Optical Music Recognition

Emma Weberyd, Jennifer Bedhammar, Julius Kördel, Oliver Johansson

Abstract

Optical Music Recognition is an image processing application which makes it possible for a computer to read music of a digitized note sheet. This report describes a software implementation of an OMR system in MATLAB. The system takes in an image of a note sheet then finds and separates the music symbols. Each symbol is then assigned musical meaning and a description of the note sheet is made.

1 Introduction

Digitizing old papers has been a useful and common tool for preserving information for a long time. However, deciphering the contents of an image still requires the human eye to some degree. A computer can not tell an image of a note sheet, a drawing or a picture apart unless they have labels or software analyzing them. Optical Character Recognition (OCR) is an application of image processing to automate character classification from a scanned or photographed text. Optical Music Recognition (OMR) is a branch of OCR specifically used for reading music. This report describes an implementation of automatic optical music recognition for scanned sheet music.

1.1 Aim

The goal of the project is to settle an algorithm that takes an image of sheet music and outputs a string containing the notes from the sheet, in the order they would be played.

1.2 The Task

The first thing that needs to be considered is the condition of the image. The system is unlikely to be able to restore missing information. However, the system can take into account the varying quality of different images and compensate for that in order to receive a better result. Image pre-processing is the notion of preparing an image by making transformations that will benefit further processing. In the case of OMR, examples of pre-processing could be filling in disconnected lines and filling out missing parts of noteheads that have disappeared in the scanning process. If the sheet music has been captured by a camera, pre-processing could involve compensating for lens distortion or uneven lighting. If the sheet is handwritten there could be inconsistencies in note sizes, staff line distance and tilt to take into account.

After pre-processing the OMR-system can be divided into three sections [1]:

1. Recognition of music symbols
2. Musical notation reconstruction
3. Final representation

The first step requires different operations to improve the quality of the symbols and separate them. To improve recognition, the staff lines should be detected and removed to get an image with only the music symbols. Staff line thickness and distance between the lines also gives a useful scale for size comparisons. The symbols are then separated and lastly classified. In the second step the detected symbols from step one are deciphered and assigned musical meaning by using known musical syntax and graphical features. In the third step the music information, obtained from the previous step, is used to generate a description of the music sheet.

1.3 Scope

This OMR implementation and project does not cover all possible music symbols that can be found on a music sheet. The software implementation will be limited to detecting quarter and eighth notes. It is assumed that each staff row contains five staff lines and starts with a Treble Clef, and that the pitch ranges from G1 to E4 in the scientific pitch notation, which is depicted in Figure 1. The implementation is focused on scanned note sheets and will not include the pre-processing that is required to read images of photographed note sheets. The project is implemented in MATLAB.

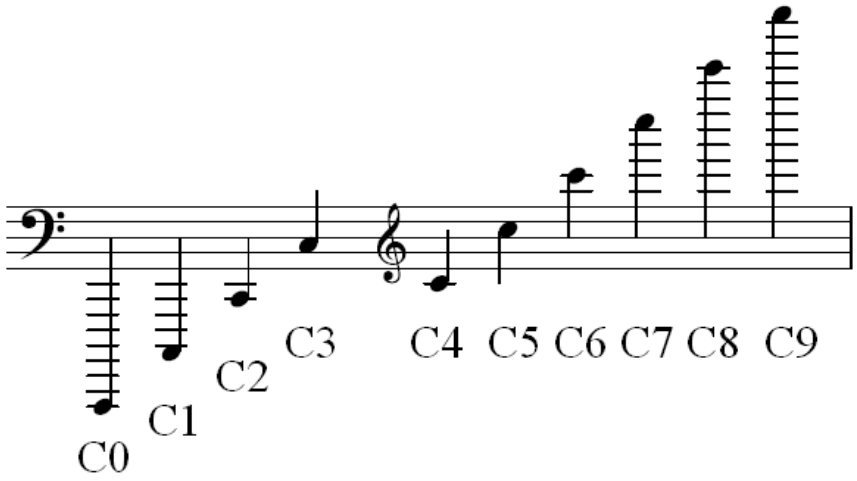


Figure 1 Different octaves of C in Scientific Pitch Notation [2].

2 Implementation

This section describes the software implementation of the OMR-system in MATLAB, what methods were used and why. In order to follow all the operations a sample image from the training set will be used as an example throughout the chapter, see Figure 2.

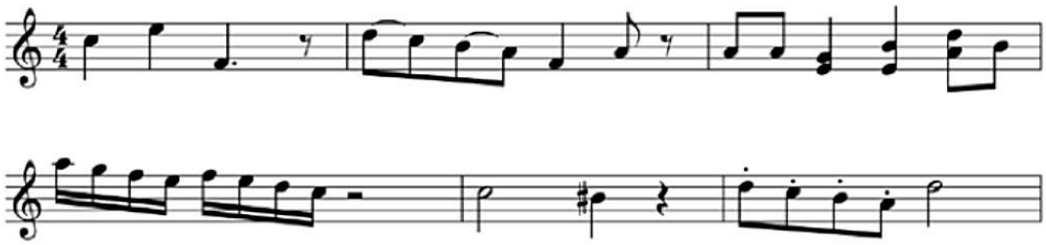


Figure 2 The sheet music that will be used as an example throughout the chapter.

