Image Processing

Final Report

QUEIROZ, Breno Cunha
11218991

NÚÑEZ, Henrique Hiram Libutti
11275300

MOREIRA, Maria Eduarda Kawakami
11218751

July 2020

1 Introduction

This project aims to extract notes from a picture of a music sheet, and be capable of playing and saving the music. To solve this problem it is necessary to create a robust algorithm that is able to differentiate objects that are expected in the image from foreign objects, such as texts and strange images. It is also necessary to identify in which positions the lines are, since they vary from image to image and are extremely important for determining the musical note.

2 Program Pipeline

The program execution pipeline can be divided into 4 stages. In the first, there is the processing of the sheet music image and the segmentation of the notes. In the second, the notes are classified according to duration. In the third, note tone is detected. At last, the note list is played (Figure 1).

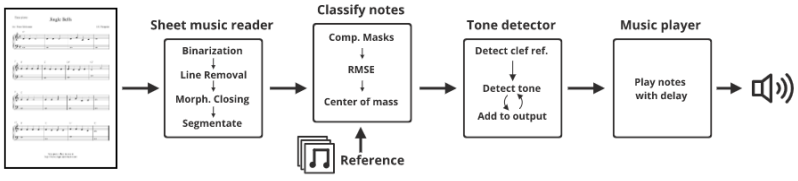

Figure 1: Program pipeline from raw image to music song.

3 Input Images

We got the input images in the website https://musescore.com/ Here are two examples of music score:

Canon in D[1]

La Danse Macabre[2]

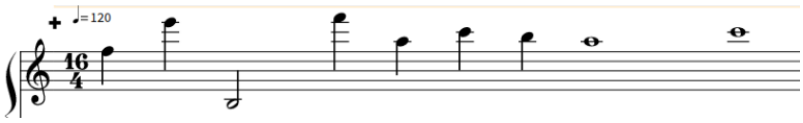

Figure 2: Original sheet music image to play.

4 Image processing

4.1 Binarization

For binarizing the image we used Otsu's threshold, known as the most widely used method. It computes the optimal threshold by separating the intensities in two classes that minimizes intra-class variance and maximizes variance among classes.

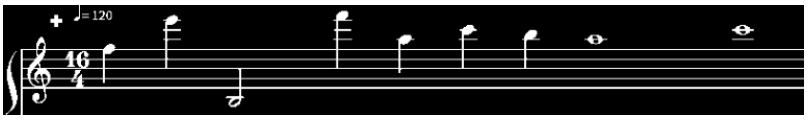

Figure 3: Image after binarization with Otsu's threshold.

4.2 Line removal

For removing the lines of the music sheet we first detected the lines with a rectangular kernel and performed erosion and dilation, which was useful afterwards (Tone Detection) for having the heights of each line in the image.

Figure 4: Lines detection.

After that, we just removed the found lines in the original image.

Figure 5: Image after line removal with kernel.

4.3 Morphological closing

After removing the lines in the music sheet, it's noticeable that the notes have little gaps where the lines were. In order to fix that, we used the morphological operation closing, which consists of dilation followed by erosion and it will fill the gaps in the notes.

Figure 6: Image after morphological closing.

4.4 Segmentation

After binarizing the image the only step missing to segment the image is to distinguish different notes, which can be done attributing a seed to each set of black pixels. To do this we are searching for black pixels on the image and attributing a seed to the set of pixels in which it is contained. Then the procedure for region segmentation by conquering is called.

4.4.1 Segmentation strategy

This procedure uses a conquering strategy, by evaluating the distance between the mean intensity value of region until now and the value of the pixels contained in a 4-neighbourhood of the current location. In case the calculated distance satisfies a given threshold, the procedure is then called recursively in one of the four directions mentioned earlier. With a then computed region, it is possible to extract useful information from the original music piece. We now must do processing for achieving two aspects: the tone and the tempo.

4.5 Note recognition

To recognize the notes segmented we are comparing the set of pixels with each seed using a kernel of note recognition. If the set of pixels matches with a

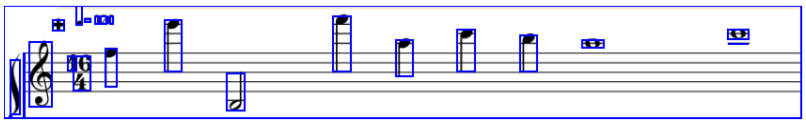

Figure 7: Bound Boxes found after segmentation.

note, we calculate the x and y of the note's gravity center, the round mark that defines its tone.

4.5.1 Symbol classification

The symbol classification is done by gathering information from the region that is being analyzed (all of the segmented regions pass through this step) and comparing against some reference. The decision of attributing a symbol class to a region is done by two different approaches.

One of them by using overlapping the region and one of the reference images, and counting how many equivalent pixels are there in both images.

- Both images are resized and binarized once again, so we can compare whether a pixel is black or white.

- The number of equivalent pixels is then divided by the total number of pixels in the resized region, so there is a percentage of similarity between the two images.

- This result is then recorded.

After running this algorithm through all reference images, the maximum values for each class is then taken, and the maximum over the classes is taken again, so we have the best fit among all data. The best fit class is now taken as the symbol class. If this maximum percentage of similarity is then greater than 70%, the region is able to pass through the next step.

Chained with the last mentioned, the other approach is to extract features from the segmented regions, so we can have more numbers to compare.

The features are: White/total pixel count, Aspect ratio, 'Mass spread'.

Because of the resizing during the overlap, some absurd may occur, such as a very thin region with some white pixels on one end getting expanded and then matching the pixels of a 'crotchet'. In order to avoid that, we take the root squared error of the aspect ratio to identify if the region was too thin, and discard it, if this is the case.

Also, the mass spread ratio is a good metric for deciding whether the previous classification was an absurd or not, by what we observed, in the cases of thin regions.

4.5.2 Tone detector

For tone detection, we used as reference the note G4 for the G-clef. After that, finding the distance between the lines enabled us to classify the the height between each tone. Then we got the distance of the gravity center to the reference note. After doing some math, we classified the tone of the note, and put it in the format that the sound player we are using.

4.6 Playing the piece

After getting all the tones and delay between each tone, we can start playing the music. To do this we are using mingus, which is a python library to play tones using songfonts. With this library we can download songfonts (sound from piano, guitar, violin, among others) from the internet and play the same music with different instruments.

4.6.1 Note player

The music player uses two python libraries to play the music: mingus and fluidsynth. This step has as input a vector with the notes that must be played. Each vector item has the tone and duration of the note to be played. Each item in this vector is reproduced sequentially.

4.7 Results

In this project we were able to apply some of the fields of Digital Image Processing and better understand its tools and its applications.

We implemented filters, binarization, kernels, used morphological processes and segmentation. In addition, the group was also able to apply feature extraction techniques, and even though actual data science was not applied, despite what was told in the partial report, the classification aspects of this project were within our expectations.

As results, we were able to detect clefs apart from notes and also to properly classify actual notes from segmentation artifacts, although the symbols themselves were not as diverse as predicted, probably due to the need of more features to be extracted and the use of other classification algorithms.

4.8 Students' roles

Breno: found a library for playing the music (Mingus), worked in the note player which uses a vector with the specifications to play the desired note and acquired a dataset for the note classification.
Henrique: worked in image segmentation, and implemented feature extraction and symbol classification.
Maria Eduarda: worked in initial steps of image processing (binarizing, removing horizontal lines, closing) and implemented tone detector.

References

[1] *Canon in D.*
https://musescore.com/user/88585/scores/105013

[2] *La Danse Macabre.*
https://musescore.com/user/54180/scores/90702