SigSys Final

Spring 2014

Music Visualization and Analysis

Authors:
Forrest Bourke
Adam Coppola
Evan Dorsky

Professors:
Oscar Mur-Miranda
Siddhartan Govindasmy
Scott Starks

Abstract

In this paper, the authors discuss several music-related ideas and methods, such as applying a Finite Impulse Response (FIR) filter to a signal, and using a method called beat spectrum analysis to determine the tempo of a sample of music. Issues with the discrete samples of digital music are discussed, such as maximum frequency based on sample rate and proper use of windowing to avoid false high-frequency artifacts.

Last Updated: May 6, 2014

Contents

1 Introduction

The signal used for this project was a 5.6-second sound clip from the song "Music Sounds Better with You" by Stardust and a 10 second sample of ATC's "Around the World (La La La)." The Startdust sample was chosen because it has a very powerful repeating four-on-the-floor house beat and a repetitive synth line. The section from "Around the World (La La La)" was chosen because it involves the end of a breakdown followed by a buildup to a repetitive beat.

2 Low-Pass FIR Filter

Finite-impulse response filters(FIR) are filters whose impulse responses are finite in time. Because they are finite in time, it is impossible to create a sharp-cornered ideal low-pass filter. However, the advantage over infinite-impulse response filters is that FIR filters only require some of the information about a signal at a time. Due to this, FIR filters are very often used in digital signal processing. Since they require only a finite amount of data, they are easy to build and they provide opportunity for greater control, as will be shown later.

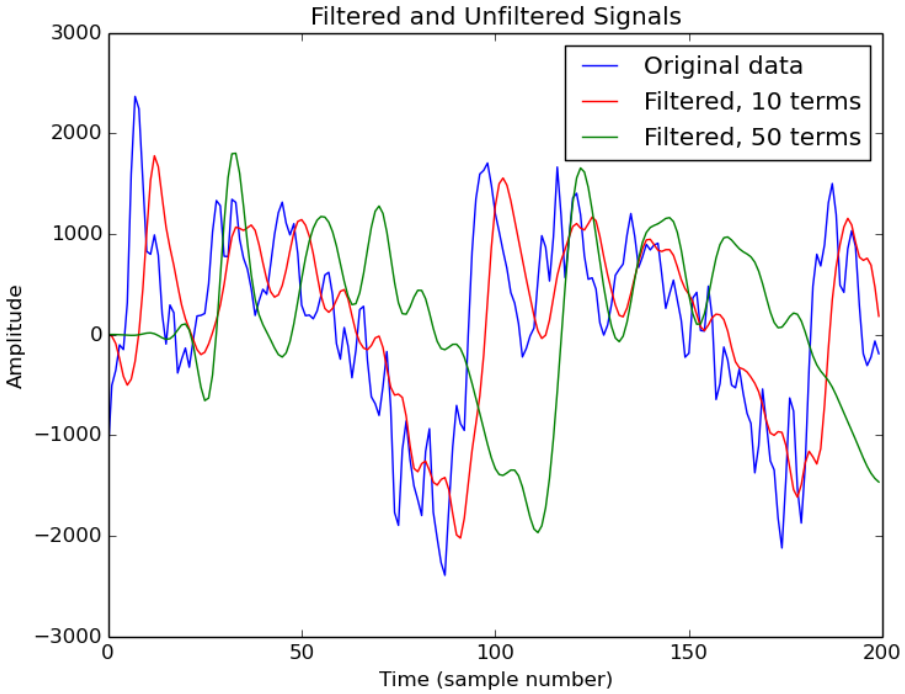


Figure 1: Digitally filtering an audio signal

Figure 1 shows the results of our digital low-pass FIR filter for 200 samples of a signal, for a total of 4.5ms of music. The blue line is the original sound signal, the red line is a filtered signal, and the green line is another filtered signal. The red line, which does a poor job acting as a low-pass filter, was constructed with fewer terms than the green line, which does a better job. It is interesting to note that the "better" filter does induce more phase shift, about 0.68ms of phase shift, compared to the 0.11ms of the 10-term filter.

