Final project report

EECS 452
Fall 2015

Title of Project: Sheet Music Translator

Team members: Ruohan Luo, Yuxiang Mu, Ziyou Su, Rui Zhou

1. Introduction and overview of project

As a necessary and useful skill in music learning, reading sheet music is a prerequisite for playing instruments. However, music amateurs are always having difficulties remembering music symbols and notations and they might not know how the notes should sound like. Sometimes it takes a long time to remember what these symbols and notations mean. More often than not, they have no idea of which key or string they should press, neither do they know how long the pitch should last. Sometimes they learn to play a song by remembering the sequence of keys or strings they press without knowing which symbol each key or string corresponds to.

Therefore, we designed an embedded device, based on a Raspberry Pi, a five mega pixel camera, and a player, which can translate handwritten or printed sheet music into acoustic sound. The input is a photo of sheet music, and the output is the corresponding music played by Raspberry Pi audio part along with a WAV file which can be used in other digital devices. The users can also choose the output instrumental sound between pure tone and bell tone. It is a useful device for music beginners, especially a useful educational tool for kids, and it can be re-implemented as a cellphone or PC software afterwards. People can use our device to learn sheet music, or they can use it to test their self-composed songs.

The project is mainly based on image processing technique. After capturing a picture by camera, the translator firstly binarize the image and corrects the image distortion using Hough Line Detection function proposed in [1]. Then it identifies the positions of staff lines and segment each symbol apart by vertical and horizontal projection. After that, it detects the shape of each symbol and the pitch of each note based on our own feature detection algorithm. Finally, it output the instrumental sound under user's choice through DDS.

During the expo, we let the viewers draw music notes on a blank sheet or randomly picked a pre-written sheet music, then we used the camera to take a picture of the sheet music, and output the acoustic sound and the corresponding WAV file. The accurate rate during the EXPO was beyond 90%.

Due to the limited amount of time we had, the current version of our project cannot process very complex sheet music. Overall, our final product meets the basic expectations we proposed in the beginning of the course.

2. Project Description

2.1 Concept Overview

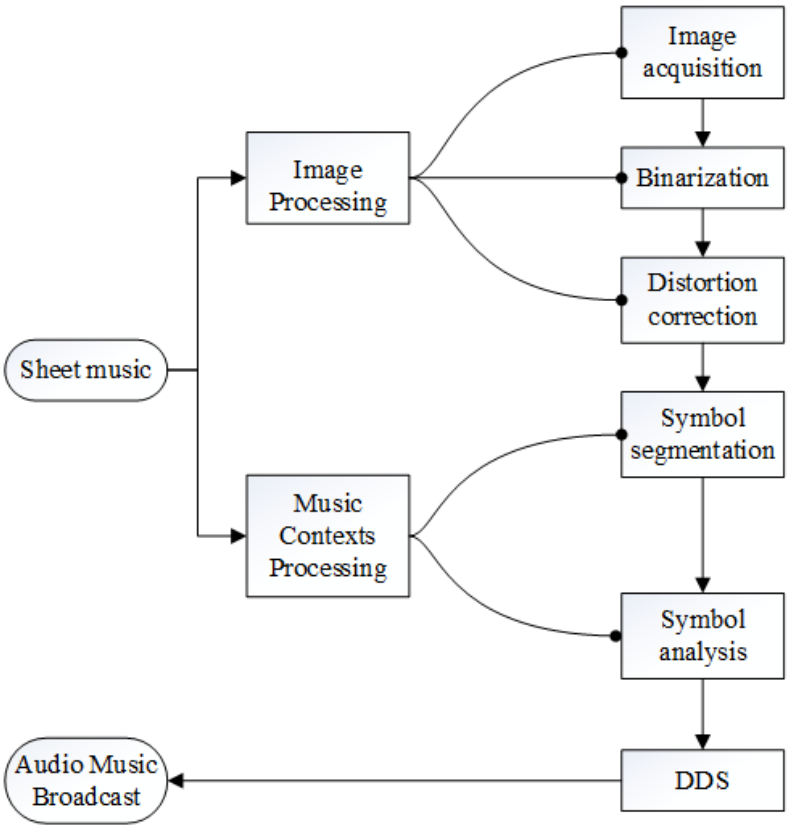

Figure 1. Design Overview

The Project realizes translation from sheet music to audio music through three stages: image processing, music contexts processing and music broadcast. Each stage consists of separate steps as listed in figure 1.