2.1 Image pre-processing

To increase the accuracy of the result pre-processing is used.

2.1.1 Binary Images

The first method used on the note sheet in the OMR-system makes the image binary and inverted, see Figure 3. A binary image is a black and white image that only contains zeros and ones. In MATLAB ones represent “objects” within an image and zeros represent empty space. This makes it easier to distinct notes from the background. However, an original note sheet is constructed of black notes on white paper, which transfers to MATLAB (when binarized) as one large item with small note-shaped holes. What is desired is small note-shaped objects and therefore the binary image is inverted.

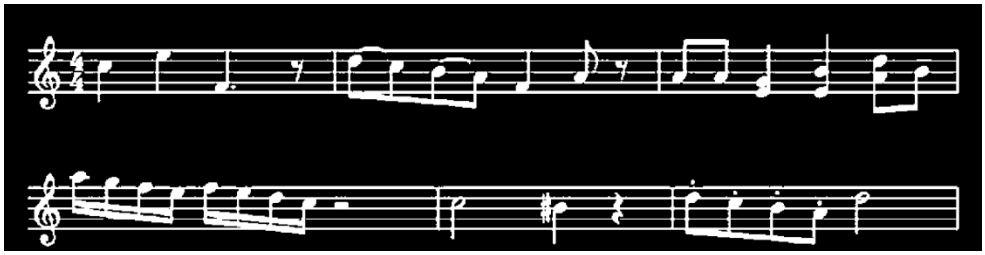


Figure 3 The inverted binary image of Figure 2.

2.1.2 Rotation

In the pre-processing stage the image is rotated to ensure horizontal lines. Horizontal lines are important for the detection and removal of staff lines, as well as deciding the pitch of the notes. The rotation angle is found by using Hough transform [4]. Hough transform (*hough* function in MATLAB) locates straight lines in the image, and by using the functions *houghpeaks* and *houghlines* the location and rotation of the most distinct lines are given.

The rotation of the longest line is then used to rotate the image, since this is most likely a staff line.

2.1.3 Morphological operation

Before staff lines and other elements in the note sheet are removed some objects needs to be saved for future use. The beams of the connected eighth and sixteenth notes get damaged at the subsequent algorithms and therefore the beams are captured exclusively in another image to be able to repair them later. To target the beams, a horizontal oriented (4 by 20 pixels) rectangle is created and applied to a morphological opening operation. The morphological operation only saves objects in the image that can fit the rectangular created object once or multiple times. Hence the only objects that can fit the rectangle inside itself are the beams (and some minor slurs, ties and dynamic indicators) which will result in an image with exclusively those items. The irrelevant elements that are included in the created image will be removed in the later stages and will not interfere if kept in the meantime. The reason why only the horizontal elements of the beam are stored and not the vertical or angled elements is due to loss of horizontal information when removing the staff lines. The stored horizontal elements of Figure 3 can be seen in Figure 4.

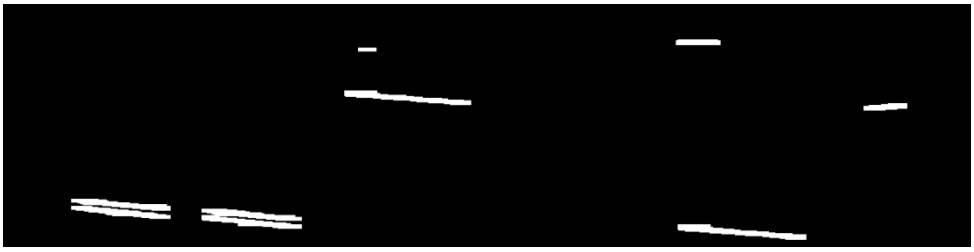


Figure 4 Result of opening with horizontal element.

2.2 Staff line detection and removal

For OMR the stafflines need to be identified. The stafflines are the longest horizontal element in the musical sheet. Even if stafflines are the longest horizontal element they vary in size from image to image. In a musical sheet there are other horizontal elements that are not staff lines. To find how long the horizontal elements are and where they are placed in the image, horizontal projection is used. Horizontal projection corresponds to horizontal summations and for Figure 3 the result is Figure 5.

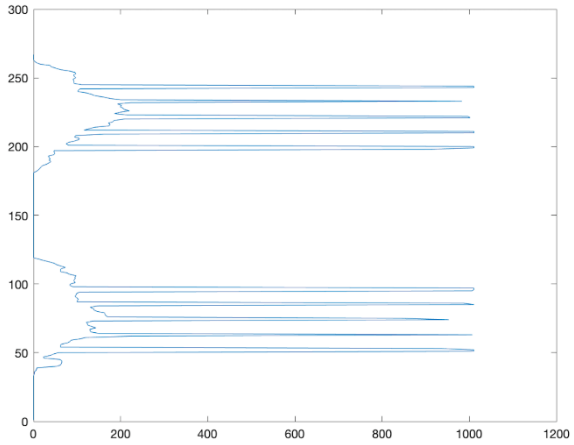


Figure 5 Histogram where the vertical axis has the row-values in the image, and the horizontal axis constitutes the sum of all pixel values in each row.

