

Optical Music Recognition of handwritten scores using structural pattern recognition

Bachelor project

Report by: Tom Doesburg, s1581791, t.doesburg@student.rug.nl

Project realized in collaboration with Michiel Bergmans, under the supervision of Marius Bulacu

Abstract: Optical music recognition (OMR) is a form of optical character recognition (OCR). Instead of automatic recognition of text, like OCR, OMR focuses on the recognition of musical scores. Commercial software that performs OMR on machine printed scores already exists and produces great results. When the same software gets handwritten music as input, its performance drops dramatically. In this report we explore the recognition of handwritten musical scores. With the use of connected components, structural- and statistical pattern recognition to deal with the variability of handwritten musical symbols we obtained a total precision of 96.2% and a recall of 85.1%. This article focuses on the structural recognition of the music, while the article of Michiel Bergmans (2008) focuses on the statistical part. Furthermore some issues concerning the use of connected components will be discussed as well as the poor recognition of some symbols.

1. Introduction

1.1 Optical Music Recognition in Artificial Intelligence

Artificial intelligence (AI) is a field that is focused on understanding and building intelligent entities (agents). AI can be divided in different research fields. On the one hand, some fields focus on the 'human' part, like modelling brain functions with the help of neuroscience, psychology and linguistics. On the other hand, there are very practical uses of AI, like building robots and intelligent computer programs for solving specific problems, using logic, mathematics and computer engineering.

The aim of the research project was to build such an intelligent agent. A basic intelligent agent consists roughly out of the following three parts: a perception part, a reasoning part and some actuators. The optical music recognition software we developed is

focused on the first two parts. First we worked on the perception part, the reading of symbols from a scanned image. Secondly, we developed on the reasoning part, using logic and shape information to decide how a symbol should be classified. After recognition, we used an existing program to render the results as musical scores that were compared with the original handwritten input. This corresponds, roughly, to the actuators.

1.2 Earlier research

Optical music recognition (OMR) is a form of optical character recognition (OCR). OMR has a big advantage compared to typical OCR, because it is bounded by a lot more rules. Rules concerning position, size and the low number of available symbols are constraining the possible classifications of symbols. There has already been done a lot of research on OMR on machine printed music. Rossant and Bloch (2002, 2004, 2007) for example have



Figure 1: Example of an input image

used template matching for recognizing machine printed music and this approach has proved to be successful. There are multiple solutions to the problem of recognizing musical scores, Bainbridge and Bell (2003) for example used graph-search and grammars for OMR. The results, from these studies and others, get up to 99 percent recognition rates. There is already commercial software like Photoscore

(<http://www.neuratron.com/photoscore.htm>) that gives similar results. However these programs fail on handwritten pages of music. Figure 1 shows an example of input to our program. Programs like Photoscore work well on machine printed music because, in machine printed music, the notes and other symbols are printed exactly the same within a certain typeset. Unfortunately this is not the case with the handwritten music. Each handwritten symbol is written with great variety in shape and size across the pages. Figure 2 shows some example of handwritten musical symbols and makes clear the shape variability. Because of this great variety, there are currently no programs that can deal properly with the offline recognition of handwritten music. Also little or no research has been done



Figure 2: Shape variation within a class

on the offline recognition of handwritten music. There has been some research on online recognition, but this concerns writing with a digital pen on preset staves. Also the online writing gives a lot of information on the shape of a symbol during the writing process, information that is not available in the offline variant. The thesis of Taubman (2005) is an example of online recognition of handwritten music.

1.3 Classified symbols

The symbols we attempt to classify are listed in Figure 3. These symbols are machine printed for clarity reasons. Table 1 shows the distinction we made between structurally classified symbols and statistically classified symbols.








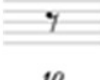










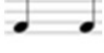
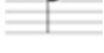

Notes		Rests	
	Stave		Whole or semibreve
	G-clef or treble clef		Half or minim
	F-clef or bass clef		Quarter or crochet
	C-clef or alto clef		Eight or quaver
	Bar line		Multiple measures
	Whole or semibreve		Legato
	Half or minim		Flat
	Quarter or crochet		Sharp
	Eight or quaver		Natural
	Beamed notes		Staccato dot
			Duration dot

Figure 3: Different classified musical symbols

Type of symbol	Structurally classified	Statistically classified
Notes	Stem Flag Beam Open note head*	Closed note head Open note head* Open note head**
Rests	Multiple measure rest Semibreve (whole rest) Minim (1/2 rest)	Crochet (1/4 rest) Quaver (1/8 rest)
Accidentals		Flat Sharp Natural
Clefs	Treble clef (g-clef)	Bass clef (f-clef) Alto-clef (c-clef)
Other musical symbols	Bar line Staccato dot Duration dot Slur	
Non-musical symbols		Letters*** Digits***

