

Improved cadenCV: An Optical Music Recognition System with Audible Playback

Abstract

Built upon the cadenCV, improved cadenCV handles the input music with only one line and improves the detection accuracy drastically.

1. Background

Teaching the computer to read and interpret sheet music and producing a machine-readable version of the written music score has always been the goal in the optical music recognition (OMR) field. The possible applications evolve using machine to help musicians to analyze music faster and providing an alternative tool for initial music learners to practice without the cost of human time. As the student at MIT, Afika Nyati designed a python package, cadenCV. cadenCV is an optical music recognition system written in the Python programming language which reads sheet music and sequences a MIDI file for user playback.

For machine to recognize music, it needs multiple detection and recognition processes. It at least needs to detect and distinguish elements including clefs, staff, notes, rests, accidentals, ornaments, dynamics and articulations. Figure 1 is a visualization of these elements.

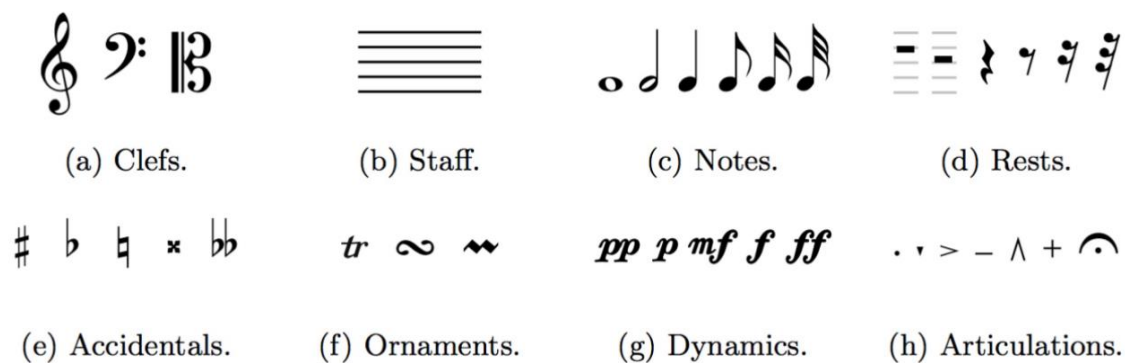


Figure 1

To detect the above elements, detecting the staffs could be a starting point, since each line in the staff is supposed to be horizontal, and the row with the largest number of black pixels in the input picture has large possibility to be one of those lines. After detecting the staff, it is easier to detect other elements based on their relative positions with each staff line.

cadenCV has four steps to do the whole task: image processing, music primitive detection, music primitive classification and semantic reconstruction. Image processing involves noise removal and binarization to avoid possible interference and delete unnecessary information such as color, since only black and white convey meanings in music scores. The system also employs Otsu's global thresholding method. "A global thresholding method was chosen over an adaptive one due to the fact that the input image would have a uniform background, thus it would not need to threshold each pixel using local neighborhood information." This simplifies the image processing process and reduces the processing time. Music primitive detection uses run-length encoding (RLE) to detect the staff line thickness and staff space height. Music primitive classifications recognizes primitives by "comparing an image segment to a library of labelled primitive templates, staff-by-staff, primitive by primitive, and classifies as a specific music primitive if its similarity exceeds a specified threshold." And each pitch is determined by its note head's relative position with the staff lines. The final semantic reconstruction process, the detected primitives are sorted with the sequence they appear on the score. And the note pitch is transformed to MIDI note value for the final output.

2. Code problems and improvements

When I tried to run the existing code with the paper, it only worked with the example picture of the score *Mary had a little lamb*. Running my own scores caused errors in the console and

prevented the code giving an output. I tired pictures from the internet and used MuseScore to make simple lines of music by myself. Neither of them worked.

By experimenting with different examples, I found out the code could not work for scores containing only one line as the figure 2 shows.



Figure 2

By line-by-line debugging, I found that there are certain places in the code where the matrices call their second element without checking the size first, which causes index out of bound error. For example, line 716 in figure 3 is the place this kind of bug first appears:

```
716     half_dist_between_staffs = (all_staffline_vertical_indices[1][0][0] -
717                               all_staffline_vertical_indices[0][4][line_width - 1])//2
```

Figure 3

When the input contains only one line of music, `all_staffline_vertical_indices` does not contain `[1]`. I added a condition check for this kind of code: if the input is only one line, use `[0]` instead of `[1]`.

When there are multiple lines of music, the next step is to crop the image into different groups: each staff is one group. But when there is only one line, there is no need to crop the image. Cropping it causes error when process the image using `cv2.cvtColor`. Again, I added another condition in the code to let it deal with the music with only one line.

When the error disappeared, I found it cannot recognize my input clef and time signature correctly. I simply added more source pictures for it to compare with.

The most important element, the pitches of notes, were completely wrong. I changed the formula for calculating the relative position of the notes.

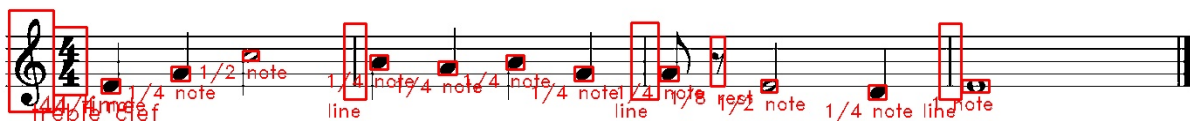
Now the code can both deal with input score with multiple lines and with a single line.

3. Results

With figure 2 as the input picture, it now detects staff correctly.



And it can detect the bar lines, staves and time signatures correctly compared to it failed to detect the elements in the old version.



```
[INFO] Corrected note values after accidentals:  
E4, G4, C5, line, A4, A4, G4, line, G4, rest, E4, D4, line, E4,
```

In terms of the correction of notes, instead of the 0% accuracy rate, it only has 2 mistakes for the same pitch in these 11 notes. It detects B4 as A4, which was quite close. This problem happens with the original code when running *Mary had a little lamb*. When A4 and B4 are adjacent to each other, it failed to detect B4. I think it involves the classification of the direction of the note stems.



I tried another example with 2 lines which has large gaps between each note. The pitches it detected were completely wrong. But when I input only the first line of the music, all notes are correct. Because I mainly worked on the case when the input image contains only one line of the score. This comparison shows that my improvement is quite effective.



4. Next steps

Since detecting one line using my improved version of code is much accurate now, next steps involve applying the change to case with multiple lines of input. Another option is to crop the image first and do the remaining detection with only one line of music.

Moreover, using source pictures for clefs and time signatures is not efficient and accurate enough. If there is database for the music elements, it is a good idea to train the modal with CNN.

Bibliography

Nyati, A., 2017. *GitHub - afikanyati/cadenCV: An optical music recognition system built as a final project for 6.819: Advances in Computer Vision course at MIT. Takes sheet music as input and returns a MIDI file of written music..* [online] GitHub. Available at: <https://github.com/afikanyati/cadenCV>.

Nyati, A. cadenCV: An Optical Music Recognition System with Audible Playback. 2017. https://firebasestorage.googleapis.com/v0/b/afika-nyati-website.appspot.com/o/design%2Fcadency%2Fcadencv_afika_nyati.pdf?alt=media&token=a5aa2413-32c0-4bc7-8222-06342b822096