AVIS Final Report

Appalachian State University
Computer Science Department
CS 5548

Abstract

Experiencing music at home traditionally uses only one sense, hearing. However, at a concert, there are many senses involved in experiencing the music being played live. For example, seeing the light show, watching the band playing their respective instruments, and jostling around amongst the other concert-goers at a particularly climactic section are all extra senses that are rarely used when experiencing music at home. Over the course of this project, I have developed a tool, AVIS, that aims to stimulate a listener's sight along with their hearing. Utilizing audio analysis externally obtained from Spotify's API, AVIS presents the listener with visual representations of the pitch and timbre at the current section of the song being played. Not only can the listener listen to a song but they also watch as the timbre and pitch visualizations change in sync with the audio.

Introduction

The experience of listening to audio at home often requires only one of the listener's senses, their hearing. While this is not necessarily a problem, I feel that the experience of listening to a song can be enhanced with the addition of visuals. This project aims to provide visuals to the listener alongside the audio.

Spotify's audio analysis of a given song provides data that can be displayed visually to the listener. The data that Spotify provides is broken up into sections, bars, segments, and tatums. The segment data includes, among others, pitch and timbre vectors for each segment of the song. Using these two vectors, it is possible to visualize the change in pitch (frequency) and timbre (texture/tonal quality) over time. AVIS does exactly this. As the song is played, these pitch and timbre vectors are visualized and presented to the listener in sync with the audio.

I chose to use a graphical user interface (GUI) as a canvas to present the listener with the aforementioned visualizations. Specifically, I chose TKinter, a very common Python package for developing GUIs.

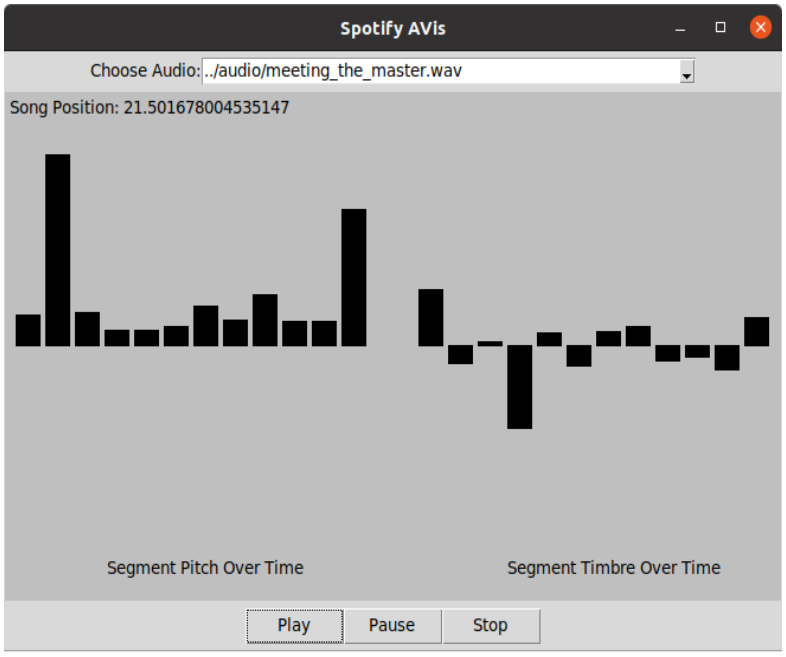
Design & Development

I've kept the design of the GUI simple for two reasons 1) I don't like complicated GUIs and 2) this project requires very few GUI elements to function properly. TKinter is a very simple GUI library and it provides all aforementioned necessary elements. The simple user interface (UI) makes the tool easy to use.

The underlying architecture includes three total threads. The first is responsible for running the TKinter GUI's main loop and scheduling an update function to update the pitch and timbre visualizations. Another is responsible for playing the audio as chosen by the user. The third is the TKinter main loop itself. There is very little communication between threads. The only data necessary to share between threads is the current frame count shared between the aforementioned first and second threads. This is required as the first thread needs to know how to update the visualizations as they are dependent on the current position (time) of the audio being played.

Development of this project was straightforward and guided by the suggestions of Dr. Parry and other class-mates. I used GitHub to log progress made throughout the development process. I found that being able to switch between previous and latest git commits was useful in determining whether a certain feature I was working on was, in fact, working properly.

Final Result



The final result of the project is shown in the screenshot above. The program visualizes the pitch and timbre vectors as provided in the audio analysis obtained from Spotify in sync with the audio currently playing.

How To Use AVIS

Due to issues with Anaconda's PyAudio package, it is necessary to run the program using a python virtual environment or system python found at */usr/bin/python3* on most Linux systems. The program has been tested and is functional on Python v3.8.10; therefore, results may vary on alternative versions of Python. The following steps to run the program assume that */usr/bin/python3* is being used:

1. Navigate to the *src* directory.
2. Install dependencies if necessary:
    a. PyAudio
    b. TKinter
    c. Wave
3. Run the program with the following command:
    a. /usr/bin/python3 -m avis
4. Select a song from the drop down menu at the top of the window
5. Press the *Play* button at the bottom of the window
6. In order to play another song, press the *Stop* button, select another song, and press the *Play* button.

Conclusions and Future Work

As this project did not aim to solve a research question. Conclusions are very relative. However, I found that using AVIS was very satisfying. Being able to watch the pitch and timbre fluctuate in sync with the audio I was listening to was very rewarding and visually pleasing. I found that, as I was listening, I gave more attention to the tonal quality and pitch changes that occurred throughout the song. In that way, listening to the music became more intentional as opposed to something that I did in the background.

I would like to improve a few aspects of the project in the future:

1. Add expected functionality to the *Pause* button (currently it functions identically to the *Stop* button).
2. Eliminate the necessity of obtaining Spotify's audio analysis externally.
    a. As it stands, the sample rate of the audio file and Spotify's audio analysis must be the same. Also, those two files must have the same name and exist in the same directory (for example, *audio/meeting_the_master.wav* alongside *audio/meeting_the_master.json*).
3. Add functionality to correct any discrepancy between the sample rate of the *.wav* file provided and its corresponding audio analysis.
4. I would also like to add the ability to handle multiple audio formats as opposed to only *.wav* audio files.