

Beat Tracking by Dynamic Programming

Daniel P.W. Ellis

LabROSA, Columbia University, New York

July 16, 2007

Abstract

Beat tracking – i.e. deriving from a music audio signal a sequence of beat instants that might correspond to when a human listener would tap his foot – involves satisfying two constraints: On the one hand, the selected instants should generally correspond to moments in the audio where a beat is indicated, for instance by the onset of a note played by one of the instruments. On the other hand, the set of beats should reflect a locally-constant inter-beat-interval, since it is this regular spacing between beat times that defines musical rhythm. These dual constraints map neatly onto the two constraints optimized in dynamic programming, the local match, and the transition cost. We describe a beat tracking system which first estimates a global tempo, uses this tempo to construct a transition cost function, then uses dynamic programming to find the best-scoring set of beat times that reflect the tempo as well as corresponding to moments of high ‘onset strength’ in a function derived from the audio. This very simple and computationally efficient procedure is shown to perform well on the MIREX-06 beat tracking training data, achieving an average beat accuracy of just under 60% on the development data. We also examine the impact of the assumption of a fixed target tempo, and show that the system is typically able to track tempo changes in a range of $\pm 10\%$ of the target tempo.

1 Introduction

Researchers have been building and testing systems for tracking beat times in music for several decades, ranging from the ‘foot tapping’ systems of Desain and Honing [1999], which were driven by symbolically-encoded event times, to the more recent audio-driven systems as evaluated in the MIREX-06 Audio Beat Tracking evaluation [McKinney and Moelants, 2006a]; a more complete overview is given in the lead paper in this collection [McKinney et al., 2007].

Here, we describe a system that was part of the latter evaluation, coming among the statistically-equivalent top-performers of the five systems evaluated. Our system casts beat tracking into a simple optimization framework by defining an objective function that seeks to maximize both the “onset strength” at every hypothesized beat time (where the onset strength function is derived from the music audio by some suitable mechanism), and the consistency of the inter-onset-interval with some pre-estimated constant tempo. (We note in passing that human perception of beat instants tends to smooth out inter-beat-intervals rather than adhering strictly to maxima in onset strength [Dixon et al., 2006], but this could be modeled as a subsequent, smoothing stage). Although the requirement of an *a priori* tempo is a weakness, the reward is a particularly efficient beat-tracking system that is guaranteed to find the set of beat times that optimizes the objective function, thanks to its ability to use the well-known dynamic programming algorithm [Bellman, 1957].

The idea of using dynamic programming for beat tracking was proposed by Laroche [2003], where an onset function was compared 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. Peeters [2007] developed this idea, again allowing for tempo variation and matching of envelope patterns against templates. By contrast, the current system assumes a constant tempo which allows a much simpler formulation and realization, at the cost of a more limited scope of application.

The rest of this paper is organized as follows: In section 2, we introduce the key idea of formulating beat tracking as the optimization of a recursively-calculable cost function. Section

3 describes our implementation, including details of how we derived our onset strength function from the music audio waveform. Section 4 describes the results of applying this system to MIREX-06 beat tracking evaluation data, for which human tapping data was available, and in section 5 we discuss various aspects of this system, including issues of varying tempo, and deciding whether or not any beat is present.

2 The Dynamic Programming Formulation of Beat Tracking

Let us start by assuming that we have a constant target tempo which is given in advance. The goal of a beat tracker is to generate a sequence of beat times that correspond both to perceived onsets in the audio signal at the same time as constituting a regular, rhythmic pattern in themselves. We can define a single objective function that combines both of these goals:

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

where $\{t_i\}$ 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. For instance, we use a simple squared-error function applied to the log-ratio of actual and ideal time spacing i.e.

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

which takes a maximum value of 0 when $\Delta t = \tau$, becomes increasingly negative for larger deviations, and is symmetric on a log-time axis so that $F(k\tau, \tau) = F(\tau/k, \tau)$. In what follows, we assume that time has been quantized on some suitable grid; our system used a 4 ms time step (i.e.

250 Hz sampling rate).

The key property of the objective function is that the best-scoring time sequence can be assembled recursively i.e. to calculate the best possible score $C^*(t)$ of all sequences that end at time t , we define the recursive relation:

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

This equation is based on the observation that the best score for time t is the local onset strength, plus the the best score to the preceding beat time τ that maximizes the sum of that best score and the transition cost from that time. While calculating C^* , we also record the actual preceding beat time that gave the best score:

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

In practice it is necessary only to search a limited range of τ since the rapidly-growing penalty term F will make it unlikely that the best predecessor time lies far from $t - \tau_p$; we search $\tau = t - 2\tau_p \dots t - \tau_p/2$.

