Printed Music Sheet Recognition

Felicia Chandra
fchandr2@illinois.edu

Hao Zhang
hzhang49@illinois.edu

Yaning Liang
liang33@illinois.edu

Abstract

Optical Music Recognition is a topic that has been studied extensively in the field of Computer Vision. Using the commonly used algorithm, our project aims to interpret a music sheet into playable music. We will take a printed music sheet as an input, process it, and play the musical tones as an output.

1. Introduction

The goal of our project is to take an image of a clean printed music sheet, recognize the musical notations in the content, and turn them into a playable music using MATLAB. The process is broken down into the following tasks.

1. Staff Lines Detection and Removal

   Detect the location of all staff lines and remove them to get the musical symbols. The staff lines location will be stored for Segmentation and Note Identification.

2. Segmentation

   Perform segmentation to extract the individual symbol.

3. Symbol Recognition

   Perform template matching on each symbol to determine the musical notation.

4. Note Identification

   For each note, determine the pitch by comparing the symbol location and the staff lines location. Also determine the duration based on the result from Symbol Recognition.

5. Music Transformation

   Create a playable music based on the pitch and duration of the notes.

The process will be explained further in the next section.

2. Details of the Approach

2.1. Staff Lines Detection and Removal

As the first step of the process, when the printed music sheet is imported, as shown in Figure 1, the image is converted into grayscale and double. Then based on the threshold τ, each pixel is adjusted to either black (0) or white (1).

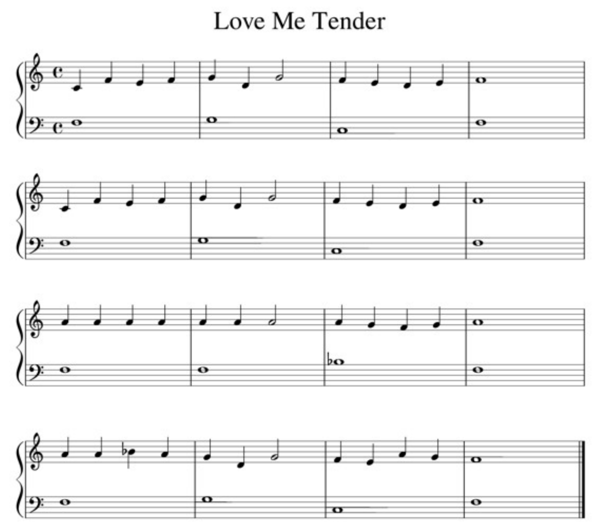

Figure 1: Original Music Sheet

The detection and removal of staff lines follows the algorithm as outlined below:

2.1.1 Detect Location of Staff Lines

We iterate through each row and count the number of black pixels. If the black pixel count is larger than the threshold α, we assume that it is a staff line. The location for these staff lines are stored for Segmentation and Note Identification process.

We explored the different values for τ and α to get the thresholds which would produce the most optimal result. This experiment will be discussed further in the Experiments and Results section.

2.1.2 Remove Staff Lines

After detecting the location of the staff lines, we remove them by changing the black pixel to white pixel.

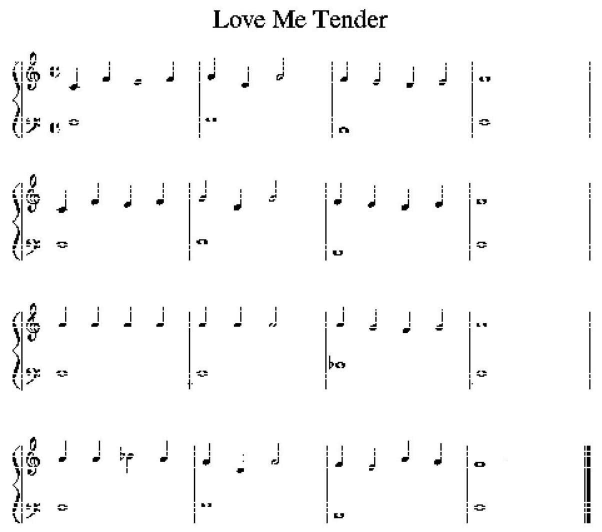

Figure 2: Image After Staff Lines Removal

2.1.3 Recover Notes

In some cases, the staff lines cut through the notes and their removal creates some gaps within the notes. Thus, it is necessary to fill in the gaps to recover the original shape. We iterate through each column and check whether the pixel is surrounded by black pixels above and below it. If this is the case, we fill in the gap with black pixel.