In Figure 5 there are clear peaks after the horizontal summation. The lower peaks can be explained as noise. The noise can be removed by thresholding with a value related to the highest peaks value. The highest peaks will not always be grouped by five, it just happened to be so for the shown case. If there are more than five candidates for being staff lines some of the peaks has to be removed due to the limitation of five lines per staff line. This can be done by choosing the five largest peaks. If this method is insufficient the distance between the peaks is also used to determine the five peaks. The steps of sorting out stafflines can be seen in Figure 6.

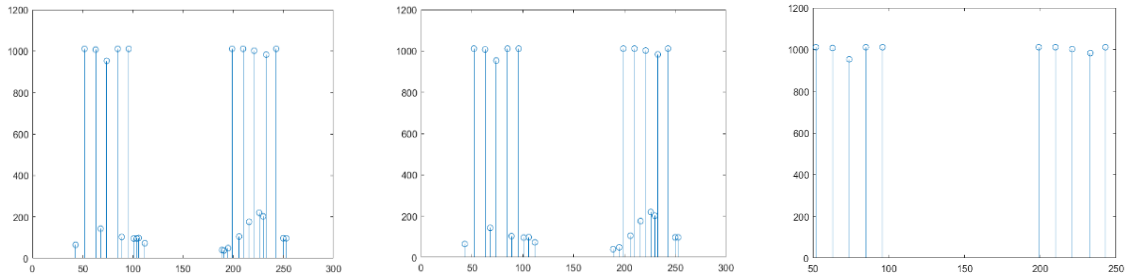


Figure 6 The process of removing dubblets, noise and grouping the lines in groups of five

Once the staff lines are identified they are stored and then removed from the image by setting all heights where the lines exist to zero, which corresponds black. However, this method would break the stems and therefore the algorithm follows along the staff lines and checks for the existence of black pixels above and below the line [3]. This complement results in less data, related to the notes, being lost when removing stafflines. When removing stafflines there is a risk of removing beams placed at same height as the lines. The beams are needed when determining the note value. To prevent this loss of information the horizontal beams has been stored beforehand and can be added to a new image with removed stafflines. The difference between adding beams afterwards can be seen in Figure 7 and Figure 8.

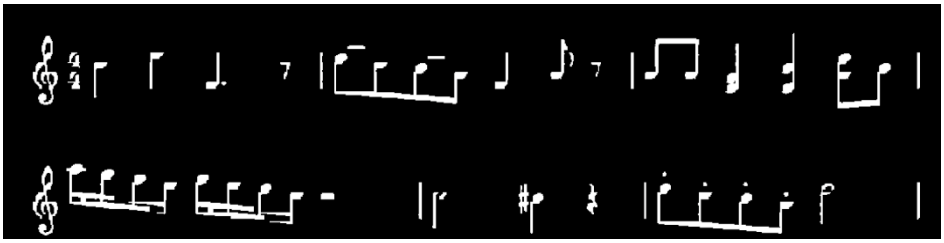


Figure 7 After removing the staff lines, the beams become damaged.

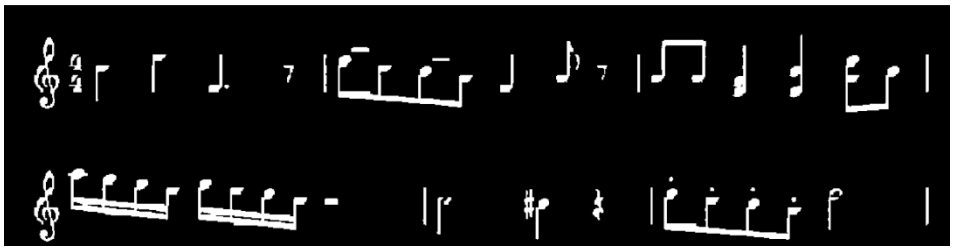


Figure 8 Image without staff lines combined with the beams that were saved from before.

2.3 Symbol detection and segmentation

After the pre-processing the symbols can be identified and separated. This is done by dividing the image, labeling the objects and erasing information not related to the notes.

2.3.1 The g-clef

To separate the notes they have to be located. However, before searching for notes some element needs to be discarded and the first element that is removed is the g-clef. The g-clef always appear in every note sheet and only in the beginning of every staff, which gathers all g-clefs in a vertical row on the far left side in the note sheet. A vertical projection is applied on the note sheet to be able to find where all the g-clefs are stacked and the image is cropped on the left side until all g-clefs are removed. The result is shown in Figure 9.

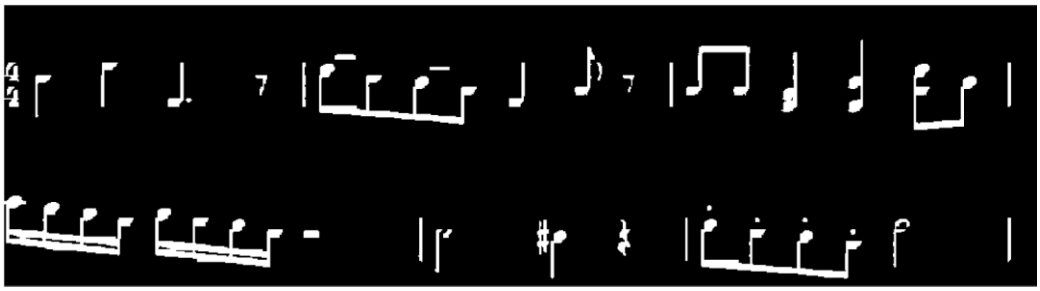


Figure 9 The inverted binary image where the Treble Clefs have been eliminated.