These filters can be defined using the following difference equation.

$$y[n] = \sum_{i=0}^{M} b_i x[n-i] \tag{1}$$

$x[n]$ is the input signal and $y[n]$ is the output signal. Because this is an FIR filter, there is no feedback. You'll note that because this only considers $i > 0$, it introduces a time delay half as large as M. The equation represents a convolution of the audio signal with a function defined by these coefficients, b_i. Therefore, if the coefficients compose a sinc function, then the filter will behave as a low-pass filter.[1]

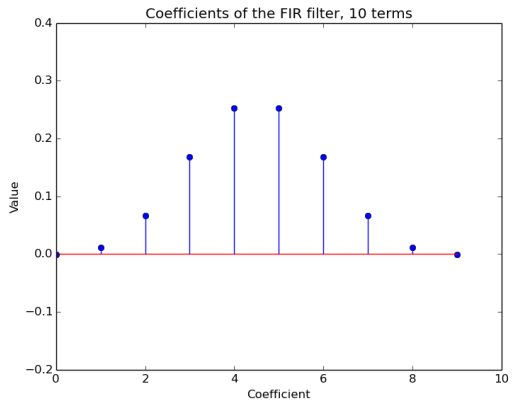


Figure 2: Ten sample filter coefficients

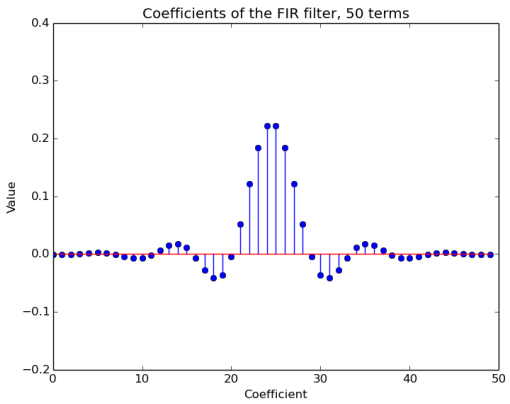


Figure 3: Fifty sample filter coefficients

The filters worked with in this project are defined in Figure 2(red signal) and Figure 3(green signal). These are both discretized portions of a sinc function, but the first only has ten taps, and the second has fifty. In the time domain, the effects can be seen in Figure 1 on the previous page. However, translating these to the frequency domain reveals why more taps are more effective.

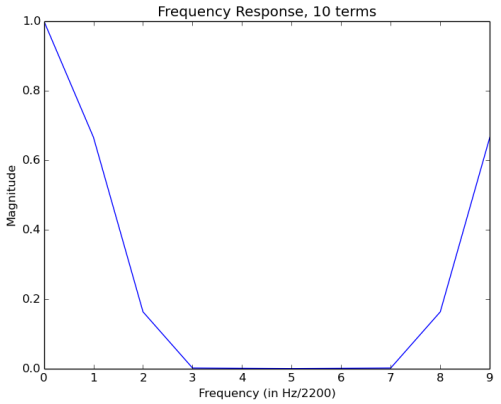


Figure 4: Ten sample filter coefficients. The cutoff frequency is supposed to be 5000Hz

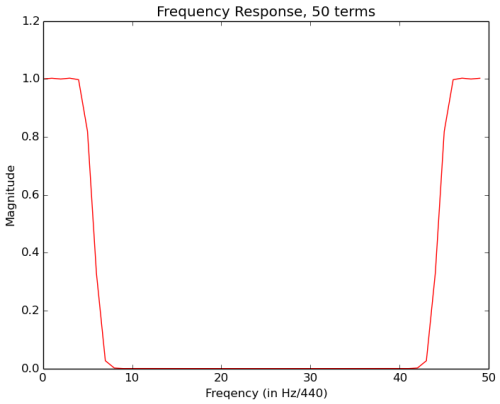


Figure 5: Fifty sample filter coefficients. The cutoff frequency is supposed to be 5000Hz

Figure 4 and Figure 5 show the Fourier transform of the filters, the transfer function of the filter. As expected, nothing is transferred at higher frequencies. With only ten taps, the filter is much less effective, and does not actually have a portion of full transfer. It actually chops something from all frequencies. As more taps are added, it looks more like the ideal. If the taps were infinite (which would in fact be an IIR), then we could see an ideal filter.