The output after this step is shown in Figure 3.

2.2. Segmentation

After removing the staff lines, segmentation is performed to extract music symbols from the music sheet. This is done following the steps below.

2.2.1 Segment Into Rows

Using the staff line locations detected from the previous step, we are able to calculate the height between two lines. There is a possibility of notes located above or below the staff, so we segment the rows using twice the line height above and below the staff.

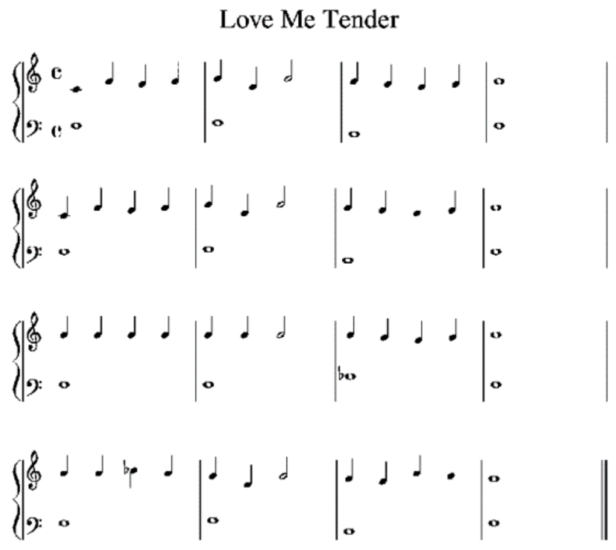

Figure 3: Image After Notes Recovery

2.2.2 Detect Boundaries

For each row, we slide through the columns from left to right and check whether the column contains at least one black pixel. We detect the start and end of the black pixels as the left and right boundaries of one symbol. Afterward, we follow the same process to detect the top and bottom boundaries by sliding through the rows from top to bottom.

Figure 4 shows the segmentation boundaries of each symbol as denoted by the red boxes.

2.2.3 Extract Symbol

After detecting the boundaries, we extract each symbol from the music sheet.

2.3. Symbol Recognition

With all the symbols extracted from the image, we perform template matching in this step. The goal of this process is to assign a label for each symbol based on the training images.

2.3.1 Build Training Dataset

In order to perform template matching, we first establish a training dataset to serve as ground truth. We load the training images and pre-process them in the same way as the music sheet, i.e. convert the pixels to grayscale and adjust them to either black or white. Examples of our training images are shown in Figure 5 and 6.

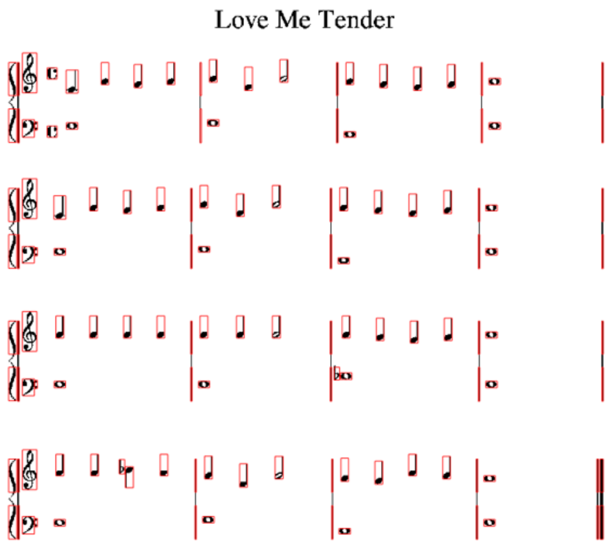

Figure 4: Image With Segmentation Boundaries

Figure 5: Training Image 1

Figure 6: Training Image 2

2.3.2 Calculate Matching Score

With our training dataset established, for every extracted symbol, we compare it against every image in the training dataset. In order to achieve this task, we first resize each extracted symbol to match the size of the training image, and then calculate the count of matching pixels. Due to the different size of the training images, we normalize the matching score by dividing it with the total number of pixels.