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 complete, we look for the largest value of C^* (which will typically be within τ_p of the end of the time range); this forms the final beat instant t_N – where N , the total number of beats, is still unknown at this point. We then ‘backtrace’ via P^* , finding the preceding beat time $t_{N-1} = P^*(t_N)$, and progressively working backwards until we reach the beginning of the signal; this gives us the entire optimal beat sequence $\{t_i\}^*$. Thanks to dynamic programming, we have 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 t_i , so the entire best-

scoring sequence up to time t_i can be calculated and fixed at time t_i without having to consider any future events. By contrast, a cost function where events subsequent to t_i could influence the cost contribution of earlier events would not be amenable to this optimization.

To underline its simplicity, figure 1 shows the complete working Matlab code for core dynamic programming search, taking an onset strength envelope and target tempo period as input, and finding the set of optimal beat times. The two loops (forward calculation and backtrace) consist of only ten lines of code.

3 The Beat Tracking System

The dynamic programming search for the globally-optimal beat sequence is the heart and the main novel contribution of our system; in this section, we present the other pieces required for the complete beat-tracking system. These comprise two parts: the front-end processing to convert the input audio into the onset strength envelope, $O(t)$, and the global tempo estimation which provides the target inter-beat interval, τ_p .

3.1 Onset Strength Envelope

Similar to many other onset models (e.g. Goto and Muraoka [1994], Klapuri [1999], Jehan [2005]) we calculate the onset envelope from a crude perceptual model. First the input sound is resampled to 8 kHz, then we calculate the short-term Fourier transform (STFT) magnitude (spectrogram) using 32 ms windows and 4 ms advance between frames. This is then converted to an approximate auditory representation by mapping to 40 Mel bands via a weighted summing of the spectrogram values [Ellis, 2005]. We use an auditory frequency scale in an effort to balance the perceptual importance of each frequency band. The Mel spectrogram is converted to dB, and the first-order difference along time is calculated in each band. Negative values are set to zero (half-wave rectification), then the remaining, positive differences are summed across all frequency bands. This signal is passed through a high-pass filter with a cutoff around 0.4 Hz to make it locally zero-

```

function beats = beatssimple(localscore, period, alpha)
% beats = beatssimple(localscore, period, alpha)
% Core of the DP-based beat tracker
% <localscore> is the onset strength envelope
% <period> is the target tempo period (in samples)
% <alpha> is weight applied to transition cost
% <beats> returns the chosen beat sample times.
% 2007-06-19 Dan Ellis dpwe@ee.columbia.edu

% backlink(time) is best predecessor for this point
% cumscore(time) is total cumulated score to this point
backlink = -ones(1,length(localscore));
cumscore = localscore;

% Search range for previous beat
prange = round(-2*period):-round(period/2);
% Log-gaussian window over that range
txcost = (-alpha*abs((log(prange/-period)).^2));

for i = max(-prange + 1):length(localscore)

    timerange = i + prange;

    % Search over all possible predecessors
    % and apply transition weighting
    scorecands = txcost + cumscore(timerange);
    % Find best predecessor beat
    [vv,xx] = max(scorecands);
    % Add on local score
    cumscore(i) = vv + localscore(i);
    % Store backtrace
    backlink(i) = timerange(xx);

end

% Start backtrace from best cumulated score
[vv,beats] = max(cumscore);
% .. then find all its predecessors
while backlink(beats(1)) > 0
    beats = [backlink(beats(1)),beats];
end

```

Figure 1: Matlab code for the core dynamic programming search, taking the onset strength envelope and the target tempo period as inputs, and returning the indices of the optimal set of beat times. The full system also requires code to calculate the onset strength envelope and the initial target tempo, which is not shown.

