
**Cairo University Faculty of Engineering**


**Credit Hours system**

# Optical Music Recognition

## Image Processing

Team 12


Team members:

- Abdelrahman Mamdouh 1170480
- Ahmed Hatem 1170139
- Osama Yahia 1170141
- Mohamed Elsayed 1170239

## Workload Distribution:

Abdelrahman Mamdouh:

- Classification

Ahmed Hatem:

- Testing on docker
- Binarization
- Dataset preprocessing

Mohamed Elsayed:

- Lines detection
- Image rotation

Osama Yahia:

- Staff line removal
- Segmentation

# Outline:

- Introduction
- Pipeline
- Code structure
- Binarization
- Lines Detection
- Image Rotation
- Segmentation
    - Staff lines removal
- Symbols Detection
- Classification
    - Dataset
    - HOG features
    - Neural Network
    - Template Matching
- Our proposed approach steps combined
- Future Improvements
- References

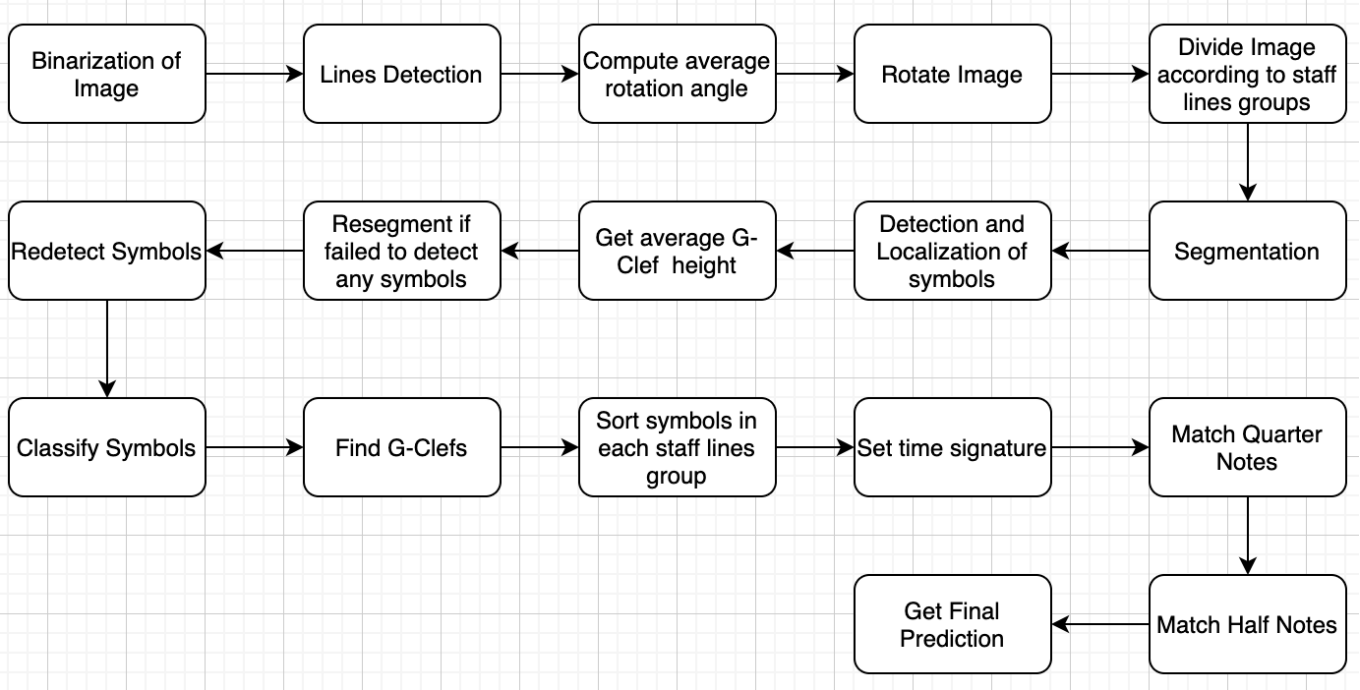
# Introduction:

In this project we propose a method for solving the problem of optical music recognition using image processing. Our method worked successfully on scanned test cases and ready to work on handwritten test cases, but we had the problem of staff lines removal in skewed images only.

