15-112 Term Project Proposal

Yuyi Shen (yuyis1) Section N

February 18, 2017

1 Required python modules

- numpy
- pyaudio
- pygame
- wave

2 The Problem

2.1 Summary

This term project will be constituted of two components, an audio processing component and a gameplay component. The objective of the audio processor is to take in a .wav file and write the time stamps associated with new notes into a text file, 'beats.txt'. The game will play back the music used to form the contents of 'beats.txt' and have the player guess at the weights for a Fourier series approximation of the music at the times stored within 'beats.txt'.

2.2 Usage Description

When termProject.py is run from command prompt, a pygame window will be generated containing a rudimentary file selection screen. The top option will always be the aforementioned 'beats.txt', whereas the following options will be any .wav files detected within the term project program's directory.

Selecting 'beats.txt' will bring the user to the game, utilizing the current contents of 'beats.txt'. Selecting one of the .wav files will cause termProject.py to utilize its audio processing function on it to generate new contents for 'beats.txt', and then bring up the game. Note that the audio processing function requires significant computing resources and may take longer than half an hour due to the size of audio files.

3 The Solution

3.1 Note Detection

3.1.1 Algorithm Summary

The underlying principle of this term project's note detection algorithm is the inability of the Fourier series to accurately model non-periodic signals. In other words, to detect sudden changes in an audio signal, such as the onset of a note, one should generate a sinusoidal model of that signal using the Fourier transform and determine the error associated with that model as a function of time. Spikes in the error will generally mark the onset of musical notes.

3.1.2 Fourier Analysis of the Audio

To synthesize a sinusoidal model of the audio, the Short Time Fourier Transform of that audio is required. Essentially, a function outputting the FFT of the audio as a function of time needs to be generated. In order to accomplish this, I first wrote a Kaiser window function, to serve as the FFT window for the audio data. Then, I wrote an algorithm that slid the Kaiser window across the entirety of the audio data and took the FFT of that windowed data.

3.1.3 Peak Continuation

Once the short time Fourier transform was taken of the audio, my program fed it into a peak continuation algorithm to generate stable frequency trajectories. To generate these trajectories, I wrote a function to serve as a peak detector for the STFT results. Then, I fed those peaks into a function that iterated over the STFT, generated frequency "guides" that followed frequency

peaks over time, and returned the list of guides.

3.1.4 Sinusoidal Model Generation and Error Calculation

With the list of frequency "guides" to serve as a map to the audio waveform over time, I wrote a function to calculate what the audio waveform would be according to the frequency "guides" and store the results in a numpy array. Then, I subtracted the modeled waveform from the actual waveform, took the absolute value of the resulting residuals, and passed the error values into an envelope detector followed by a low pass filter. Lastly, I used a peak detector function to find error spikes, and wrote the time stamps associated with those spikes into the text file 'beats.txt'.

3.2 Gameplay

3.2.1 Summary

The user has the option to play a game utilizing the beat detection results after successfully processing a .wav file. The game will play back the audio utilized to produce the contents of 'beats.txt' and, at the moments the audio processor detected notes, have the player guess at the weights of a Fourier series based approximation of the audio.

3.2.2 Guess Mechanism

The Fourier series based approximation of the audio will be graphed in a manner of bubble that will be popped when the player successfully guesses at the weights of the series. These guesses will be made via two sliders on the left hand side of the game display, which will appear when the player selects the "bubble" with the mouse.

3.2.3 Algorithm

At the time stamps listed in 'beats.txt', the game will take the FFT of the audio. A peak detection algorithm will be used to determine the 2 highest peaks in the FFT, and a function of these 2 most significant frequency components will be generated to serve as an approximation to the audio waveform. To guess at the weights utilized for these frequencies, the player will adjust sliders to the side of the game screen until the graph of his/her guess matches

the graph of the audio waveform approximation. At this point, the score is incremented and the waveform approximation disappears.