BEAT DETECTION BY DYNAMIC PROGRAMMING

Racquel Ivy Awuor

University of Rochester
Department of Electrical and Computer Engineering
Rochester, NY 14627
rawuor@ur.rochester.edu

ABSTRACT

A beat is a salient periodicity in a music signal. It provides a fundamental unit of time and foundation for the temporal structure of music. The significance of beat tracking is that it underlies music information retrieval research and provides for beat synchronous analysis of music. It has applications in segmentation of audio, interactive music accompaniment, cover song detection, music similarity, chord estimation and music transcription.[7]

The goal of this project is to implement a beat tracker system and to demonstrate the performance with creative output such as, but not limited to drumming, pop music, or flickering lights. This paper begins by exploring the underlying theory of Dynamic programming and why it is a preferred method of beat tracking compared to earlier methods of beat detection. It then proceeds to demonstrate the implementation of the beat detection system and concludes with results demonstrating the efficiency of the system and other possible tasks that can be performed by a beat tracking system.

Index Terms— Dynamic Programming, Beat tracking, Tempo estimation, Beat Detection

1. INTRODUCTION

Over the years, researchers have built and tested systems for beat tracking in audio signals. These range from the 'foot tapping' systems of Desain and Honing [1999], which were largely comprised of symbolically-encoded event times, to the more recent audio driven systems as evaluated in the MIREX-06 Audio Beat Tracking evaluation [McKinney and Moelants, 2006], and more recently, implementations using dynamic programming algorithms [Ellis, 2007][1] which implements a well-known algorithm first proposed by Bellman [1957][4].

The idea of using dynamic programming for beat tracking was first proposed by Laroche [2003][5] where the onset function is equated to a predefined envelope spanning multiple beats that incorporated expectations concerning how a particular tempo is realized in terms of strong and weak beats; dynamic programming efficiently enforced continuity in both beat spacing and tempo. Since then, the idea has further been pursued by researchers such as Peeters [2007][6] who used the idea, while allowing for tempo variation and matching the envelope patterns against templates, as well as Ellis [2007] [1]who, in contrast to Peeters, implemented a relatively simple system, which assumes a constant tempo which allows a much simpler formulation and realization, at the cost of a more limited scope of application.

This work focuses on demonstrating the effectiveness of dynamic programming in the implementation of a simple beat tracking system. This paper is organized as follows. In section 2, the idea of formulating beat tracking as the optimization of a recursively-calculable cost function is introduced. In the following section (section 3), the implementation of the beat tracking system including details of how the onset strength function is derived, is described. Section 4 describes the details of the results of applying the system compared to data collected from users (tapping using Sonic Visualizer data and a score comparison function). The final section is a conclusion on the effectiveness of the dynamic programming algorithm as well as future advancements that can be made to improve the system in future.

2. DYNAMIC PROGRAMMING FOR BEAT DETECTION

Assuming we have a constant target tempo which is given in advance, we can specify the goal of the beat tracking system to generate a sequence of beat times that correspond to both the perceived onsets of the audio signal as well as the rhythmic pattern of the audio signal, which is related to the tempo of the system. We can define a single function that achieves both of these aims as follows[1]:

$$C(\{t_i\}) = \sum_{i=1}^{N} O(t_i) + \alpha \sum_{i=2}^{N} F(t_i - t_{i-1}, \tau_p) \quad (1)$$

In the above equation, $\{t_i\}$ is the sequence of N beat instants found by the tracker, $O(t)$ is an "onset strength envelope" derived from the audio, which is large at times that would make good choices for beats based on the local acoustic properties, α is a weighting to balance the importance of the two terms, and $F(\Delta t, \tau_p)$ is a function that measures the consistency between an inter-beat interval Δt and the ideal beat spacing τ_p defined by the target tempo. In this work, the consistency function is as derived by Ellis [2007] , where it is a simple squared-error function applied to the log-ratio of actual and ideal time spacing[1] i.e.

$$F(\Delta t, \tau) = -\left(\log \frac{\Delta t}{\tau}\right)^2 \quad (2)$$

The function takes a maximum value of 0 when $\Delta t = \tau$. This function becomes negative for larger values of Δt .

To calculate the best possible score of all sequences, we define a recursive relation as follows[1]:

$$C^*(t) = O(t) + \max_{\tau=0\ldots t} \{\alpha F(t-\tau, \tau_p) + C^*(\tau)\} \quad (3)$$

This is based on the observation that the best score for a given time t is the local onset strength plus the best score to the preceeding beat time τ that maximizes the sum of that best score and the transition cost from that time. While calculating the best score, we also keep track of the preceeding beat time that gives the best score[1].

$$P^*(t) = \arg\max_{\tau=0\ldots t} \{\alpha F(t-\tau, \tau_p) + C^*(\tau)\} \quad (4)$$

While it is only necessary to search a limited temporal range of the signal we search the range of $\tau = t - 2\tau_p$ to $t - \tau_p/2$. This is because it is unlikely that the best predecessor time lies outside the defined range [1].