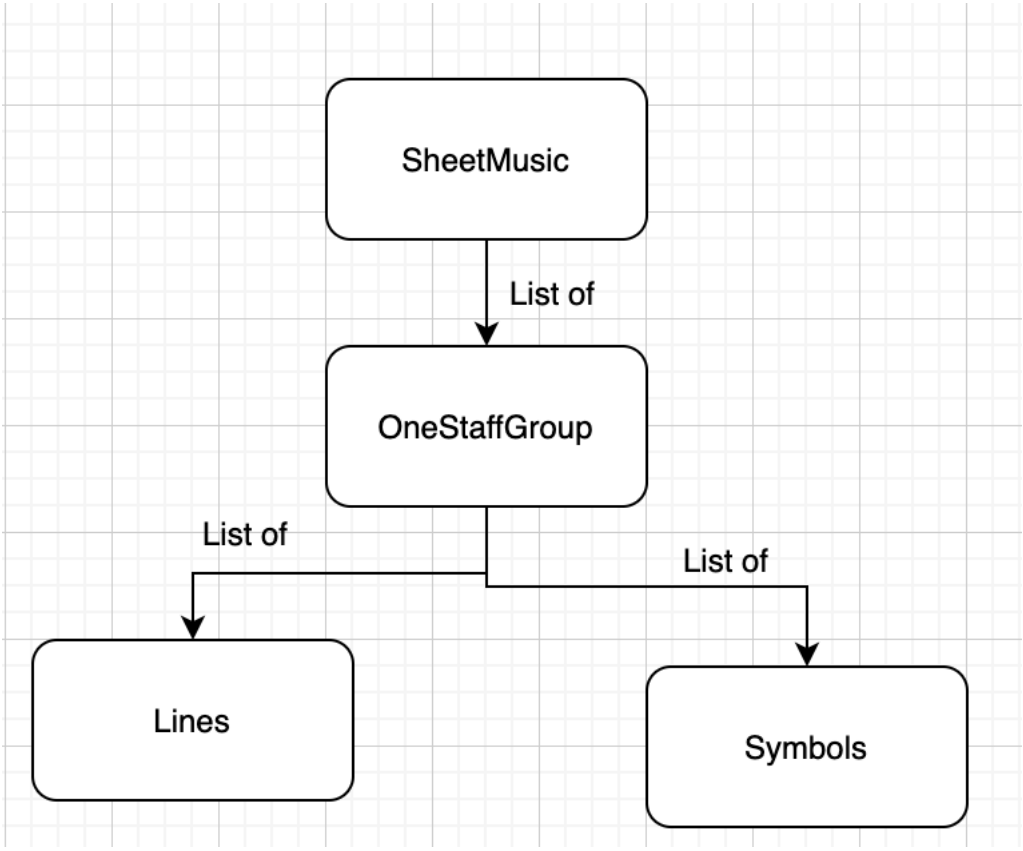
This document will first present how we used and implemented each functionality in our pipeline, then will show all the combined steps that enables us to reach our final prediction.

Our project is tested the scanned public test cases, and we were able to classify correctly all the images in this scanned dataset except only 2 images that we had some error in our classification.

# Pipeline:


Binarization of Image
Lines Detection
Compute average rotation angle
Rotate Image
Divide Image according to staff lines groups
Segmentation
Detection and Localization of symbols
Get average G-Clef height
Resegment if failed to detect any symbols
Redetect Symbols
Classify Symbols
Find G-Clefs
Sort symbols in each staff lines group
Set time signature
Match Quarter Notes
Match Half Notes
Get Final Prediction

## Code structure:




SheetMusic: represents the whole musical sheet image, and it has multiple staff groups.

OneStaffGroup: SheetMusic is divided into multiple staff groups (each 5 staff lines form a group).