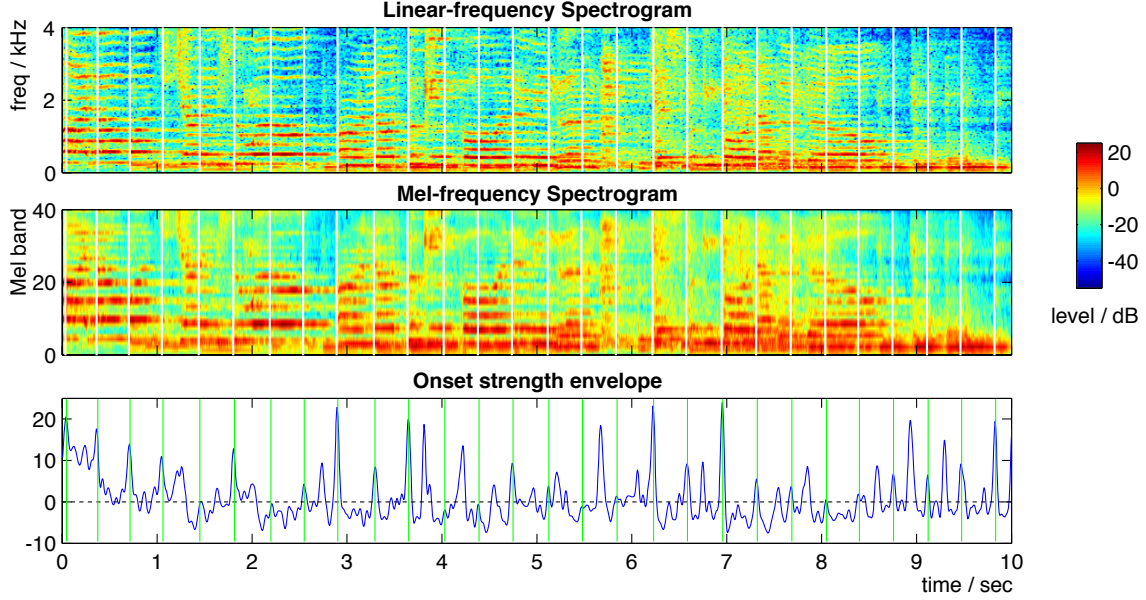


Figure 2: Comparison of conventional spectrogram (top pane, 32 ms window), Mel-spectrogram (middle pane), and the onset strength envelope calculated as described in the text. Vertical bars indicate beat times as found by the complete system.

mean, and smoothed by convolving with a Gaussian envelope about 20 ms 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. Figure 2 shows an example of the STFT spectrogram, Mel spectrogram, and onset strength envelope for a brief example of singing plus guitar (‘train2’ from the MIREX-06 beat tracking data [McKinney and Moelants, 2006a]). Peaks in the onset envelope evidently correspond to times when there are significant energy onsets across multiple bands in the signal.

Since the balance between the two terms in the objective function of eqn. 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

The dynamic programming formulation of section 2 was dependent on prior knowledge of a target tempo (i.e. the ideal inter-beat interval τ_p); here, we describe how this is estimated. Given the onset strength envelope $O(t)$ of the previous section, autocorrelation will reveal any regular, periodic structure i.e. we take the inner product of the envelope with delayed versions of itself, and for delays that succeed in lining up many of the peaks, a large correlation occurs. 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 perception (as might be examined by asking subjects to tap along in time to a piece of music [McKinney and Moelants, 2006b]) is known to have a bias towards 120 BPM. We apply a perceptual weighting window to the raw autocorrelation to downweight periodicity peaks far from this bias, then interpret the scaled peaks as indicative of the likelihood of a human choosing that period as the underlying tempo. Specifically, our tempo period strength is given by:

$$TPS(\tau) = W(\tau) \sum_t O(t)O(t - \tau) \quad (5)$$

where $W(\tau)$ is a Gaussian weighting function on a log-time axis:

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

where τ_0 is the center of the tempo period bias, and σ_τ controls the width of the weighting curve (in octaves, thanks to the $\log_2$). The primary tempo period estimate is then simply the τ for which $TPS(\tau)$ is largest.

To set τ_0 and σ_τ , we used the MIREX-06 Beat Tracking training data, which consists of the actual tapping instants from 40 subjects asked to tap along with a number of 30 s musical excerpts [McKinney and Moelants, 2006a, Moelants and McKinney, 2004]. Data were collected for 160

excerpts, of which 20 were released as training data for the competition. Excerpts were chosen to give a broad variety of tempos, instrumentation, styles, and meters [McKinney and Moelants, 2005]. In each of these examples, the subject tapping data could be clustered into two groups, corresponding to slower and faster levels of the metrical hierarchy of the music, which were separated by a ratio of 2 or 3 (as a result of the particular rhythmic structure of the particular piece). The proportions of subjects opting for faster or slower tempos varied with each example, but it is notable that all examples resulted in two distinct response patterns. To account for this phenomenon, we constructed our tempo estimate to identify a secondary tempo period estimate whose value is chosen from among $(0.33, 0.5, 2, 3) \times$ the primary period estimate, choosing the period that had the largest *TPS*, and simply using the ratio of the *TPS* values as the relative weights (likelihoods) of each tempo.