To find the set of beat times that optimize the objective function for a given onset envelope we start by calculating C^* and p^* for every time starting from zero. Once this is completed, we can find the largest value of the score. This forms the final beat instant of the given signal. We can then trace P^* finding the preceding beat time and progressively work backwards until we get to the start of the signal. This gives the entire optimal beat sequence $\{t_i\}^*$.

As demonstrated above, dynamic programming effectively searched the entire exponentially sized set of all possible time sequences in a linear time operation. This was possible because, if a best scoring beat sequence includes a time t_i , the beat instants chosen after t_i will not influence the choice (or score contribution) of beat times prior to the defined time [1]. This means that the best scoring sequence can be determined at a fixed time without having to consider any future events. As such dynamic programming represents a fairly simple way of completing a relatively complex audio processing task as beat detection.

3. THE BEAT DETECTION SYSTEM

This work borrows heavily from the work proposed by Ellis [2007]. The system works by searching for the globally-optimal beat sequence and using these to reconstruct a final output of a signal comprised of the detected beats mixed into the original signal. The block diagram of the implemented system is as follows:

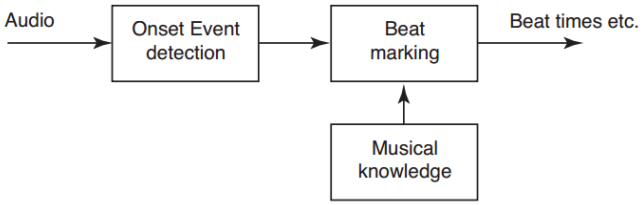

Figure 1: Block diagram of the beat detection system

3.1 Onset Strength Envelope

The envelope is calculated using a crude conceptual model, which has been demonstrated by onset models presented by previous research work [1][2][3]. First of all, the input sound is resampled to 8 kHz. The output is then used to calculate the short-term Fourier transform (STFT) magnitude (spectrogram) using 32 ms windows and 4ms advance between frames. This is then converted to an approximate auditory representation by mapping it to 40 Mel bands, via a weighted summing of the spectrogram values [Ellis,2005]. This is followed by an auditory frequency scale in an effort to balance the perceptual importance of each frequency band. The Mel spectrogram is then converted to d B and the first order difference along time is calculated in each band. Negative values are set to zero (half-wave rectification), then the remaining differences (positive ones) are summed across all frequency bands. This signal is then passed through a high pass filter with cutoff around 0.4Hz to make it locally zero mean, and smoothed by convolving with a Gaussian envelope about 20ms wide. This gives a one dimensional onset strength envelope as a function of time that responds to proportional increase in energy summed across approximately auditory frequency bands.

Since the balance between the two terms in the objective function of equation 1 depends on the overall scale of the onset function, which itself may depend on the instrumentation or other aspects of the signal spectrum, we

normalize the onset envelope for each musical excerpt by dividing by its standard deviation.

3.2. Global Tempo Estimate

Given the onset strength envelope $O(t)$ of the previous section, autocorrelation can reveal any regular periodic structure. For a periodic signal, there will also be large correlations at any integer multiples of the basic period (as the peaks line up with the peaks that occur two or more beats later), and it can be difficult to choose a single best peak among many correlation peaks of comparable magnitude. However, human tempo estimation is known to have a bias towards 120 BPM. We apply a perceptual weighting window to the raw autocorrelation to down-weigh periodicity peaks from this bias, then interpret the scaled peaks as indicative of the likelihood of a human choosing that period as the underlying tempo. Specifically, the tempo period strength is given by[1]:

$$TPS(\tau) = W(\tau) \sum_{t} O(t)O(t-\tau) \quad (5)$$

$W(\tau)$ is a Gaussian weighting functionon a log time axis[1]:

$$W(\tau) = \exp\left\{-\frac{1}{2}\left(\frac{log_2\tau/\tau_0}{\sigma_\tau}\right)^2\right\} \quad (6)$$

In this case τ_0 is the center of the tempo period bias, and σ_τ controls the width of the weighting curve (in octaves). The primary tempo period estimate is then the time difference for which the TPS has the largest value.

4. RESULTS

The system was implemented as the GUI shown in the figure in figure 4. Among the functionalities included in the GUI are an audio player function, a beat detection function, a beat randomizer function (which randomizes the placement of the beats, like an audio mixer) and a beat randomizer with metre (this randomizes the placement of the beats detected in the signal, while maintaining a temporal continuum in the perception of the signal, i.e., the recurring pattern of stresses or accents that provide the audio signal with the pulse or beat of the music is maintained.). While this paper doesn't directly focus on the details of beat randomization and its implementation, these functionalities are just an example of the possible ways by which we can expand the scope of the beat detection system implemented in this paper.