2.2 Pre-image Processing

a. Image Acquisition

To start with, we need to capture an image of the sheet music. This process is realized using the Vilros 5MP camera module, which is compatible with the Raspberry Pi. The software we developed prompts a request for the user to start capturing an image and provides previewing until the camera is adjusted to the desired position. The image captured from the camera is stored in RGB format as shown in figure 2.

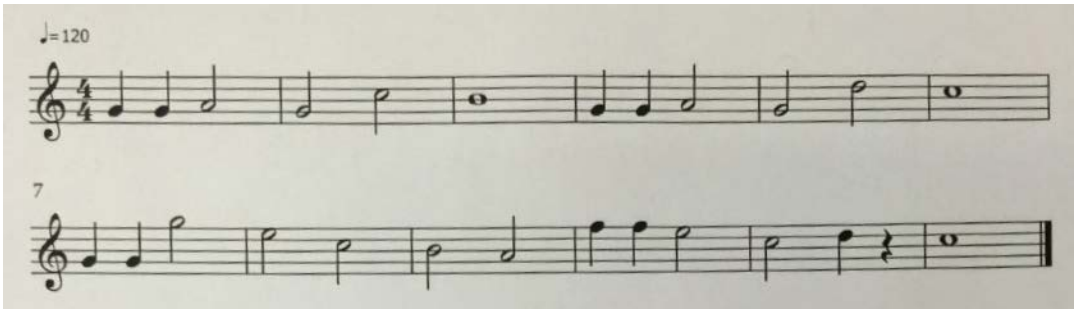

Figure 2. Raw image in RGB format

b. Binarization

The raw image obtained after the first step is in color, which is not necessary for the process of music contexts identification and even adds to the complexity. To remove this extra complexity, we employ image binarization to convert the initial color image to a single-bit black & white image. To accomplish this, we first transformed the 3-channel RGB image to a single channel of gray values with the formula below.

$$Y = 0.2989R+0.587G+0.114B$$

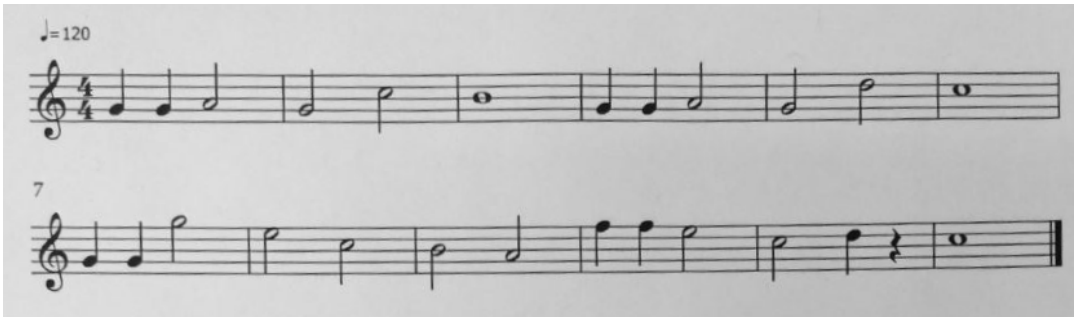

Figure 3. Single-channel grayscale image

With the image in grayscale (figure 3), the simplest way of binarization is to choose a global threshold value for the entire image, and classify all pixels with values above the threshold as white (0), and all other pixels as black (1). However, since the illumination varies spatially in real life, our project employed the more robust adaptive thresholding technique. The threshold is determined as the mean, medium or a weighed sum (Niblack's technique, Sauvola's technique) [2] of the pixel values within a small sized window and varies dynamically over the entire image. The project finally uses local mean as the dynamic threshold value to achieve relative higher processing speed.

