Music Sheet Understanding and Tones Transposition

Khoa Truong,Minh Dinh,Triet Huynh,Khoa Nguyen and Vinh Dinh
CSE,Vietnamese-German University

August 17, 2021

Abstract

The field of Optical Music Recognition (OMR), a subfield in Artificial Intelligence, is aimed at automating the translation or understanding of music sheets [1]. However, there is a lack of applications to solve the problem of musical tones transposition (the process of moving a collection of notes up or down in pitch by a constant interval). Such a problem in tones transposition is highly labor-intensive if the musician has to reorder the notes manually, often impossible if done during a performance. Our work proposes a way in which musicians can perform tones transposition by scanning a music sheet, input the number of shift tones or semitones required, and then the programm will output an audio file or a new music sheet with all of the notes shifted to the required pitch.

1. Introduction

The topic of recognizing musical sheets, i.e., Optical Music Recognition (OMR), is not a novel field of research. The term OMR first appeared in a paper written by MIT scientists in the 60s. During the last three decades until now, OMR is an ever increasingly developing field and is capable of solving many music related problems [4].

More specifically, the current OMR systems of today are capable enough to recognize a printed musical sheet and digitize it. The resulting output could be a .midi file, or other types of sound files such as .way, .mp3. The vast majority of those researches are dedicated to the common user, even for users who are not educated on musical theory, but there is still a lack of products that can be used by professional or enthusiast musicians. In reality, a common problem that is encountered is the transposition of music tones, i.e., shifting up or down a constant interval of semitones or tones for the whole music sheet. Currently to obtain a music sheet with a few tones higher or lower the musician has to manually retype the entire musical sheet by hand, which is labor-intensive and time-consuming.

2. Proposed Method

The process includes the first stage, Staff Line Preprocessing, to remove the lines on each staff for ease of musical note detection. The second stage is Note Translation to translates the detected notes into scientific pitch notation. The third stage, Tone Transposition will transpose the semitones or tones of the song to give the final output.

2.1. Staff Line Preprocessing

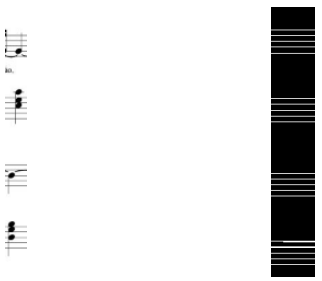
For ease of note recognition, first, the staff lines must be detected and removed. This process includes two stages. First, detection of the staff lines and their thickness; second, removal of those staff lines.

2.1.1 Staff Line Detection

To detect the staff line in the music sheet we first need to remove all of the notes. This is done for ease of staff line detection.

Our staff's line detection algorithm will first grayscale the image and then invert the image colors so that the lines are now white and the background is black. The color inverted music sheet is then run with the OpenCV built-in dilate() function, where the horizontal lines will be expanded so that its width is increasingly larger, and any white pixels, i.e., notes, that does not belong to the horizontal line, will be flipped to 0 and becomes the background thus eliminating all of the notes.

Result:



(a) before applying dilate function (b) after applying dilate function

the dilated image above will give an output that is fairly accurate except for a few lines at the last staff line. this is why we then use Gomez's method [2] and run length encoding (RLE) to determine the lines' thicknesses and their distance between one another. RLE will be applied vertically along a random column of the dilated image above:

This image was taken from Gomez's original paper [2] which was done on a note removed but not color inverted music sheet. which is why the line is in black and the distance is in white, opposite to our dilated image.

RLE will group adjacent pixels of the same color into one group with a number to indicate how many pixels of the same color is within that group and a character to indicate that group is either a group of black pixels "b" or a group of white pixels "w". through which the staff line thickness and the distance between two staff lines are then obtained. more specifically, to extract the distance between each staff line, e.g., our RLE output might be 3b,2w,4b,2w,3b,3w,3b,2w, the black group (because our music sheet is color inverted so black is the distance) that appear most often "3b" with the value of three pixels is chosen. therefore, the distance between two staff lines is three pixels. the same method is also applied to get the thickness of the lines.