Lines: Each OneStaffGroup must know the position and orientation of the lines that it has.

Symbols: Each OneStaffGroup contains a list of symbols that lie on the its staff lines. Symbol contains the segmented symbol from the whole sheet music image.

## Binarization:

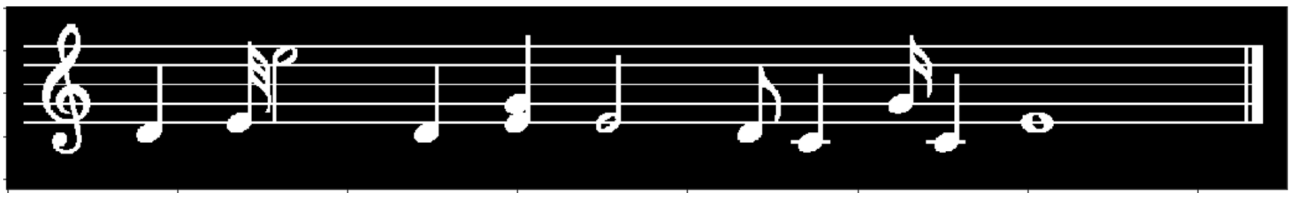
It is an important step in the pipeline, as from it we can detect lines and its orientations.

The main method used for binarization is otsu's method, and for some cases for segmentation we used yen's method. Both methods are already implemented in the skimage library.

Used functions from skimage:

- skimage.filters.threshold_otsu
- Skimage.filters.threshold_yen

In general, threshold is a better choice for both scanned and handwritten test cases.



*Binarization using yen's thresholding*

## Lines Detection:

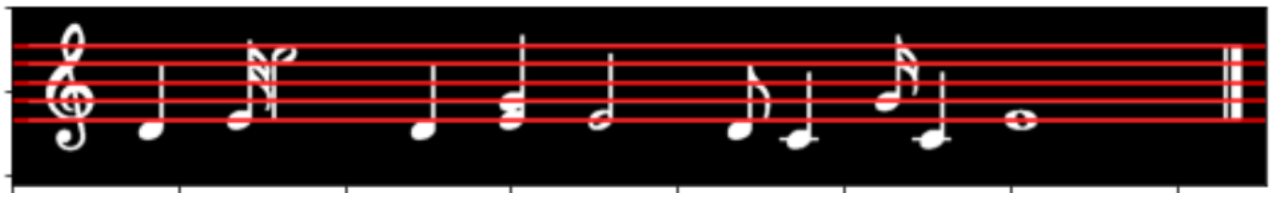
It is one of the most important steps in the whole pipeline, where in our implementation we depend heavily on the position and orientation of the staff lines, as they act as our coordinate axes that from it we construct the staff labeling method.

Used algorithms:

- skimage.transform.hough_line
- skimage.transform.hough_line_peaks

Other approaches experimented but not used in the project:

We experimented with other approaches like the probabilistic hough transform (cv2.HoughLinesP()), but eventually did not use it, as we only work on the scanned version of test cases.



*Five lines detected in the image*

## Image Rotation:

For our method to work correctly, staff lines should be as horizontal as possible so that we can efficiently remove them(will be discussed in the Segmentation section). In order to detect the skewness of the image, we get the average rotation angle from the angles list that is got by hough transform.

Used algorithms:

- imutils.rotate_bound

Drawbacks:

The drawbacks of this rotation method with respect to our approach, as ew expect that the rotated image should be perfectly horizontal. But this will not be the case when rotating images in the handwritten test cases. This is a problem to us because in order to remove staff lines(will see why in the Segmentation section), we use a horizontal structural element to remove the staff lines, but if the lines were more than even +/-1 degree the structural element will not be able to detect and remove the lines.

## Segmentation:

In order to classify the musical notes, we first need to separate them from the staff lines in order to localize them accurately and separate them from the background.

In our code implementation, the only use of the segment function is to remove the staff lines and separate each musical note each as a connected component.