* When the note head is made up out of two separate halves

** When the note head consist of a single coco

*** Classified, but not processed

Table 1: Structurally vs. Statistically classified symbols

1.4 The optical music recognizer

In this project, that has been done in cooperation with Michiel Bergmans, we used, two recognition approaches. On the one hand we used structural pattern recognition using logic, as described in the book of Russel and Norvig (2003). This type of pattern recognition is discussed in this paper. On the other hand we used statistical pattern recognition as described in the book of Duda, Hart & Stork. (2001). This statistical part is discussed in the article of Bergmans (2008).

There is a lot of information on a page of music. Most importantly there are the notes that are positioned on a certain height on a stave to indicate the pitch. These notes have durations and so have the rests. Furthermore there are symbols like clefs, flats and sharps that give information about the pitch of the notes following after them. In this research our aim is to use this information to successfully process scanned pages of handwritten music and print them with the program Lilypond (<http://www.lilypond.org>) for visual assessment. We will also create a MIDI file which contains an audible representation of the found music. Structural pattern

recognition is done on symbols that are either very large or very small. These symbols have a simple structure without many details. The advantage of the structural approach is the simplicity and the speed when using it in software. In this paper only the recognition of structural classified symbols is described. For all other symbols we use statistical pattern recognition that is described in the article of Bergmans (2008). By using statistical classification, one is able to deal with shape variation and still produce good results, while structural classification fails in this case. The technique is more computation-intensive than the structural variant and it is suitable for symbols with shape details. For simple and large symbols we prefer structural classification. Furthermore some musical knowledge is used for better recognition. The program was tested with a dataset of 200 staves with handwritten monophonic music. The staves were machine printed and the music was all written by the same writer. We evaluated the software by counting the errors and computing the precision and recall rates per symbol.

2. Method

2.1 Preprocessing the image

Input for our software is a scanned handwritten musical score consisting of several pages. An example is given in Figure 1. Before we can start recognizing musical symbols we need to do some preprocessing, because the images in our dataset contain multiple pages. We have a page by page approach: we have cut the images into separate pages and stored them in memory. After cutting one page, the program processes the page and writes the recognized music to a Lilypond music file.

The Lilypond file is a text file that represents the recognized music in such a way that the Lilypond program is able to build a MIDI file that contains the audible representation of the music on the page.

2.2 Finding staff lines

Before we will be able to detect any music, we need to know where to look for musical symbols. Therefore we have to detect the staff lines in the preprocessed pages and, after we have found them, we need to store their positions in memory. These positions are used

later to determine the position of a symbol on a staff. In this way, for example, we can establish the pitch of a note. When the staff lines are found we remove them from the picture, so we can use a connected components approach that will be discussed later. First we will discuss the method to detect the staff lines and in the second part the removal of the found staff lines.

2.2.1 Detecting staff lines

For the detection of the staff lines we have developed an algorithm that searches for five equally spaced maxima within a preset vertical range. To find these maxima we use projections of horizontal lines. Projections are represented by an array of which the indices are the positions of horizontal lines in the picture. The values in the array are the average gray scale values of a horizontal line within the range. Because the lines tend to be a bit skewed sometimes, we look for the staff lines separately at the left and the right side of the page, see also Figure 4. We use up to a fourth of the page width. To find these 5 starts and 5 ends we use a vertical projection to find the start and the end of the staff lines. When we have made the vertical projections we need



Figure 4: Staff with marked ranges for finding maxima

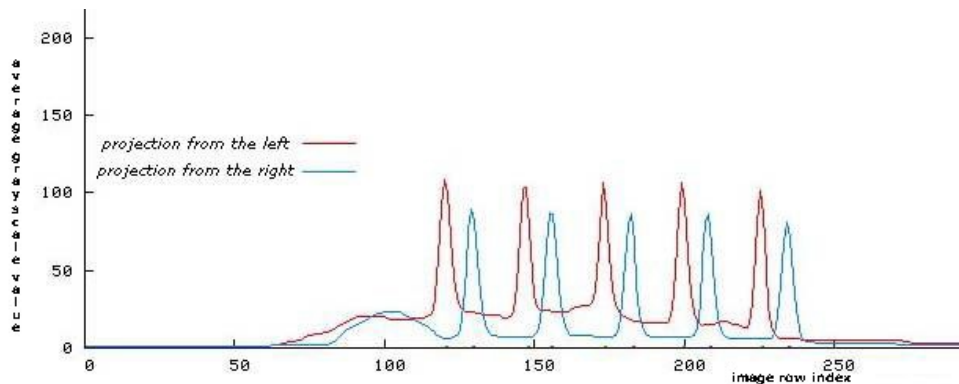


Figure 5: example of the smoothed projections



Figure 6: Found staff lines are drawn with red on the original picture

to find a very steep slope in the projection. The steep slope indicates a start or an end. Then we create an horizontal projection. After creating the horizontal projections we need to find the maxima.

