CHORD AND BEAT TRACKER IN MATLAB

Isaac Mosebrook

University of Rochester Department of Audio and Music Engineering

ABSTRACT

Digital signal processing is a useful tool for helping musicians practice and learn their instrument. This project's aim is to build a Matlab application that provides visual ques to a keyboard player to help them learn or play a song. The user can pick any song they have on their computer and the program will analyze and play back what chords they should play at what time in the song. In the paper the specific details of this analysis are described, specifically the three step process of getting a chromagram, detecting onsets, and then displaying the results in a graphical user interface.

***Index Terms*—** Audio, STFT, Chromagram, Onset, GUI

1. INTRODUCTION

To implement the above application, two features of the input audio need to be visualized. During the analysis phase, the harmonic content and note onsets are extracted from the input. Once this is complete, the features can be visualized while the song is played using Matlab's GUI features. The goal is to make the interface as similar to a real keyboard as possible to make it as easy to use as possible.

2. SHORT-TIME FOURIER TRANSFORM

The first step is to transform the song the user selected from the time domain into the frequency domain. To do this we can use the Short-time Fourier Transform on the input signal. This was done using the built in Matlab function, which provides the frequency content as a function of time.

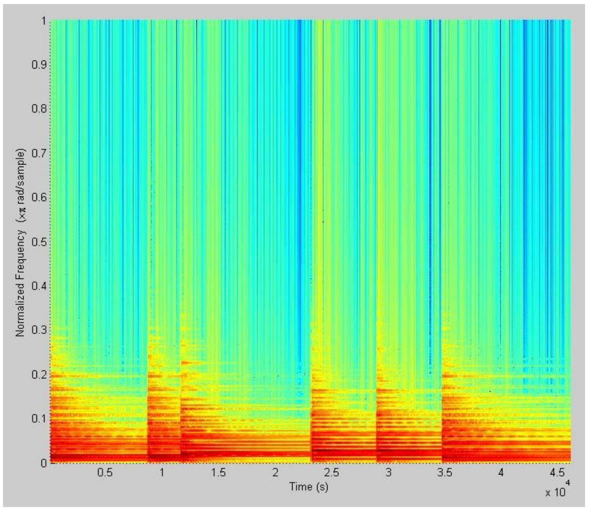

Figure 1: Short-Time Fourier Transform of Input

3. CHROMAGRAM

Now we have the information of what frequencies are present in the song at what time. However, what we need is to know what musical notes these frequencies are represented by on the keyboard. We can obtain this information instead by multiplying the STFT by a conversion matrix.

A conversion matrix uses knowledge of the relationship between a frequency and its corresponding note to sum all the octaves together into 12 separate note bins. The matrix is created by knowledge of the equal temperament system in music, described by the following equation.

$$f_m = f_{ref} \times 2^{m/12}$$

From here, all that is needed is to know which f_m each frequency bin corresponds to. This can be discovered with the following formula where F_s is the sampling rate, M is the number of frequency bins, and k is the current frequency bin.

$$f_m = \frac{F_s}{M} k$$

Now there is a complete system to convert the frequencies into notes. The resulting conversion matrix is as follows.

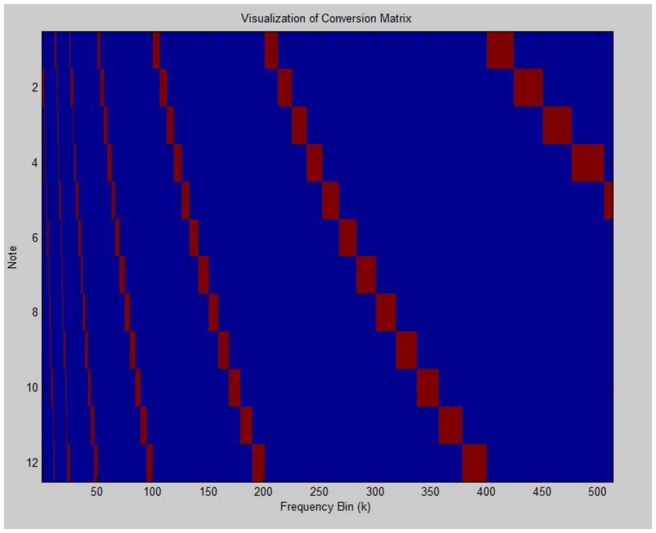

Figure 2: Visualization of Conversion Matrix

Left multiplying the STFT with this conversion matrix results in a chromagram, shown in the following picture.

