TNM034 - Optical Music Recognition (OMR)

Linnea Bergman, Emma Broman & Ingela Rossing

Linköping University

December 2018

Abstract This paper describes a project made in the course *TNM034 - Advanced Image Processing* at Linköping University, Sweden, with the aim of creating an algorithm for solving a simplified optical music recognition (OMR) problem. Firstly, the problem is introduced and the limitations explained and then the theory and challenges related to OMR are presented. After that, the method used for solving the problem is described, beginning with the pre-processing followed by segmentation and classification. The result is an OMR system that works reasonably well for scanned images of music scores, but is in need of some improvements and fine-tuning.

Contents

1 Introduction

For centuries, music have been shared and remembered by two traditions; aural transmission and in the form of musical scores, which is written documents of the music. Many old musical works are only available as original manuscripts or as photocopies and are in danger of being lost through the ravages of time. To preserve these works digitalization and a transformation into a machine-readable format is necessary. The process of interpreting scanned, written or photographed music sheets is known as *Optical Music Recognition* (OMR). OMR is an extension of *Optical Character Recognition* (OCR) and interprets sheet music into an editable form. Once captured digitally, the sheet music can be saved into more commonly used and maintainable file formats.

1.1 Aim

The aim of this project is to implement a simplified version of OMR that given an input image containing music scores detects and registers eighth (1/8) and quarter (1/4) notes using methods for image processing and produces an output string based on the result. A requirement given for the implementation is that it shall not require any user interaction to produce the result.

1.2 Limitations

As mentioned in the aim, only 1/8 and 1/4 notes are considered in this OMR implementation. Any other notes or symbols are merely ignored. It is also assumed that each row in the music sheet starts with

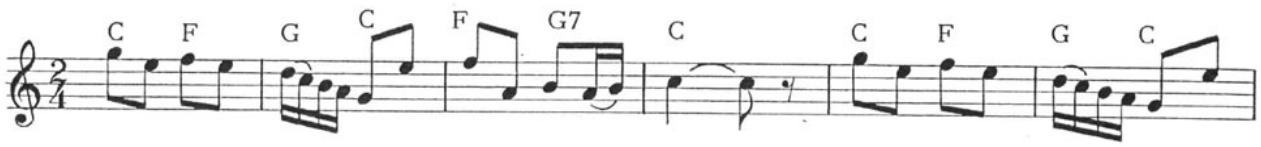


Figure 1: Example of one row of a scanned training image. The correct output of the OMR in this case would be: 'g3e3f3e3g2e3f3a2b2C3c3g3e3f3e3g2e3n'.

a G-clef. All notes are classified only with tones in the major C tone scale. All half pitches are ignored.

Another limitation of this project is that the OMR system is limited to images of scanned quality and not images photographed with a camera, since these require additional pre-processing. Pre-processing of photographed images will be left to the end of the project, and only considered if the system works sufficiently good and there is time over for extra development.

1.3 Output Specification

In the result, each note is encoded using a letter and number giving the pitch of the note. The quarter notes (filled notes without a flag) are encoded using uppercase letters and the eighth notes (filled notes with one flag or bar) are encoded using lowercase letters (see figure 2 and 3). The rows on the score sheet are processed left to right and any double notes are processed separately, from top to bottom. The end of a row on the score sheet is recorded in the output string using the character n. An example of a resulting output string is demonstrated in figure 1.

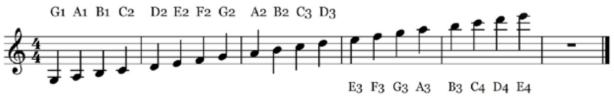


Figure 2: Encoding of the quarter (1/4) notes.

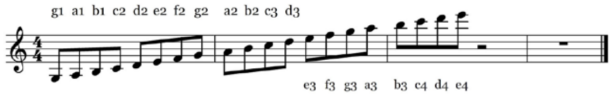


Figure 3: Encoding of the eighth (1/8) notes.

2 Theory

Despite solutions for OMR being commercially available since the 1990's, there still exists no standardised solution. However, most systems follow a similar general approach [1, 2, 3, 4]. First some sort of preprocessing is performed to achieve an image quality that is possible to work with. Even lighting, straight lines etc. can be created using image processing techniques. After the preprocessing there is usually some sort of segmentation to extract the musical symbols of the music sheet. This might include the removal of the staff lines, since these intersect the symbols and aggravates the symbol extraction. After segmentation the symbols are analysed and classified and lastly there is some sort of decision process and output generation.