2.3.3 Assign Label

We establish a threshold σ for the minimum matching score prior to assigning labels. The choice of σ will be further described in the Experiments and Results section. We consider an extracted symbol to be non-musical notation if all the matching scores are below σ. After filtering out the non-musical notations, we assign the remaining symbols with labels from the highest matching score.

Figure 7 illustrates some examples of the recognition results.

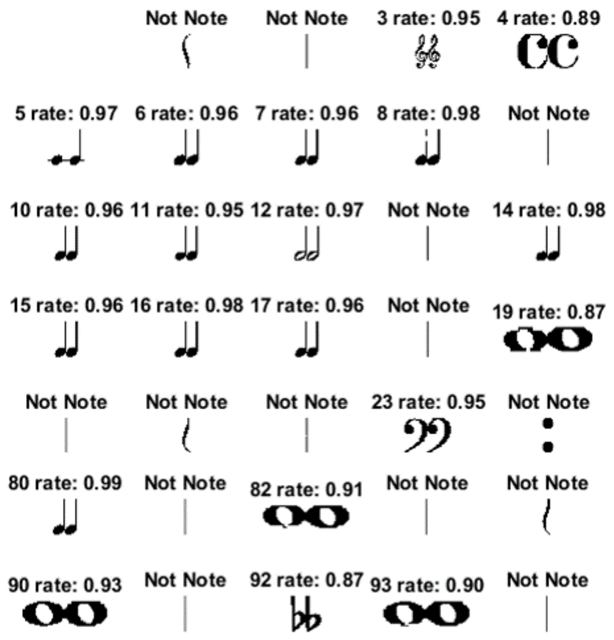

Figure 7: Symbol Recognition Results

2.4. Note Identification

To be able to play the music, we have to identify the pitch and duration for each note. The following steps are used to complete this process.

2.4.1 Calculate Center Point Coordinate

To calculate the center point, we first remove the stem portion of a note if it exists. Using the same algorithm as the segmentation process, we recalculate the boundary of the adjusted image. From there, we calculate the center point's x-coordinate by averaging the left and right boundary values, and the center point's y-coordinate by averaging the top and bottom boundary values.

Figure 8 displays the result of center point calculation.

2.4.2 Determine Pitch

Using the previously stored staff lines location, we compare the center point's y-coordinate of each symbol against the staff line y-coordinates to get the pitch of the node. Since the notes will have a different octave

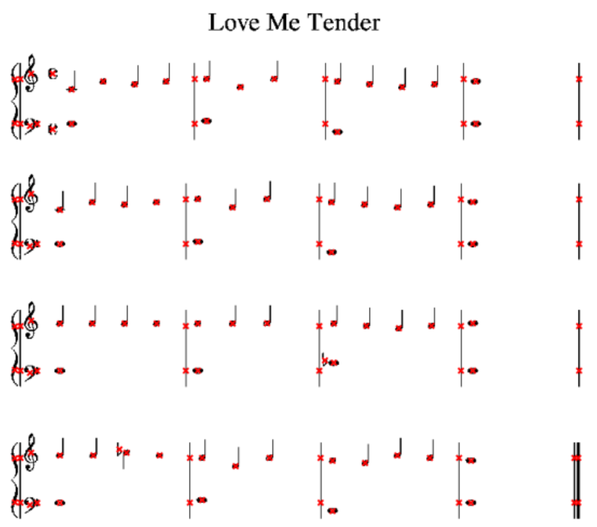

Figure 8: Center Point Coordinates

depending on the clef it belongs in, we take into account whether the notes come from C clef or G clef. Based on the information, we transform the notes into their character representation, such as "C", "D", "E", etc. Additionally, if there is a sharp of flat symbol beside the note, we will increase or lower the note by a semitone (half step).

