

Capstone Program Report

Music Sheet Maker

Submitted by:

Haocong Wang (mw814)

Sifan Yuan (sy609)

Advisor[s]:

Prof. Yingying Chen

Yilin Yang

May 7, 2020

Electrical and Computer Engineering Department

Rutgers University, Piscataway, NJ 08854

1. Introduction

The Transcription of music refers to the generating of the music note from a given part of music. The traditional music transcriptions have been written by hands. This is absolutely a time-consuming job, and, it also requires the people who do this has a very well musical knowledge base. What’s more, it also has several extra requirements, for example, the place should not be very crowdy. These limitations of the traditional music transcription make people hard to write or record the music they want to. To let everyone be able to “write” music, we want to find a way to do music transcription automatically.

Our goal is to develop an application to write music sheet automatically with a given part of the audio on a web-based application.

After the research of the related work, we find that there are many excellent works to refer till now, but there are still few application, especially mobile application, exists. When it come to the application, there are more things to concern for example, how to get the audio, how to generate the noise of the data. To solve these and also do a good transcription of music, we need to do more.

The whole project can be probably divided into 3 main parts: the audio data fetching and preprocessing part, the music transcription part, and the output and presenting part. In the first part, we will focus on how to get the data from the device and to generate the dataset which will be used in the transcription part. In the second part, our goal is to fetch and analyze the features of the data. The last part is to transcript the features into a MIDI file, which can be directly read and understood by computers as music sheet.

The introduction should include:

**2. Methods and Results**

Our program can be divided into three parts: audio recording, audio data fetching and analyzing and creating a MIDI for the audio. In this section, we will introduce our methods and approaches for each part with details.

**2.1 Audio Recording**

In this step, we first play a short part of the piano audio, which is played by GarageBand App on iPad. Then we make use of the pyaudio lib in Python to record the audio. The sampling rate is set at 44100 Hz, which is high enough to avoid alias for most music signal. The audio will be recorded into a wav file finally.

**2.2 Audio Data Fetching and Analyzing**

In this part, we use Spectrogram [6] and Chromagram [7] to do the data preprocessing part. Then, we extract the detail data from these grams to do the analyzing and transcription work. Next, we will describe the method in detail.

**2.2.1 Spectrogram**

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams. It is used extensively in the fields of music, linguistics, sonar, radar, speech processing, seismology, and others.

The spectrogram we used has two geometric dimensions: one axis represents time, and the other axis represents frequency; a third dimension indicating the amplitude of a particular frequency at a particular time is represented by the intensity or color of each point in the image.

With spectrogram, we can connect the frequency domain and the time domain together. This will help us to do the later splitting work.

A picture containing indoor, electronics

Description automatically generated

**2.2.2 Chromagram**

In music, the term chroma feature or chromagram closely relates to the twelve different pitch classes. Chroma-based features, which are also referred to as "pitch class profiles", are a powerful tool for analyzing music whose pitches can be meaningfully categorized (often into twelve categories) and whose tuning approximates to the equal-tempered scale. One main property of chroma features is that they capture harmonic and melodic characteristics of music, while being 图片包含 游戏机, 画

描述已自动生成robust to changes in timbre and instrumentation.

There are many ways for converting an audio recording into a chromagram. For example, the conversion of an audio recording into a chromagram may be performed either by using short-time Fourier transforms in combination with binning strategies or by employing suitable multirate filter banks.

**2.2.3 Data Analyzing**

We have two sets of data (chromagram and spectrogram). What we should do now can be divided into two parts, the first part is the splitting and the second part is the transition. The splitting work is to divide the whole audio by every single note. The transition part is to find out the frequency of every split note then compare this frequency with the standard library we set up before to get the exact pitch of these notes.

It seems easy to do this, but when we start the work, we faced a lot of problem.

The main problem we met is that the harmonics occurring at whole-number multiples of the fundamental frequency, in another word, every same note’s frequency will reflect in all octaves. The frequency of a single pitch will reflect on different frequency bands. So, it is difficult to determine which octave (from a lower ***do*** to a higher ***do***) does the pitch in. This also can’t be solved by the chroma gram too. Chromagram can get the exactly pitch of different frequency, but it can’t figure out which octave does this pitch in.

To solve this problem, we set up a sliding-window, which have a static length. We first find out the maximum window from all windows we get. Because when a note is played, it will create noise in the frequency nearby. The true note’s frequency volume adds up with the volume of the nearby frequency will definitely be the maximum window. After finding out the maximum window, we then find out the frequency with the largest volume in this window, and that will be the one we need.

With all of the resource above, we can now extract some simple piano music.

屏幕上写着字

描述已自动生成

The pitches and times we extracted

**2.3 Creating a MIDI**

**2.3.1 MIDI**

MIDI is the abbreviation of Musical Instrument Digital Interface. It is a technical standard that describes a communications protocol, digital interface, and electrical connectors that connect a wide variety of electronic musical instruments, computers, and related audio devices for playing, editing and recording music.