The values returned by this model were then compared against ground truth derived from the subjective tapping data, and τ_0 and σ_τ adjusted to maximize agreement between model and data. “Agreement” was based only on the two tempos reported and not the relative weights estimated by the system, although the partial score for matching only one of the two ground-truth tempos was in proportion to the number of listeners who chose that tempo. The best agreement of 77% was achieved by a τ_0 of 0.5 s (corresponding to 120 BPM, as expected from many results in rhythm perception), and a σ_τ of 1.4 octaves. Figure 3 shows the raw autocorrelation and its windowed version, *TPS*, for the example of fig 2, with the primary and secondary tempos marked.

Subsequent to the evaluation, examination of errors made by this approach led to a slight modification in which, rather than simply choosing the largest peak in the base *TPS*, two further functions are calculated by resampling *TPS* to one-half and one-third, respectively, of its original length, adding this to the original *TPS*, then choosing the largest peak across both these new

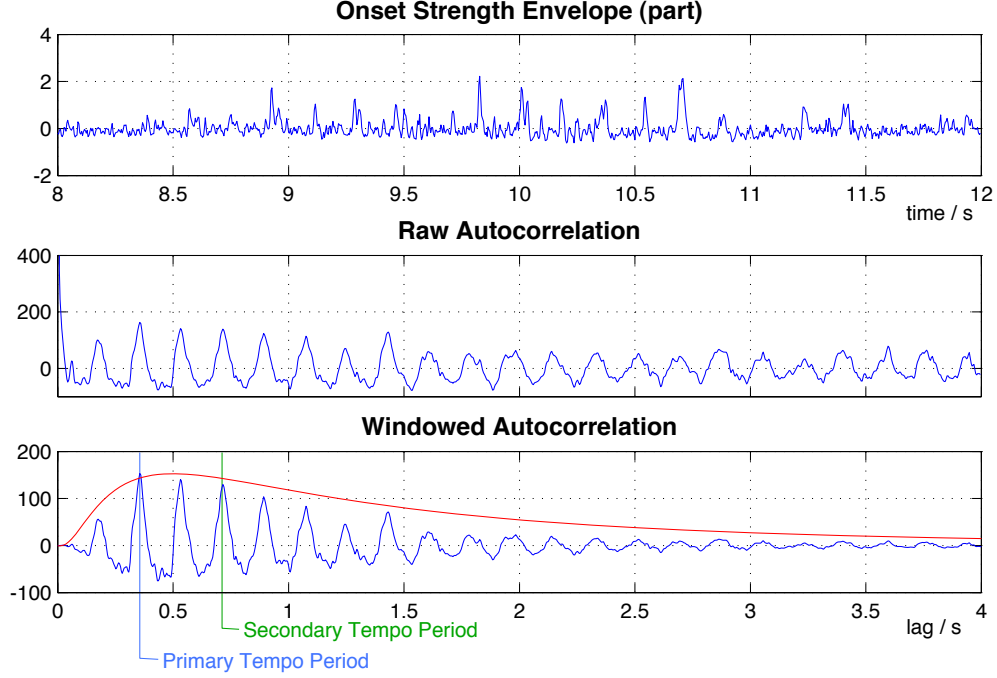


Figure 3: Tempo calculation. Top: Onset strength envelope excerpt from around 10 s into the excerpt. Middle: Raw autocorrelation. Bottom: autocorrelation with perceptual weighting window applied to give the *TPS* function. The two chosen tempos are marked.

sequences i.e. treating τ as a discrete, integer time index,

$$TPS2(\tau) = TPS(\tau) + 0.5TPS(2\tau) + 0.25TPS(2\tau - 1) + 0.25TPS(2\tau + 1) \quad (7)$$

$$TPS3(\tau) = TPS(\tau) + 0.33TPS(3\tau) + 0.33TPS(3\tau - 1) + 0.33TPS(3\tau + 1) \quad (8)$$

Whichever sequence contains the larger value determines whether the tempo is considered duple or triple, respectively, and the location of the largest value is treated as the faster target tempo, with one-half or one-third of that tempo, respectively, as the adjacent metrical level. Relative weights of the two levels are again taken from the relative peak heights at the two period estimates in the original *TPS*. This approach finds the tempo that maximizes the sum of the *TPS* values at both metrical levels, and performs slightly better on the development data, scoring 84% agreement. In this case, the optimal τ_0 was the same, but the best results required a smaller σ_τ of 0.9 octaves.

4 Experimental Results

4.1 Tempo estimation

The tempo estimation system was evaluated within the MIREX-06 Tempo Extraction contest, where it placed among the least accurate of the 7 algorithms compared. More details and analysis are provided in McKinney et al. [2007].