To avoid finding noise, we smooth the projections using a 3-point averager. In Figure 5, the average gray scale value is plotted against the image row index. The five red peaks in Figure 5 represent the starting points of the staff lines, the five blue peaks represent the ends. The blue peaks are positioned more to the right than the red peaks, this reveals that the staff lines are a bit skewed. The found maxima are used to make a representation of the different staff lines. A straight line is calculated from the first peak in the left projection to the first peak in the right projection. Figure 6 shows the result when drawing the calculated line. After finding and storing the staff lines in memory, we need to remove them from the image.

2.2.2 Removing staff lines

When the staff lines are found it is quite simple to remove them. Using a vertical run-length while going over the found staff lines enables us to remove the staff lines completely without damaging the objects on the staves. Figure 7 illustrates this process. If the vertical run-lengths within the ranges are shorter than a set length they will be removed, if they are too long they won't be removed. We also take into account that it is possible to have one or two white pixels above or below before there are more black pixels. This approach makes sure that the objects stay intact and only the staff lines are removed. The result of this removing process is shown in Figure 8.

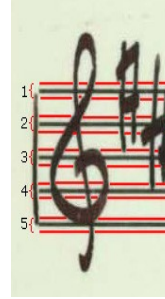


Figure 7: Using vertical runlength to remove lines



Figure 8: Objects are preserved after removing staff lines

2.3 Detecting connected components

After removing the staff lines from the image, what is left are meaningful musical tokens and some garbage. To find and use the different symbols on the page we use the connected components algorithm. This works as follows. For every black pixel in the image all adjacent black pixels get the same id. All the white



Figure 9: All connected components get a different id number

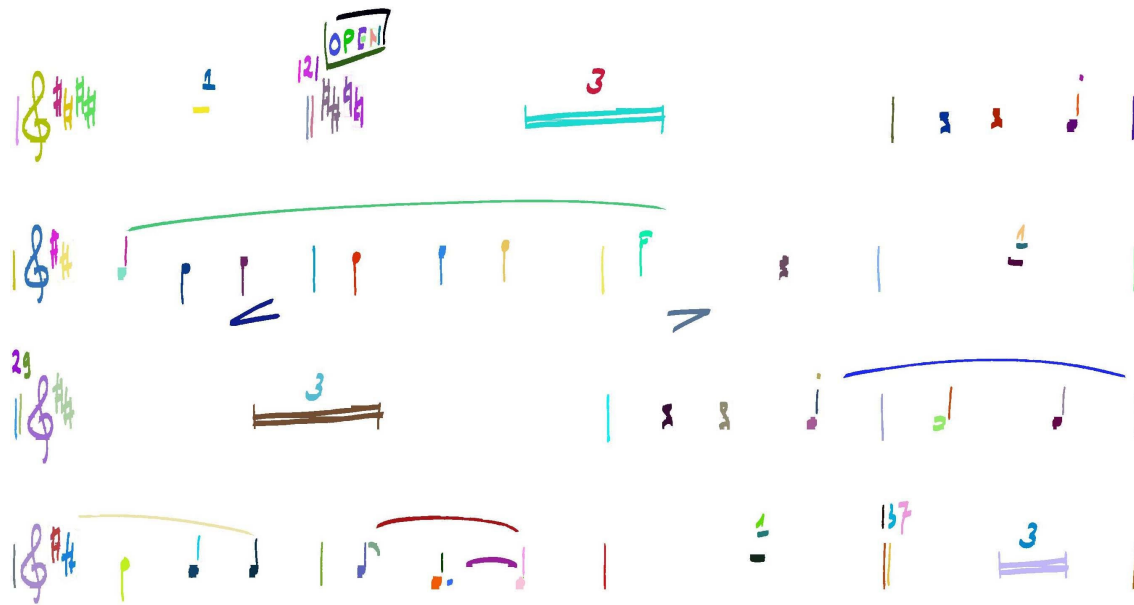


Figure 10: Connected components obtained after removing the staff lines

pixels are set to zeros. Figure 9 shows an example of this approach where for clarity reasons the tokens are colored green and red instead of black. As is visible in Figure 9, this leads to clusters of pixels with the same value. Most of these connected components contain musical symbols or parts of them.

