Musical Tempo and Beat Estimation

Evan Lucas
Electrical and Computer Engineering
Michigan Technological University
Houghton, Michigan, United States

***Abstract*—The author attempted to recreate the results of Alonso in [1]. As the author did not have access to professional musicians to estimate beat, the tempo estimation was not verified. They attempted to use an existing MATLAB toolbox [2] to get a second estimate, but it was written a long time ago and is not very compatible with the current state of MATLAB. Some sentences which are the original authors work are borrowed from the project proposal in whole without alteration, particularly in the problem statement.**

***Index Terms*—music, music information retrieval, signal estimation, acoustic signal processing**

I. Introduction

I decided to work on a fun signal processing project, recreating the work of Alonso in [1]. I have been a musician for most of my life (although I don't make time to play as often as I should these days). This document was created on Overleaf (an online LaTeX editor) using an IEEE Transactions template.

II. Problem Statement

Being able to identify the beats in music is useful for many applications such as: automated music processing, procedurally generated music (especially in the case of a generated accompaniment), automated multimedia editing, and improved music compression algorithms. [3]

It is not difficult to imagine non-musical applications for this type of algorithm. For example, many marine surveillance systems include underwater acoustic sensors. Being able to determine a beat frequency may help to classify an underwater source as a certain type of craft.

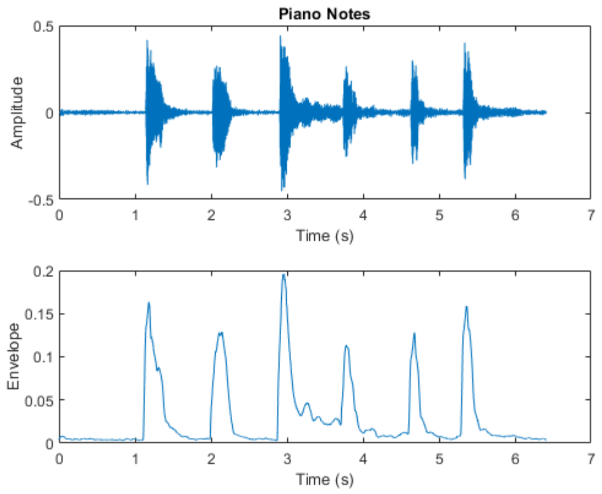
This problem comes in two large parts and a third smaller one. First, one must detect the beats in the musical signal, sometimes also called the musical onset. Then, one must determine how these beats form a tempo. Finally, one must find the time locations of the beats.

III. Methods

A. Mathematics behind beat detection

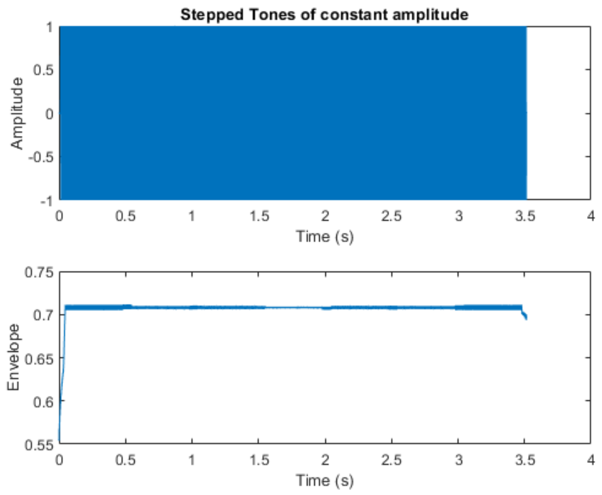
There are multiple methods that have been proposed to solve this problem. The simplest methods rely on the impulsive nature of many musical instruments. [3] Consider a piano, which creates sound by striking a wire with a hammer. The sound created will start at a given volume, which will decay with time. If one looks at the energy envelope of this signal, there will be a clear series of pulses that a tempo could be derived from. An example of this is given in Figure 1.

