MULTI-FUNCTION AUDIO SYSTEM DESIGN

Youmeiyi Pan, Yuxuan Wang, Zhuohuang Zhang

University of Rochester Department of Electrical and Computer Engineering

ABSTRACT

Every piece of audio has its own properties, such as onsets, beats, the amplitude and the timbre, etc. These qualities can influence the feelings of the listeners. This project is aimed to design a multi-function audio system based on MATLAB to visualize some properties of the input audio and make it possible for users to manipulate the audio with some functions, for instance, adding reverberation effects to the output audio.

In this paper we will introduce the three functions our system can realize, which are waveform and FFT synchronized display, the LED simulation of onsets together with the amplitude and reverberation simulation of different environments. Some pictures of our designed GUI will be shown in the result part.

Index Terms— Audio, Waveform, FFT, Onsets, LED simulation, Reverberation, GUI

1. INTRODUCTION

For our design, the objective is to visualize some properties of the input audio. We extract four features of the input audio, waveform, FFT figure, onsets and the amplitude. For the first two properties, we make the display synchronized with the audio. For onsets and the amplitude visualization, we use sparkling and color density changing, respectively. Besides, users can generate reverberation effects of different environments with four options we provide. All the functions mentioned above will be realized on a MATLAB graphic user interface.

2. WAVEFORM

The waveform is the basic visualization of the audio file. It simply shows the value of each sample, or called amplitude, in the input signal. we can analyze the periodicity, the frequency, the amplitude and the transients of the signal from the waveform figure.

2.1. Realization

Our system need to show the waveform of the input signal when the audio is being played. The ideal result is that the output plot will update itself as time changes. However, the sampling rate of ordinary music is very high, generally 44100 Hz. The execution time for the program would be very long if we plot the points one by one. To speed up, we decide to plot a series of points at one time and synchronize the plots with the output audio using certain delays. We have tried two methods to realize this function.

The first one is concerning with axis moving and it can only be used to deal with the offline audio signal. When a piece of audio is input, the system will analyze the audio file and get the information such as the signal length, the sampling rate and the matrix containing all the elements of the signal. Then the system will have the entire plot of the whole signal. What we need to do is plotting the current part of the audio on GUI. To make sure the waveform figure can display synchronously with the music, the speed for MATLAB of changing the axis of the plot figure should match the speed of audio playing. So we calculate the actual playing time of the corresponding part of the signal and add a *tic toc* function to know the execution time for MATLAB to run this part. Then a *pause* function is added to create a time delay to make sure the output figure is synchronized with the current part of the audio. This method is the one we use in our system since the processing time is faster and the delay between audio playing and the figure plotting is ignorable. The figure below is an output sample of the audio waveform.

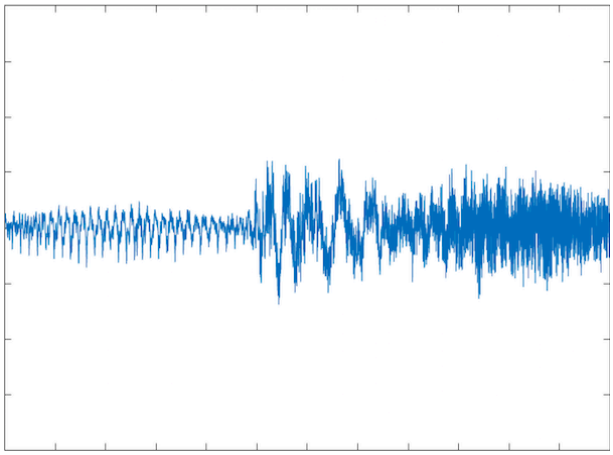

Figure 2-1 The sample audio waveform

The second method is using a buffer to analyze the audio. The general idea here is using a buffer to store the data of current part of the audio, for example, a buffer with length of three thousand points. Then the system will plot the contents of the buffer on GUI. After that, the buffer moves on to store the

next part of the audio and the system will replace the old figure with the new plot of the buffer on GUI. Since the buffer will update itself as the audio plays, this method can also be used to process the real-time audio signal. The flow chart of the procedures is shown as follows:

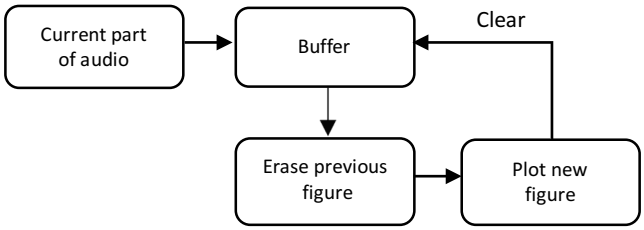

Chart 2-1 Flow chart of the second method

This method can deal with both the offline and real-time audio signal. However, the running time of this method is much longer than the former one. So we can choose this method to deal with real-time audio signal and use the previous method to process the offline audio signal.