There are already a large amount of existing methods to read MIDI as the music sheet in computer. So, in another word, MIDI itself is the music sheet that can be read and understood by the computer.

手机屏幕截图

描述已自动生成

How MIDI looks like in computer

**2.3.2 How to create MIDI**

In music, there are three basic elements: pitch, rhythm and volume. In this project, we didn’t put volume of sound into consideration. However, we considered pitch and rhythm carefully. As mentioned above, in music, different notes represent different pitches and in science, different pitches represent different frequencies. Rhythm represents how fast the music is played, which, in another word, is the speed of the music.

In the previous step, we have already successfully fetched the list of frequencies and the list of time for each note, which means we have the information of how each note sound like and last for. In this step, we can take advantage of these two lists to create our MIDI file for the given part of the piano audio.

There are many libs that can create MIDI file. Most of them requires user to label pitch and lasting time for each note, performing as the same way as writing music sheet by hand. In our project, to create MIDI, we utilized the mido [4] lib in Python.

In mido, each note has its own serial number. However, it is not given by the official document, so we have to test it by ourselves. After several experiments, we found most note’s serial number. The figure below shows an example of the results.

图片包含 黑色, 大, 监控, 侧面

描述已自动生成

After this step, we now have a database of all serial numbers. Besides this database, we also created another list for frequencies of each note, as shown below:

蓝色的标志

描述已自动生成

Now with this two collections and the two lists achieved from the previous step, we can now write our own music sheet. We will discuss results in the next section.

**2.4 Results and Problems**

Our program runs well on most cases. Here are some samples of the MIDI file we created successfully and accurately.

手机屏幕截图

描述已自动生成

图片包含 游戏机

描述已自动生成图片包含 游戏机

描述已自动生成

However, sometimes there are some errors. The most commonly happened error is octave error. For example, when we play C3, known as the middle C, the computer sometimes recognized it as C4, one octave higher than C3. This is because one note carries frequencies of the same notes in different octaves, so sometimes the computer may recognize it as another note. Apart from octave errors, two notes that are close to each other, such as *do* and *re*, could be recognized wrongly.

**3.** **Cost and Sustainability Analysis**

Our final product is a web-base application. With a given part of an audio, out project can record it into a wav file and output its music sheet, which is the MIDI file.

**3.1 Android Application**

Our original plan for the final product is to make an android application. However, due to the limitation of time and hands, we cannot finish the design and implementation of android application, so we created a web-based application instead. In the future work, we can focus on the android application, which can be used for commercial purpose.

**3.2 Future Improvement**

In the future work, we need to make our program more accurate and deal with more complicated music. Right now, our project can only write music sheet for polyphonic [1] [3], but the real music will be much more complicated than that. We also need to deal with all the errors mentioned above to increase the accuracy. Besides, our program can only deal with slow music now. That is another problem we need to solve.

**3.2 Research Development**

In our original plan, our project should be able to recognize human voice and different musical instruments. Sometimes the music could be sung by human instead of played by instruments, so we need to recognize human voice and write music sheet for it. Apart from that, we also planned to recognize different musical instruments [5]. These two parts both require machine learning skills and can be used for research purpose.

**4. Conclusion**

In conclusion, we created a web-based application that can record the given part of an audio and write music sheet for it automatically. The programming language we used is Python. Our program runs well on most cases. Future development includes developing an android application, recognizing human voice and different musical instruments.

**5. Acknowledgments**

We sincerely thank for our advisor, Prof. Yingying Chen and her PhD student, Yilin Yang.

**6. Reference**

[1] M. P. Ryynanen and A. Klapuri, "Polyphonic music transcription using note event modeling," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, New Paltz, NY, 2005, pp. 319-322, doi: 10.1109/ASPAA.2005.1540233.

[2] Benetos, Emmanouil, et al. "Automatic music transcription: challenges and future directions." *Journal of Intelligent Information Systems* 41.3 (2013): 407-434.

[3] Martin, Keith D. "Automatic transcription of simple polyphonic music." *Proceedings of 3rd Joint meeting of the Acoustical Society of America and Japan*. 1996.

[4]<http://p1htmlkernalweb.mybluemix.net/articles/%E7%94%A8python%E7%9A%84mido%E5%BA%93%E6%92%AD%E6%94%BE%E9%9F%B3%E4%B9%90_3894581_csdn.html>

[5] Sarnatskyi, Vladyslav, et al. "Music Transcription by Deep Learning with Data and" Artificial Semantic" Augmentation." *arXiv preprint arXiv:1712.03228* (2017).

[6] J. Dennis, H. D. Tran and H. Li, "Spectrogram Image Feature for Sound Event Classification in Mismatched Conditions," in IEEE Signal Processing Letters, vol. 18, no. 2, pp. 130-133, Feb. 2011, doi: 10.1109/LSP.2010.2100380.

[7] Müller, Meinard. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.