Fig. 1. Example of using a Hilbert Envelope to detect beats



This problem is harder with sounds that are less impulsive, such as a violin or in a worst-case, a simple tone generator. A good example of this is given in Figure 2, where I attempted the same method with some stepped tones.

Fig. 2. Another example of using a Hilbert Envelope to detect beats



B. Mathematics behind tempo estimation

Once the beat has been detected, it must be turned into a tempo. This is relatively straightforward to think about, as tempo is generally expressed in the units of beats per minute.

Naively, one could process this the same way you process a tachometer pulse signal or by using a cepstrum, but the authors of the paper had other ideas. They suggest two different methods taken from one of their earlier papers. [4] The first is an autocorrelation based method, which can find repeating content as well as the delay between repeats. The second is a spectral product method, which assumes that there will be content at integer multiples of the primary beat frequency. This assumption allows for faster, in-rhythm notes (such as eighth or sixteenth notes in a 4/4 time signature) to sum into the tempo estimation.

C. *Mathematics behind beat location*

Finally, after tempo has been estimated, the location of each beat must be determined. To detect the location of the initial beat, a synthetic series of beats is generated and used as a matched filter (cross correlating) with the detection function.

Once the first beat has been located, each subsequent beat is searched for at it's expected location (the last beat, plus the time lapse expected by the current tempo). If the beat location varies by more than some value (the paper uses ten percent), a new beat phase is assumed and the algorithm goes back to searching for a first beat.

IV. Results

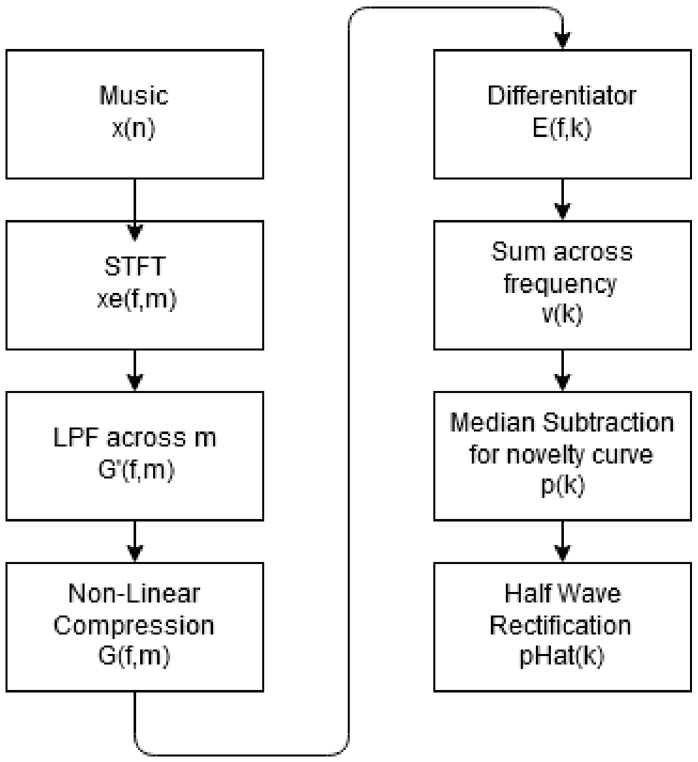
A. *Beat Detection*

The paper selected attacks the problem of beat detection by using the concept of Spectral Flux, which is the change in frequency content as a function of time. The signal processing workflow is shown in Figure 3. First, the spectrogram (aka Short Time Fourier Transform - STFT) is taken, representing the sound as a sequence of frequency spectrums. This is lowpass filtered along the time block axis; a half-Hanning window was used following one of the suggestions in the paper. A non-linear compression is then performed; multiple methods were chosen but the inverse hyperbolic sine function was found to work well.Then the absolute value of the signal was taken.

The next step was one that I struggled with. A differentiator filter is applied across the time block axis to find changes in frequency content. I attempted to use several different types of differentiator filters, include a simple first order one [-1 1], a five point differentiator, and a differentiator designed using the Remez exchange algorithm as described in the paper. Ultimately the five point differentiator filter was found to be the most robust over most of the music samples used. Once differentiation is complete, which finds the changes in spectral energy, the spectral flux is summed across frequency evenly. (As an aside, for certain applications, it may be useful to weight different frequencies differently. The human ear doesn't have a flat frequency response, so spectral weighting may improve results for perceived beat.)