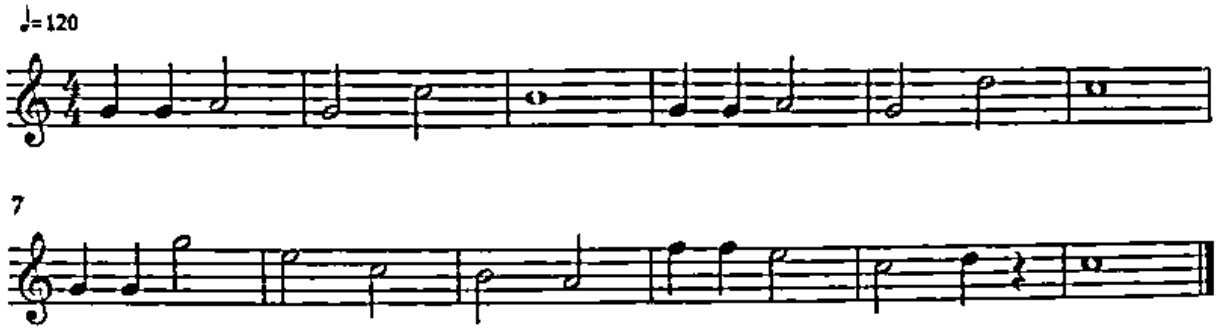

Figure 4. Single-bit binary image

After binarization, each pixel of the image can be expressed in binary values 0 & 1, where the white background is expressed as 0s and the black music contexts (foreground) are expressed in 1s.

c. Distortion Correction

After the image is classified into background and foreground, the next step is to modify the image in case of any distortion resulted from improper camera placement. When the camera sensor is rotated, the staff lines are skewed and not parallel to the top/bottom of the image. This step promises that all staff lines are horizontally placed which is crucial for further implementation.

Hough line detection algorithm is employed here to identify the straight lines in the image. The detected lines whose length exceeds a specific threshold are considered to be staff lines. The average angle of inclination of these staff lines are used as the rotation angle of the entire image.

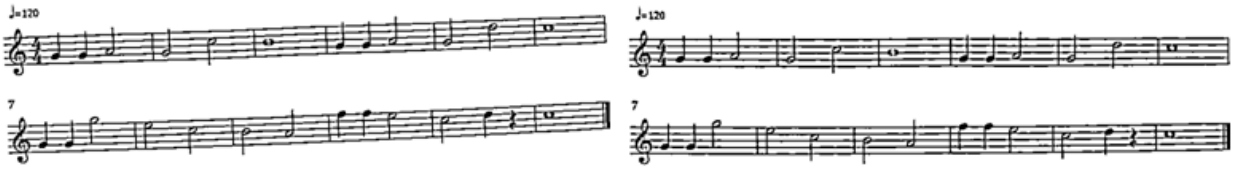

Figure 5. Image before & after distortion correction

2.3 Symbol Segmentation

a. Staff Line Extraction

With the processed image provided, every 5 consecutive staff lines are detected and extracted for later analysis. To achieve the staff line detection, the distribution of pixels is analyzed.

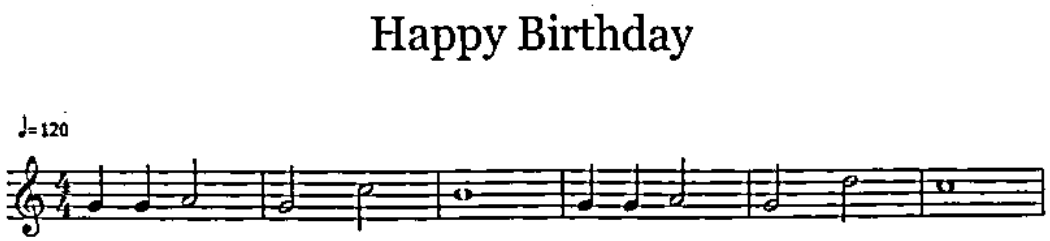

Figure 6. Staff lines after pre-image processing

As observed from figure 6, the staff lines contain much more black pixels than other lines. Therefore, we project the image horizontally and sum up the number of black pixels in each horizontal line.

Figure 7. Horizontal projection intensity diagram

The projection intensity diagram is shown in figure 7, from which we can obtain the positions of regions of the staff lines. Then we can extract the images of every 5 consecutive staff lines. Figure 8 shows our extraction results of another example.