2.1 Challenges of OMR

OMR is a two-dimensional problem, since the vertical position of a symbol also affects its meaning, e.g. the pitch of a note. This is one of the reasons that add to the complexity of the problem and part of what makes it more challenging than for example OCR, which could be considered as a problem in a one-dimensional domain.

The main challenges of OMR are extensively described by Bainbridge et al. in a paper from 2001 [4], in which the authors highlight the challenges related to the properties of musical features, as well as the steps of the OMR implementation and how to compare the accuracy of different OMR systems. Some of their conclusions will be briefly described in the following sections.

2.1.1 Challenges Related to Properties of Musical Features

For individual music features, a property that cause challenges is as mentioned that the same symbol has different meaning depending on its vertical position. Also, the graphical representation of the same musical feature can vary a lot. For example, the graphical representation of a single 1/8 note is done using a flag, but subsequent 1/8 notes are connected by beams. In addition, these beams might be sheared or stretched. Symbols might also be rotated, or consisting of multiple components. Figure 4 shows examples of properties that adds to the complexity of recognising musical features.

Other properties of the musical features that propose a challenge are related to the layout of the features. Music often rely on complex two-dimensional relationships between the features, i.e. one symbol might affect another depending on its placement on the sheet. For example, a clef affects the pitch of every subsequent note until another clef is encountered. These types of two-dimensional relationships can be

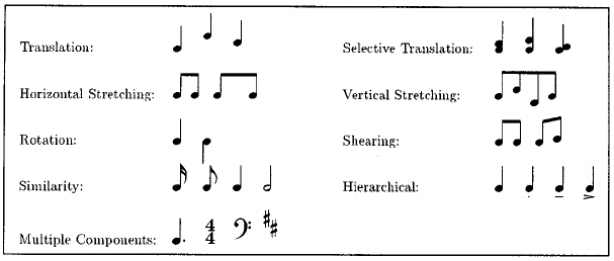


Figure 4: Some properties of music features that contribute to the complexity of the problem.

difficult for a computer to process. The order of the musical shapes can also be written in an ambiguous manner to simplify the layout. In these cases it can be difficult for a computer to determine the order to process the musical features.

2.2 Challenges Related to Implementation and Image Processing

The challenges related to the actual implementation and image processing aspects of OMR will be furthered explained in the method section of this report (see section 3). Challenges can be related to the segmentation of individual objects and how to deal with poor image quality and broken objects. Also, the implementation of the classification of the different objects are far from trivial and thus a challenge in itself.

If the OMR system shall be able to handle photographed images further challenges are introduced. For example, photographed images might suffer from bad lighting conditions or distortions and thus requires a pre-processing step to achieve the quality of a scanned image. Also, the music sheet has to be extracted from the picture and any background removed, since it might complicate the recognition of the music symbols.

2.3 Other Challenges

Apart from implementation and challenges related to the properties of musical symbols, there exist a number of additional challenges in OMR. For example, another challenge is how to compare the solutions of different OMR systems. Bainbridge et al. even claims that this might be the hardest task in OMR [4], since different systems might address varied problems of the field, incompatible sets of music features or different output formats. In addition, a related challenge is to determine how the accuracy of the result should be computed. For example, the output of the OMR system implemented in this project is a string with literals and a challenge that has to be addressed is to compare different results to decide which was more accurate. However, comparing strings is not a trivial task, especially not in the OMR case since different errors, e.g. unregistered or wrongly classified notes, have different impact on the resulting string.

3 Method

The implementation of the OMR system in this project was done in Matlab and can be divided into three different steps: *pre-processing*, *segmentation* and *classification*.

3.1 Pre-processing

First of all, to prepare for further processing, the input image was converted to a grey scale image and binarized using Otsu's method. It was also inverted, so that the background corresponds to the value zero and the objects (notes, staff lines, etc.) to one.

3.1.1 Skew Compensation

An important part of the pre-processing is compensating for possible skewing, i.e. rotating the image so the staff lines are horizontal. This was done using the Hough transform, which detects straight lines in an image. The peaks of the transform was used to find the angle θ, that describes the rotation of the lines as seen in figure 5.