When the connected components (cocos) are found we have an image that is similar to Figure 10. In Figure 10 the different connected components are visible and colored to illustrate them. The connected components are stored in memory with their top, bottom, left and right positions in the image. This is similar to drawing a box around the connected component. From this point on it is possible to classify a connected component, or a group of connected components as a musical symbol.

2.4 Recognition of connected components

The found connected components enable us to classify a connected component or a group of connected components as a symbol. The different symbols discussed in the following paragraphs are the only symbols that are

classified with structural pattern recognition. For other symbols recognized using statistical methods please read the paper by Bergmans (2008) on this subject.

2.4.1 The treble clef

The treble or 'G' clef is shown in Figure 11. This key is classified according to its size and position. First of all it is important that the symbol is positioned on a stave. The top of the symbol must lie above the stave and the bottom must lie below the stave. Also the width is important. If it is too narrow it could be a bar line for example. So we set the desired

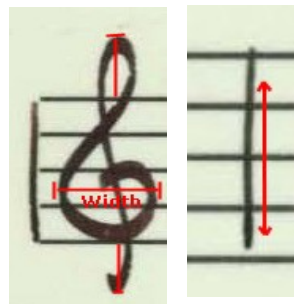


Figure 11: Left: Treble Clef, right: Bar line

width between 50 and 80 pixels. These constraints are sufficient to successfully detect the treble clefs.

2.4.2 Bar lines

The bar lines are simple approximately vertical lines. The important factor here is the position and the height of the lines. The line must be on a staff and the height must be approximately the same as the stave height. It is important not to be too strict about this because the writer varies the height a lot. Therefore the top must be higher than the height of the first staff line plus 15 pixels and the bottom lower than the fifth staff line minus 15. This is shown in Figure 11 with the red double arrow. The width of the imaginary box drawn around the bar must be smaller than 15 pixels and bigger than 3 pixels to be classified as a bar line.

2.4.3 Whole and half measure rests

The whole and half measure rests shown in Figure 12 are fairly small objects for structural pattern recognition in comparison to treble clefs and bar lines, but because of the restriction of being small and rectangular it is possible to detect these without the use of statistical pattern recognition. Furthermore the position of the whole and half measure rests are of crucial importance for the classification. A small connected component with a rectangular shape that lies between the first and the third staff line could be a whole rest. If



Figure 12: On the left: Whole rest or semibreve and on the right: Half rest or minim

the middle of the connected component is closest to the second line it will be classified as a whole rest. For the half rest a similar method is used, but in this case we check if a connected component lies between the second and fourth staff line and the closest staff line should be the third staff line.

2.4.4 Multi-measure rests

A multi-measure rest is shown in Figure 13. The big rectangular shape is easily recognized because of its size. Multi-measure rests are always positioned on staves. And they mainly consist out of two semi-horizontal parallel lines. To recognize these symbols we check for big connected components that lie on a stave. Then we check the middle of the connected component (see the red line in Figure 13) for the parallel horizontal lines. If there are black pixels followed by white pixels and again followed by black pixels it will be classified as a multi-measure rest. We have chosen to ignore the numbers above the multi-measure rests that indicate the number of measures to rest, because of the time consuming and fairly complicated job it will be to recognize them. This could be an interesting item for further improvements.

2.4.5 Legatos

The writer of our dataset always puts a legato (Figure 14) above the staves. Because of the big size and fixed position of the legatos it is fairly simple to recognize one. We check the sides of the connected component for square angles. This is done because the, in our dataset rarely seen, Volta-brackets are also large and

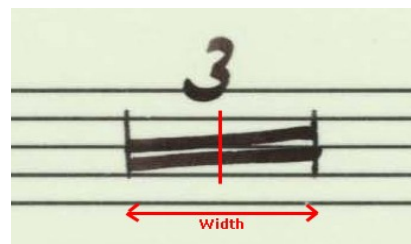


Figure 13: Multi-measure rest



Figure 14: Legato and Volta bracket

above a staff. For an example see Figure 14. The big difference however is the square angle on the left and sometimes on the right as well. So if the sides do not contain square angles, the connected component is very wide and it lies completely above the second staff line it is classified as a slur. This is a premature classification, because it is important to know if the ends of the slur are near a note. This issue involves introducing some musical knowledge and the notes must be classified already. This will be further discussed in paragraph 2.5.

