BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2025
http://excel.fit.vutbr.cz

Excel@FIT Poster Commentary

Jura Rusnák

Abstract

Real-time music visualizers often rely on basic FFT-based feature extraction, resulting in visuals that poorly reflect the structure and energy of modern electronic music, especially drum and bass (D&B). This project implements a real-time system that extracts advanced audio features including RMS, Spectral Flux, Spectral Predictivity, and melody separation using Harmonic-Percussive Separation. These features are integrated into the ProjectM visualizer, adding to its original rigid analyzer to enable visuals that respond to melody, transients, and basslines. The result are visuals that accurately depict the songs energy, while ramining engaging and interesting.

*xrusna08@stud.fit.vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Live visualizations of electronic music performances are often disconnected from the actual musical content. While large venues may pre-sync visuals to music, smaller clubs lack the funds to spend on pre-generated visuals and pre-recorded sets.

Current visualizers are made to be compatible with all genres of music. They use basic analysis to extract features from the track. The visuals are of differing quality. Some are too boring, others are too fast. Furthermore, the visual may not react very well to the track. Some detail will be inherently lost. I aim to create a visual that will truthfully and interestingly represent the track.

ProjectM and similar programs compute FFT and extract max amplitudes for each of three band (Bass, middle, treble). While computationally efficient, these techniques ignore harmonic predictability and transients. Recent research [Lerch, Müller][1][2] shows how RMS, spectral flux, and harmonic separation can produce more meaningful data.

I implemented a custom analyzer inside ProjectM that computes RMS, spectral flux (SF), spectral predictivity (SP), and simplified harmonic-percussive separation (HPS). These features are normalized, attenuated, and passed to ProjectM's visual shaders. New presets use these values to enhance responsiveness to melody, rhythm, and bass.

2. Feature Extraction

In order to keep the original functionality of existing presets inside ProjectM I left the bass, middle and treble as they are. ProjectM has a lot of preset already made and it would be shame if they became unusable in my version.

Figure one visualizes how the values for bass, middle and treble are extracted from one frame of spectrum.

2.1 Root Mean Square (RMS)

Figure two shows the value of RMS over time and it's equation.

I adjusted the calculation of volume that was previously calculated as the mean of the bass, middle and treble values. Instead, I used an RMS with a low-pass filter for its calculation. RMS is a simple equation that can be calculated from the waveform. I have done this because I wanted to capture the energy of the song. In D&B the highest energy sections are the ones with the loudest bass.

2.2 Spectral Flux (SF)

Figure three shows the values of spectral flux on a spectrum over time.

Spectral flux tracks the change in spectral energy between consecutive frames. It is useful for detecting transients such as drum hits. I use it in the renderer to scale the waveform so that the drum hits are more prominent.

2.3 Spectral Predictivity (SP)

Figure four shows the values of spectral predictivity over time.

Spectral Predictivty calculates the consistency of the frequency content over time. Making it suitable for adding harshness to the visual. It also modulates a low pass filter on waveform with a high cutoff frequency so with the increasing value of spectral predictivity the waveform contains more top frequencies.

2.4 Waveform and Spectrum

Along with these values the waveform and spectrum is also sent to the renderer. I scale and filter the waveform for more engagement.

The spectrum has a bit more going on. Specifically, the harmonic percussive separation (HPS) algorithm is used to punch out the transients in the spectrum. In order to reduce latency I simplified the HPS algorithm.

Figure five shows the simplification of the HPS algorithm.

In D&B the melodies are not composed of consistent harmonic notes and include more noise, so the horizontal filtering can be omitted, and just the vertical filtering is implemented. The resulting algorithm lets some transients pass, especially when the song does not contain louder melodies. With more intensive melodies the algorithm is successful at removing the transients.

3. Integration with ProjectM Visualizer

The extracted features (RMS, SF, SP, frequency bands, waveform and spectrum) are passed to ProjectM, a MilkDrop-style visualizer. The default analyzer, which only supplies bass, mid, treble, waveform, and spectrum was replaced with the custom analysis pipeline.

My custom presets access the new features and use them to control various parameters such as scale, color, and warping. For instance, bass influences zoom and pulse, while spectral flux creates twitchy motion synced to drum hits.

An example preset is shown in the lower section of the poster (figure seven), along with a code snippet that demonstrates how these parameters are used.

4. Analyzer and Renderer

The analyzer takes in a waveform as input and spits out an object that contains all the analysis data. This data is then used by the renderer to move, scale, and zoom the visual. Figure six point one and Figure six point two shows an example preset. The preset is color-coded for better ease of explanation.

The red and blue chunks of code represent an init code that is executed only once when the preset is loaded in. The more interesting yellow and green chunks are the per frame equations that use the values from the analyzer to move the preset. The yellow code is linked to the blue shape code that moves the shape, in this case an octagon that shrinks and grows based on the middle value extracted from the track.

The green code is the global per frame code, that zooms, rotates and sets the decay on the preset. Decay determines how much of the previous frame is shown on the next frame.

There are parts of the preset code that are left out. The 2 biggest are the stock waveform and and my custom waveform code. The stock waveform is shown in a light green color on the preset example image. It is drawn in dots and handles most of the movement. Waveform is suitable for this, as it reacts to the all of the frequencies.

The red circle in the center is my custom waveform that adds another layer to the preset.

These waveforms are colored based on time. The equations can be seen on the first part of the green chunk of preset code.

5. Shaders

Shaders add another level to the preset. A shader in my example creates a kaleidoscopic effect, by mirroring a a chunk of the screen eight times.

Acknowledgements

I would like to thank my supervisor Ing. Oldřich Plchot Ph.D. for their willingness to supervise my unconventional idea, their help and understanding in my time of difficult studies.

References

[1] Alexander Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, Inc., 1 edition, 2012.

[2] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing, 1 edition, 2015.