2.3.2 Labeling of music symbols

To be able to classify certain notes, they should be isolated from the rest of the image. In order to map the classified notes to their original position in the image, the image is labeled using the matlab function called *bwlabel*. It takes in the inverted binary image, see Figure 9.

that has been cleared of staff lines and irrelevant symbols and outputs an image where each connected element has its own label, see Figure 10.

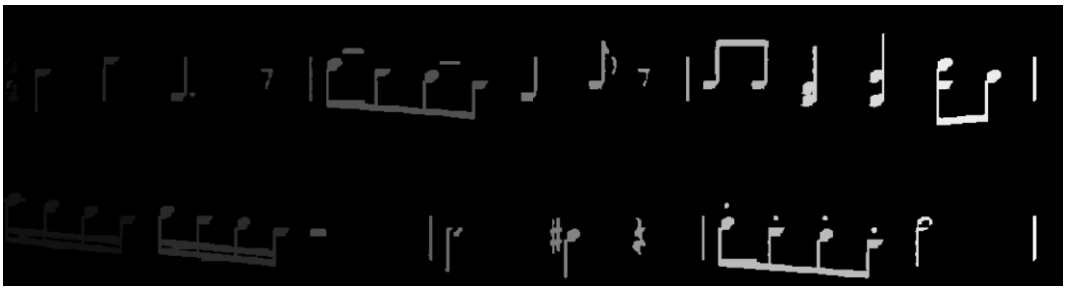


Figure 10 Image returned from bwlabel where each note is made up of pixels with the value of its label. The notes are labeled top to bottom and then left to right.

Each element is originally labeled from top to bottom and then left to right in the image, meaning that the order of the notes is not consistent with the order in which the notes should be played. To fix this issue, the image is first divided into parts where each image part constitutes one row of notes in the sheet, see Figure 11.

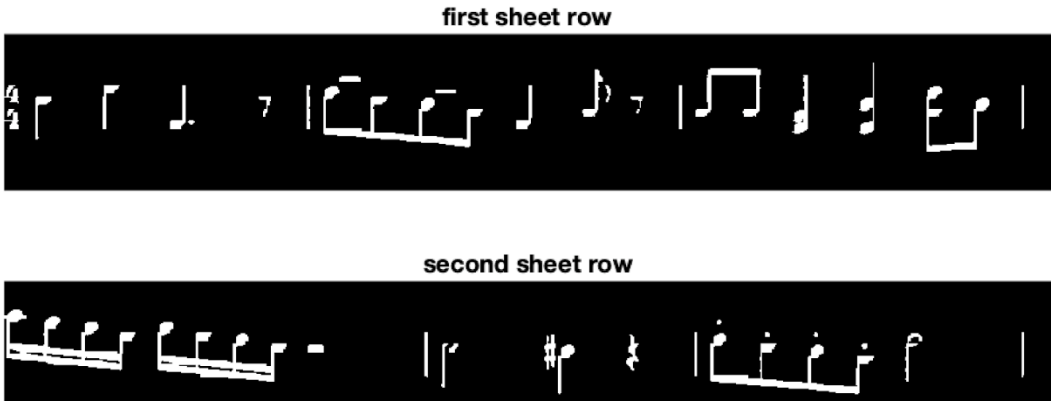


Figure 11 The binary image divided into each row of notes.

The labeling can then be carried out uninterrupted from left to right and then top down as one would read the music. After each part has been labeled they are stitched together again. The final result can be seen in Figure 12.

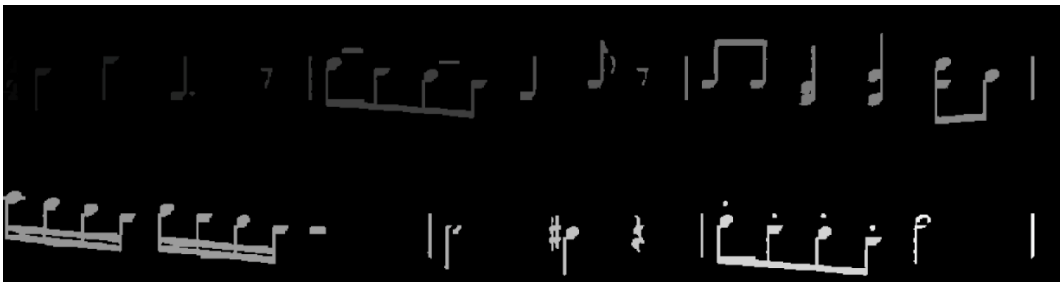


Figure 12 The final labeled image, which is labeled in the order that is consistent with how the music should be played.

The image is then divided into smaller images where each image consists of one labeled element. These images are put into an array, which is as big as there are labeled elements in the image. A depiction of the first ten array elements can be seen in Figure 13.