For an alternative comparison, we also ran the tempo estimation system on part of the data used in the 2004 Audio Description Contest for Tempo [Gouyon et al., 2006]. Specifically, we used the 465 “song excerpt” examples which have been made publicly available. The attraction of using this data is that it is completely separate from the tuning procedure used to set the tempo system parameters. In 2004, twelve tempo extraction algorithms were evaluated on this data against a single ground truth per excerpt derived from beat times marked by an expert. The algorithm accuracy ranged from 17% to 58.5% exact agreement with the expert reference tempo (“accuracy 1”), which improved to 41% to 91.2% when a tempo at a factor of 2 or 3 above or below the reference tempo was also accepted (“accuracy 2”).

Running the original tempo extraction algorithm of section 3.2 (global maximum of TPS) scored 35.7% and 74.4% for accuracies 1 and 2 respectively, which would have placed it between 5th and 6th place in the 2004 evaluation for accuracy 1, and between 3rd and 4th for accuracy 2. The modified tempo algorithm (taking the maximum of $TPS2$ or $TPS3$) improves performance to 45.8% and 80.6%, which puts it between 1st and 2nd, and 2nd and 3rd, for accuracy 1 and 2 respectively. It is still significantly inferior to the best algorithm in that evaluation, which, as in 2006, was from Klapuri.

4.2 Beat tracking

The complete beat tracking system was evaluated against the same data used to tune the tempo estimation system, namely the 20, 30 s excerpts of the MIREX-06 Beat Tracking training data,

and the 40 ground-truth subject beat-tapping records collected for each excerpt (a total of 38,880 ground-truth beats, or an average of 48.6 per 30 s excerpt). Performance was evaluated using the metric defined for the competition, which was calculated as follows: For each human ground-truth tapping sequence, a true tempo period is calculated as the median inter-beat-interval. Then algorithmically-generated beat times are compared to the ground truth sequence and deemed to match if they fall within a time ‘collar’ around a ground-truth beat. The collar width was taken as 20% of the true tempo period. The score relative to that ground-truth sequence is the ratio of the number matching beats to the greater of total number of algorithm beats or total number of ground-truth beats (ignoring any beats in the first 5 s, when subjects may be ‘warming up’). The total score is the (unweighted) average of this score across all 40×20 ground truth sequences. Set as an equation, this total score is:

$$S_{tot} = \frac{1}{N_G} \sum_{i=1}^{N_G} \frac{\sum_{j=1}^{L_{A,i}} \min_k (|t_{A,i,j} - t_{G,i,k}|) < 0.2\tau_{G,i}}{\max(L_{G,i}, L_{A,i})} \quad (9)$$

where N_G is the number of ground-truth records used in scoring, $L_{G,i}$ is the number of beats in ground truth sequence i , $L_{A,i}$ is the number of beats found by the algorithm relating to that sequence, $\tau_{G,i}$ is the overall tempo of the ground-truth sequence (from the median inter-beat-interval), $t_{G,i,k}$ is the time of the k^{th} beat in the i^{th} ground-truth sequence, and $t_{A,i,j}$ is the time of the j^{th} beat in the algorithm’s corresponding beat-time sequence. This amounts to the same metric defined for the 2006 Audio Beat Tracking evaluation although it is expressed somewhat differently. Results would differ only in the case where multiple ground-truth beats fell into a collar width, in which case the competition’s cross-correlation definition would double-count algorithm-generated beats.

Our system has three free parameters: the two values determining the tempo window (τ_0 and σ_τ , described in the previous section), and the α of eqn. 1 which determines the balance between the local score (sum of onset strength values at beat times) and inter-beat-interval scores. Figure 4 shows the variation of the total score with α ; we note that the score does not vary much even

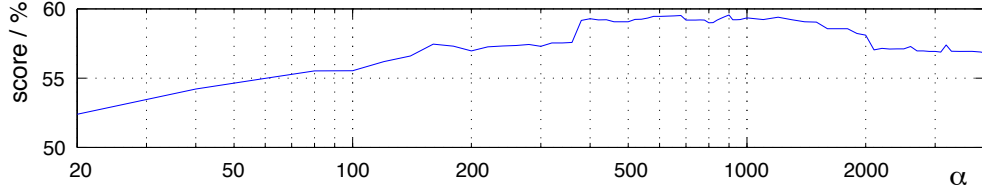


Figure 4: Variation of beat tracker score against the 20 MIREX-06 Beat Tracking training examples as a function of α , the objective function balance factor.

over almost three orders of magnitude of α . The best score over the entire test set is an accuracy of 58.8% for $\alpha = 680$. A larger α leads to a tighter adherence to the ideal tempo, since it increases the weight of the ‘transition’ cost associated with non-ideal inter-beat intervals in comparison to the onset waveform. A very large α essentially obliges the algorithm to find the best alignment between the onset envelope and a rigid, isochronous sequence at the estimated global tempo. The details of the test data set – consisting of relatively short excerpts chosen to be well described by a single tempo – may make the ideal value of α appear larger than would be the case for a more typical selection of music. Of course, this algorithm will not be suitable for material containing large variations or changes in tempo, something we return to in section 5.1.