If the music symbol is labeled as a rest notation, a stop character "Q" will be added to the resulting array.

This process will generate two output arrays which correspond to the two different clefs.

2.4.3 Determine Duration

The duration of each note or rest is determined according to the musical notation rules. Different recognition labels correspond to different duration, ranging from 0.5 to 4. If the symbol is labeled as "dot", the duration of the previous symbol will be multiplied by 1.5.

This process will also generate two output arrays; each of them corresponds to its pitch array.

2.5. Music Transformation

The last task in this project is to play a song based on the pitch and duration we previously calculated. The steps for this process are outlined below.

2.5.1 Generate Sound Data

The sound data is represented as a vector of numbers. Based on the pitch and duration, we generate a sound vector for each note using the mathematical sine function. The formula we use for the sound vector generation is adapted from a MATLAB forum post [1]

For the rest symbols, we create a vector of zeroes where the length is calculated from the symbol's duration.

2.5.2 Play Song

Once the sound data is generated, we can play the song using a MATLAB built-in function called soundsc. This function allows both G clef and C clef's sound data to be played simultaneously.

3. Experiments and Results

3.1. Parameter Selection

In our project, we introduced the following three parameters.

1. Threshold for black and white pixel conversion τ

   To facilitate operations on pixels, we convert all image pixels to either black (0) or white (1). All pixel values that are equal to or above τ are assigned as white (1), while the pixel values below τ are assigned as black (0).

   We initialized τ at 0.5 and incremented by 0.1 in each experiment. We ended up selecting 0.9 for τ since it produced the optimal result.

2. Threshold for staff line detection α

   We consider a row to be a staff line if it contains a number of black pixels greater than a certain threshold. We define the threshold to be the width of the image W multiplied by factor α.

   We used 0.8 as a starting value for α, and decremented by 0.1 in each experiment until we reached 0.3. We discovered that having α with the value 0.5 produced the optimal result.

3. Threshold for minimum matching score σ

   The symbols we extract from the music sheet are not necessarily musical notations. To eliminate non-music symbols, we introduce a threshold for minimum matching score.

   We initialized σ to be 0.5 and incremented by 0.1 in each experiment. We decided that the value 0.8 produced the best recognition result.

A summary of the selected parameter values is shown in Table 1.

Table 1: Selected Parameter Values

Parameter	Value
τ	0.9
α	0.5
σ	0.8

Table 2: Training Images Breakdown

Type	Number of Images
4-Beat Note	2
2-Beat Note	4
1-Beat Note	4
0.5-Beat Note	4
4-Beat Rest	1
2-Beat Rest	1
1-Beat Rest	1
0.5-Beat Rest	1
G Clef	1
C Clef	1
Sharp	1
Flat	1
Total	**22**

3.2. Training Images

For symbol recognition, we use a total of 22 training images. The breakdown of these images is provided in Table 2.

3.3. Staff Lines Detection Experiment

When we first worked on the staff line detection, we tried to explore the use of Hough Transformation. The Hough Transform generates a Hough matrix by transforming the input image into parameter spaces. Then it uses Hough Lines block to determine where the longest line intersects the edges of the original image.

Figure 9 shows a sample of the output image produced using this algorithm. The green lines represent the detected staff lines. We can easily interpret from this image that a lot of the staff lines are not detected properly. Thus by comparison, as shown in Figure 2, the final method we decided to pursue produces a better result.