Figure 13 The first ten labeled elements in the array. The objects are not comparable in size.

2.3.3 Removing irrelevant objects

The labeled object array is then traversed to find notes and to remove irrelevant objects such as rests, random lines and open noteheads (half and whole notes). The removal is based on size comparisons and notehead template matching. Objects smaller than the template are removed because they can not be matched with. Template matching is done with the MATLAB function *normxcorr2* and thresholding. If there is a match the note is saved, otherwise it is removed, see Figure 14 and Figure 15 respectively.

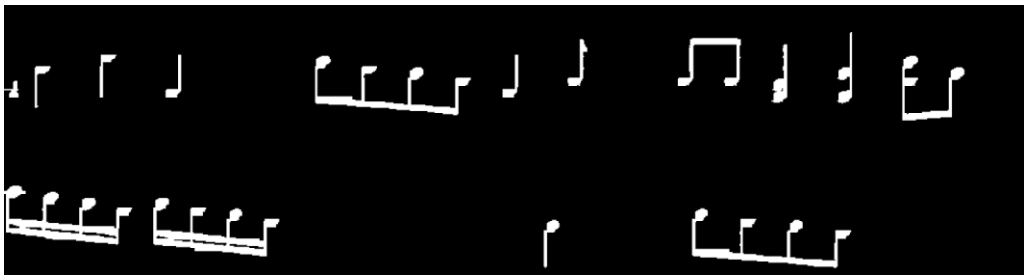


Figure 14 Objects that matched the notehead template.

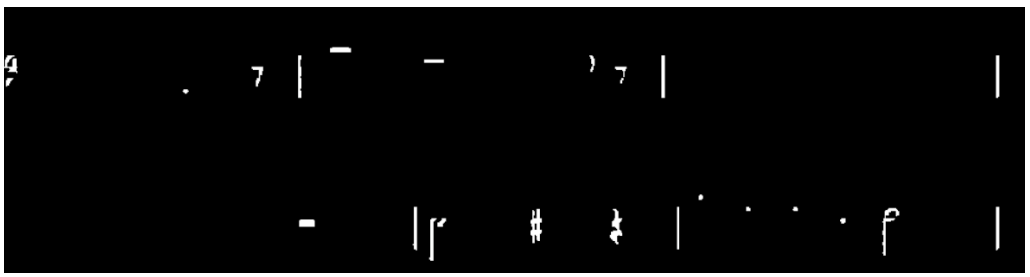


Figure 15 Objects that were either too small or did not match with template.

2.3.4 Locating the noteheads

To find what pitch each note has, the height of each notehead has to be mapped in relation to the staff lines. In order to know where the noteheads are, they need to be extracted from the image in Figure 14. This is done using template matching with an image of a notehead 70% the size of the average notehead in the image. The result is thresholded with 0.6 so that the elements that have a probability of 0.6 of being noteheads are determined to be noteheads. The size of the template notehead and the threshold value have been experimentally determined. The result of this operation is shown in Figure 16.

Figure 16 Result of template matching with notehead template.

Since the template matching results in small noteheads, the confidence in determining the pitch is quite low. It's difficult to determine what parts of the noteheads that remain, and if they are not center parts then the pitch can't be determined properly. The problem of extracting the noteheads was then instead approached by performing the morphological matlab operation *imopen* with a disc element, approximately the size of an average notehead from the image in Figure 14. The result of this operation is shown in Figure 17.

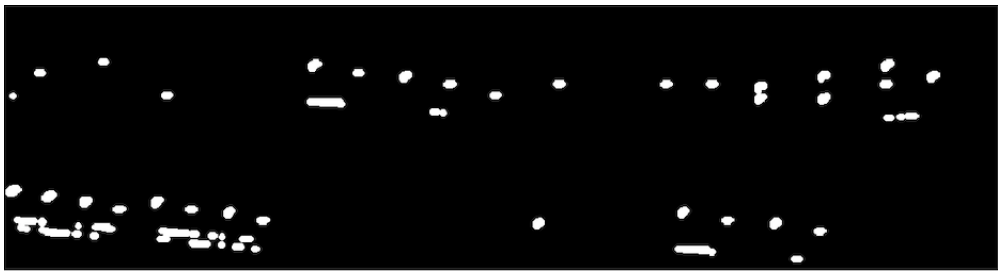


Figure 17 Result of matlab operation imopen with disk element

With this approach the notehead preservation is better. However, there is a lot of unwanted debris. The two approaches were then combined so that each object in Figure 17 which is also represented in Figure 16 will be kept. This results in Figure 18.

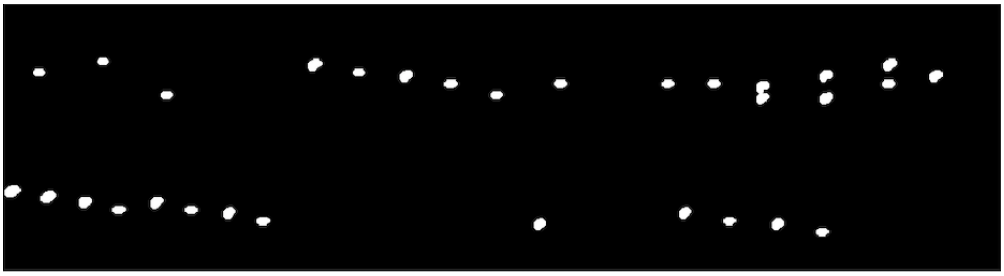


Figure 18 Result of combining template matching and opening with a disk element.