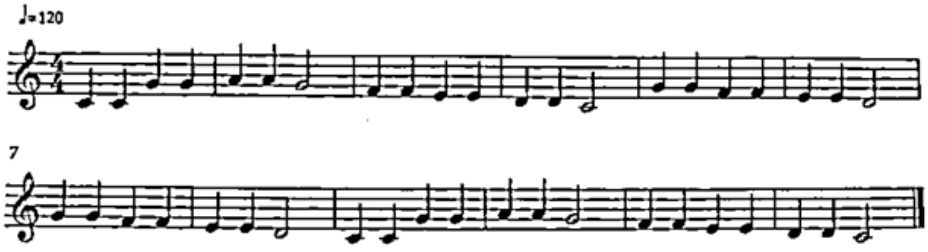

Figure 8. Staff lines before extraction

As shown in figure 8, there are two musical lines which contain 10 staff lines. The following figures shows the result images of staff line extraction.

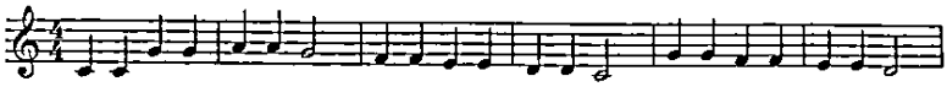

Figure 9. Extraction of the first 5 consecutive staff lines

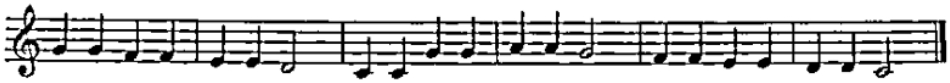

Figure 10. Extraction of the second 5 consecutive staff lines

As shown above, we successfully extract every 5 consecutive lines. We will use these plots of staff line extraction in our symbol segmentation.

b. Symbol Extraction

Once the music lines are extracted, we segment the music symbols in a similar way. Since the music symbols are consisted of black pixels, we project the image vertically to find the position of each symbol.

Figure 11. Symbol Segmentation

As illustrated in figure 11, the entire music line is segmented with each music symbol in a separate block, which can be used in later analysis.

2.4 Symbol Analysis

Now that we have the segmentations containing single music note, we are able to analyze the notes and translate the pitch and duration values for music broadcast. The segmentation is presented as a 2-dimensional grayscale array containing either 1 (black) or 0 (white). We accomplish our analysis through the following steps.

a. Staff line detection

Our work start with detection of staff lines and record their vertical location. The staff line is mostly composed by black pixels so we simply count black pixels in each line. We select such lines that black pixels account for 90% of whole pixels in line as staff lines and store their vertical locations, which act as reference for following detection.

b. Define location of notes

To start with, we define that our detection region contains 18 notes ranging from low G to high C. The notes are numbered from 0 to 17.

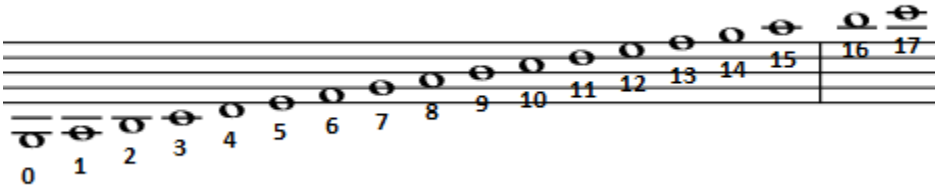

Figure 12. Range of notes

Our task is to define all of 18 tones' vertical location in our segmentation. At first glance, we can easily find that several tones lie exactly on the staff lines, whose locations are already known. Thus, we first define the vertical location of tones 5, 7, 9, 11 and 13 as same as the location of corresponding staff lines.

Next, we need to find the average gap between two continuous staff line. Basically, five staff lines have four gaps. We calculate the average gap by subtracting the location of the lowest staff line by location of the highest staff line and divided by four. Notice that the distance between two continuous tones is half of this gap. Hence we can define the vertical location of rest of the tones. The result is visualized below.