To increase the performance, the Hough transform was applied in two steps: first with a rough resolution for θ and then with a higher resolution around the resulting angle from the first step. Once the angle was found, the original version of the image was rotated accordingly using bicubic interpolation. Worth noting is that after the image was rotated it had to be binarized and inverted once more.

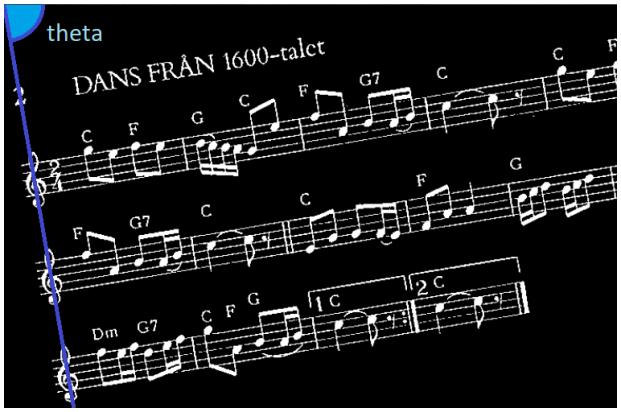


Figure 5: Example of the angle theta that is computed using the Hough transform, shown on a binarized and inverted version of a highly rotated input image.

3.1.2 Compute Relevant Staff Measures

Next, the distance between the staff lines, d, and the line thickness, n, were computed. These were

later used for determining the sizes of the structuring elements used for the morphological operations in the pre-processing, segmentation and classification process.

The computation of the measures was done as described by Bellini et al. in a study from 2001 [5]: the size of the sequences of black and white pixels per column in the image are counted as a function of their dimension. The most occurring values for the white and black pixel sequences respectively give the values for the desired distances, d and n. To be able to account for possible variations and noise a range was computed for each measure, i.e. each measure is represented by two values: $d = [d_{min}, d_{max}]$ and $n = [n_{min}, n_{max}]$. See page four in [5] for a more detailed explanation of how d and n are computed.

3.1.3 Staff Detection

It is worth noticing that finding the correct line indices is crucial for the following steps of the process, which are directly dependant on the line indices being correctly identified. The aim was to get one row index per staff line. By a horizontal projection each staff line results in a higher value. In figure 6 the horizontal projection can be seen, showing the sum of each row in the binary image. Using a threshold around $0.4 * largestValue$ separated the lines, even if the last line sometimes is shorter.

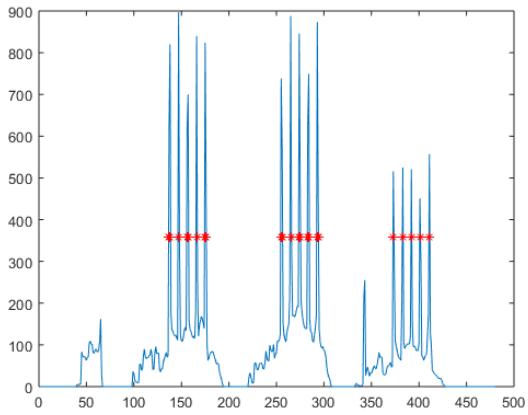


Figure 6: A horisontal projection of "DANS FRÅN 1600-talet" with stars marking each staff line at a suitable threshold-value

This method is however very sensitive to small angle changes, making the maximas from the lines differ less from the rest. If the lines are thicker multiple values could also get through the threshold filter and to find only one center of each staff line the function *findpeak* was used. The built in function returns indices with local maxima, which worked well on most images. Some images with higher resolution got more than one maxima per staff line. One solution was to smoothen using a low pass filter. In the end it was not part of the end result since a threshold matching the whole image was hard to find for the same images. These images had a small angle difference between the lines at the top and bottom of the image.

For later functions to work it became more important to return the correct number of indices than actually giving a very exact position of them. To estimate the correct number and locations the staff groups were found by closing all groups and projecting them. As all groups contain exactly five lines an alternative could be used if the number of previously found lines seemed incorrect. A quite good result was found by just distributing the lines evenly in the found group intervals. This alternative could possibly make the pitch intervals slightly misplaced, but was only needed as a last resort for a few images.

3.1.4 Divide Into Subimages

To simplify later operations, the music sheet was divided into subimages; one per group of five staff lines. All following steps of the OMR process were then done for each subimage individually.