The accuracy of the beat detection system was evaluated in comparison to beat detection figures derived from human subjects, using the Sonic Visualizer software (http://www.sonicvisualiser.org/download.html). An audio signal was uploaded and the subjects recorded the perceived beats using the ';' key on the keyboard. The recorded beats were then played on Sonic Visualizer, alongside the beats determined by the beat detection system to assess the accuracy of the system in general. It was generally observed that for audio files that were highly rhythmic, the beats detected matched closely the beats detected by the human subject. For a signal that had a more randomized rhythmic sequence, the beat detection algorithm produced a beat sequence that was slightly delayed compared to the beat sequence perceived by the human subject. Further the accuracy of the detection system was evaluated in terms of the number of beats detected by the algorithm compared to the number of beats detected by the human subject. Although this is a more rudimentary way of testing accuracy, the evaluation was in favor of the accuracy of the algorithm implemented, as shown in the table below:

Song	Human-Detected tempo	Machine detected tempo	Difference (absolute)
Song 1	110	109.89	0.11
Song 2	110	109.5	0.5
Song 3	185	186.19	1.19
Song 4	91	55.98	35.02
Song 5	78	59.72	18.28
Song 6	152	88.56	63.44
Song 7	119	80.06	38.94
Song 8	139	140.33	1.33
Song 9	118	118.13	0.13
Song 10	110	112.42	2.42
Average			16.136

Table 1: Showing the performance of the beat detection system in comparison to human beat detection data acquired via Sonic Visualizer system.

While the expected difference is ideally 0, the system does have some deviation from the intended function. Overall, out of 10 songs, I observed dismal performance for 4 songs, which were comprised of a variation of beats and therefore it was fairly difficult to standardize the global tempo for the signal, which leads to a poor performance of the system. However, the average performance for a total of 1212 beats, the system had a variation of 161.36, which 13.31% of the overall system.

Inasmuch as an error of 13.31% is not small, the system overall proves to be robust for audio signals that have a more predictable rhythm. In addition, it demonstrates versatility in potential work that can be done using a beat detection system (i.e., can potentially be transformed into an audio mixing system.

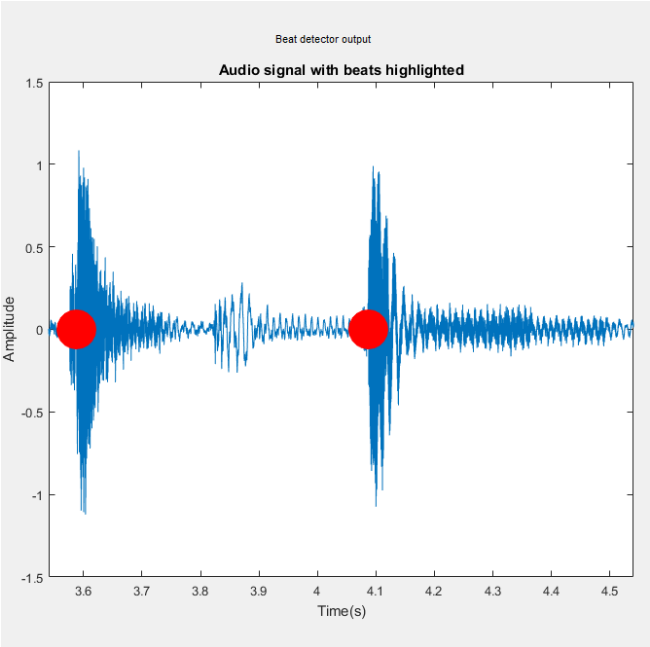

Figure 2: Beat detection output. Beats are highlighted in red, while audio signal is in blue.