2.4 Note deciphering and musical description

Once the notes have been localised and separated, they can be deciphered and a musical description of the note sheet can be made.

2.4.1 Classifying note value

The result will only include quarter and eighth notes, however all notes still needs to be classified. A note can have several flags or beams and for each flag or beam the note value is doubled. The quarter note has neither flags nor beams.

To be able to classify the value of the notes their position in the input image need to be known. The positions of the notes are localised by a vertical projection. A vertical projection finds the stems of each note. This combined with the noteheads' position relative to their respective stem makes it possible to decide what part of the image to analyze for classifying the note. Examples of input images for the classifications can be seen in Figure 19.

Figure 19 Example of images that the classification is made on.

The first thing to identify is where the noteheads are located in the image. This is done by opening with a disc element with half the size of a notehead. The noteheads need to be stored and later removed to lessen the risk of wrongly classifying a note. This opening operation will sometime match with beams because the disc element also fits in the beams. Matched noteheads in beams need to be sorted out from our notehead image and to do this the position of the first notehead is identified. Depending on the position of the matched note the next note can begin on the proper half of the image, meaning if the last match was on the top half of the image then the search continues top to down.

When noteheads has been localised they can be removed from the image before doing vertical projection. The heads are not needed when deciding how many notes there are in the current sequence and removing them lessens risk of detecting a vertical line that is

unwanted. The vertical projection also need to remove overlapping lines, noise and lower peaks as done in the horizontal projection. When the position of the noteheads and stems are known a local horizontal projection is made as in Figure 20 where flags or beams occur related to the notehead.

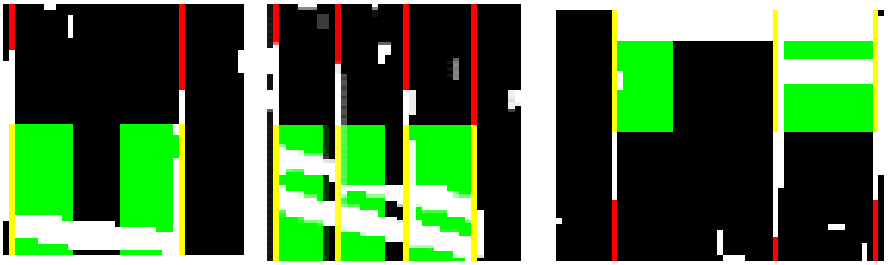


Figure 20 The green area represent the area the local projection is made on and red lines are the vertical lines corresponding to the stems.

To get a better classification the green area seen in Figure 20 is used on the image with the added horizontal elements of the beams and no noteheads. If there are beams in the green area of this image and they are tilted, the image will be rotated for a proper horizontal projection. Depending on the number of peaks found by the projection the note is classified with a value. This is repeated for every note in the input images. The classifications are stored in an array and is later used when deciding the note value for the output.

2.4.2 Pitch and print out

To be able to decide the pitch of the noteheads a matrix with reference points is created. Each row of the matrix corresponds to a row of staff lines in the image and the columns represents the height of each pitch for the specific row, sorted highest to lowest pitch, E4 to G1. The height of each pitch is then calculated by using the distance between the staff lines. The final implementation uses the distance between the first and second staff line but average staff line distance was also tested, however the first was more stable and therefore the final choice.

The final step in the algorithm is to combine the previous results and print out the note's pitch and value. The pitch of the notes are calculated by comparing the notehead's positions with the reference points for the pitches. With a nearest neighbour method the closest pitch reference is established for each notehead and assigned to respective note. The note value is established by the note classification array gathered in a previous algorithm. Each notehead is matched with the array of classifications and given its note value. If a note does not have a corresponding note value it is assigned as a quarter note. Thereafter each note's pitch and note value is printed as a long string, where a lower case letter represented an eighth note, a higher case letter represents a quarter note and the combinations of a number and a letter the pitch.

3 Results

The following is three different note sheets that represent three different levels of success. The result of the OMR is represented by a string where each new line is marked with a *n.*

Note Sheet 1

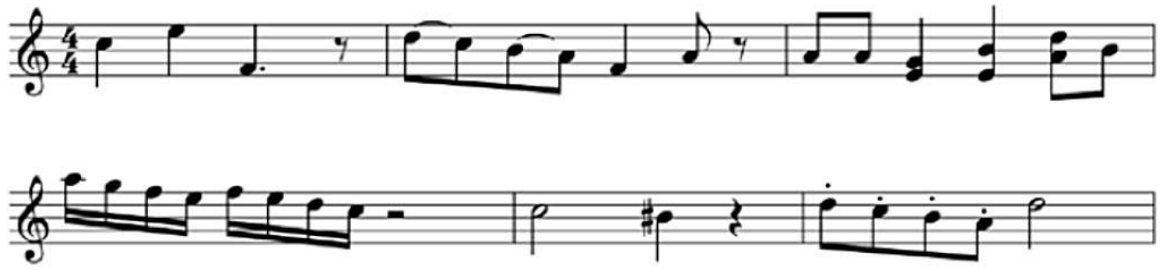


Figure 21 Test image 1

Output string – Note sheet 1

Result	C3E3F2d3c3b2a2F2a2a2a2F2B2E2a2d3b2nB2d3c3b2a2
Correct answer	C3E3F2d3c3b2a2F2a2a2a2G2E2B2E2d3a2b2nB2d3c3b2a2