The standard deviation of the difference between system-generated and ground-truth beat times, for the ground-truth beats that fell within 200 ms of a system truth beat, was 46.5 ms. This, however, appears to be mostly due to differences between the individual human transcribers which is of this order.

Because of the multiplicity of metrical levels reflected in the ground-truth data (as noted in section 3.2), it is not possible for any beat tracker to score close to 100% agreement with this data. In order to distinguish between gross disagreements in tempo and more local errors in beat placement, we repeated the scoring using only the 344 of 800 (43%) of ground-truth data sets in which the system-estimated tempo matched the ground-truth tempo to within 20%. On this data, the beat tracker agreed with 86.6% of ground-truth events.

One interesting aspect of the dynamic programming search is that it can use a target tempo from any source, not only the estimator described above. We used the two ‘ground truth’ tempos

provided for each training data example (i.e. the two largest modes of the tempos derived from the manual beat annotations) to generate two corresponding beat sequences, then scored against the 747 (93.4%) of ground-truth sequences that agree with one or other of these. (The remaining 53 annotations had median inter-beat-intervals that agreed with neither of the two most popular tempos for that excerpt). This achieved only 69.9% agreement. One reason that this scores worse than 86.6% achieved on the 344 sequences that agreed with the system tempo is that the larger set of 747 ground-truth sequences will include more at metrical levels slower than the tatum, or fastest rate present. When the tempo tracker is obliged to find a beat sequence at multiples of a detectible beat period, it runs the risk of choosing “offbeats” in-between the ground-truth beats, and getting none of the beats correct. When left to choose its own tempo, it is more likely to choose the faster tatum, allowing a score of around 50% correct for a duple time against those same ground-truth sequences.

5 Discussion and Conclusions

Having examined the performance of the algorithm, we now comment on a few issues raised by this approach.

5.1 Non-constant tempos

We have noted the main limitation of this algorithm is its dependence on a single, predefined ideal tempo. While a small α can allow the search to find beat sequences that locally deviate from this ideal, there is no notion of tracking a slowly-varying tempo (e.g. a performance that gets progressive faster). Abrupt changes in tempo are also not accommodated.

To examine the sensitivity of the system to the target tempo, we experimented with tracking specific single-tempo excerpts while systematically varying the target tempo around the “true” target tempo. We expect the ability to track the true tempo to depend on the strength of the true beat onsets in the audio, thus we compare two excerpts, one a choral piece with relatively weak onsets,