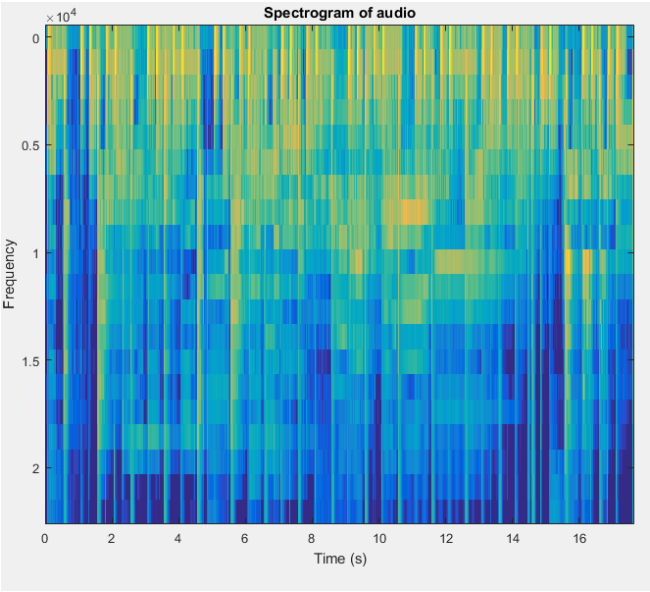

Figure 3: Output spectrogram of the audio signal

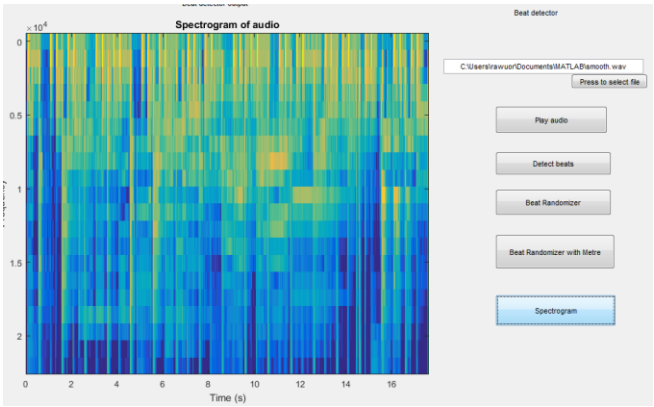

Figure 4: GUI of implemented beat detection system

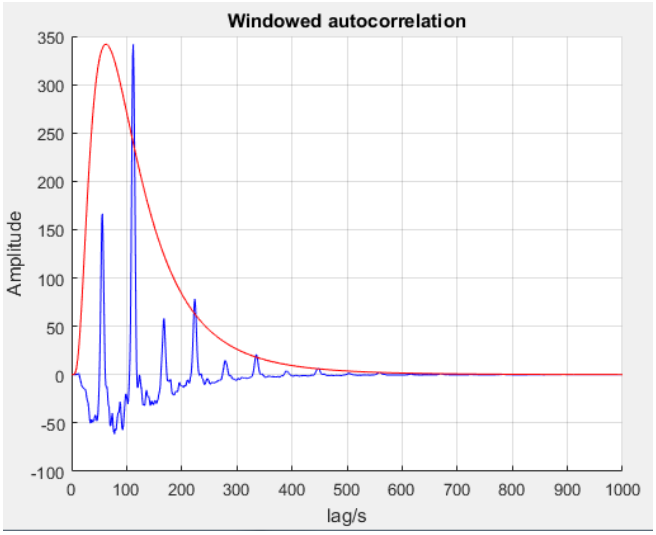

Figure 5: Showing the windowed autocorrelation window plotted against the weighting window applied to give the TPS function, for audio file 'Pop.wav'

5. CONCLUSION

This project successfully demonstrates the ability of dynamic programming in implementing a beat detection system. While it is a rudimentary version of an ideal system, it can be further expanded to a stand-alone audio mixing system. In addition, further improvements can be made to the proposed algorithm to allow for finer beat detection even in systems with complex rhythm. Nonetheless, this project demonstrates that commercially viable and fairly accurate beat detection systems can be implemented using dynamic programming.

6. REFERENCES

[1] D.P.W Ellis, "Beat Tracking by Dynamic Programming," *Journal*, Publisher, Location, pp. 1-10, Date.

[2] P. Desain, H. Honing, "Computational models of beat induction: The rule-based approach",*Journal of New Music Research*, 28(1):29-42, 1999.

[3] M.F. McKinney, D. Moelants, M. Davies, and A. Klapuri, “Evaluation of audio beat tracking and music tempo extraction algorithms”, *Journal of New Music Research*, 2007.
[4] R. Bellman, “Dynamic Programming”, Princeton University Press, 1957
[5] J. Laroche, “Efficient tempo and beat tracking in audio recordings”, *Journal of the Audio Engineering Society*, 51(4):226-233, April 2003.
[6] G. Peeters. “Template-based estimation of time-varying tempo”, *EURASIP Journal on Advances in Signal Processing*, 2007(Article ID 67215):14 pages, 2007, URL 10.1155/2007/67215.
[7] D. Levitin, S. Hainsworth, D. Ellis, M. Plumbley, S.Dixon, M. Muller, “IEEE Signal Processing Cup 2017”, Retrieved: https://piazza-syllabus.s3.amazonaws.com/ip7857wq1zi19q/ASSP_TC_SPCup_2017.pdf?AWSAccessKeyId=AKIAIEDNRLJ4AZKBW6HA&Expires=1494043938&Signature=41dnBJv34SBbGhqHZFSHLxBkWbE%3D, 05/05/2017.
[8] M.E.P Davies, “Introduction to musical beat tracking and creative transformations in MATLAB”, Retrieved: http://xxi.sinfo.org/index.php/home/schedule/workshops/matthew-davies, 05/05/2017