



Vietnamese-German University

MUSIC SHEET UNDERSTANDING AND TONES TRANSPOSITION

August 9, 2021

Contents

I	Research team members	2
II	Disclaimer	2
III	Abstract	3
IV	Introduction	3
V	Proposed Method	4
	i Line Removal	4
	ii Note Translation	6
	iii Tones transposition	9
VI	Future work	11
VII	Expected outcome	11
VIII	Cost	11
IX	Timeline	12
	i Task description	13

I Research team members

- Team Leader: Truong Minh Khoa (VGU CSE 2018)
- Team Member: Dinh Cong Minh (VGU CSE 2019)
- Team Member: Nguyen Tho Anh Khoa (alumni of HCM University of Technology K14)
- Team Member: Huynh Minh Triet (VGU CSE 2019)
- Supervisor Dr.Dinh Quang Vinh,CSE (Faculty of VGU)

II Disclaimer

This report is a product of our team's work, unless otherwise referenced. In addition, all opinions, results, conclusions, and recommendations are of our own and may not represent the policies or opinions of the Vietnamese-German University's Department of Engineering or the University as a whole.

III 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 program will output an audio file or a new music sheet with all of the notes shifted to the required pitch.

IV 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 problems that involve music [5]

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 .wav, .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.

V Proposed Method

The process includes: first removing the lines on each staff for ease of musical note detection. The second stage includes note recognition to translate the detected notes into scientific pitch notation. The third stage is where an additional python function will transpose the semitones or tones of the song to give the final output.

i Line Removal

By using the method in Gomez and Sujatha [2] the staff's line removal algorithm will first grayscale the image then invert the color of the image so that the lines are now white and the background is black. Finally, using a kernel of the form:

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

then run with the `dilate()` function built-in OpenCV, 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) music sheet before line removal



(b) music sheet after line removal

Finally, flipping the output image will result in a music sheet with lines removed:

Lại Gần Hôn Anh

hungmusic.com

Nhạc Ngoại Lời Việt: Phạm Duy
Soạn Piano: Việt Hùng (Hùng Music)

The image displays a piano score for the song "Lại Gần Hôn Anh" (Come Closer to Me). The score is written for piano and is in 4/4 time. It consists of five systems of two staves each. The first system starts with a treble clef, a key signature of one sharp (F#), and a 4/4 time signature. The melody is in the treble staff, and the accompaniment is in the bass staff. Chords are indicated above the treble staff: Am, Dm, G, C, F, Dm, and F7. The second system continues the melody and accompaniment. The third system starts with a new key signature of two sharps (F# and C#) and a common time signature. The fourth system continues the melody and accompaniment. The fifth system continues the melody and accompaniment. The score is written in a clean, professional style with clear notation and chord symbols.

ii 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 [4] method 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.

Eliminating overlapping 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 Rosebrock [4] method call Faster Non Maximum Suppression so that each note will only have one position entry, avoiding duplication in our list of note positions.

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 ¹. Any notes' vertical position that fell in the range from those previously mentioned points will be considered as being in the same staff

¹the 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

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

E.g:

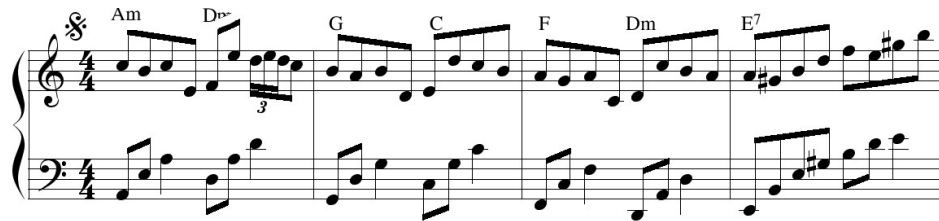


Figure 2: Two staves that need to be played simultaneously

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_ORDERED`. 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 `ReorderedStaffs()` 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 number 2

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

Figure 3: The fourth position has a (0,0) tuple because there is no corresponding fourth note on the sub staff

Digitalization of the notes

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

To achieve this we have create four lists containing scientific pitch notation. 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 4: notes below the first staff line of the main staff

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

Figure 5: 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 6: notes below the first staff of the sub staff

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

Figure 7: 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 chosen from. This problem is resolved by the following formula:

$$D = \frac{2 \cdot (\text{note_position}[1] - \text{headlines}[i])}{d}$$

D is the output

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 ²

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

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

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.

iii Tones transposition

After the previous process the output is a list of scientific pitch notation, 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:

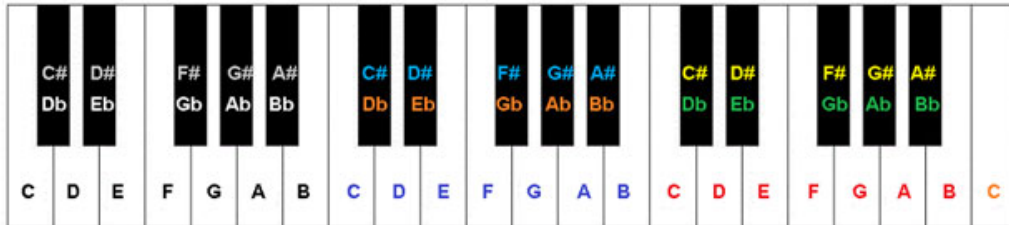


Figure 8: layout of a piano keyboard

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

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 8.

```
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', 'B7']
```

Figure 9: list of notes pitch in ascending order

In the list above in figure 9, 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 9. 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.