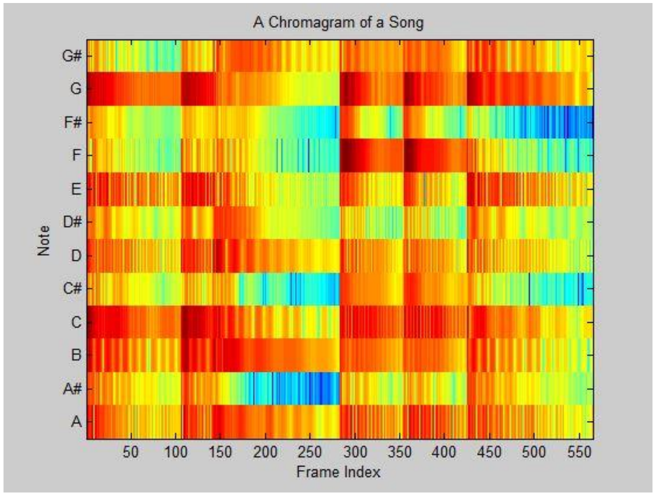

Figure 3: Chromagram of Input

4. ONSET DETECTION

Now we have the keys that need to be played in the song, but we still need to find when to play them. To do this, we can use an onset detection algorithm. Since we already have the spectrogram, we don't need to get the envelope and can skip right to taking the derivate. To obtain the rate of change we can use the following formula.

$$\Delta_{\text{Energy}}(n) := |\mathrm{E}_w^x(n+1) - \mathrm{E}_w^x(n)|$$

When a key sudden goes from no energy (not being played) to high energy (starts playing), the derivative will be a large value. If the value is high enough, then we can safely assume that the key started playing at that time. After some experimenting, I picked a threshold which seemed to give the best results. If the derivative was above this threshold, then a one is put into an onset array to indicate when that key starts playing.

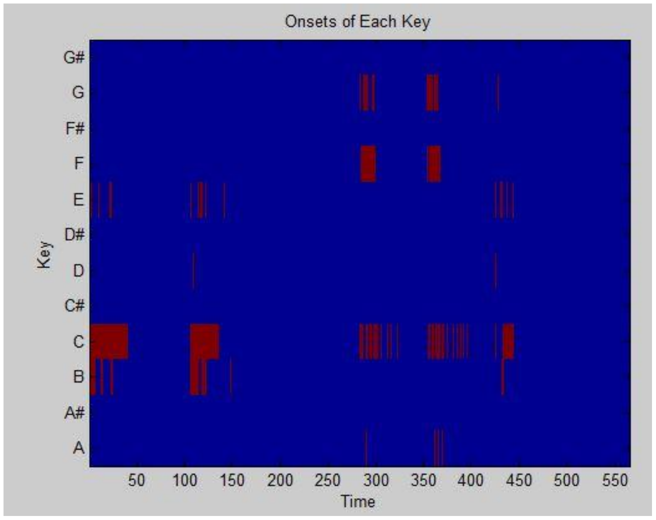

Figure 4: Onsets for All Twelve Notes

Now we have 2-d array with the onsets for all twelve keys on the keyboard. If we also want to view the overall beat of the song, we can simply sum all twelve of these vectors to see all the beats at once.

One important note to make is that the time axis on the onsets is not one sample at a time. Since the onset detection algorithm looks at the spectrogram, it is looking at the windowed signal. Therefore, each index in the onset array is one hop size apart. This information is critical to playing back the onsets and music in sync.