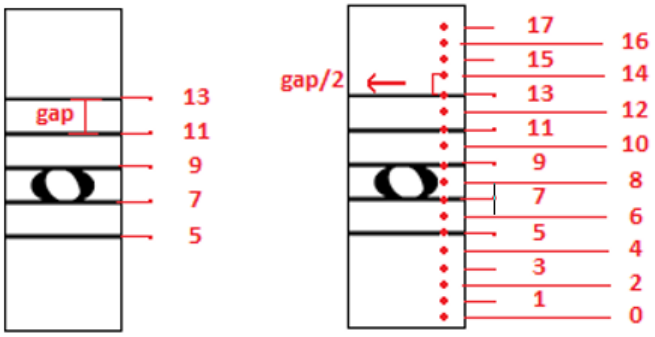

Figure 13. Vertical locations of tones

c. Delete staff lines

After we define the location of tones, the staff lines become useless so we delete them. Using the location of the staff lines we recorded before, we keep black pixels that go across the music note and tune other black pixels into white.

Figure 14. Deletion of staff lines

d. Determine kinds of symbols

Thanks to the deletion of staff lines, our segmentation contains pure music symbols only, which makes our detection much more straightforward. Firstly, we can classify our symbols by counting the vertical region. The figure below shows the difference of these symbols.

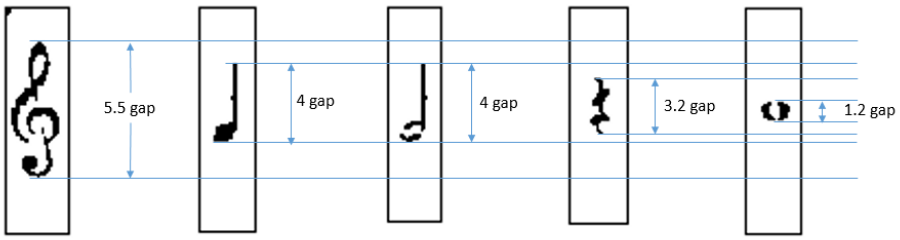

Figure 15. Musical symbol classification using vertical range

Taking advantage of this difference, we can detect the vertical length of the pure symbols and identify the clef, rest mark, the whole note and half note/quarter note, the only two symbols that we cannot separate are half note and quarter note. We need more specific recognition between these two.

The main difference half note and quarter note is that a half note has hollow ellipse head while a quarter note has a solid one, which leads to a difference in black pixels distribution. Therefore, we select a rectangular area surrounding the head part of the note and calculate the percentage of black pixels in this area. The figure below shows calculation result of both a standard half note and a standard quarter note. The difference is obvious so we can separate them easily.

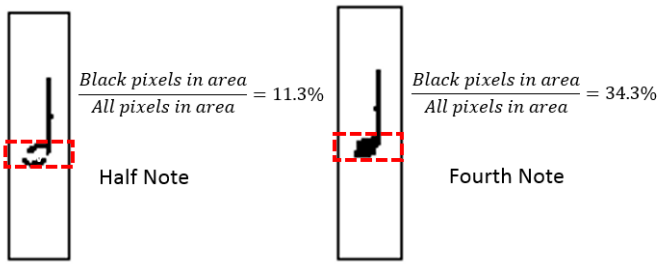

Figure 16. Musical symbol classification using black pixel distribution

e. Determine the pitch of music note

For quarter notes, half notes and whole notes, we also need to analyze the pitch. We notice that these notes all have ellipse heads and the pitch is determined by the vertical location of the heads. Hence we take advantage of the elliptical shape and detect the upper bond and lower bond of the ellipse head and calculate the vertical location of the center.

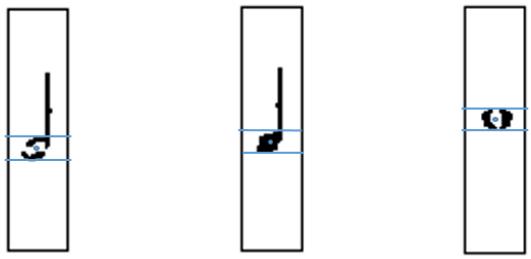

Figure 17. Music notes with different shapes

Remember that we have defined vertical locations of all 18 tones previously. Now we can compare the center location with them and determine which tone is the closest to the center.

