PULSEBOX: A SELF-CONTAINED BEAT VISUALIZER - TEAM ID: 24690

Creed Gallagher, *Student Member, IEEE,* Josh Preuss, *Student Member, IEEE,* William Heimsoth, *Student Member, IEEE.*
University of Maryland - College Park

Video demonstration: https://www.youtube.com/watch?v=KPwFnY6bJNI

ABSTRACT

We describe a self-contained audio processing system for determination and display of the musical beat in live music. The system uses two Raspberry Pi microcomputer modules (one for audio processing calculations, the other for display control) and a Teensy 3.5 microprocessor as display driver. The display is a 9-inch cube with 245 individually addressable RGB LED's arrayed on the surface. Given external power and music into its external microphone, the system calculates the tempo and the beat, refreshing these calculations every 11 milliseconds, and displays each beat using colorful rhythmic patterns from the LED's. The system, trained using a dataset of 149 musical excerpts, is capable of handling a diverse set of songs. The beat finding algorithm has been designed to function even when the tempo varies or is unstable.

Implementation was carried out in the following manner: We began by obtaining the hardware and constructing the cube. We originally planned to program one Raspberry Pi in C to perform all necessary tasks. As the complexity of the system became clear, we decided to divide the tasks among the 3 devices as described above. We also favored Python over C for most complex tasks, including all of the beat processing code. Implementing the beat tracking software involved continually improving and refining our methods. We gauged our performance by comparing our algorithm's detected beats to the pre-verified known beats for a sample set of songs. Finally, once we were satisfied with the beat tracking performance we designed various light patterns to visualize the beat.

1. INTRODUCTION

Real-time beat tracking is an audio processing challenge with deep relevance to the fields of music theory, psychoacoustics and digital signal analysis. There are also practical connections in such areas as clinical heart monitoring, neurology and artificial intelligence. Some early approaches to this challenge utilized simplified input, such as MIDI-encoded music and prerecorded music selected for regular rhythm. A paper in 2001 by Masataka Goto (Journal of New Music Research 30:159-171) describes an approach that handles some of the irregularities of live music. In that paper, the input signal is Fourier transformed and parsed into bands for beat detection; further processing establishes a prediction for the next beat, and a method of improving its predictions over time as the music progresses. Here, we have taken those basic concepts as starting suggestions, and developed a new more complex algorithm that incorporates features of machine learning. These features enabled the optimization of about 30 parameters, over hundreds of training runs on well-annotated (i.e., with beats known) songs, to give the algorithm greater flexibility in handling a wide variety of music with challenging, irregular and weak beats.

2. METHODS

Briefly described, our algorithm uses autocorrelated power onsets in the frequency parsed input to find the tempo and then uses those onsets along with the tempo to find the beats, refining its value of tempo every 350ms and refining its prediction of the next beat every 1ms. The algorithm can be outlined in the following 3 steps.

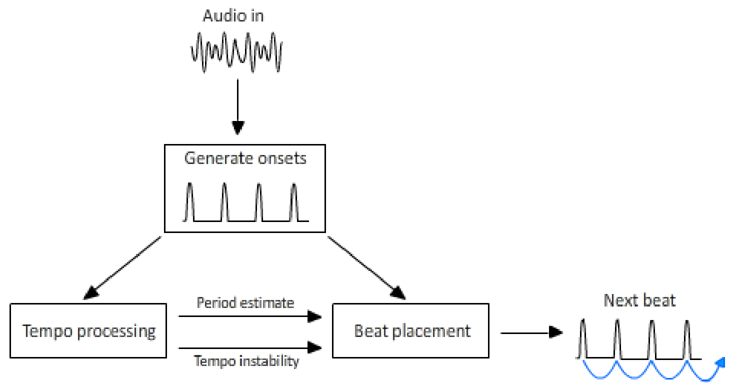

Fig. 1, overview of modules

2.1 Tempo estimation

The general approach is to find the tempo first and then the beats, because individual beat times are sometimes quiet and not easy to define without the framework of a tempo. First, the tempo is determined by finding power onset events in four frequency bands, and calculating autocorrelations of these onsets events. An onset is defined as positive changes in power, e.g. as described in Goto's paper [1]. Each onset data point represents one 46ms Hanning windowed frame, continuously shifted by 11.6 ms (256 samples at 22.05kHz).

Although the onsets are candidates for beat events, the beats themselves are not sought at this stage, only the tempo. Autocorrelation finds the repeating patterns and calculates a best guess for the tempo. This autocorrelation is performed for each frequency band and on various lengths of time (short, medium and long term). For each frequency band and duration, the strongest correlation peaks are cast as votes for the current tempo guess. Weightings and other parameters involved in combining this data to give the overall tempo estimate were trained using machine learning methods. In general, this approach of deconstructing the audio, processing it, and reconstructing with proper weightings and techniques, proved to give a more reliable tempo estimate than simply picking the largest peak of the overall power autocorrelation.

The initial tempo guess is usually available after a few seconds, and is often incorrect, but the algorithm is designed to continually incorporate new audio data to refine the guessed tempo. As time progresses, the tempo algorithm will analyze trends from past tempo data. If the algorithm is believed to have settled on a reliable tempo, it will generally resist large tempo changes.

The tempo as determined above, and at any point in time, corresponds to a period with a value in seconds. Importantly, it also has an associated confidence level, or stability. For a simple song with a clear beat, this tempo stability number will rise quickly and remain high. In more complex songs, this number will vary as the tempo shifts or beats become unclear. Discrepancies between the short, medium, and long term correlations are one indicator of instability.