The subimages were extracted by locating the lines (as described in the previous section) and use the result to "cut" the image above and below the staff at a certain distance from the center line of each group, as shown in figure 7. We use a distance that is three times the distance from the first to the third line, which gives enough space to fit any possible note. To avoid cutting outside of the image it is firstly padded in the top and bottom direction using the same distance.

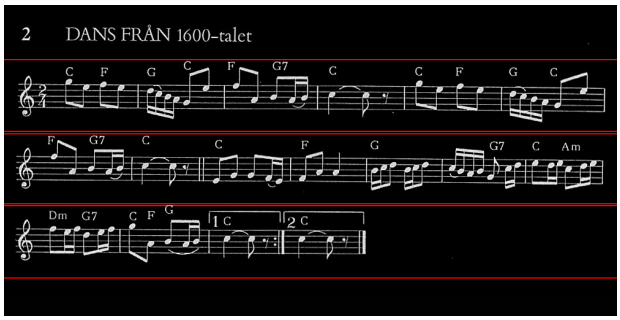


Figure 7: Illustration of how the sub images were created by "cutting" the image a certain distance above and below the center line of each group of staff lines.

An assumption that is made when dividing the music score into subimages is that the staff lines are correctly identified. Thus, if something goes wrong when detecting the lines the division into subimages will fail and since the rest of the process is dependent on the subimages being correctly created the OMR process will be unsuccessful for that music score. The current solution used to address this problem is to try to

guarantee that the number of found staff lines are always divisible by five and matches the number of line groups on the sheet, which seems to work in most cases.

3.1.5 Staff Line Removal

To prepare for the classification and segmentation the music sheet had to be cleared from the horizontal staff lines. The removal consists of two steps; One for removing all lines and one for trying to mend accidentally broken objects. An example of the result is shown in figure 9.

(a)

(b)

Figure 8: Result for one subimage before and after removing lines.

The removal of the lines was done using morphological opening with a vertical line structuring element with a size that is twice the average line width, n_{avg}. In practice this means that the image is processed column by column and all segments that are shorter than the structuring element, i.e. the staff lines, are removed, but longer segments, i.e. the notes and their bars, are kept. We played around with a few different sizes of the structuring element and $2 * n_{avg}$ gave the best result, successfully removing all lines without breaking other objects of importance.

After the lines were removed an attempt to mend possibly broken objects were performed using morphological closing. This was done in two steps: first a general closing to mend broken bars and similar objects, and then and a closing using a vertical line structuring element with the length as an attempt to fix notes with broken stems. This also closed the gap between double beams, which made them easier to remove when isolating the note heads (see section 3.3.2).

3.2 Segmentation

To find separate objects in the binary image the built in function *CC = bwconncomp(BW);*, which uses less memory than the similar function *bwlabel*, was used. Properties, such as area or center off mass, for the individual objects were found using the built in function *STATS = regionprops(CC, properties)*.

Before the segmentation it is important that objects are in one piece and not broken and split into separate objects, since this can lead to them being miss-classified or removed by filtering operations.

At this stage many small uninteresting objects were still left in the image. All notes of interest has relatively large area and a function for removing the smaller objects was made to reduce the number of objects to classify. To remove the small objects, the *'BoundingBox'* property, which returns a rectangle surrounding the object, and *'Area'* was used. These were then used to filter objects based on their area and height, using the thresholds:

$$areaThresh = round(r * r * \pi * 1.8)$$

$$heightThresh = d_{max} * 3$$

where $r = d_{max}/2$ is approximately the radius of a note head. The constant 1.8 was found experimentally, until the result was considered good enough. To base the thresholds on the maximum possible line distance, d_{max} was necessary for removing objects in images with different resolutions.

To remove the objects in the binary image the whole region given by the bounding box was set to black. In figure 9(a) we can see the objects large enough to pass the filtering, with their bounding boxes. The result of the filtering can be seen in figure 9(b).

(a) Bounding boxes surrounding the objects with large area and height

(b) The image after applying the filtering on area and height

Figure 9: Segmentation of interesting objects.

3.3 Classification

The classification of a note is determined both by its pitch and duration, in this case whether it is an 1/8 or 1/4 note. Any other duration was classified as "not interesting" in this project.

3.3.1 Duration Determination

To determine the note duration each object from the segmentation was investigated separately. Firstly, the object to be classified, given by its bounding box, was extracted from the full subimage. The classification was then performed by passing the image through a tree structure of operations and tests.