Record that tone as the pitch of this music note. That is all for our analysis for single segmentation.

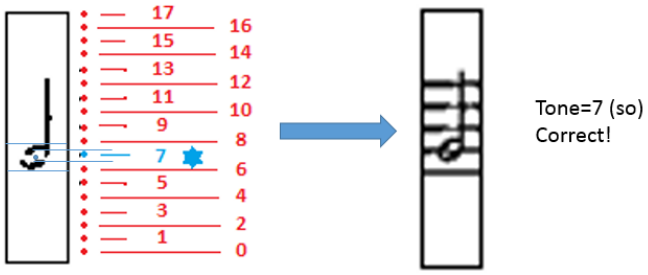

Figure 18. Pitch determination of music notes

We repeat this process to all of the segmentation and accumulate the tunes and beats information. Finally, we successfully decode the information in the sheet music and ready for music broadcast.

2.5 Music Broadcast

With information of the pitch & duration of each note known, we generate a pure tone sound of the sheet music using the direct digital synthesis method. The frequency of the sine wave generated is characterized by the pitch of the corresponding note.

Pitch	G3	A3	B3	C4	D4	E4	F4	G4	A4	B4
Frequency	196	220	246.9	261.6	293.7	329.6	349.2	392	440	493.9

Figure 19. Different pitches with corresponding frequencies

We also provide another more complex bell sound as the output with a combination of waveforms rather than a single sinewave.

3. Milestones

For milestone 1, our original goal was to build a complete model in Matlab. In reality, we built almost all the algorithms in Matlab except the symbol recognition part.

For milestone 2, we originally wanted to translate our Matlab code into the Simulink model and implement it into the Raspberry Pi. However, it turned out that many algorithms in Matlab are not compatible with Simulink. Therefore, we rewrote our algorithms in Python and implement them into the hardware. Fortunately, the performance is still good.

4. Demonstration

We demonstrated our Sheet Music Translator at the Design Expo and at the final presentation. The following figure shows the setup of our device.