The effects of this filter on the filter can be found in Figures 6,7, and 8. Note that these sample spectrograms are not from the same audio sample (because there is a delay), but they still demonstrate how

[1] *Introduction to Digital Filters with Audio Applications*, Julius O. Smith III

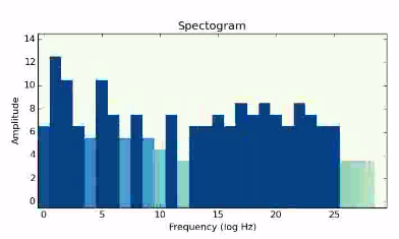


Figure 6: The unfiltered sound sample

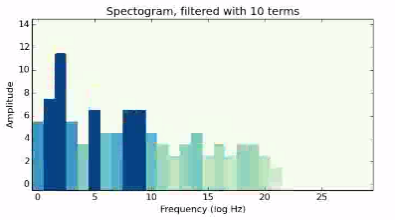


Figure 7: The "poorly" filtered sound sample

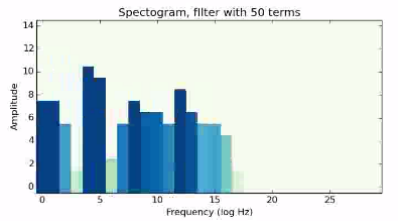


Figure 8: "Well" filtered sound sample

the filter influences the signal. In the poorly filtered sample, the amplitudes are more diminished as the frequencies get larger. In the well filtered sample, there is a more clear cutoff.

These FIR filters proved to be very effective in breaking down the signal and helping us understand digital signal filtering. However, the beat detection algorithm was a much more effective way of analyzing BPM and creating visualizations, so this strategy was not used in our final visualization.

3 Beat Detection

Though beat detection is easy for humans, (humans have difficulty *not* walking to a beat[2]) it's not an easy task for computers. Simply looking at the amplitude and when it increases dramatically (called "onset detection") is not sufficient. Even filtering by frequency, this method is not sufficient for anything but the simplest of 4-on-the-floor electronic music. Instead of using onset detection, we chose to detect beats using a method called the "beat spectrum."[3] This method compares a signal's similarity to itself, delayed by varying amounts. First, a window is chosen and compared to successive windows of increasing delay. The lag-time of successive peaks in similarity occurs at the same tempo as the music.

3.1 Windowing

The first step is to split the music into windows. We used hamming windows (Figure 9 on the following page) whose width was depending on the sample rate. The Hamming window was chosen (as opposed to a rectangular window) to reduce the "ringing" inherent in taking the transform of a square-windowed function. When the Fourier transform of a square-windowed function is taken, the result is the true spectrum multiplied with a sinc function. The Hamming window reduces the effect of windowing by having a specific taper designed to minimize the amplitude of frequencies close to, but outside the window.

3.2 Similarity Analysis

To find the similarity between two windows, the dot product of the windows (each consisting of a 1-D vector of time-based amplitude values) is taken with another window, then normalized. The equation is shown in Equation (2).

$$\text{Similarity}(i,j) = \frac{\vec{v_i} \bullet \vec{v_j}}{|\vec{v_i}||\vec{v_j}|} \tag{2}$$

The similarity of each element to each other element is plotted with the time of the beginning of $\vec{v_i}$ and $\vec{v_j}$ on each axis.

For an example of this technique, we used a short (10 second) sample of ATC's "Around the World (La La La)" starting at around 2:33. In this section of music, it builds up with looping snare drums to a only cymbal hit for a second, then the beat begins again. The signal is shown in Figure 10a on the following page. A spectrogram is shown in Figure 11a on page 6. Though not strictly relevant to beat detection, it is

[2] *Activating and Relaxing Music Entrains the Speed of Beat Synchronized Walking*, Leman, M. et. al
[3] *The Beat Spectrum: a New Approach To Rhythm Analysis*, Foote J., Uchihashi S.