2.1.2 Staff Line Removal

In our music sheet, places where there is only lines will have matrix of this type:

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Figure 2: an example of a kernel with only line and no notes

Please note that the rows of ones represent the thickness of the line, which can vary from one to multiple rows, the same thing goes for the row of zeros, distance between each line.

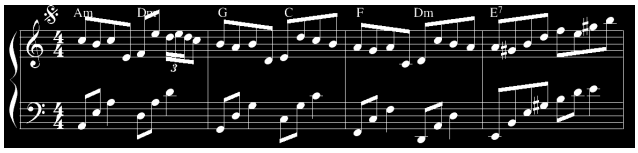


Figure 3: The current color inverted music sheet

To remove the staff lines on each music line, a check vector of length N will be run on each NxN kernel of the image:

$$\begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 & 1 & \dots & 1 \end{pmatrix}$$

Figure 4: Check vector of length N

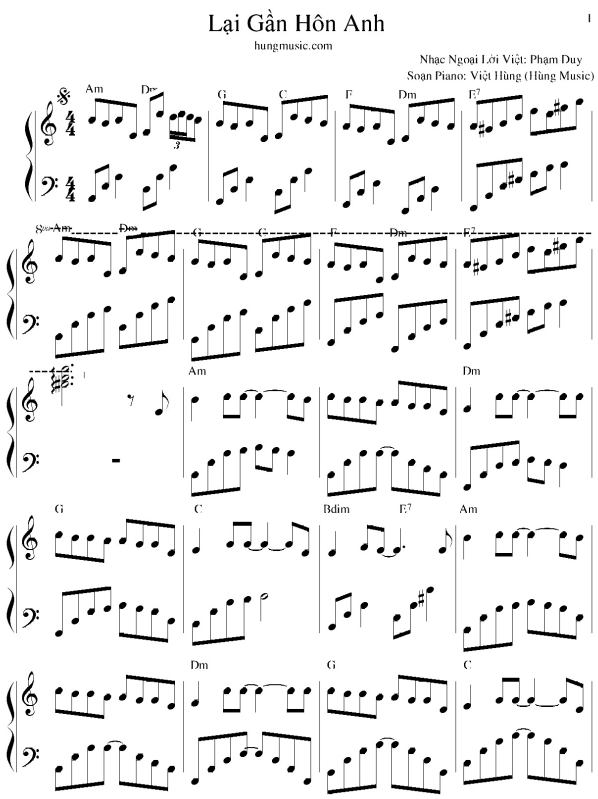
The number of zeros and ones in figure 4 depends on the thickness of each line and the distance between two staff lines respectively, which was obtained in the previous section. For example, with the kernel in figure 2 the check vector will be of the type:

$$\begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$$

This check vector is then multiplied with every NxN kernel of the image. If the kernel is of the type of a line, the result will be a vector of 0s, otherwise, the vector result will contain at least one positive integer among its elements.

For every time we encounter an area that is a line by using the check vector, the found area will be replaced by the NxN matrix which is full of 0s, else it will be ignored and we will continuously check the next kernel. The process will be terminated when we have checked all the possible kernels in the image.

Finally, inverting the color of the output image one last time will result in a music sheet with lines removed:



2.2. Note Translation

To translate the note from its input as pictorial data to scientific pitch notation so that the computer can process the music sheet, three steps are required. First by using Rosebrock's method [3] to eliminate overlapping note positions after running template matching. The second is to reorder the notes that are on the same staff lines into a list, for each staff line there is a corresponding list containing the positions of each note on that staff. Finally, converting the note positions into scientific pitch notation.

2.2.1 Eliminating overlaping positions

By running the template matching function built-in OpenCV on the output staff line removed music sheet will obtain a list of position of notes of music sheet. However, there is a problem with multiple note positions overlapping with each other.

Inorder to handle this we use Rosebock's method [3] call Faster Non Maximum Suppression so that each note will only have one position entry, avoiding duplication in our list of note positions.