An algorithm for finding the note heads was applied first to determine the number of heads in the object

(see section 3.3.2 for details about this). An interesting object consists of one or more note heads, possibly connected by bars and depending on how many heads were found slightly different classification algorithms were used. An issue in the beginning of the project was that the heads located to the very left in the image were overlooked by the algorithm (see figure 10(a)). To solve this problem a padding of zeroes (black) was added to the edges of the image, making it easier for the algorithm to find heads close to the edges (see figure 10(b)).

(a) (b)

Figure 10: Result after locating note heads, before (a) and after (b) padding was added. Before padding was added, notes in the edges could sometimes be overlooked.

First a check was performed to determine if the object have one or more note heads. In the latter case it was also checked if the heads are placed on top of each other, by computing the difference between the x-values of the note head positions. If the difference was smaller than the size of a note head then the note heads were considered to be on top of each other.

For objects with one note head, or where the note heads were one the same stem, the following procedure was used to classify the duration. For each note head in the object a smaller bounding box was created positioned in the middle of the object, in a way so that it contained eventual flags. With the use of vertical projection the number of flags could be identified by counting the number of peaks. If the object contained one or no flags then it was saved, otherwise it was ignored.

A similar process was done for objects containing multiple note heads on different horizontal positions, such as subsequent 1/8 and 1/16 notes. These objects contains one or more bars to symbolise the duration. First, a check was made to determine whether or not the bars were positioned over the middle line of the image or not. Depending on the position of the bars the bounding boxes for the different heads were positioned differently. Then, the note heads were removed from the object and both a horizontal projection and vertical projection was computed in order to find the number of bars associated with the note head. The result from the horizontal projection was used to determine if the object had bars at all and the vertical projection was used to determine if the object has one or more bars. If only one bar was discovered then the note was considered otherwise it was ignored. The reason why the vertical projection is needed is because if the bars are skewed or closed together (as in figure 10(b)) the horizontal projection will result in one peak even for notes with two bars. By checking if the mean of the result of the vertical projection is smaller than a threshold representing the thickness of one bar (currently d_{avg} is used for the threshold) then it could be determined if there were multiple bars or not.

A special case that had to be accounted for in the multiple-head situation was that if the second to last note was an 1/8 note, then the last note is as well since they are connected. This was checked manually using an if-statement.

The interesting notes were saved in an array containing structs representing the notes, each given by the position of the note head relative to the full sub image and a string representing the duration. The string *'note4'* was used for 1/4 notes and *'note8'* was used for 1/8 notes.

3.3.2 Finding the Note Heads

Identifying the note heads is currently one of the most important parts of the classification, since whether an object contains a head or not is the foundation for the decision of whether it is interesting or not.

To locate the heads a number of morphological opening operations were first performed to isolate the head in the image and then the centroids of the heads were found using the functions *bwlabel* and *regionprops*. Three different openings were done:

- One with a rectangular structuring element of height $2 * d_{avg}$ and width $d_{avg}/2.5$, to remove thick vertical lines.
- One with a rectangular structuring element of height $d_{avg}/3$ and width $2*d_{avg}$, to remove thick horizontal lines.
- Lastly, one with a disk structuring element with the radius $d_{avg}/3$, to isolate the head.

In addition, morphological thinning was performed twice before the opening using a disk. The reason for this was to try to break thicker objects, like the G-klef, that is not affected by the first two openings. This is also the reason for the small radius of the disk elements, since the size of the note heads get smaller due to the thinning.

3.3.3 Pitch Determination and Output

The list of interesting notes were after the duration classification passed to a function which determines the pitch of each note, based on the vertical position of the note head and the row indices representing the staff lines in the sub image.

To determine the pitch, the possible vertical positions for the pitches were computed using the average line distance and the center line. Based on the center line the pitch positions were generated by the following range:

$$pitchPos = round(centerLine + 9 * pitchDist : \\ -pitchDist : \\ centerLine - 10 * pitchDist),$$

where *centerLine* is the row index for the center staff line and *pitchDist* is half the average distance between the lines. The generated positions are then manually encoded to the correct pitch, initially given by a string containing a lowercase letter and number according to figure 11.