Finally, the novelty curve is calculated by subtracting the median amplitude. Following this, the curve is half wave rectified to emphasize peaks.

Fig. 3. Signal Processing Workflow



One of the primary focuses of my proposal was to reproduce a specific figure form it, which is included as Figure 4. The original paper includes a note that copies (in full or part) may be made for educational purposes, so it is included here without any concerns. I attempted to replicate this figure, which is included as Figure 5, using some piano notes that I recorded in a practice room in Walker. The data could probably be improved by recording at a lower sampling rate, but otherwise it is a good replication of the original results. My version of the novelty curve algorithm includes a startup transient that is seen in every dataset processed, but I didn't spend the time necessary to remove this from the output. I have also included one intermediate step that wasn't present in the original paper, as I feel it helps aid understanding of the signal processing method.

The algorithm was also tested on the stepped tone file and the results are included in Figure 6. There is room for algorithm improvement here, but I'm fairly happy with the results. A number of other music samples were used to test the robustness of the algorithm, and some of the more interesting results are included here. I discovered the the algorithm is extremely sensitive to changes in frequency content. The impact of this can be seen in Figure 7, which is a file of me playing some notes and chords on a piano. There is some paper shuffling or scraping noise at 1 second and 10 seconds, which shows up on the novelty curve. This is quite interesting and not a bad result, but I think some pre-processing to remove spurious noises could help improve the results for this