3.4. Sample Results

We created two videos to showcase the result of our project. They can be viewed at the following links:

- Love Me Tender
- Mary Had a Little Lamb

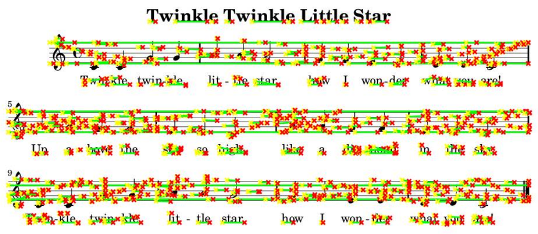

Figure 9: Staff Line Detection Using Hough Transformation

4. Future Work

Interesting future work include incorporating handwritten music sheets as input and leveraging classification methods.

4.1. Expand the Scope of Input Music Sheet

Since the current scope of our project is limited to printed, handwritten music sheets only, we can explore the possibility to incorporate other forms of input including handwritten music sheet and music sheets that are with lower quality, blurry, slanted, scanned with low resolution.

4.1.1 Incorporate Handwritten Music Sheets

The scope of our project is restricted to clean, printed images of music sheets. Due to the fact that a lot of musicians produce hand write their music sheets, it would be beneficial to extend our work so that handwritten music sheets can be processed. We can leverage techniques mentioned in [2] to study recognition methods for recognizing handwritten symbols.

4.1.2 Incorporate Low Quality Music Sheets

Pre-processing of the input image can also be enhanced to recognize images of low quality. There are cases that the input image is blurry or the scanning process produces low quality images. We can explore possibilities to process images with low resolutions and transform these to higher quality. Regarding music sheets that are tilted, we can leverage MATLAB functions like resize, rotate and locating putative matching points to produce images of the correct orientation to be processed in later steps.

4.2. Leverage Classification Methods

Existing classification methods like Naive Bayes, K Nearest Neighbors (KNN) and Convolutional Neural Network (CNN) can be leveraged in our framework for classification. [3] We can directly use the training images we have to build the training image set. Naive Bayes involves a learning process thus the offline is relatively longer compared to KNN. The entire KNN process is online, and no training is required. CNN requires longer training time compared to the Naive Bayes and KNN but produce relatively better results.

Our current classification methodology is relatively straightforward. In order to deal with more complex input data, we can explore the possibility of incorporating an existing classification framework to render better results for symbol recognition.

5. Discussion and Conclusion

5.1. Discussion

Since we implemented our entire framework on MATLAB, we researched a lot about MATLAB functions. For example, we researched the *soundsc* function, which scales the audio signal, and then plays the audio of the sound through speaker. Another function we researched and leveraged greatly was the *subplot* function, with which we can combine multiple images, and plot all of them in one. Also, during the classification phase, we first used a small set of training images (~ 20), and learned that template matching cannot be done when the size of the training images is too small.

With these techniques learned, we feel more comfortable with MATLAB functions, image recognition techniques and classification steps. We feel more proficient to use Computer Vision techniques in other projects and MATLAB to aid us with our future research.

5.2. Conclusion

In this project, we managed to implement our original goal, which is to take an image of a printed music sheet as input, perform staff line detection and removal, segmentation, symbol recognition, note identification and music transformation, and play the song as the output. The entire project is done on MATLAB.

Additional work can be done to extend the scope of input data to include low-quality images and classification algorithms can be incorporated to render better classification results.

6. Statement of Individual Contribution

Below is a list of contribution made by each team member.

1. Felicia Chandra implemented the segmentation and music transformation.
2. Hao Zhang implemented the staff lines detection and removal, and note identification.
3. Yaning Liang collected music sheets and training images, and implemented symbol recognition.

References

[1] Matlab answers: create a piece of music using matlab?? `http://www.mathworks.com/matlabcentral/answers/48717-create-a-piece-of-music-using-matlab`.

[2] S. Levin, P. Clough, and M. Sanderson. Assessing the effectiveness of pen-based input queries. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 437–438, New York, NY, USA, 2003. ACM.

[3] R. Zhong and T. Tezuka. Parametric learning of deep convolutional neural network. In *Proceedings of the 19th International Database Engineering &#38; Applications Symposium*, IDEAS '15, pages 226–227, New York, NY, USA, 2014. ACM.