3. FFT PLOT

The Fast Fourier Transform converts the discrete time-domain signal into a discrete frequency-domain signal. In the frequency domain, we can analyze the properties of both the low frequency and high frequency part to know some features of the audio signal.

3.1. Theory

To calculate Fast Fourier Transform, we can use the formula

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (k = 0,1,\dots,N-1)$$

In MATLAB, we can use the *fft* function to do the Fast Fourier Transform to each part and use *fftshift* function to move the origin point to the middle of the figure.

3.2. Realization

Our system also need to plot the FFT figure synchronized with the output audio. Since we need to show the FFT figure of current part of the signal, so we choose the second method mentioned before, which updates the buffer with the current part of the audio signal. The sample output FFT figure is shown in Figure 3-1.

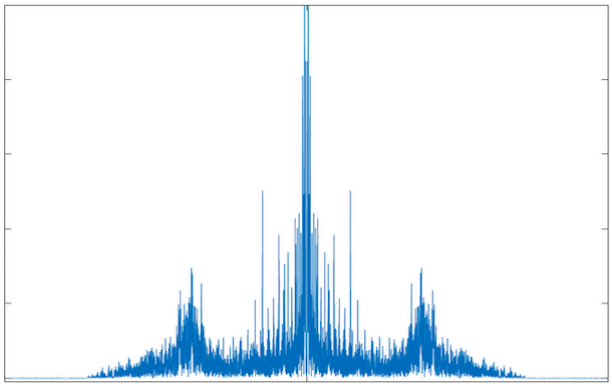

Figure 3-1 The sample FFT figure

4. LED SIMULATION

The second function of our multi-function audio system is to simulate stage lighting effects. For this function, we simulate two LEDs with images to represent some features of the input audio, the onsets and the amplitude. The general shape of the LEDs is shown as Figure 4-1.

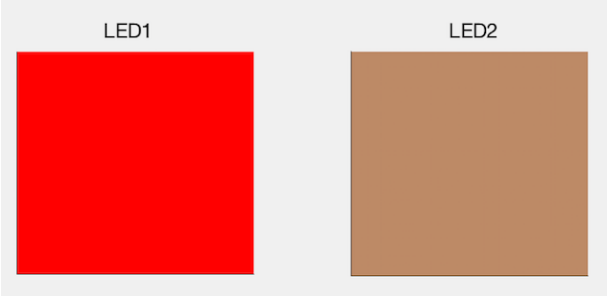

Figure 4-1 The sample of two LEDs

We also plot the overall onsets detection result on the GUI, which is shown as Figure 4-2.

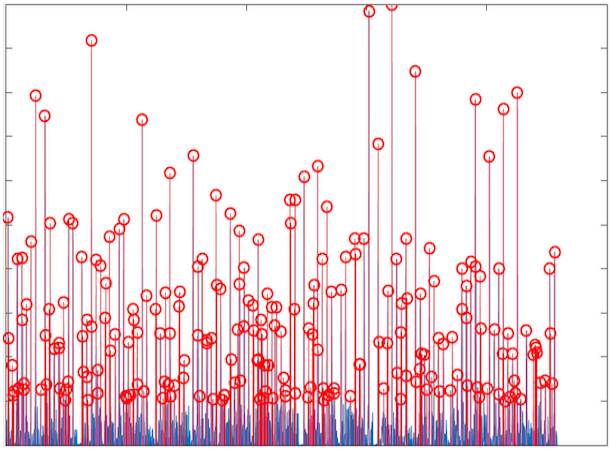

Figure 4-2 The onsets detection result

4.1. Theory

The simulation of LEDs is based on the onsets detection of the input audio. Our system has two different methods to do the onsets detection, which are energy based and spectral based.

The idea of the energy based method is quite straightforward [1]. Firstly, the system chooses a window and takes the frames of the input signal. It takes the square value of each frame and then sums then up to represent the value for each frame. Then the system will subtract each frame's value by its previous value. If the output is greater than 0, the result will be kept. Otherwise, the value will be set to 0. The next step is to set a threshold and do the peak finding in order to get the onsets, together with the corresponding frames. We have also transformed the index of the onsets from frame number to time second. The spectral based onsets detection is almost the same as the previous one but processes in the frequency domain.

Another feature we extracted from the audio is the current amplitude of the audio. We build a buffer to store current amplitude value of the audio signal. Then we do the averaging on that buffer to get a mean value to represent the current signal and classify the result with different levels. Thus we are able to use different colors to represent the amplitude levels of the audio signal.

4.2. Realization

The key part of this function is to synchronize the output audio with the LED display. Since the running time for each loop is unpredictable in MATLAB, it is very hard to synchronize the LED output with the audio signal. However, we already have the time indices of each onset. So, the trick we apply in our algorithm is using *tic toc* functions in MATLAB to calculate the running time of each loop and adding corresponding delays with *delay* function to make sure the simulated LED is synchronized with the output audio.