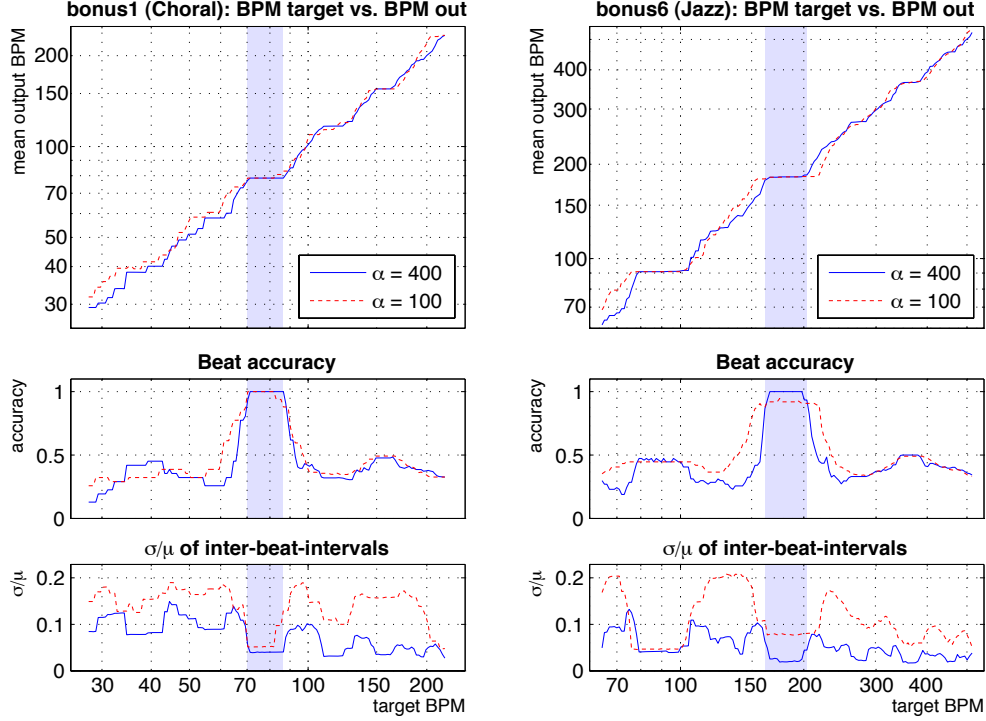


Figure 5: Effect of deviations from target BPM: For examples of choral music (‘bonus1’, left column), and jazz including percussion (‘bonus6’, right column), the target tempo input to the beat tracker is varied systematically from about $1/3\times$ to $3\times$ the main value returned by the tempo estimation (78.5 and 182 BPM, respectively). The shaded background region shows $\pm 10\%$ around the main value, showing that the beat tracker is able to hold on to the correct beats despite tempo deviations of this order. Top row shows the output tempo (mean of inter-beat-intervals); second row shows the beat accuracy when scored against the default beat tracker output; bottom row shows the standard deviation of the inter-beat-intervals as a proportion of the mean value, showing that beat tracks are more even when locked to the correct tempo (for these examples). In each case, the solid lines show the results with the default value of $\alpha = 400$, and the dotted lines show the results when $\alpha = 100$, meaning that less weight is placed on consistency with the target tempo.

and one a jazz excerpt where the beats were well defined by the percussion section (respectively “bonus1” and “bonus6” from the 2006 MIREX Beat Tracking development data [McKinney and Moelants, 2006a]). Figure 5 shows the results of tracking these two excerpts.

In order to accommodate slowly-varying tempos, the system could be modified to update τ_p dynamically during the progressive calculation of equations 3 and 4. For instance, τ_p could be set to a weighted average of the recent actual best inter-beat-intervals found in the max search. Note, however, that it is not until the final traceback that the beat times to be chosen can be distinguished

from the gaps inbetween those beats, so the local average may not behave quite as expected. It would likely be necessary to track the current τ_p for each individual point in the backtrace history, leading to a slightly more complex process for choosing the best predecessors. When τ_p is not constant, the optimality guarantees of dynamic programming no longer hold. However, in situations of slowly-changing tempo (i.e. the natural drift of human performance), it may well perform successfully.

The second problem mentioned above, of abrupt changes in tempo, would involve a more radical modification. However, in principle the dynamic programming best path could include searching across several different τ_p values, with an appropriate cost penalty to discourage frequent shifts between different tempos. A simple approach would be to consider a large number of different tempos in parallel i.e. to add a second dimension to the overall score function to find the best sequence ending at time t and with a particular tempo τ_{p_i} . This, however, would be considerably more computationally expensive, and might have problems with switching tempos too readily. Another approach would be to regularly re-estimate dominant tempos using local envelope autocorrelation, then include a few of the top local tempo candidates (peaks in the envelope autocorrelation) as alternatives for the dynamic programming search. In all this, the key idea is to define a cost function that can be recursively optimized, to allow the same kind of search as in the simple case we have presented. At some point, the consideration of multiple different tempos becomes equivalent to the kind of explicit multiple-hypothesis beat tracking system typified by Dixon [2001].

5.2 Finding the end

The dynamic programming search is able to find an optimal spacing of beats even in intervals where there is no acoustic evidence of any beats. This “filling in” emerges naturally from the backtrace and is desirable in the case where the music contains silence or long sustained notes. However, it applies equally well to any silent (or non-rhythmic) periods at the start or end of a