5. GUI

Now the analysis phase is complete and it's time to visually represent the features of the input audio. There should be 12 different graphical objects to represent each key, which are controlled by their respective row in the onset array. If the array has a one at the current time instant, then the graphical object representing that key should change state to indicate it is to be played.

This can be done in Matlab by plotting the onset array. When a one is reached, that key will fill the vertical space indicating that it is to be played. Additionally, the overall beat will flash in the next plot location next to the keyboard in a different color. The graph is refreshed every hop size as mentioned in the previous section.

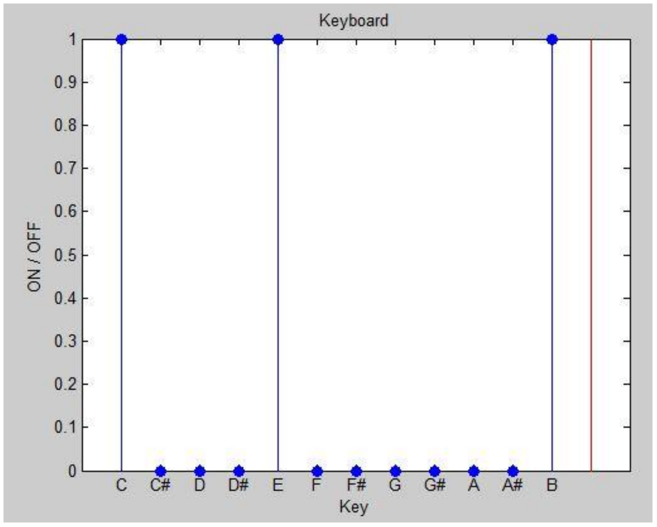

Figure 5: GUI

6. RESULTS

Overall, the program succeeds in doing what it needs to do. However, there is one slight complication in each area that prevents it from being a fully practical program.

For the chromagram, the accuracy of note detection isn’t quite good enough. For example, when only C is playing, the program may also detect a large energy at C#. This issue is caused by the construction of the conversion matrix, but the bug was unable to be identified.

The onset detection works well. It may seem like it detects incorrect notes, but this is only because of the chromagram as just mentioned. The only issue is that the threshold needs to be set for each song because of different loudness levels. A solution for this may be to create an adaptive threshold that analyzes the average energy over a certain span of time and picks a threshold based on this average and the max peak.

The final issue is with the synchronization of the GUI and the song playback. Currently the plot is not able to update at the proper rate to keep up with the song, so it is much slower than the song playback. Otherwise the GUI seems to work fine.

7. FUTURE WORK

One problem with the current implementation is that we only know when to start playing each note, but not when to stop playing them. One naïve solution is to have the program hold the note until the next note starts playing. While this generally would work, but sometimes the right and left hands play different rhythmic parts. A much better solution would be to implement some decay onto each note based on the velocity with which that note was hit.

A second improvement that can be made is the overall beat tracker. Currently it shows the beats of how the song was performed in the recorded, but this is only relative to the actual tempo. A better solution would be to perform tempo detection on the song and display the detected tempo as a metronome instead, since it is more consistent than the recorded performance.

8. CONCLUSIONS

This paper outlined the steps to create a chord and beat tracking program in Matlab. It starts with a Short-time Fourier Transform of a song. Then a chromagram is obtained by multiplication with a conversion matrix. The onsets for each note are then analyzed. Finally, the data is presented in a GUI using Matlab plots.

Audio signal processing is very useful and this project is an example of the benefits it can provide. With the bugs fixed and the future work added in, this program is actually a very handy tool for learning songs or at least getting a quick idea of what is going on in a song.

12. REFERENCES

[1] ZoÌˆ lzer, Udo. *DAFX: Digital Audio Effects*. Chichester: Wiley, 2011. Print.

[2] Zölzer, Udo. *Digital audio signal processing*. Chichester: Wiley, 2008. Print.