5. REVERBERATION

This is the third function for our audio system. This function can create different reverberation effects for different types of rooms. In our project, there will be four types of rooms which are the large "bright" room, the large "dead" room, the small "bright" room and the small "dead" room. Shown as Figure 5-1.

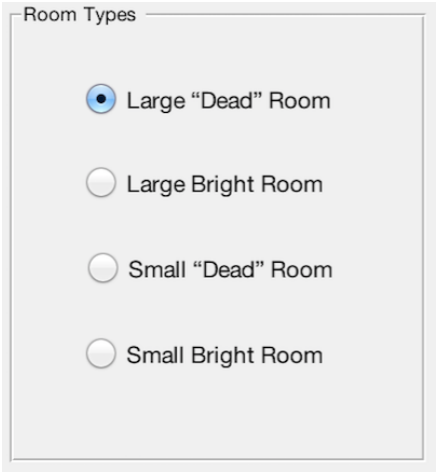

Figure 5-1 Options for reverberation

Here "bright" means the room is very empty, the absorption rate is relatively low and there will be many echo. The word "dead" stands for the opposite meaning, which means high absorption rate. In this GUI, the user can simply choose the room type they want to simulate and click on 'play' to hear the simulation result.

5.1. Theory

The main idea is inspired by the paper of Schroeder in 1962. The reverberation system has a general architecture shown in Figure 5-2 [2].

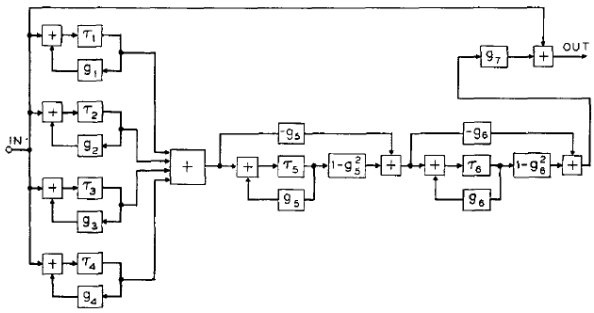

Figure 5-2 Architecture of Reverberation System

In general, the system can be divided into two parts. The first part is just a parallel structure of four comb filters, which are used to create delays and then the output will pass through the second part consisting of two cascaded all-pass filters to enhance the echo. Finally, we combine the processed signal with the original signal to create the reverberation effect.

For different sizes and types of rooms, we need to calculate the T60 first. The equation is shown below [3]:

$$RT_{60} = 0.163 * \frac{V}{\sum_n \alpha_n * S_n}$$

After calculating the T60 based on the room size and type, we can set the coefficients for all these filters. From what we have learned from the paper, we can fix the ratio between τ1 and τ4 to 1:1.5, set the values of τ5 and τ6 to 5 and 1.7 ms,

respectively. Also we fix g5 and g6 be 0.7. To calculate g1 to g4, we apply the following equation [2]:

$$T = 3\tau_n/(-log_{10}g_n)$$

Thus we are able to calculate all the coefficients for different types of rooms. By applying these filters to the audio signal, the system can simulate the reverberation effects.

5.2. Realization

The key part for this function is to calculate different parameters for each type of room and build different kinds of filters to simulate the system. For offline implementation, we can use the entire input signal to do the operation and get the simulation result. However, it is feasible for real-time audio processing on DSP board. We can build a buffer and update the contents of the buffer with current signal, then use the buffer to convolve with the comb filters and the all-pass filters. Therefore, we can create the real-time reverberation effect for the input audio.

6. RESULTS

To combine the functions together, we build a graphical user interface (GUI) for our program with one tutorial interface and three function interfaces. The tutorial interface is shown as Figure 6-1.

Figure 6-1 The tutorial interface

As the figure shown above, the tutorial interface can give instructions to users about how to use our system.

6.1. Waveform and FFT plot

The GUI for this part of function is shown as Figure 6-2.

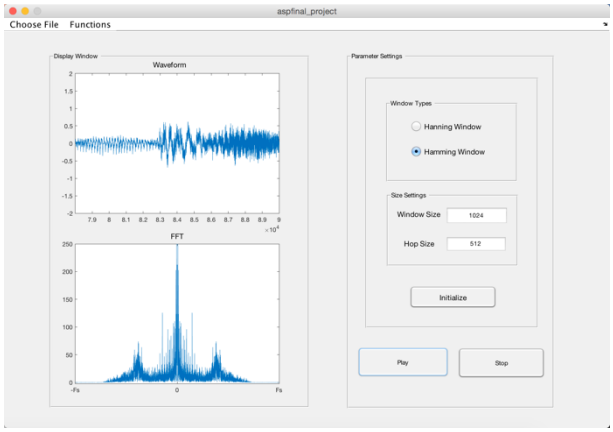

Figure 6-2 GUI of Waveform and FFT plot

The user need to choose the input audio file then click on the 'initialize' button and the 'play' button to start the audio. Then the figures on the left will change correspondingly to the current audio.