2.4.6 Notes

After the structural classification of the symbols we discussed above we remove them from the image, leaving the notes and the other, unclassified symbols. The recognition of notes is done in three stages. We recognize in the following order possible stems, note heads and beams or flags. First we use a horizontal run-length algorithm to find all thin vertical lines in the image and store them in a new image. After running the connected components algorithm on the new image, we remove all the small garbage, while leaving the longer lines. In our images the cocos that are possible stems have a height between 25 and 130 pixels. Where the lines are found, we remove them from the original image (see Figure 15) and this leaves the note heads and the flags intact.



Figure 15: Notes and stems. Possible stems found by algorithm are colored gray



**Figure 16:
Broken
open note
head**

The closed note heads and the intact open note heads are classified by a statistical classifier that is discussed in the article of Bergmans (2008). The open note heads that consists of a top part and a bottom part, see Figure 16, are classified in a structural way. If we encounter two small pen strokes that are roughly vertical aligned and very close to each other, then we merge them to an open note head. This combined picture is then classified statistically.

If the closed note heads have a stem directly above or below them, the head and the stem are merged to a note with a duration of $\frac{1}{4}$. An open note head does not have to have a stem. So if a stem is found, the head and the stem are merged and the duration is set to $\frac{1}{2}$. If no stem is found, then the duration is set to 1.

The whole and half notes are now fully classified and stored. For quarter notes there is the possibility of a flag or beam on the end of the stem. We check for parts of cocos in

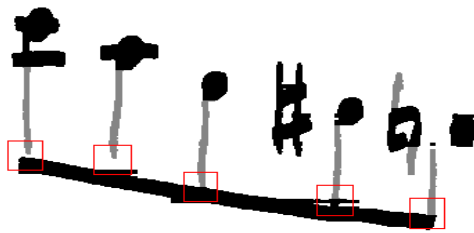


Figure 17: Detecting beams in the red box

a box of 30 by 30 pixels. If a connected component is found and if the size is large enough (wider than 20 pixels, higher than 8 pixels), see Figure 17, the duration of the notes with the flag or the same beam is set to $\frac{1}{8}$ and the notes are merged with the beams. If different notes have the same beam, then they are connected notes. Notes with more than one beam or flag do exist, but in this project we chose for a limitation of a single flag or beam. Notes with multiple beams or flags are classified as eighth notes. In fact this is incorrect, but for now it is sufficient to show that the basic principle is working. The notes are now fully classified. The classified notes are now like the structural classified symbols removed from the image.

The rest of the image is classified using template matching. A nearest-neighbor classifier is built by collecting and labeling a reference dataset of patterns. More details are found in Bergmans (2008).

When the statistical classification is finished we need to check for some symbols in the vicinity of the notes, like legatos, staccatos and duration dots. Also the pitch of the notes is still not found, we will discuss this in the next paragraph.

2.5 Introducing musical knowledge

After symbols are classified by their size, shape and position, we use some musical knowledge to detect the pitch of notes, staccato dots, notes that belong to legatos and duration dots. In the following paragraphs we

will discuss the different types of knowledge used for the extraction of this information.

2.5.1 Assessing pitch

To assess the pitch of a note, we have three sources of information: the key, the height of the note head and the detected staves. In music scores the key assigns a pitch to each staff line and the gaps between them. When the key is correctly classified, we are able to use the information of the height of the note head, or in other words the position of the note head on the staff. For closed note heads we simply use the middle of the note head as the height. The middle is found by looking for the highest density in the note head. For open note heads we use the middle of the empty white middle portion of the head as the height. When we compare the height and horizontal position of the head with that same point on the staves, we find the pitch that is dictated by the clef.

2.5.2 Staccato dots

Staccato dots are little dots that are positioned directly above or below a note head. The easiest way to detect these dots is not just to find them, but to look for them only directly above or below a note head. Using the knowledge of the restricted position of a staccato dot it is possible to distinguish staccato dots from duration dots, but also from ordinary noise in the image.

2.5.3 Duration dots

Duration dots are found very similar to the staccato dots. The difference here is that we are not looking directly above or below the note head, but directly at the right of it. Again we only use the position in relation to the note head.

2.5.4 Legato starts and -ends

Legatos are meaningful when the start and the end of the legato is placed above a note.

Because of this restriction the legatos are relatively roughly classified. Then we use this musical knowledge to look for the notes that the legatos are connecting to. At the start and at the end of the legatos we look for the closest note head on the same stave as the legatos using the middle of the note head. The calculated distance is the horizontal distance. The closest note at the start and the closest note at the end are connected by the legatos.