Fig. 4. Figure from Paper

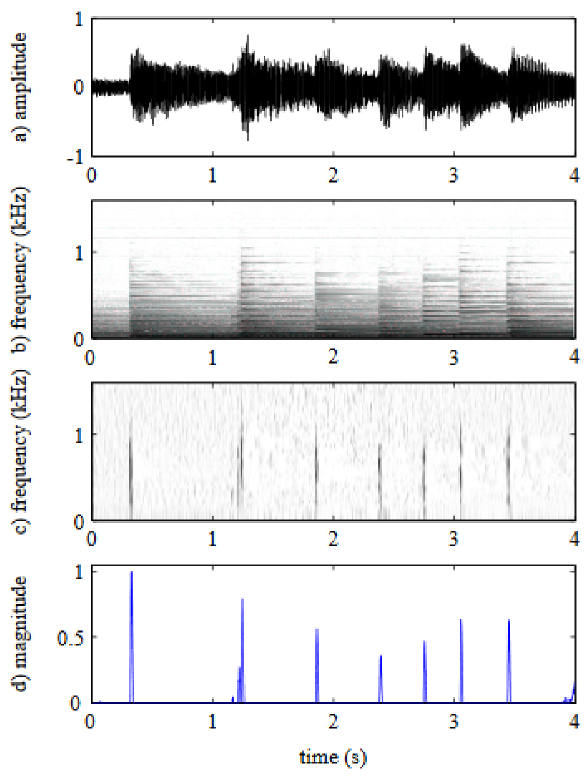


Fig. 5. Sample Beat Detection with pianoNotes.mp3 file

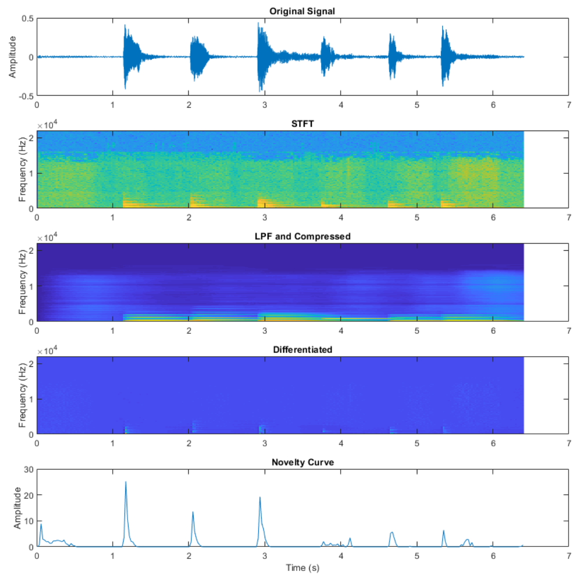
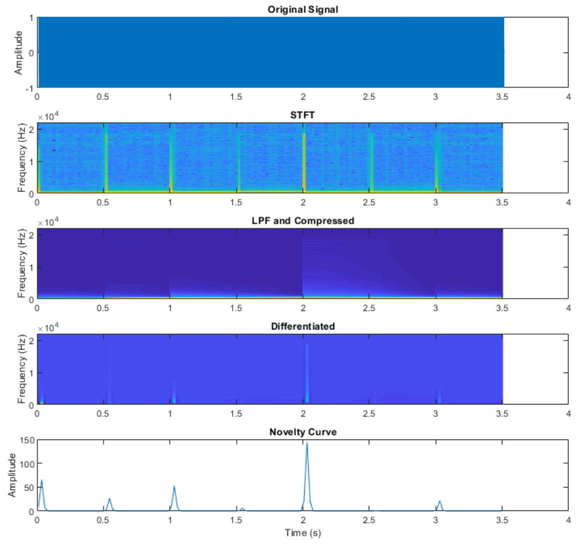


Fig. 6. Sample Beat Detection with steppedTones.mp3 file



application. As an aside, all audio samples used for this paper are included in the files uploaded. Any copyrighted music included is sampled to be twenty seconds or shorter, so as to comply with standard Fair Use practices.

B. Tempo Estimation

I spent very little time on the tempo estimation aspect of the project. I started playing with some old code I had written to process tachometer speeds, which is a very similar problem - converting pulses into a a pulse frequency. I ended up simplifying this code into a very simple algorithm of: peak picking, finding the time delta between peaks, inverting this, constraining it to be less than 220 BPM, and smoothing with a median filter. An example of this is displayed in Figure 8, which shows a metronome recording that increases from 110 BPM to 130 BPM. Real music doesn't give results nearly this clean, but seems to work fairly well outside of the startup/ending transients typical with a median filter.

This is different than the methods described in the paper, which I didn't implement. I started working on the correlation method, but gave up due to other time priorities.

One other issue that was observed with tempo estimation can be seen by using the file containing the beat from Omegah Red's remix of Books of War - the song is played at a beat of roughly 90 BPM, but because of the changes in frequency and impulsive events that occur at higher multiples of the song rhythm, the algorithm wants to say that the song is at a tempo of roughly 180 BPM. This can be seen in Figure 9. Tighter constraints of upper tempo would fix this, but it's hard to imagine a way to do it without humans checking the computers work.

C. Beat Location

I ended up not attempting this part of the project. Although it is straightforward to create a synthetic pulse train and convolve it with the detection curve (as one does for a matched filter), I had no way of verifying my results or using them. If

