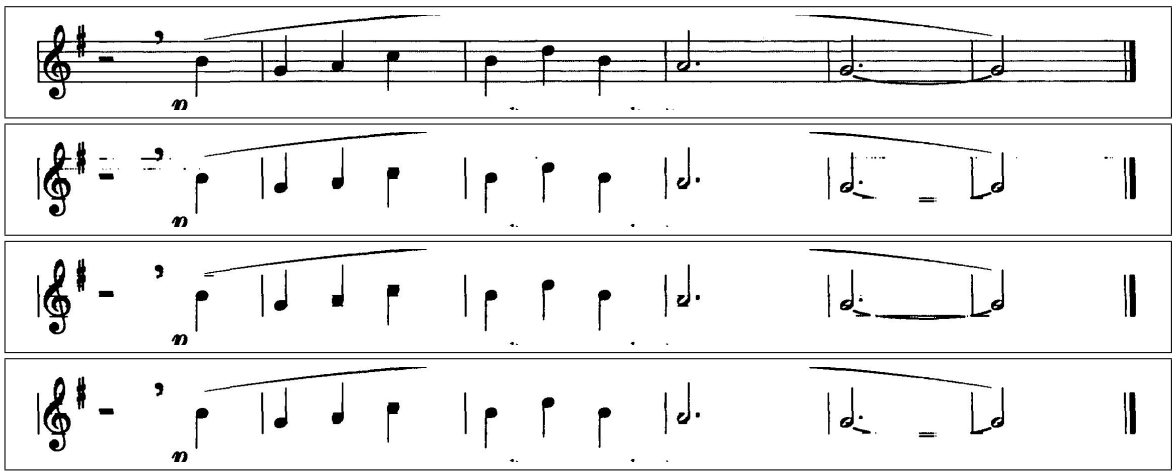


FIGURE B.4: The eighth staff of *Den fyrste song*; before removing the lines, and after removing the lines using Method A-C described in Section 5.1.4.

Bibliography

F. Albregsten. Non-contextual thresholding and adaptive thresholding. Available at: `http://www.ifi.uio.no/~inf5300/2008/thresholding-13022008.pdf`, February 2008.

J. Anstice, T. Bell, A. Cockburn, and M. Setchell. The design of a pen-based musical input system. In *Proceedings on the Sixth Australian Conference on Computer-Human Interaction*, pages 260–267, November 1996.

D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.

D. Bainbridge and T. Bell. A music notation construction engine for optical music recognition. *Software-Practice & Experience*, 33(2):173–200, 2003.

D. Bainbridge and K. Wijaya. Bulk processing of optically scanned music. *Seventh International Conference on (Conf. Publ. No. 465) Image Processing And Its Applications*, 1:474–478, July 1999.

S. Baumann. A simplified attributed graph grammar for high-level music recognition. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 2, pages 1080–1083, August 1995.

P. Bellini, I. Bruno, and P. Nesi. Optical music sheet segmentation. In *Proceedings of the First International Conference on WEB Delivering of Music (WEDELMUSIC '01)*, pages 183–190, Florence, Italy, November 2001.

D. Blostein and H. S. Baird. A critical survey of music image analysis. *Structured Document Image Analysis*, pages 405–434, 1992.

N. Bogdan. *Pax Vobis for trompet og orgel.* Musikk-Husets Forlag A/S Oslo, M-H 3116 edition, 2003.

M. E. Bonds. *A History of Music in Western Culture.* Pearson Education, Inc., 2006.

G. Chen and S. Xia. The study and prototype system of printed music recognition. In *Proceedings of the 2003 International Conference on Neural Networks and Signal Processing*, volume 2, pages 1002–1008, December 2003.

B. Couasnon and J. Camillerapp. A way to separate knowledge from program in structured document analysis: application to optical music recognition. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 2, pages 1092–1097, August 1995.

N. Efford. *Digital Image Processing - a practical introduction using Java.* Pearson Education Limited, 2000.

A. Fornés, J. Lladós, and G. Sánchez. Primitive segmentation in old handwritten music scores. *Graphics Recognition. Lecture notes in Computer Science*, 3926:279–290, 2006.

B. R. Hanning. *Concise History of Western Music.* W. W. Norton & Company, Inc., 1998.

M. Luckner. Recognition of noised patterns using non-disruption learning set. *Sixth International Conference on Intelligent Systems Design and Applications, 2006. ISDA '06*, 1:557–562, October 2006.

K. Ng. Music manuscript tracing. *Lecture Notes in Computer Science*, 2390:330–342, 2002.

K. C. Ng, D. Boyle, and D. Cooper. Low- and high-level approaches to optical music score recognition. *IEE Colloquium on Document Image Processing and Multimedia Environments*, pages 3/1–3/6, November 1995.

J. C. Pinto, P. Vieira, M. Ramalho, M. Mengucci, P. Pina, and F. Muge. Ancient music recovery for digital libraries. *Research and Advanced Technology for Digital Libraries, Proceedings. Lecture Notes in Computer Science*, 1923:24–34, 2000.

A. Rebelo, A. Capela, J. F. P. Costa, C. Guedes, E. Carrapatosos, and J. S. Jaime. A shortest path approach for staff line detection. *Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, 2007. AXMEDIS '07*, pages 79–85, November 2007.

K. T. Reed and J. R. Parker. Automatic computer recognition of printed music. In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR '96)*, volume 3, pages 803–807, Vienna, Austria, August 1996.

F. Rossant and I. Bloch. A global method for music symbol recognition in typeset music sheets. *Pattern Recognition Letters*, 23(10):1129–1141, 2002.

F. Rossant and I. Bloch. A fuzzy model for optical recognition of musical scores. *Fuzzy Sets and Systems*, 141(2):165–201, 2004.

F. Rossant and I. Bloch. Optical music recognition based on a fuzzy modeling of symbol classes and music writing rules. In *Proceedings of IEEE International Conference on Image Processing (ICIP '05)*, volume 2, pages 538–541, Genoa, Italy, September 2005.

F. Rossant and I. Bloch. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *Eurasip Journal on Advances in Signal Processing*, 2007.

E. Selfridge-Field. Optical recognition of music notation: A survey of current work. *Computing in Musicology: An international directory of applications*, 9:109–145, 1994.

M. C. Su, H. H. Chen, and W. C. Cheng. A neural-network-based approach to optical symbol recognition. *Neural Processing Letters*, 15(2):117–135, 2002.

M. Szwoch. Guido: A musical score recognition system. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:23–26, September 2007.

P. Vieira and J. C. Pinto. Recognition of musical symbols in ancient manuscripts. In *Proceedings. 2001 International Conference on Image Processing*, volume 3, pages 38–41, October 2001.

K. Wijaya and D. Bainbridge. Staff line restoration. *Seventh International Conference on (Conf. Publ. No. 465) Image Processing and Its Applications*, 2:760–764, July 1999.

Wikipedia. Modern musical symbols. Available at: `http://en.wikipedia.org/wiki/Modern_musical_symbols`, May 2009.