2.2.2 Note reordering

Next musical notes that belong to the same staff will be grouped into the same list, for each staff line there is a corresponding list containing its note. To do this, from the position of the first line(the bottom line of each staff) move down for a distance of half a staff height and then create a second point which is from the current point up to two staff heights (reason for moving half of a staff height from the first line down and then moving two staff height up is to account for notes that are on the staff's lines and notes that are on the ledger lines). Any notes' vertical position that fell in the range from those previously mentioned points will be considered as being in the same staff.

In music theory, if two staves are connected by a curly bracket on the left that means they must be played simultaneously.

E.g:

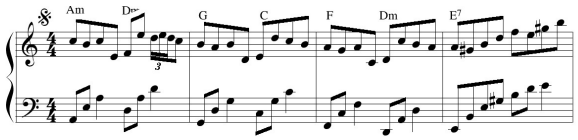


Figure 5: Two staves that need to be played simultaneously

In the example above the staff above with the treble clef is called the main staff and the staff below with the bass clef is called the sub staff. The algorithm will now initialize two lists MAIN[] and SUB[] to store the two staves respectively.

Now the algorithm will move simultaneously through both staves (main staff and sub staff) and check the notes iteratively. For example, our first note MAIN[0] and SUB[0], if they are vertically aligned, meaning that they need to be played simultaneously, the algorithm will then move MAIN[0] and SUB[0] to MAIN_RE_ORDERED and SUB_RE_ORDERD. By the word "move", the algorithm will cut the note position from the original MAIN list and paste it into the MAIN_RE_ORDERED, same goes for SUB_RE_ORDERED.

However, since some music sheet has minor errors during printing which will result in minor misalignment of the note, meaning they are still supposed to be play simultaneously but their horizontal (x-axis) position are not exactly the same. The function ReoderedStaffs() has a threshold value of 5, meaning if the two notes deviate from each other, either to the left or right, less than 5 pixels will still be considered as being played simultaneously.

There will arise a case, in which there is only one note on one of the staff, meaning only one note is needed to be played at that moment, the tuple (0,0) will be added into the staff that doesn't have a note as a filler note. In the case that any one of the staff ran out of notes before the other staff, continue to move through both of the staves like before, but the tuple (0,0) will be filled in as notes for the staff that

ran out of notes first. Doing this will assure that the two lists MAIN_RE_ORDERED and SUB_RE_ORDERED will always have the same number of elements in their list.

This is the result of the first five notes of the two staves in figure where there are two staves that need to be played simultaneously. 5:

(216, 253), (242, 260), (270, 253), (298, 285), (325, 278)
(216, 412), (271, 368), (298, 381), (0, 0), (353, 368)

2.2.3 Digitalization of the notes

With the list of notes's position, to be able to make note transposition possible they need to be translateed into scientific pitch notation.

To achieve this we have create four lists containing scientific pitch notaion. Two list for the main staff and two for the sub staff(one for notes above the first staff line and one for notes below the first staff line).

The two list for the main staff:

['E5','D5','C5','B4','A4','G4','F4','E4','D4','C4','
B3','A3, 'G3','F3','E3','D3','C3']

Figure 6: notes below the first staff line of the main staff

['E5','F5','G5','A5','B5','C6','D6','E6','F6','G6']

Figure 7: notes above the first staff line of the main staff

The two list for the sub staff:

['G3','F3','E3','D3','C3','B2','A2','G2','F2','E2','D2,
'C2', 'B1','A1','G1','F1','E1']

Figure 8: notes below the first staff of the sub staff

['G3','A3','B3','C4','D4','E4','F4','G4','A4','B4']

Figure 9: notes above the first staff line of the sub staff

Whenever a note is encountered on the staff (scanning from left to right), a number will be generated which will be used as the index to get the note notation. But since each staff has two lists which can be choosen from. This problem is resolved by the following formula:

$$D = \frac{2.(note\_position[1] - headlines[i])}{d}$$

D is the ouput
note_position[1] is the vertical position of the note
headlines[i] is the position of the first staff line of each staff
d is the distance between two staff lines in a staff [1]

If D < 0 the algorithm will use the list for notes below the first staff line and D ≥ 0 will use the list of notes that are above the first staff line.

$$I = \begin{cases} \text{D}, & \text{if } D \geq 0 \\ \text{-D}, & \text{otherwise} \end{cases}$$

Continue to repeat this staff by staff for all staves in the music sheet will result in a list of scientific pitch notations for each note in the music sheet organized in chronological order.