Note Sheet 2

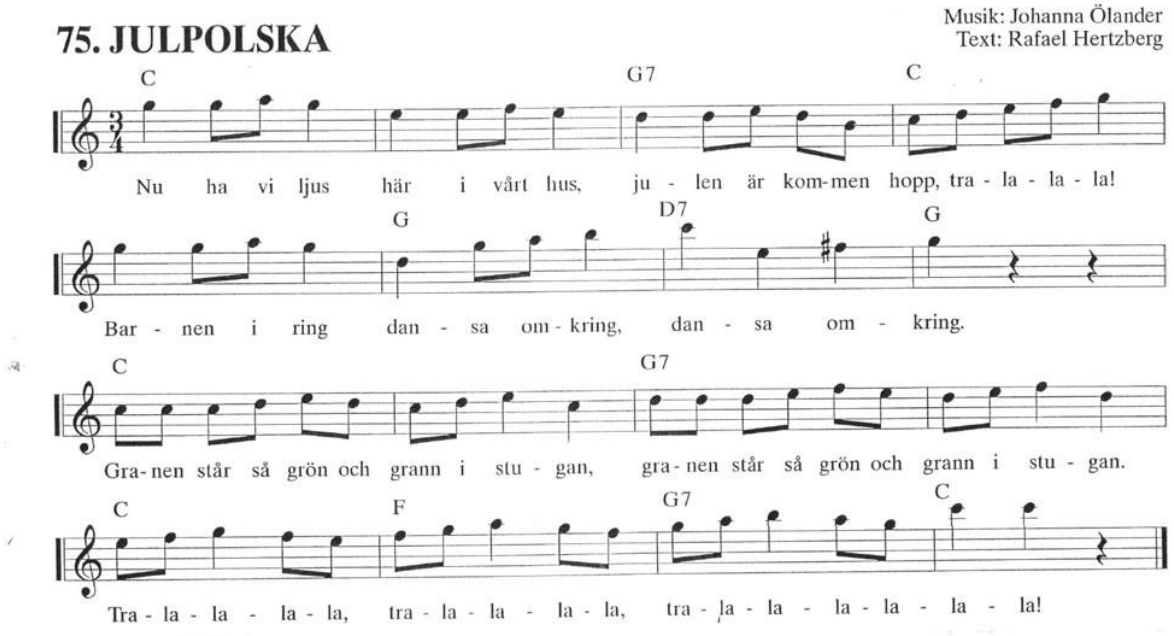
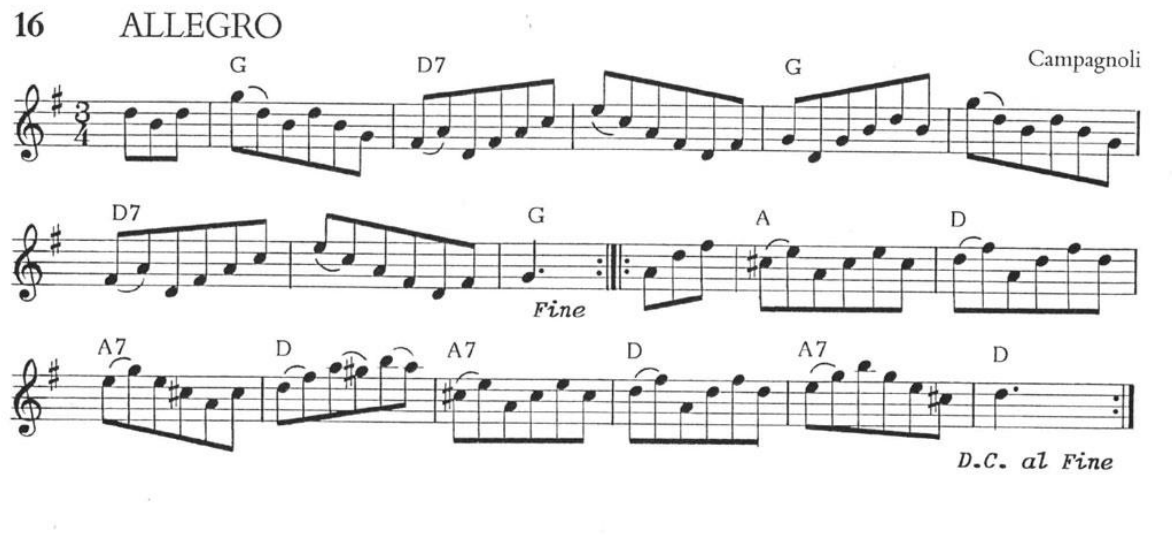