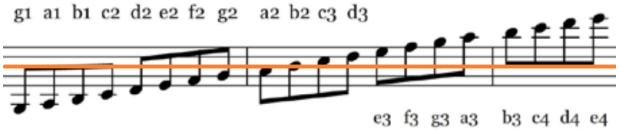


Figure 11: The pitch encoding for 1/8 notes, showing the center line in orange. There are ten possible pitches above the center line and nine below.

To actually determine the pitch of the note head the difference between the vertical position of the head every possible pitch position was computed. Then, the index of the pitch position resulting in the minimum difference was used to select a pitch encoding. As described in the introduction, the output of the system is a string where 1/4 notes are represented by uppercase letters and 1/8 notes by lowercase letters. Thus, after the pitch is determined follows a check whether the note is a 1/4 note, based on the result from the duration classification. If that is the case the pitch encoding is converted to an uppercase letter. Lastly, the encoding is concatenated with the string containing the result.

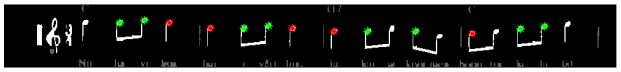
A possible issue related to determining the pitch is that the lines are not always completely straight, but might be slightly distorted. In these cases, the notes to the far left and right of the score may be encoded with a slight offset in the pitch.

4 Results

Most notes are classified correctly. Verification of the duration of chosen objects was easiest done visually. In figure 12 a star marks the notes classified as 1/4 or 1/8 note. The classification of other objects is not considered in this project and other objects should hence not be marked. The objects in grey are filtered away and are not are not considered worth classifying.

Figure 12: A correct result with green stars for 1/8 notes and red for 1/4 notes. Grey objects have been removed by the size-filter.

Broken objects sometimes disappeared in filters making it important to add them in the visualisation. In figure 13 some different errors that are left to fix can be seen.



(a) Small note heads ignored

(b) The 1/8 note in triple-heads ignored

(c) Objects wrongly classified as interesting

Figure 13: Results with various errors

The pitch is in most cases correct and was verified by comparing the output strings with the correct solution. How alike two strings are was determined by reading them. A missed object will make rest of the string displaced, making the simpler character-wise comparison misleading.

5 Discussion

Due to the lack of time compromises had to be made in the end of the project and the solution is far from perfect. In hindsight there are some things that we would have like to do differently, spent more time on or improved. The following sections discusses and analyses the method and results, and covers the desired improvements and future work.

5.1 Results

The solution works well in most cases and based on the limited time available and workload of other courses we are reasonably satisfied with the result.

As mentioned, measuring the accuracy of the result is hard. Visual test were simple to understand and gave most information of what could have gone wrong. The pitch was harder to determine as the resulting string is hard to read and compare well. It is also harder to connect the error to the actual cause. It does however give a possible way to measure the accuracy in numbers given a suitable comparison to solutions.

In this limited project it was not possible to see if the duration matched the score. Combined all notes separated by the thin vertical lines have the same duration, which can be used for testing in full OMR solutions. This gives hints about ignored, miss-classified or extra elements and further investigations can be done in the classification. This solution has no way of knowing how unstable it is without the given solution. Even the possibility that not a single note should be printed to the result can be excluded. A lot of things could go wrong due to all dependencies in the code without a single way of testing the outputs credibility.

5.2 Method

One of the greatest issues is the identification of the staff lines, which has a big impact on later steps. It is also the foundation for the division into sub images, and if this step fails the entire recognition of that music score will fail. In hindsight, It would have been better to implement the sub image creation in a way that is not dependent on the exact staff lines, but rather divide the image using the group intervals and then find the exact line indices in separate sub images. This would further minimise the effect of deficiencies in the subdivison algorithm.

Furthermore, in the current solution the skew compensation is only done for the entire image and not for the subdivided images. It would probably be better to find the perfect rotation for each line group, rather than the full image. Currently, the maximal angle for the full image is used for the rotation, but if this would have been done on each sub image instead and as a result minimise the risk of errors due to misrotation. However, this would mean that the rotation algorithm would have to be applied on several images rather than one and since it is one of the most time consuming steps of the system it could possibly slow down the process significantly.

Morphological operations worked well to close broken objects, but there is always a chance that objects very close by get merged. A musician would easily understand that a # is not part of the note even if they touch, but the algorithm might not. Balancing the kernel sizes for fixing artefacts while still keeping objects separate is difficult as all art pieces differ.