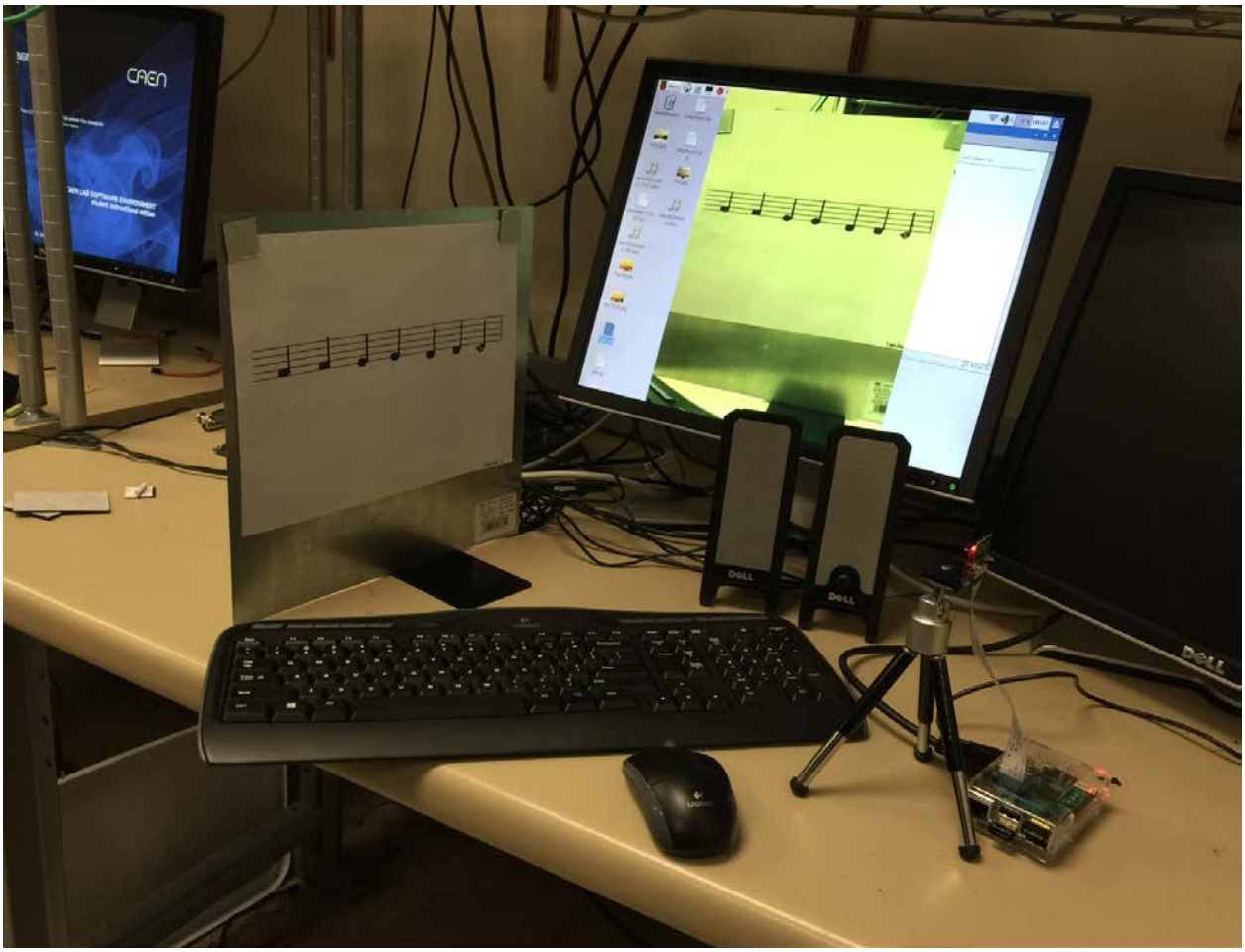

Figure 20: Setup of our device

We connected our Raspberry Pi with the monitor. The Raspberry Pi was controller by the wireless keyboard and mouse. The Raspberry Pi Camera was fixed on a tripod. The music sheet was fixed on a vertical steel board. The speaker was also connected to the Raspberry Pi. During

our demo, we firstly took a picture of our music sheet. Once we pressed our keyboard, the Python program allowed us to see the camera image on the monitor. Then we pressed the keyboard to take a proper picture of the music sheet. Next, the Python program began to process the image and translate it into audio music. Finally, the audio music could be broadcasted by the speaker.

During our Design Expo and final presentation, we tried to broadcast several music sheets (such as “Happy Birthday” and “Twinkle Twinkle Little Star”) in pure tone. The results of translated audio music were all correct. Then we also tried to display the translated audio music in bell tone. The sound became more beautiful. In conclusion, our demonstrations of project in the Design Expo and the final presentation were successful.

5. Contributions of each member of team

Describe the contributions of each team member to the project in the form of a table as indicated in the example below

Team Member	Contribution	Effort
Ruohan Luo	Feature recognition algorithm Instrumental sound output	25%
Yuxiang Mu	Symbol segmentation Feature recognition algorithm	25%
Ziyou Su	Image binarization Neural network design Hardware setup	25%
Rui Zhou	Image distortion correction Neural network design Hardware setup	25%

6. Future Improvement

The project has not reached its full potential due to the limited time and resources we have. Some aspects that can be further improved are listed below.

- Enhance implementation speed with more efficient code & hardware with higher speed
- Enlarge range of pitches using larger segmentation blocks

- Develop better recognition algorithms to distinguish more types of symbols
- Provide more options for output sound

6. References and citations

[1] Hough Line Transform. (n.d.). Retrieved December 17, 2015, from http://opencv-pythontutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

[2] J. Sauvola and M. Pietikainen, "Adaptive document image binarization," Pattern Recognition 33(2), pp. 225–236, 2000.

[3]Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. In IEEE Transactions on Systems, Man and Cybernetics, Vol SMC-9, No.1

[4] OpenCV: Canny Edge Detection. (n.d.). Retrieved December 17, 2015, from http://docs.opencv.org/master/da/d22/tutorial_py_canny.html#gsc.tab=0