Figure 22 Test image 2

Output string – Note sheet 2

Result	G3g3a3g3E3e3F3e3D3d3e3D3B2d3e3f3g3nG3G3A3g3D3G3A3b3C4E3F3G3nC3c3e3d3c3d3E3C3d3d3d3E3f3E3D3e3F3D3ne3f3G3f3e3f3g3A3g3G3A3B3a3g3C4
Correct answer	G3g3a3G3E3e3f3E3D3d3E3d3b2c3d3e3f3G3nG3g3a3G3D3g3a3B3C4E3F3G3nc3c3c3d3e3d3c3d3E3C3d3d3d3e3f3e3d3e3F3D3ne3f3G3f3e3f3g3A3g3f3g3a3B3a3g3C4C4

Note Sheet 3



Figur 23 Test Image 3

Output string – Note sheet 3

Result	d3b2d3g3d3b2d3b2g2f2a2d2f2c3A2E3C3A2F2D2F2g2d2g2b2d3b2g3d3b2d3b2g2nf2a2d2f2a2c3e3C3A2F2D2f2g2a2d3f3c3e3a2c3e3c3d3f3a2d3f3d3ne3g3e3c3c3A2f3a3g3D3a3c3e3a2c3c3E3e3g3b3e3G3D3
Correct answer	d3b2d3g3d3b2d3b2g2f2a2d2f2a2c3e3c3a2f2d2f2g2d2g2b2d3b2g3d3b2d3b2g2nf2a2d2f2a2c3e3c3a2f2d2f2G2a2d3f3c3e3a2c3e3c3d3f3a2d3f3d3ne3g3e3c3a2c3d3f3a3g3b3a3c3e3a2c3e3c3d3f3a2d3f3d3e3g3b3g3e3c3D3

4 Discussion

This section will discuss the results of the OMR system and some alternative approaches and possible improvements.

4.1 Results

The result has a varying range of success when comparing the output of the system with the actual note sheet. The OMR can get a fully correct result as seen in Note Sheet 1, however it may also get a very unsuccessful result in both pitch and value as in Note Sheet 3.

The system's biggest flaw is its inconsistency. It occasionally misses a note or adds one that should not be there, represents chords were the notes are connected as single notes, and sometimes it determines the wrong pitch. If a pitch is wrongfully declared it might be due to removal of data from the notes which in turn adjusts the centre of the note when comparing to the pitch's reference points. Another example is that two notes with the same pitch, value and potentially similar order in a sequence can give different outcomes. These errors may also be caused by the misclassifications of the note values. All these mentioned faults are not the cause of a random factor but rather incorrect handled situations that results in accumulation of problems throughout the process.

The accuracy of the result can be hard to measure because it depends on what is measured. It can be the number of note matches, correct pitch or note classification value. For a good implementation all of these are wanted. However all the mentioned aspects can vary for different note sheets, meaning that sometimes an implementation can get all the notes right and in correct pitch but for an other image it might get everything wrong. There is no simple way to compare two implementations were one sometimes gets a high accuracy result and sometimes very low accuracy result and the other implementation has a stable average result.

4.2 Alternative approaches and improvements

Template matching was helpful for finding objects but it took a lot of trial and error to find parameters (size, thresholding) with good enough results. It was difficult to find a template that worked for most of the images when every image had its own shape and size of notes and g-clefs. Removing all g-clefs with template matching seemed to be impossible, even with three different templates of g-clefs and therefore cropping was implemented instead.

To improve the stability of the system a database of templates could have been created. Instead of choosing a single notehead as the template, several noteheads could have been chosen, for example one for each training image. The matching could then be done by using the nearest neighbour method which has been proven to give good results when it comes to

OMR [1]. Notes with flags could have been found by using the same method and that could be used to increase the accuracy of the classification for notes with flags.

The current classification of note values is based on the peaks. This method can be altered to instead looking at how many edges there are or how often it changes sign over a selected area. Either of the peaks, edges or sign change could been improved if the area looked at was selected in a way that it only contained relevant information. The area could have also been manipulated before decisions were made on it. Such manipulation could be to remove smaller unconnected areas or decrease the thickness of all lines in the area so that peaks, edges and sign changes would be more distinct.

5 Conclusion

Some notes disappear due to problems with image quality, processing and template matching. Recognition and classification of fourth and eighth notes also fails at times for the same reason, but mostly because of processing. With the improvements and different approaches mentioned in the discussion a better result could be obtained. Overall the result capture the majority of notes and determine the correct pitch of the found notes and therefore the goal of the project has been fulfilled.

References

1. *Optical recognition of music symbols - a comparative study*. Rebelo, A. Capela, G. Cardoso, Jaime S. Springer-Verlag, 2009.
2. *Scientific Pitch Notation.* Wikipedia. December 2018. [online] Available: https://en.wikipedia.org/wiki/Scientific_pitch_notation [Accessed: 10 December 2018].
3. *The Challenge of Optical Music Recognition*. Bainbridge, David. Bell, Tim. Computer and the Humanities 35: 95-121, 2001.
4. *Hough Transform*. Mathworks. [Accessed: 19 November 2018] https://se.mathworks.com/help/images/hough-transform.html#buh9ylp-26