Staff line removal:

*Staff lines removed*

In order to remove staff lines, the horizontal lines (or the image in general) must be perfectly horizontal, so that our horizontal structural can detect it and remove it. We first apply a horizontal structural element of size (25,1). After removing the lines, some parts of the musical notes may have some of their parts removed due to the morphological operation, so in order to fix that, we used a vertical structural element of size (1,8) so that it can repair the damage that happened to the symbols.

Used functions:

- cv2. getStructuralElement
- cv2.morphologyEx
- cv2.findContours

Drawbacks of this staff removal method:

Any small skewness in the staff lines or in the image in general will result in not being able to detect the lines or even remove them. A better approach may be to calculate the run length encoding at different angles and iterate through the image column-wise and see the dominating angle to detect the line.

## Symbols Detection:

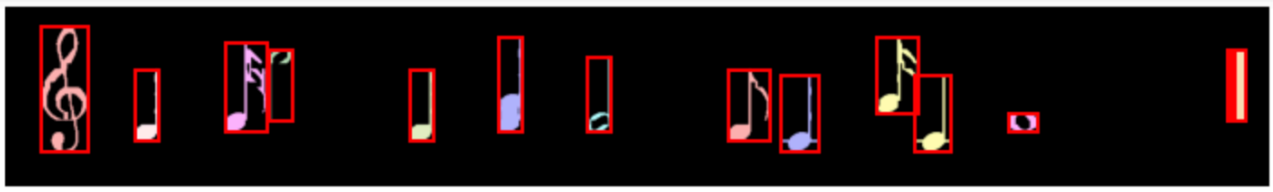
In our approach, symbols detection is a continuation of the segmentation method idea.

Since we were able to detect and remove staff lines successfully in the scanned test cases, we used the connected components algorithm to detect and localize each musical symbol in the image. Another check we make, is to see the region area of the detected connected component, if is a greater than a certain threshold then it is an actual symbol, else it is a false segmentation.

Used functions:

- skimage.measure.label
- skimage.measure.regionprops

This is a very robust approach only if staff lines are removed correctly with no false removals.



*An example of correct segmentation and symbols detection*

## Classification:

To classify a symbol, we depended on several methods to gain confidence about a certain classification. First we make initial guesses using the 2 layer neural network we trained on HOMUS dataset, and in order to have a general rule for classifying scanned and handwritten symbols, we apply thinning on the symbol so that will be no difference between a scanned symbol and a handwritten symbol. Then we use template matching to match quarter notes and half notes, as they are the most important musical note symbols as they appear frequently. We give more credit to the template matched results since we are working on scanned images only in our case. And if detected quarter notes using template matching, we see the classification using the neural network, if the neural network detect any type of notes class('Sixteenth-Note','Eighth-Note',or 'Thirty-Two-Note') we than append this to our classification as the duration label, else then this symbol is only a quarter note.

Algorithm for getting final prediction:

1. Extract HOG features from each symbol
2. Pass the features into the neural network
3. Apply template matching on the whole image in the staff group and find quarter-notes(full noteheads) and half-notes(empty noteheads).
4. For each symbol in symbols:
    1. If full noteheads present (using template matching):
        1. Get the center position and get it's staff label position(ex: "a1","b1",…)
        2. See if the neural network classifier detected one of the following ("Eighth-Note","Sixteenth-Note","Thirty-Two-Note"):
            1. If so, then this is the duration label ("/8","/16","/32")
            2. Else, then this is surely only a quarter note with duration label ("/4")
        3. Check if multiple noteheads in this symbol:
            1. If vertical multiple noteheads, then Chords are present
            2. Else if horizontal multiple noteheads, then beams are present

2. Else If empty noteheads present:
    1. Then get the center position of the note head and gets it's staff label position and duration must be "/2" as this is the only case in the given dataset to test on
3. Else if check for the presence of dots using the aspect ratio of white to black pixels in the symbol
4. Else take the classifiers guess (which is a weak method!)

Feature Extraction:

With some research, we found that HOG features were the most successful for optical music recognition. We also compared it with only using raw pixels of the symbol, but its accuracy was far below the HOG features.