3. Results

To check the performance of our algorithms we have tested the program on 200 staves. We selected staves that contain monophonic music. The selected staves are all non-piano staves. We selected staves with different clefs and staves with note heads that are not more than one extended staff line above or below the standard stave. In short, we have selected staves that contain symbols our program should be able to recognize. The output of the program is compared to the original image of the stave by hand, ignoring the indication of the key in which the music is written, because the program is not yet equipped to deal with that kind of information. The results of the symbols that are (partly) structurally classified and discussed in the method section are found in Table 2. The results for the statistical part are found in appendix A and in the article of Bergmans (2008). See below for a legend of the different terms. The results in Table 3 show the number of complete staves that the program failed to remove, the number of staves where not all five of the lines were removed and the total amount of staves tested.

Hit: *number of correct classifications*

Miss: *number of missed symbols*

FA: *false alarm, symbols assigned wrongly to class*

OK: *the start and end of a slur correctly found*

Dur: *number of symbols with the correct class and correct duration.*

Pitch: *number of symbols with the correct class and correct pitch*

Total: *total of all symbols, structurally and statistically classified*

Relevant: *total symbols of class on paper, hit + missed*

Retrieved: *total symbols of class found, hit + FA*

Precision: *hits divided by the number of retrieved symbols of class*

Recall: *hits divided by the number of relevant symbols of class*

G_clef	Hit	82	Relevant	87	Precision	95.3%
	Miss	5	Retrieved	86	Recall	94.3%
	FA	4				
Bar line	Hit	848	Relevant	1039	Precision	99.9%
	Miss	191	Retrieved	849	Recall	81.6%
	FA	1				
Slur	Hit	137	Relevant	170	Precision	98.6%
	Miss	33	Retrieved	139	Recall	80.6%
	FA	2				
	OK	104				
Measure Rest	Hit	60	Relevant	71	Precision	98.4%
	Miss	11	Retrieved	61	Recall	84.5%
	FA	1				
Whole rest	Hit	14	Relevant	15	Precision	53.8%
	Miss	1	Retrieved	26	Recall	93.3%
	FA	12				
Half rest	Hit	8	Relevant	9	Precision	44.4%
	Miss	1	Retrieved	18	Recall	88.9%
	FA	10				
Staccato Dot	Hit	9	Relevant	18	Precision	75.0%
	Miss	9	Retrieved	12	Recall	50.0%
	FA	3				
Duration Dot	Hit	223	Relevant	236	Precision	96.1%
	Miss	13	Retrieved	232	Recall	94.5%
	FA	9				
Open note	Hit	303	Relevant	384	Precision	95.6%
	Miss	81	Retrieved	317	Recall	78.9%
	Dur	300			Perc. Dur	99.0%
	Pitch	290			Perc. Pitch	95.7%
	FA	14				
Closed note	Hit	756	Relevant	826	Precision	94.5%
	Miss	70	Retrieved	800	Recall	91.5%
	Dur	734			Perc. Dur	97.1%
	Pitch	693			Perc. Pitch	91.7%
	FA	44				
Total	Hit	2822	Total relevant	3318	Total Precision	96.2%
			Total retrieved	2934	Total Recall	85.1%

Table 2: Results for structural pattern recognition

nr. staves not found	2
nr. lines not found	5
total staves	202
total lines	1000

Table 3: Results for staves

4. Discussion

The results shown in section 3 show the performance of the program on the 200 staves it was presented with. In general the program had a very high precision of 96.2%. This can be attributed to the fact that the most common symbols were very well recognized and the poor recognized symbols were fairly rare. The overall recall rate of 85.1% is considered to be pretty good. A big portion of the missed figures was caused by a fundamental problem of the connected component approach. When two different symbols become a single coco because they overlap, they will often be missed or falsely classified. An example shown in Figure 18 shows a bar line, that is missed because it is intersected by a slur. This intersection causes the creation of a large coco that is not recognized by the program. By solving problems caused by this phenomenon it is possible to improve the recall rate significantly. Another big factor causing the missing of symbols is premature commitment to a certain classification. By removing classified symbols from the image, the program commits to a certain classification which can lead to errors. For example, a separated open note head on a staff line on which whole and half rests are found as well, is partly removed when classified as a rest. This makes it impossible to correctly classify the open note in Figure 19, in which the top half of the note head is classified as a half rest. By not removing some classified symbols and delaying the commitment to a classification, it may be possible to detect the open note head and to discard the earlier classification of a half rest. Some objects like G-clefs can still be removed because of the already very good classification. If we do not remove them, some of them have a part that is classified as a note. This shows that both approaches have their own advantages and disadvantages.



Figure 18: Slur intersects a bar line



Figure 19: Premature commitment

The recognition of the staves was very good, only 2 out of 202 staves were missed. Both times this was a result of finding too many line starts or -ends. However, with a recall of 99.0% percent we can say, that the staff finding algorithm works very well. The precision was also very high, when a staff was found only 5 of the 1000 lines were not removed. This means that 99.5% of the found staff lines were successfully removed.