6.2. LED Simulation

This is the second function of our multi-function audio system. In the GUI for this part, users can simply choose an input audio file, set up the parameters and click on 'initialize' and 'play' to enjoy the stage lighting simulation. The overall user interface is shown as Figure 6-3.

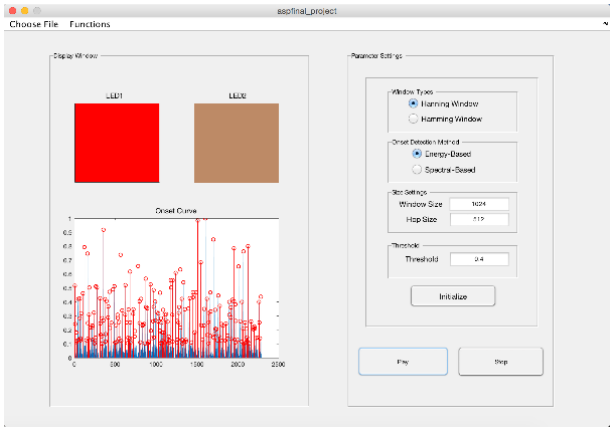

Figure 6-3 GUI of LED Simulation

As shown in the figure above, the LED on the left is used to represent the onsets detection. When an onset is detected, it will spark once. However, in real life, we think another way to represent this effect rather than just sparkling. For instance, it could be the rotation of the stage light or the intensity change of lighting. Another LED on the right side is used to represent the current amplitude of the audio wave and the program sets different color intensities to represent the five different levels of the input audio signal. When the amplitude of the signal is getting lower, the color for that LED will turn from yellow, orange, red, dark red and finally into brown.

This is a good way to simulate the background lighting color in real life stage lighting control.

6.3. Reverberation

The GUI for this part of function is shown as Figure 6-4.

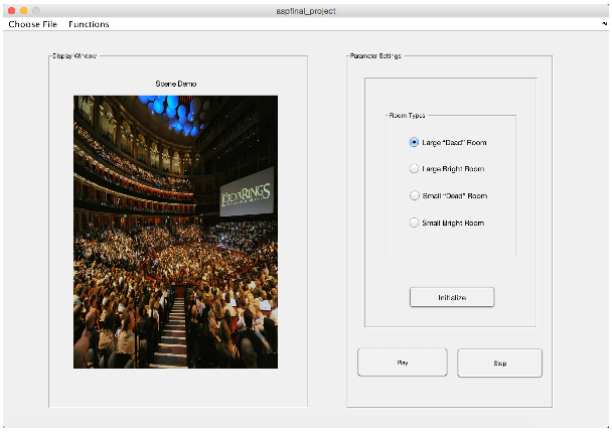

Figure 6-4 GUI of Reverberation [4]

As shown in the figure above, when the user selects different type of the rooms, there will be a sample image on the left side illustrating what the room looks like. For instance, a picture of a concert full of audience stands for a large “dead” room. The echo for this room is not so strong as a large “bright” room. Then the user can click on the ‘play’ button and enjoy the performance of reverberation.

7. FUTURE WORKS

To refine each function, there are still some remaining works to do. First of all, we can implement the algorithm of each part to the real-time audio processing. The most important part here is to get familiar with the MATLAB audio toolbox, then we are able to realize all these three functions with real-time input audio in MATLAB.

Furthermore, the onsets plot on the LED simulation interface is not dynamic right now, we can make it synchronized with the output audio, just like the waveform and FFT plot.

For the third function, the reverberation part, there could also be some refinements. We plan to allow users to DIY the room size and type themselves and our system can automatically calculate each parameter and output the audio with corresponding reverberation effect. This part can also be implemented as a real-time algorithm both in MATLAB and on DSP board.

8. REFERENCES

[1] ECE 472 Assignment HW5.

[2] Schroeder, M. R. (1962). Natural sounding artificial reverberation. *Journal of the Audio Engineering Society*, *10*(3), 219-223.

[3] ECE 472 Class notes and Assignment HW7.

[4] Image from web: http://www.theonering.net.

[5] Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, M. B. (2005). A tutorial on onset detection in music signals. *Speech and Audio Processing, IEEE Transactions on*, *13*(5), 1035-1047.

[6] Goto, M., & Muraoka, Y. (1999). Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions. *Speech Communication*, *27*(3), 311-335.

[7] MATLAB GUI Tutorial. Web source from: http://www.mathworks.com/videos/creating-a-gui-with-guide-68979.html.