2.3. Tones Transposition

After the previous process the output is a list of scientific pitch notaion, e.g., C3, B2, C3, E2, F2, E3, D3, C3.

In music theory shifting a musical note up or down a tone or semitone follow a predictable pattern:

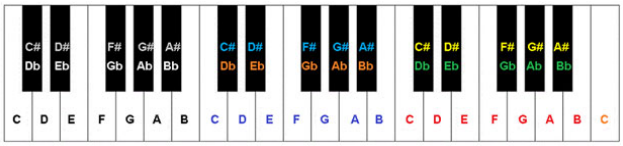


Figure 10: layout of a piano keyboard

According to the graph above, the music note C3 shifted up a tone would become C#3, same goes for all other notes, they become a sharp note. Except for the note E and B which when shifted up a semitone becomes F and C respectively.

To solve this problem, we create a list of scientific pitch notation arrange in increasing order similar to figure 10.

notes_height= ['E1', 'F1','G1','A1', 'B1', 'C2','D2', 'E2', 'F2','G2','A2', 'B2', 'C3', 'D3', 'E3', 'F3','G3', 'A3', 'B3', 'C4', 'D4', 'E4', 'F4', 'G4','A4','B4', 'C5', 'D5', 'E5', 'F5', 'G5', 'A5', 'B5', 'C6', 'D6', 'E6', 'F6','G6','A6','B6' ,'C7','D7', 'E7', 'F7','G7','A7', 'B2',]

Figure 11: list of notes pitch in ascending order

[1] the full equation is: $D = note\_position[1] - headlines[i]/d/2$ with d/2 divided by two because each note is separated with just half of the distance between each staff line

In the list above in figure 11, each note will have a note that is one semitone higher on the left, e.g., E1 and F1. Therefore to shift a note tone up a semitone the note's position in the list, i.e., its index, must be determined; then the new shifted note is at the index of the old note plus one.

A for loop is used to loop over the entire list and check within each iteration whether the current iteration's note notation from the list above matches with the note that the program is working with , if yes return the index of the note.

```
def findNoteinNote_height(note):
    for i in range(len(notes_height)):
        if(note == notes_height[i]):
            return i
```

Once obtained the index of the note, to shift the tone's note up a semitone another for loop is used:

```
if(chord.name!="E" and chord.name!="B" and
chord.sharp==False):
    chord.sharp = True
else:
    chord.sharp = False
    chord.name +=1
# E + 1 = F, F+1 = G, etc
```

The variable "chord.name" is the index of the new note in the list in figure 11. Each time the loop above is run it will increase a note on the current staff by one semitone, so if the end-user wants to shift the music sheet by three semitones, the for loop above will be run three times for each note.

The same process is used to shift the music notes down but instead of increasing "chord.name" by one, "chord.name" will be decreased by one. In addition, there are special cases for when the note is either F or C when shifting the music note down a semitone just like the cases B and C above.

3. Future work

With the help of Nguyen Tho Anh Khoa we are planning to use differential binarization for musical note detection and CRNN for note recognition. With these methods in text recognition, we are hoping to build our new version to be more robust and capable of handling badly handwritten music sheets, or music sheets that have staff lines not perfectly straights or equally parallel.

4. Expected outcome

Our outcome includes a demo source code that can demonstrate tone transposition and at least one conference paper.

References

[1] Jorge Calvo-Zaragoza, Jan Hajic, and Alexander Pacha. Understanding Optical Music Recognition. *ACM Computing Surveys*, 53(4), 2020. 1

[2] Ashley Antony Gomez and C N Sujatha. Optical Music Recognition: Staffline Detectionand Removal. 6(5):48–58, 2017. 2

[3] Adrian Rosebrock. (Faster) Non-Maximum Suppression in Python - PyImageSearch. 3

[4] Elona Shatri and György Fazekas. Optical Music Recognition: State of the Art and Major Challenges. jun 2020. 1