VI 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.

Future version will implement the model in Pacha and Eidenberger [3] to bring this method to android devices.

VII Expected outcome

Currently, the outcome of our group's project is to produce a paper to attend a scientific conference. We plan to submit to the Nafosted conference, and if time allows, other Machine Learning conferences.

Hardware: Our final product does not include any hardware components.

Software: After attending the Nafosted conference as well as finishing our paper, we would like to build an app implementation of the method capable of running on Android tablets and PCs.

VIII Cost

Total estimated cost will be 20 Million Vietnam Dong, which will be used to pay for the conference's entry fee and other miscellaneous fees related to attending such conferences.

	Budget items	Total estimated budget		Breakdown of requested budget by year	
		Amount (VND)	%	From VGU	Others
1	Man-power				
2	Materials & Supplies				
3	Equipment, instrument				
4	Conference, publication fee	16,499,000 ³	100	16,499,000	
5	Travel, field expenses				
6	Outsourcing, services				
7	Other direct costs				
	Total:	16,499,000 ⁴	100	16,499,000	

IX Timeline

TABLE 1 Timeline

Sep/2021 - Oct/2021	•	Detecting Staff lines.
Sep/2021 - Oct/2021	•	Staff line removal.
Mid Sep/2021 - Dec/2021	•	Note and Note Type Detection.
Dec/2021 - Jan/2022	•	Grouping Note to Staff.
Dec/2021 - Mid Feb/2022	•	Detecting additional symbols.
Feb/2022 - Mar/2022	•	Grouping Symbol to Staff.
Mar/2022 - Mid Apr/2022	•	Note Translation.
Apr/2022 - Mid May/2022	•	Note Transposition.
Mid May/2022 - Mid Jul/2022	•	Build an app to display the result.
May/2022 - Sep/2022	•	Write paper.

³one VGA MSI GeForce RTX 3060 LHR VENTUS 2X 12GB GDDR6 OC

⁴cost may be subject to change

i Task description

Detecting staff line: staff line detection is the first essential step to obtain the coordinates of each line from which the note's pitch can be derived.

Staff line removal: Run Length Encoding or other methods are used to remove every line of a line-detected music sheet from the previous process or an independent music sheet; this is done for ease of symbols and note recognition.

Note and note type detection: At this stage, template matching is then used to recognize the shape of each music note as well as categorize each note to determine whether it's a whole note, half note, or a quarter note, etc.

Grouping Note to Staff: This process will arrange each note into its respective staff. From then on, any operations on the music sheet will not be done on the entire sheet but it is then done staff by staff; this is a divide and conquers method

Detecting additional symbols: A music sheet doesn't just contain music notes, but it might also include accidentals and key signatures to indicate beat and tone. Recognition and understanding of the semantic of each symbol enable the checking of whether the staff has the number of beats required.

Grouping symbol to Staff: Similar to Grouping Note to Staff but with symbols such as accidentals and key signatures.

Note Translation: At this stage, each note will be converted to scientific pitch notation, e.g., A, B and C, etc. based on its positional coordinate.

Note Transposition: This process shifts the note up or down based on its scientific pitch notation to result in a transposed music sheet.

Build an app to display the result: at this stage, the final product, i.e., the transposed music sheet is completed; we will build an app to apply and display the final result on mobile devices.

Write paper: During the last two months of the project, we will start the process of writing a scientific paper; as a way to contribute to the scientific community and concluding a year of work.

Bibliography

- [1] Jorge Calvo-Zaragoza, Jan Hajic, and Alexander Pacha. “Understanding Optical Music Recognition”. In: *ACM Computing Surveys* 53.4 (2020). ISSN: 15577341. DOI: 10.1145/3397499. arXiv: 1908.03608.
- [2] Ashley Antony Gomez and C N Sujatha. “Optical Music Recognition: Staffline Detection and Removal”. In: 6.5 (2017), pp. 48–58.
- [3] Alexander Pacha and Horst Eidenberger. “Towards self-learning optical music recognition”. In: *Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017* 2017-Decem (2017), pp. 795–800. DOI: 10.1109/ICMLA.2017.00–60.
- [4] Adrian Rosebrock. *(Faster) Non-Maximum Suppression in Python - Py-ImageSearch*. URL: <https://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-python/>.
- [5] Elona Shatri and György Fazekas. “Optical Music Recognition: State of the Art and Major Challenges”. In: (June 2020). DOI: 10.5281/zenodo.4105964. arXiv: 2006.07885. URL: <https://arxiv.org/abs/2006.07885v2> <http://arxiv.org/abs/2006.07885>.