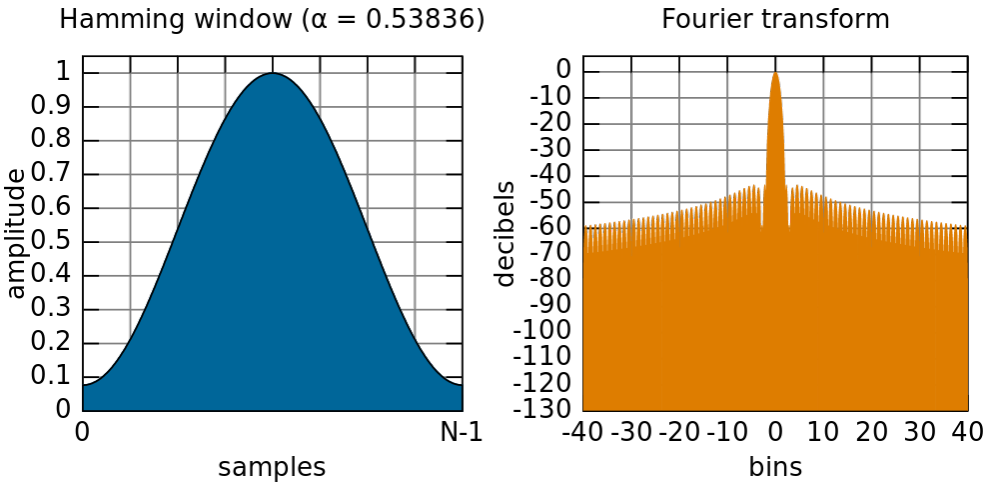
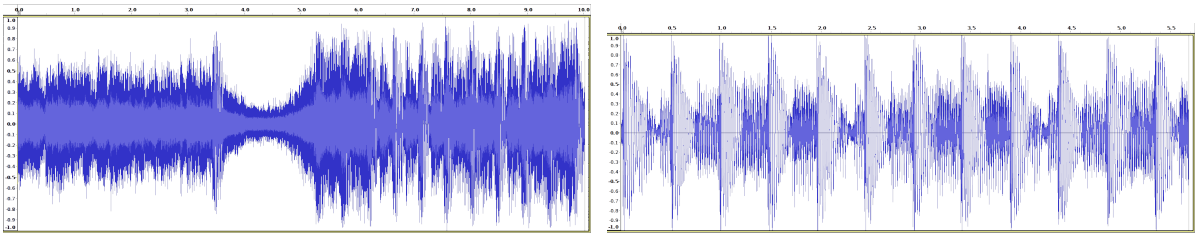


Figure 9: Hamming Window and Fourier Transform (Wikipedia)



(a) 'Around the World (La La La)' Sample

(b) 'Music Sounds Better With You' Sample

Figure 10: Music Samples

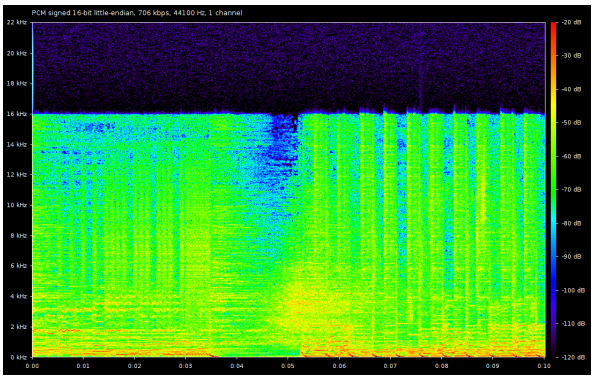
interesting to note that the spectrogram contains a hard line at exactly 16khz. Even though the sample rate of 44.1khz predicts a maximum frequency of 22khz, it appears as if the song was sampled at 32khz and then upsampled to 44.1khz resulting in nothing but noise above the previous nyquist frequency of 16khz, leaving the top 6khz of the spectrum empty (and a larger-than-necessary file size). The results of our program are shown in Figure 12a on the following page. The diagonal line represents the fact that every window is equal to itself. The "plus sign" is the cymbal hit, which is effectively only high frequency noise. Since the cymbal hit is continually changing (decaying), it displays low self-similarity. After the beat begins again, similarities are shown as diagonal patterns. The tempo of the song can be effectively determined by taking the distance between the diagonal patterns.

