Audio Viz

Aaron Karp, Sean Bowen-Williams, David Ryan
Department of Electrical Engineering and Computer Science
Northwestern University

1. Our Project

We want to create an interactive visual experience that is dynamically generated from audio files or music. Music visualization is useful for a number of different reasons, but most modern visualizers solely focus on the entertainment aspect. We aim to provide a layer of education and musical analysis not offered in other current music visualizers. The project is not entirely practical, but it does serve an entertainment purpose that will help users look at songs they enjoy in a new way. The visual interpretations of songs they are familiar with will draw focus to components of the song (beats per minute, pitch variation, etc.) that may not have been of much interest or understanding before. Current systems do not lend themselves to easy recognition of complex audio features; our project will make those more accessible through a visual experience.

2. IMPLEMENTATION

Our application visualizes preprocessed audio files that have separated foregrounds and backgrounds. This source separation is accomplished using Adaptive REPET [4], a Matlab program published by Zafar Rafii of Northwestern, which outputs two discrete files. First, a beat spectrum is generated by estimating repeating periods in the original source audio. By

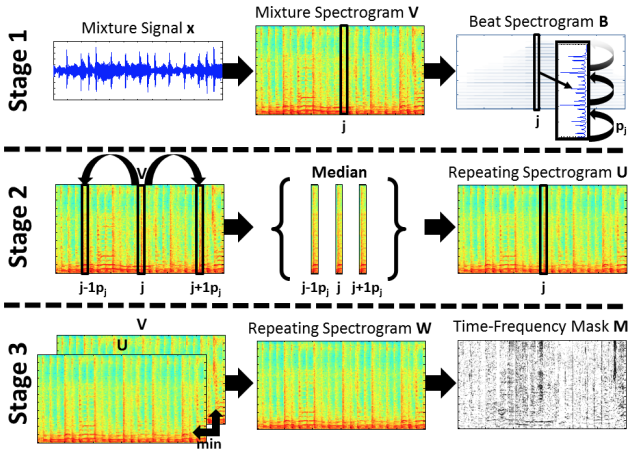

Figure 1

Reppit Processing

http://www.zafarrafii.com/repet.html

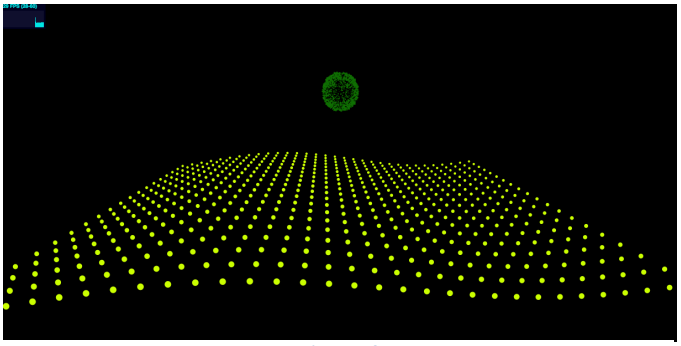

Figure 2

A screenshot from the visualizer.

filtering the mixture spectrogram and calculating a repeating spectrogram, the program derives the time frequency mask. This mask is then used to find the background and foreground sounds.

From there, we will look at the two files and examine the beat [1] of the background file and the real-time amplitude and pitch tracking [3] on the foreground file. These features are computed using Librosa, a Python library for audio analysis. Each of these components are used as inputs for our visual representation of the audio. Each audio feature is assigned to an aspect of the visual scene (see Fig. 1). The beat-tracking is shown by changing the size of the yellow spheres. The hovering sphere in the center of the screen changes color based on pitch of the foreground (see Fig. 2) and radius based on amplitude of that same file.

There is also an interactive portion of AudioViz. At the top of the screen is a slider that can be pulled right or left. The slider is an equal power crossfader between the two source-separated audio tracks. Pulling the slider to the right all the way changes the output audio to entirely background, while moving the slider to the left all the way changes the output audio to entirely foreground. This was accomplished by adding gain nodes to the source nodes that load the audio files. An in-depth explanation and a full tutorial of this can be found here: http://www.html5rocks.com/en/tutorials/webaudio/intro/#toc-xfade.