Dataset used for training (HOMUS):

We used the HOMUS dataset as it comes with large variety of symbols with 15000 images. The dataset comes original in a '.txt' format, so some preprocessing was done on the dataset before usage. First draw the image by connecting the points in the '.txt' files, then make the symbol fit the image window exactly(useful for classification).

Neural Network:

The neural network we used for classification was a 2 layer neural network with 400 neurons, and trained for 500 iterations. Training time took approximately 7-9 minutes on i7 processor 16GB RAM.

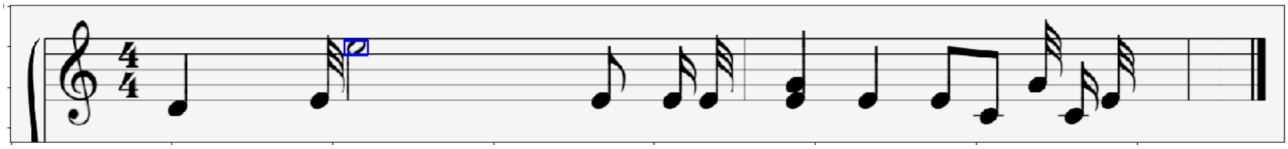
The network's accuracy was 86%, were the most confusions were in symbols that had flags especially between 'Eighth-Note' and 'Sixteenth-Note'. Other classifications methods were experimented as SVM and KNN, but neural networks got us a slightly better accuracy

Template Matching:

We used template matching mainly to detect full note heads and empty note heads, as they were the most frequently occurring in every music sheet. We cropped full note heads and empty note heads from the public dataset named “PrintedMusicSymbolsDataset”.


*Example of correct classification*


*Example of incorrect classification, should be ‘Eigth-Note’*



*Example of using template matching to detect empty noteheads*

## Proposed approach:

Using the methods described in the previous pages, we constructed our pipeline to get the final output prediction/score for a given input image. Our pipeline works as follows:

Algorithm:

1. Binarize image
2. Detect lines using Hough transform, and get list of angles
3. Compute average rotation angle, excluding outliers
4. Rotate image
5. Divide image into staff groups (if there are multiple staff clusters)
6. Segment and remove staff lines (only use of this step is get the positions of symbols if though they are dilated and corrupted, we only need to detect its presence and its location for now, will be fixed in step 10)
7. Detect and localize the symbols
8. Classify symbols using an initial guess using the neural network
9. Find the average G-Clef height, remove misclassified G-Clefs (outliers)
10. Resegement the image using the average G-Clef height to make a better segmentation and separation of symbols and better removal of staff lines
11. Redetect symbols based on the previous staff line removal method
12. Classify symbols
13. Again find G-Clefs to make sure that number of G-Clefs matches our staff line groups (assuming that each staff line group has only one G-Clef)
14. Sort symbols in each staff group
15. Set time signature for each staff group if any
16. Template matching of quarter notes
17. Template matching of half notes
18. Apply the algorithm explained in Classification section to get the final score of each symbol
19. Get final prediction

## Future Improvements:

The main improvements to make is to focus more on the staff lines removal and make a robust algorithm to work on all cases. Also, classification should be done without template matching, in some handwritten cases template matching was not able to find any matches, therefore we should depend more on a machine learning classifier trained on a bigger dataset and adding different orientations to each symbol.

## References:


- opencv documentation website
- Skimage documentation website
- HOMUS dataset: https://grfia.dlsi.ua.es/homus/
- PrintedMusicSymbolsDataset: https://github.com/apacha/PrintedMusicSymbolsDataset
- Baró, A., Riba, P., Calvo-Zaragoza, J. and Fornés, A., 2019. From Optical Music Recognition to Handwritten Music Recognition: A baseline. *Pattern Recognition Letters*, 123, pp.1-8.
- Novotný, J. & Pokorný, J.. (2015). Introduction to optical music recognition: Overview and practical challenges. CEUR Workshop Proceedings. 1343. 65-76.