Although the tempo processing portion of the algorithm requires 12 correlations (4 frequency bands and 3 durations), it only need be calculated as often as we would expect differences to appear in the tempo (we chose about 3 times per second, 350ms). Additionally, autocorrelation can be performed in O(nlogn) time and is only performed on more recent audio data making tempo processing only a small burden on the processor.

2.2 Beat placement

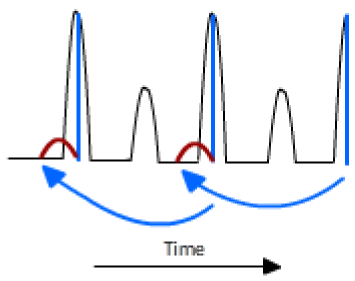

Fig. 2, blue arrows represent tempo spacing of comb. Dark red represents snapped distance to nearby peak. Snapping fixes small tempo error (as above) and handles music with irregular tempo elements

Every 11 msec, the precise time of the next expected beat is predicted, based on a "snapping comb sum" approach. This approach makes use of the period (trivially related to the tempo) and instability values from the tempo processing portion of the algorithm.

The "comb" refers to taking samples of the power onset vector at equally spaced intervals. In this case, we chose the current period estimate as the interval. At the offset where the beat lies, we would expect to find the largest sum when adding up each sample spaced at the period. If this is true in general, which it is for most songs, quiet or irregular sections of a song will still have their beats placed correctly.

One problem with this method is that a very precise tempo estimate is needed. If we imagine adding up power onset values as we backtrack (from the current time back to the beginning of the song) by the period to create the comb sum, a small tempo error will compound. Even though most of our tempo estimations were accurate to about 20ms, this would result in an error of 200ms after backtracking 10 beats, i.e. completely offbeat for a 150bpm song.

In order to deal with this, we implemented a "snapping" feature. After backtracking by the tempo, we search a range of nearby samples and select the largest peak. Then we backtrack from that peak, iteratively forming the comb and calculating the sum. Thus, the "teeth" of the comb need not be exactly evenly spaced.

Possibly the most important takeaway from our algorithm is that the instability rating from the tempo processing portion of the algorithm determines the degree to which we should snap. A song that seems stable can be tracked with a small snapping value of 20ms, as we believe the tempo is accurate. An unstable song, i.e. a live song with a tempo that is not synced to a metronome, may benefit from a snapping range of 60ms. A song with a changing tempo, if it is gradual enough, is also implicitly handled by this approach.

2.3 Optimization through Machine Learning

Thirty one parameters governing the details of those processes, were optimized through hundreds of training runs on a set of well-annotated songs. For example, there are parameters for the frequency bands and autocorrelation windows, and for how much to weight recent, vs old, information for adjusting the tempo and predicted beat values. Optimizing all these parameters is a complex problem. Many of them started with intuitive assigned values. We designed a machine learning approach in which each parameter was varied while others were fixed, cycling through all parameters many times and monitoring algorithm behavior on individual songs. Through the optimizations, our calculated scores (percentage of predicted beats meeting the criteria as discussed in the results section) rose from about 60% to over 70%.

3. LED Cube

Visual Display. For visual display of the real-time calculated beat, we constructed a square tower of dimensions 9 x 9 x 11 inches (almost a cube), and covered its 5 exposed faces with strips of LEDs. The LEDs are individually controllable with 24 bit color. Each face has 7 strips of 7 LEDs, for a total of 245 LEDs. The design enables placing all processor components (two Raspberry Pi 3's and a Teensy 3.5) inside, so that there are a few external connections: for power, the microphone, and a start switch. Please see the video that accompanies this paper.

The tower was first built using square wooden dowels, screws and glue; then covered with black posterboard. Strips of APA102 individually addressable LEDs were cut from long rolls and soldered as needed for interior connections and power/signal connectors. Finally the LED strips were attached to the tower using double-sided tape. The design gives inter-strip spacing equal to the inter-LED spacing along each strip, so that the arrays are square.

This video shows two songs – one easy and one hard – which perform well.
https://www.youtube.com/watch?v=KPwFnY6bJNI

This supplementary video shows some additional songs.
https://www.youtube.com/watch?v=mCjMvdbCPes

4. Results

Accuracy scores were determined using beatEvalator.m provided by IEEE, which followed evaluation strategies such as those in [7]. It counts beats as accurate if they are within a 17.5% tolerance for offset and tempo.

Our system predicts and displays the beat accurately for most music. Accuracy was over 70% for the dataset as a whole, which included many songs with tempo elements that made them difficult to track. Easy songs, such as dance music with strong drums, are tracked with much higher accuracy.

Accuracy scores were obtained using .wav files, but were also observed using a microphone input. Performance is reduced slightly when audio is captured via microphone, due to imperfect frequency response of the speakers, room, and microphone. This was mitigated by the use of proper equipment and mic placement.

5. References

[1] M. Goto, "An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.
[2] Y. Shiu, N. Cho, and J. Kuo, "Robust On-line Beat Tracking with Kalman Filtering and Probabilistic Data Association (KF-PDA)," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1369–1377, Aug. 2008.
[3] S. Durand, J. Bello, B. David, and G. Richard, "Downbeat Tracking with Multiple Features and Deep Neural Networks,".
[4] D. Ellis, "Beat Tracking by Dynamic Programming," Jul. 2007.
[5] E. Scheirer, "Tempo and beat analysis of acoustic musical signals," *Journal of Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.
[6] S. Dixon, "Evaluation of the Audio Beat Tracking System BeatRoot," *Journal of New Music Research*, vol. 36, 2008.
[7] S. Hainsworth, "Techniques for the automated analysis of musical audio," Ph.D. dissertation, Department of Engineering, Cambridge University, 2004.