In the next paragraphs we will discuss the individual results and class restricted errors of the structural recognized symbols. The statistical results are discussed by Bergmans (2008).

The G-clefs were found almost every time, with a few exceptions. These clefs were missed because of the intersecting symbols (Figure 18). Fortunately this happened only on a few occasions.

Bar lines were also found well, some of the errors in finding bar lines were due to the large size of some of the bar lines and again the intersecting of symbols has played its part. With some looser constraints and a solution for the connected component issue, the program will be able to find the lines almost perfectly.

Measure rests have a really good precision, but their recall may be improved. Some measure rests were missed when the horizontal lines stuck together in the middle (Figure 20). If the existing constraints of

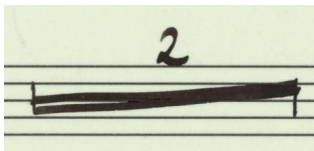


Figure 20: Slur with joined horizontal lines

finding the open part in the middle are loosened by checking at multiple points in the symbol, it will be possible to improve the recall rate.

The problem of the whole and half rests has already been explained at the beginning of this chapter. Instead of committing to a classification later on in the program, it is also possible to check a possible whole or half rest for neighboring connected components and avoid premature classification in this way. By not classifying possible rests straight away, the precision of the rest will improve radically. This will improve the recall of the open notes as well.

Staccato dots were missed often because of bad positioning by the writer. Officially a staccato dot is put above or below the note head. The writer of the sheet music we used varies this, sometimes he puts them incorrectly above or below the end of the stem instead of close to the note head. It is possible to find these dots, by also checking the ends of the stems. This is a choice between sticking to the rules and expect writers to do so too or adapting to writer habits even if they are incorrect.

Duration dots were found very well. This is because the position of the duration dot is always the same. By using this musical knowledge it is possible to get a very good precision and recall.

The precision of open notes was very good, but the recall could be improved. This has already been discussed earlier in this chapter. A few open note heads that lay above or below the staves and have an extra line

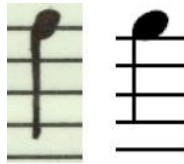


Figure 21: Pitch error

indication like the first note in Figure 16 were missed. This may be improved by giving more examples to the statistical classifier. The duration and the pitch were almost perfect and it will be hard to improve that.

Closed notes were better classified than the open notes. Some common errors were false classifications of written comments on the staves. Furthermore there were some pitch errors. These errors were fairly small, for an illustration of this see Figure 21. The same problem with closed notes above and below the staves were seen as with open notes. These errors can be improved with more examples for the statistical classifier, but keep in mind that the errors were not very common. There is a lot of room for improvement of the program. In the restricted test we performed the results were fairly good. However by dealing with multiple beams and flags, notes that lie a long way above or below the staves, polyphonic music, time restrictions and registering in which key a piece of music is, it is possible to use a lot more musical knowledge and improve the results and robustness of the program.

5. Conclusion

In this research we have tried to build a perceptive, reasoning agent to classify handwritten music scores. We have focused on image processing and pattern recognition. We have done this by using connected components and two types of pattern classification. Structural classification for large symbols or symbols with little shape variation and statistical classification for relatively small symbols with a lot of shape variation. We have used projections of the image to detect and remove staff lines. The removal of staff lines is essential to be able to use the connected component algorithm. The staff detecting and removing algorithm is able to deal with lines that are a bit skewed. After the removal of the staff lines we were left with an

image without staff lines but with all the musical symbols. These tokens were turned into objects by using the connected components algorithm and after that they were put into a structural classifier.

The structural classifier used logic to be able to reason about shape and positional information to classify some of the large and simple symbols. The classified objects were then removed from the original image. In some cases this led to premature commitment and caused some of the errors in the classification of open notes. The connected components approach also led to the missing of symbols that were intersecting. To find notes in the image we removed possible stems from the image and used the statistical classifier to detect the note heads and stems left in the image. This approach led to very high precision (open notes 95.6%, closed notes 94.5%), but because of the removal of the detected notes this also led to the missing of some symbols (recall open notes 78.9%).

The problem of the premature commitment still needs to be addressed. Not removing classified symbols will, in the current implementation, lead to multiple classifications of a single symbol. This problem could presumably be solved by delaying final classification and using a measure of confidence in a certain classification.

The detection of pitch and duration of the notes was very good, but the duration of detectable notes was limited from whole to eighth notes, classifying shorter durations as eighth notes. Using musical knowledge enables the program to detect pitch, to distinguish meaningful dots from meaningless dots and to detect notes with legatos.