3.3 Beat Spectrum Graph

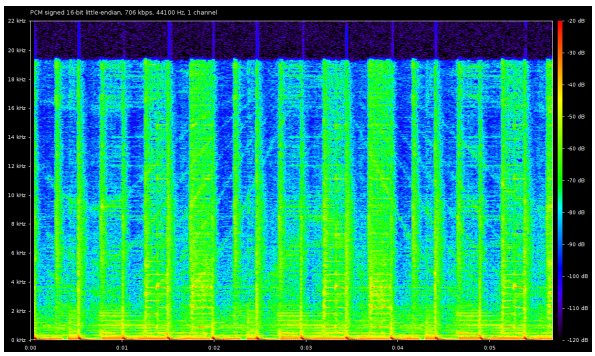
The best "punchline" graph that can be developed from this beat spectrum method is achieved by plotting the diagonal autocorrelation along a line in the similarity matrix against the distance between the line and the 1:1 equivalence line. We found good results with Equation (3), a discrete approximation of the autocorrelation used in the paper.

$$B(l) = \sum_{i,j=0}^{n} S(i,j)S(i+l,j+l) \tag{3}$$

where l is the distance from the 1:1 line, $B(l)$ is the value on the beat spectrum graph, n is the total number of windows used, and $S(i,j)$ is the function from Equation (2) on the preceding page. This song has a BPM of 132, so we expect to see a peak every 60/132, or 0.45s. As is visible on the graph, this song does exhibit the peaks we expect at ± 0.45 seconds, as well as multiples of this value.

