Cairo University - Faculty of Engineering
Computer Engineering Department
Image Processing (CMP362) – Fall 2020

Course Project
“OMR”

Presented to

Dr. Mayada Hadhoud

Eng. Youssef Ghatas

Used Algorithms

- The Optical Music Recognition problem involves images where the locus of information lies in only a subset of the input image. This leaves room for a lot of preprocessing to facilitate operation on the input image. The preprocessing step involves thresholding the image and inverting it, representing information as true values in the image.
- Preprocessing also involves deskewing the image, in case it is camera-captured. To deskew the image, we apply a **gaussian blur** then dilate the image to connect staff lines. Then we apply **edge detection** to get a box around the staff. By extracting the biggest **convex hull** of the image, we extract the border of the music sheet, which we use to fit a rectangle around the music sheet. This rectangle is then used to apply a **perspective transform** to finalize the deskewing operation.
- Staff lines, clefs, barlines, and other such elements do not affect the required output, so, we opted to remove them as our first processing step. To extract staff lines from the image, we chose to examine the **Horizontal projection** of the input image. Assuming the image is aligned properly, dominating peaks in the projection appear where staff lines exist in the input image. By iterating horizontally over the original image where these peaks are found, staff lines are extracted from the image.
- Afterwards we split the music sheet into horizontal bars, each bar containing five staff lines, the standard number, to process each bar separately. Then for each bar, we removed staff lines from the bar. Then, we examined the **Vertical projection** of the output, and found that the leftmost peak represents the clef in the music sheet. And by masking out the leftmost run, we removed the clef from the image. Afterwards, a step of **Binary opening** was necessary to remove noise from the resulting image.
- The next step was to segment the image into notes, to do so, we applied **Binary closing** to the image after removing lines to close the holes present in beamed and flagged notes, and then we retrieved the contours of the resulting image and their bounding boxes. By filtering the bounding boxes based on their size relative to the spacing between staff lines, we retrieved the bounding boxes around each note in the image. We also removed articulation dots from the image to prevent them from getting caught in detection of notes, and moved their processing to a later stage in the pipeline.

- The previous step left us with bounding boxes surrounding separate notes as well as beamed ones. In order to separate beamed notes, we extracted note heads in each bounding box which represents a beamed note. To do so, we again resorted to the **Vertical projection** of the image, where peaks in the vertical projection represented heads. All that remained was to cut the box at intermediate locations between peaks.
- After segmenting the image, we adopted an Object-oriented approach, starting all segments initially as base components, identifying them and assigning them their types by means of classficiation.
- The final steps were to link components which accompany notes to their respective notes, such as accidentals and articulation dots. Accidentals always came before their respective notes, while articulation dots always came after them.
- Afterwards, the output is written in the Guido music notation to the output file.

Classification

- We used a tree-like hierarchy for classification where the input is a base component, and the final output is a fully classified element.
- First we determine if the component is a meter and classify its time, otherwise we determine if it's an accidental and classify its type.
- If the component is neither a meter nor an accidental, it is therefore a note, and the next step is to determine whether it's beamed, flagged, or hollow. Once its type is determined, it's then passed to the corresponding classifier which classifies its timing.
- Each classifier was trained on a sizable dataset (~3000 images), which we gathered by generating sheets using the Guido Editor, extracting the preview of these sheets, and segmenting them using our algorithm. A process which we automated using JavaScript and Python. Then we serialized the trained models and saved them in a binary format, which we load each time we run the script.

Experiment Results and Analysis

- We applied the algorithm to several test cases each representing a different combination of note tones, timings, accidentals and articulation dots. The algorithm achieved a very high success rate with high quality images, and a lower success rate with low quality images, where elements in the image suffer from discontinuities as a result of thresholding.

- We also experimented with different approaches to classification of elements in the music sheet. Among the features we experimented with were: Histogram of Gradients, Horizontal Projection, Skeleton of Horizontal Projection, Ratio of White to Black pixels, etc.. We achieved the best results using the Histogram of Gradients as a feature for most classifiers, with the exception of the beamed note timing classifier, for which we used a feature we crafted, **Weighted Line Peaks**, which measures weights for each peak in the horizontal projection of the image.

Work Division

Abdelrahman Farid Sec. 1 BN. 34	Assigning note tones Extracting staff lines Extracting note heads
Ahmed Hesham Sec. 1 BN. 9	Remove braces that connect multiple staffs Trying multiple approaches in pre-processing Trying different feature sets in classification
Ahmed Nasser Sec. 1 BN. 8	Trying multiple approaches in pre-processing Generating dataset for the classifiers
Youssef Walid Sec. 2 BN. 34	Worked on segmentation module Researched multiple approaches for classifiers Trying multiple approaches in pre-processing

Accuracy and Performance

- Among the provided scanned test cases, the algorithm achieved an accuracy of 98.25%, getting only 2 notes out of a grand total of 114 incorrectly.
- However, when it comes to camera-captured test cases, the algorithm did not perform so well, and only achieved high accuracy for cases 25 through 28.

- Running the algorithm over the entire set of test cases, consisting of 32 images, took 61.26 seconds on an Intel Core i7-8700 CPU, with 16 GB of RAM running at 3200 MHz.

Conclusion

- This project was quite the learning experience for all members of the team. We did a reasonable amount of research on the subject which guided our approach to solving the problem. We experimented with many different image processing algorithms and techniques and settled on morphological operations, as they offered the best performance for the scope of our project.
- This project was also our first serious contact with training classification models, learning about the process and applying it practically helped us grasp its different aspects better and understand it further, as we spent a considerable amount of time fine tuning the classification process and experimenting with different sets and combinations of features.
- As for future work, we intend to further develop our algorithm to achieve better results with camera-captured and handwritten sheets, and to improve upon our previously achieved scores with scanned music sheets.

References

- Understanding Optical Music Recognition (Calvo-Zaragoza, Jan Haji˘c Jr., Pacha)
- Optical Music Sheet Segmentation (Bellini, Bruno, Nesi)
- cadenCV: An Optical Music Recognition System with Audible Playback (Afika Nyati: MIT)
- Detecting the Staff-lines of Musical Score with Hough Transform and Mathematical Morphology (Chen, Zhang)
- Introduction to Optical Music Recognition: Overview and Practical Challenges (Novotny, Pokorny)
- Sheet Music Reader (Harris, Verma: Stanford)