The program was tested on a total of 202 staves that were selected by hand. The results (overall precision 96.2%, overall recall 85.1%) showed that the limited program was able to perform well on the symbols it was

designed to deal with. Noting the fact that this program is not complete yet, it is not able to classify all possible musical symbols.

The results show that it is possible to use statistical and structural pattern recognition to recognize most symbols. Future improvement must be concentrated on the symbols that were ignored until now, multiple writers, issues with intersecting symbols and the problem of premature classification. Furthermore the use of musical knowledge was very limited. Presumably it is useful to use more knowledge about time constraints and the key in which the music is written to restrict the number of possible classifications. Most of these improvements concern improving the robustness of the current program.

Although a lot of improvement is needed to deal with the discussed limitations of the program, we have shown that the limited perceptual reasoning agent is able to get good results on the offline detection of handwritten music and there seems to be a bright future in to solving the problem.

References

- Bainbridge, D. & Bell, T. (2003). A music notation construction engine for optical music recognition. *Software – Practice and Experience* (33), 173-200.
- Bellini, P., Bruno, I. & Nesi, P. (2007). Assessing optical music recognition tools. *Computer Music Journal* (31), 68-93.
- Bergmans, M (2008). *Optical Music Recognition of handwritten scores, using statistical pattern recognition*. Bachelor's thesis. Department of Artificial Intelligence, University of Groningen

- Dalitz, C., Droettboom, M., Pranzas, B. & Fujinaga, I. (2008). A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (30), 753-766.
- Duda, R., Hart, P. & Stork, D. (2001). *Pattern Classification*. New York: John Wiley & Sons.
- Göcke, R. (2003). Building a system for writer identification on handwritten music scores. In M.H. Hamza (ed.), *Proceedings of the IASTED International Conference on Signal Processing, Pattern Recognition, and Applications SSPRA 2003*, pages 250-255, Rhodes, Greece, 30 June - 3 July 2003. Acta Press, Anaheim, USA.
- Nienhuys, H. & Nieuwenhuizen, J. Retrieved on July 11th, 2008 from <http://lilypond.org/web/about/>
- Rossant, F. (2002). A global method for music symbol recognition in typeset music sheets. *Pattern Recognition Letters* (23), 1129-1141.
- Rossant, F. & Bloch, I. (2004). A fuzzy model for optical recognition of musical scores. *Fuzzy Sets and Systems* (141), 165-201.
- Rossant, F. & Bloch I. (2007). Robust and adaptive OMR system including fuzzy modelling, fusion of musical rules, and possible error detection. *EURASIP Journal on Advances in Signal Processing*, 1-25.
- Russell, S. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Upper Saddle River (NJ): Prentice Hall
- Taubman, G. (2005). *MusicHand: a handwritten music recognition system*. Undergraduate thesis, Brown University.

Appendices

Appendix A

Bass_clef	Hit	62	Relevant	104	Precision	96.9%
	Miss	42	Retrieved	64	Recall	59.6%
	FA	2				
Alto_clef	Hit	8	Relevant	10	Precision	72.7%
	Miss	2	Retrieved	11	Recall	80.0%
	FA	3				
Sharp	Hit	28	Relevant	38	Precision	100.0%
	Miss	10	Retrieved	28	Recall	73.7%
	FA	0				
Flat	Hit	6	Relevant	10	Precision	75.0%
	Miss	4	Retrieved	8	Recall	60.0%
	FA	2				
Natural	Hit	14	Relevant	22	Precision	100.0%
	Miss	8	Retrieved	14	Recall	63.6%
	FA	0				
Crochet	Hit	226	Relevant	240	Precision	98.3%
	Miss	14	Retrieved	230	Recall	94.2%
	FA	4				
Quaver	Hit	38	Relevant	39	Precision	97.4%
	Miss	1	Retrieved	39	Recall	97.4%
	FA	1				
Open note	Hit	303	Relevant	384	Precision	95.6%
	Miss	81	Retrieved	317	Recall	78.9%
	Dur	300			Perc. Dur	99.0%
	Pitch	290			Perc. Pitch	95.7%
	FA	14				
Closed note	Hit	756	Relevant	826	Precision	94.5%
	Miss	70	Retrieved	800	Recall	91.5%
	Dur	734			Perc. Dur	97.1%
	Pitch	693			Perc. Pitch	91.7%
	FA	44				
Total	Hit	2822	Total relevant	3318	Total Precision	96.2%
			Total retrieved	2934	Total Recall	85.1%

Appendix A: Recognition results for the statistical approach (Bergmans (2008))