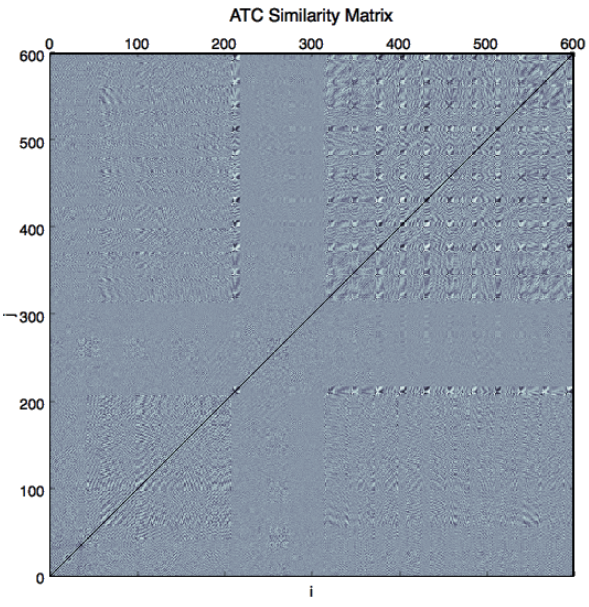


(a) 'Around the World (La La La)' Spectrogram

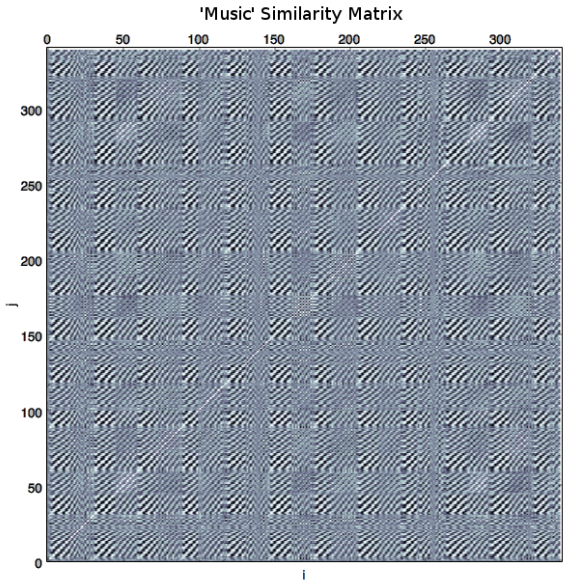


(b) 'Music Sounds Better With You' Spectrogram

Figure 11: Music Spectrograms



(a) 'Around the World (La La La)' Similarity Mapping



(b) 'Music Sounds Better With You' Similarity Mapping

Figure 12: Music Similarity Mappings

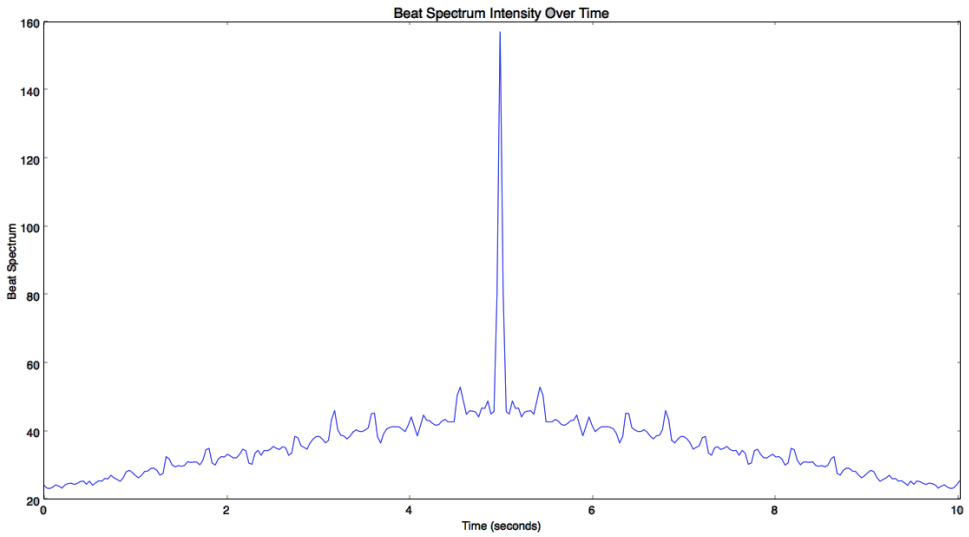


Figure 13: 'Around the World (La La La)' Beat Spectrum

An easier similarity mapping to read is also provided in Figure 14 on the next page. This is based our sample of Stardust's French house classic "Music Sounds Better With You." The beat peaks are even more visible, and (as shown on the figure) they align with the BPM we would expect of the song.

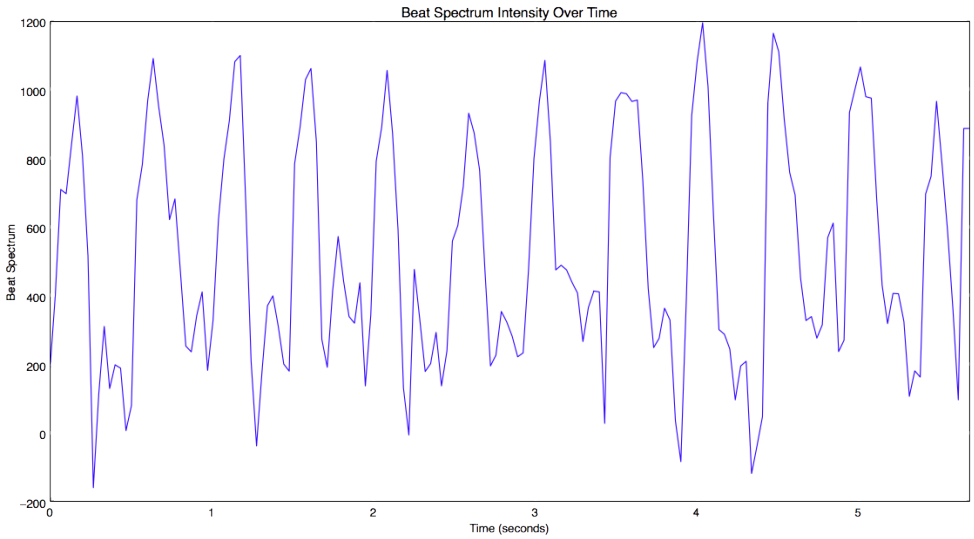


Figure 14: 'Music Sounds Better With You' Beat Spectrum

4 Conclusion

If you would like to find the repository hosting the code for this project, you can find it here: `https://github.com/EnigMoiD/SigSys-Music-Visualization`. The video showing our final visualization can be found here `https://www.youtube.com/watch?v=HJ39KpvKcCA`.