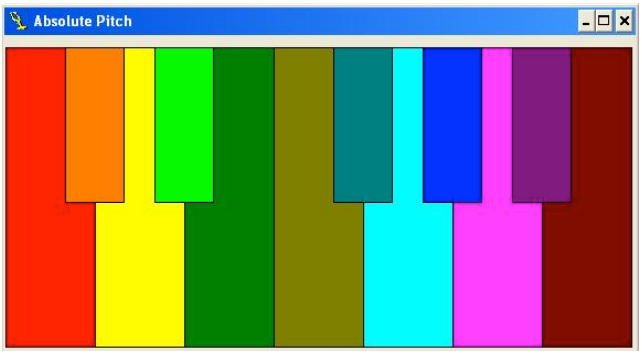

Figure 3

The color-to-pitch representation used.

https://xenophilius.files.wordpress.com/2008/11/absolutepitch.jpg

3. Testing

We approached this project by first creating test mp3 files that had the features we are looking to extract and represent. After succesfully testing the system with simple given inputs, we began testing commercial songs that have similar characteristics. This resulted in some additional problems with the source-separation, namely that the small artifacts that should have been placed in the background file were sometimes placed in the foreground file, which complicated the pitch tracking of the foreground. However, after further tweaking, this problem was fixed.

Judging the quality of our system relied on user feedback. Our main goal was to determine if the different visual elements of the experience are accurately and obviously representative of specific audio features. We ran the program with a preset song and asked the users a set of questions relating to both its visual appeal and each of the features we are representing (e.g., what component of the song did the floating points represent?). These responses were mostly positive in all aspects.

4. Conclusions

Based on the results of our user surveys and our own interpretations, AudioViz has proven to be successful in its ability to entertain users. The accuracy of the beat-tracking and its visual representation is also quite high. The biggest problem we encountered was the tracking related to the foreground. This turned out to be fairly inaccurate with pop songs. This can be traced back to inaccuracies within the source separation. There are significant artifacts from the background that are separated into the foreground audio file, which in turn confuses the pitch and amplitude tracking.

5. Further Work

There are many potential directions for expanding on our music visualizer. By writing Reppit in Python, the entire process can be streamlined into a live version where users can upload their own files to visualize in real-time. The largest area of improvement can be found in the source separation, however. The accuracy of the vocal pitch tracking and amplitude calculations would be improved with more advanced methods of source separation. There are a number of source separation styles that have been researched and implemented successfully, so we would just need to tailor them to fit AudioViz. Other audio features could also be implemented visually, such as multi-pitch tracking and overall volume of the audio track.

6. References

[1] Pikrakis, Aggelos; Antonopoulos, Iasonas; Theodoridis Sergios, “Music Meter and Tempo Tracking from Raw Polyphonic Audio“, University of Athens, Greece Department of Informatics and Telecommunications. (http://www.cs.northwestern.edu/~pardo/courses/eecs352/papers/tempo%20tracking%20audio%20-%20pikrakis.pdf)

[2] Ciuha, Peter; Klemenc, Bojan, Solina, Frank, “Visualization of Concurrent Tones in Music with Colours”, University of Ljubljana (http://delivery.acm.org/10.1145/1880000/1874320/p1677-ciuha.pdf)

[3] Talkin, David. "A robust algorithm for pitch tracking (RAPT)." *Speech coding and synthesis* 495 (1995): 518. (http://www.ee.columbia.edu/~dpwe/papers/Talkin95-rapt.pdf)

[4] Rafii, Zafar; Bryan Pardo. “REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation”, *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 21, No. 1, January 2013. (http://music.cs.northwestern.edu/publications/Rafii-Pardo%20-%20REpeating%20Pattern%20Extraction%20Technique%20(REPET)%20A%20Simple%20Method%20for%20Music-Voice%20Separation%20-%20TALSP%202013.pdf)