5.2.1 Classification

The classification still requires a lot of tweaking to handle all errors, and is the part of the system that has suffered the most from the lack of time for this project. The variation of typography makes the classification challenging, but a decent result was achieved through quite simple projections and a tree of if-statements. With consideration to the number of different types of elements to classify this would not be recommended for a more comprehensive version where other symbols than 1/4 and 1/8 notes are considered, but it works well enough for this limited case. However, the classification still fails to classify some 1/8 notes and is currently unable to exclude objects based on anything else than the number of note heads or too many bars. This is an issue for example for the G-klef, which is sometimes classified to have a note head is is thus handled like any other note. It might have been useful start the classification with first determining whether the object is a note at all, for example by using template matching or some sort of projection to determine if the object has the desired properties of a note (stem and head). Another possibility could have been to classify the G-klef separately, to be able to exclude it in the final result.

Currently, the classification is highly dependent on the algorithm for finding the note heads working correctly. However, this algorithm is also in need of improvements. First of all, it does not work at all for closely neighbouring note heads that have merged together, since they are removed by the openings removing thick horizontal and vertical lines. Thus, this algorithm is in need of some more work or another approach, for example by first use thinning to try to separate the merged note heads. In addition, it has been very challenging to determine a suitable size for the disk element used in the last opening, since the relative size of the heads varies between music sheets and if the disk element is too large the note heads will be removed by the opening.

5.3 Future Work and Improvements

There are of course a number of improvements we would have liked to make if more time had been available. For example, this project is now limited to scanned images but more time on pre-processing would have been interesting to add so that photos of music pieces could have been classified as well. The same algorithm can be used by making the photos look as scanned images using methods for noise removal, deblurring and geometric and photometric restoration.

Also, as implied by the method part of the discussion there are too many dependencies of previous steps in the process and the classification part is in need of more work. If more time had been available it would have been desireable to further look into the issues of the classification and put more effort into detecting uninteresting objects such as the G-klef, as well as rewriting how sub images are created to minimise risk of these being created incorrectly and ruining the result for a full music score.

6 Conclusion

The result of this project is an OMR system that works reasonably well for scanned images of music scores. However, the solution is not perfect and suffer from some stability and robustness issues. The greatest problem with the current solution is that there are too many dependencies on previous steps. For example, if the creation of the sub images fails then the rest of the process fails as well. In turn, the sub image creation is dependent on the line indices being found correctly. This is something we realised at the end and if there had been more time available we would have liked to reimplement these parts of the system.

References

[1] A. Rebelo, G. Capela and J. S. Cardoso. "Optical recognition of music symbols: A comparative study". *Document Analysis and Recognition*, vol. 12, no. 1, p. 19-31, March 2010. [Online]. Available: `https://www.researchgate.net/publication/220163392_Optical_recognition_of_music_symbols_-_A_comparative_study` [Accessed Dec. 6, 2018]

[2] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marcal, C. Guedes and J. S. Cardoso. "Optical music recognition: state-of-the-art and open issues". *International Journal of Multimedia Information Retrieval*, vol. 1, no. 13, p. 173-190, October 2012. [Online]. Available: `https://link.springer.com/article/10.1007/s13735-012-0004-6` [Accessed Dec. 4, 2018]

[3] F. Rossant. "A global method for music symbol recognition in typeset music sheets". *Pattern Recognition Letters*, vol. 23, no. 10, p. 1a29-1141, August 2002. [Online]. Available: `https://www.researchgate.net/publication/220647622_A_global_method_for_music_symbol_recognition_in_typeset_music_sheets` [Accessed Dec. 6, 2018]

[4] B. Bainbridge and T. Bell. "The Challenge of Optical Music Recognition". *Computers and the Humanities*, vol. 35, no. 2, p. 95-121, May 2001. [Online]. Available: `https://www.researchgate.net/publication/220147775_The_Challenge_of_Optical_Music_Recognition` [Accessed Dec. 5, 2018]

[5] P. Bellini, I. Bruno, P. Nesi. "Optical Music Sheet Segmentation" in *Proc. of the First Intern. Conference on WEB Delivering of Music, Nov. 23-24, 2001, Florence, Italy* [Online]. IEEE, 2001. Available: `https://ieeexplore.ieee.org/document/990175` [Acessed Dec. 4, 2018]