Fig. 7. Sample Beat Detection with pianoChords.mp3 file

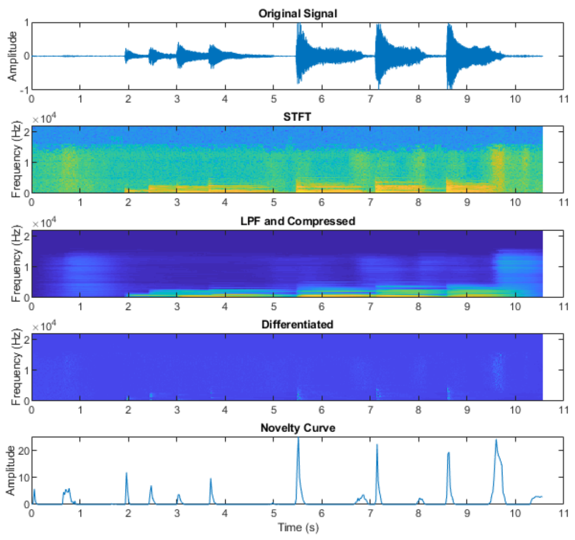
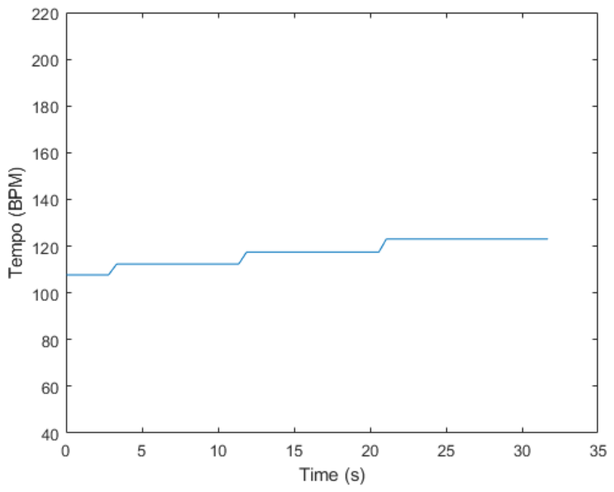


Fig. 8. Tempo Estimation using an increasing metronome signal



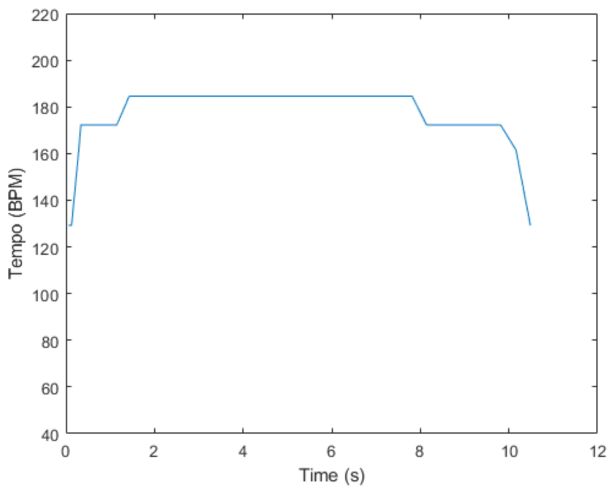
I had continued with the audio/video sync part of the project; this would have been utilized.

V. Summary

The project was fun little exploration into musical signal processing. In my past, I worked as an Noise, Vibration, and Harshness (NVH) engineer and had always wanted to try playing with musically oriented projects, but usually didn't want to sit down at a computer once I got home. This project was a great opportunity to get to do a little signal processing with some musical audio signals.

I'm not entirely satisfied with the performance of the beat detection algorithm. It needs to be tweaked for different types of music - for example the Beirut song doesn't give best beat detection unless low pass filtering is reduced/removed. The startup-transient seen on the novelty curve could also be

Fig. 9. Tempo Estimation Issue Example



improved, possibly by zero padding, but I didn't explore this possibility. An adaptive algorithm would be a logical next step if I was working on a production product.

I could have also spent more time working on the tempo estimation, but most of my efforts were spent tuning the beat detection. A small effort was put into doing visual beat detection as seen in [5], but after proving to myself that I could detect periodic visual changes I became busy and didn't return to the topic. I have included my very rough code in case you are interested in the process, as well as one demo file. If you choose to play with this, the xylophone.mp4 file included in the MATLAB demo files gives a better demo.

References

[1] M. A. Alonso, G. Richard, and B. David, "Tempo and beat estimation of musical signals." in *ISMIR*, 2004.

[2] P. Grosche and M. Müller, "Tempogram toolbox: Matlab implementations for tempo and pulse analysis of music recordings," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR), Miami, FL, USA*, 2011, pp. 24–28.

[3] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on speech and audio processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[4] M. Alonso, B. David, and G. Richard, "A study of tempo tracking algorithms from polyphonic music signals," in *4th COST 276 Workshop*, 2003.

[5] A. Davis and M. Agrawala, "Visual rhythm and beat." *ACM Trans. Graph.*, vol. 37, no. 4, pp. 122–1, 2018.