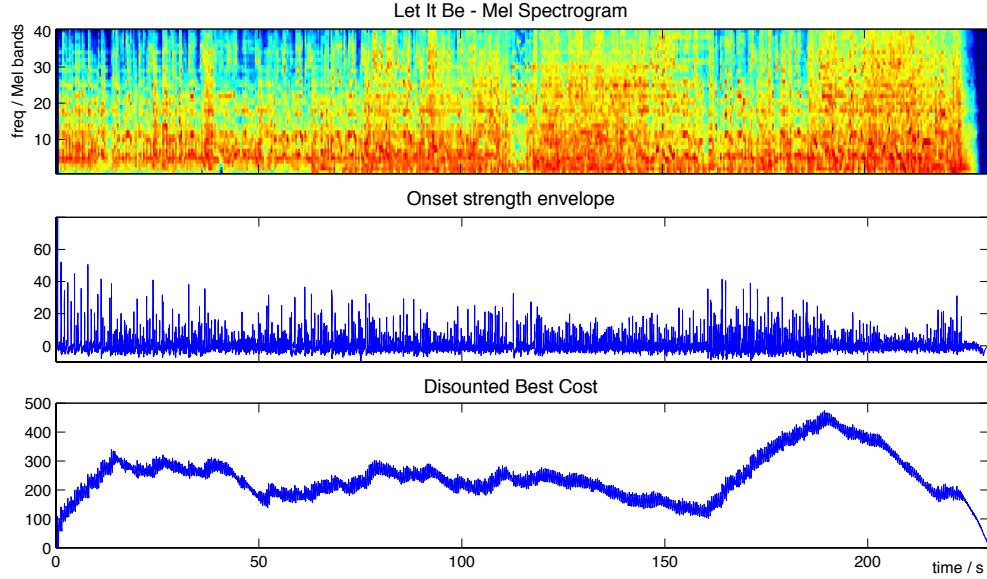


Figure 6: Comparison of Mel spectrogram, onset strength envelope, and discounted best cost function for the entire 3 min 50 s of “Let it Be” by the Beatles. The discounted best cost is simply the C^* of eqn. 3 after subtracting a straight line that connects origin and final value.

recording, when in fact it would be more appropriate to delay the first reported beat until some rhythmic energy has been observed, and to report no beats after the final prominent peak in the onset envelope.

Alternatively, some insight can be gained from inspecting the best score function, $C^*(t)$. In general this is steadily growing throughout the piece, but by plotting its difference from the straight line connecting the origin and the final value, we obtain a graph that shows how different parts of the piece contributed to a more or less rapid growth in the total score. Setting a simple threshold on this curve can be used to delete regions outside of the first and last viable beats. Figure 6 shows an example for the Beatles track “Let It Be”. Note the relatively strong score growth during the instrumental region around $t = 160 \dots 180$ s. In this region, the presence of a strong percussion track and the absence of vocals that would otherwise dominate much of the spectrum leads to particularly large values for the onset envelope, and hence large increases in the best score.

5.3 Conclusion

Despite its limitations, the simplicity and efficiency of the dynamic programming approach to beat tracking makes it an attractive default choice for general music applications. We have used this algorithm successfully as the basis of the beat-synchronous chroma features which underly our cover song detection system [Ellis and Poliner, 2007] which had the best performance by a wide margin in the MIREX-06 Cover Song Detection evaluation. In this and other related music retrieval applications we have run the beat tracker over very many pop music tracks, including the 8764 tracks of the `uspop2002` database [Ellis et al., 2003], and we have found it generally satisfactory – for this material, in the examples we have inspected, the tracked beats are very often at a reasonable tempo and in reasonable places. Compared to the other, state-of-the-art beat tracking systems evaluated in the MIREX-06 Audio Beat Tracking evaluation (discussed in McKinney et al. [2007]), we see this algorithm as far simpler, as evidenced by its simple objective function (eqn. 1) and compact code (fig. 1) while sacrificing little or nothing in accuracy. Even with a pure-Matlab implementation, its running time was faster than all but one of the C++ and Java-based competitors, and its performance had no statistically-significant difference from the best system. The Matlab code for the entire system is available for download at <http://labrosa.ee.columbia.edu/projects/coversongs/>, which also links to a Java implementation as part of the the MEAPsoft package [Weiss et al., 2006].

Acknowledgments

This work was supported by the Columbia Academic Quality Fund, and by the National Science Foundation (NSF) under Grant No. IIS-0238301. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF.

References

- R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- P. Desain and H. Honing. Computational models of beat induction: The rule-based approach. *Journal of New Music Research*, 28(1):29–42, 1999.
- S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.
- S. Dixon, W. Goebel, and E. Cambouropoulos. Perceptual smoothness of tempo in expressively performed music. *Journal of New Music Research*, 23(3):195–214, 2006. URL <http://www.ofai.at/~werner.goebel/papers/Dixon-Goebel-Cambouropoulos-MusPerc2006.pdf>.
- D. P. W. Ellis. PLP and Rasta and MFCC and inversion in Matlab. Web resource, retrieved July 11, 2007, from <http://labrosa.ee.columbia.edu/matlab/rastamat/>, 2005. URL <http://labrosa.ee.columbia.edu/matlab/rastamat/>.
- D. P. W. Ellis and G. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages IV–1429–1432, Hawai’i, 2007. URL <http://www.ee.columbia.edu/~dpwe/pubs/EllisP07-coversongs.pdf>.
- D. P. W. Ellis, A. Berenzweig, and B. Whitman. The “uspop2002” pop music data set. Web resource, retrieved July 11, 2007, from <http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html>, 2003. URL <http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html>.
- M. Goto and Y. Muraoka. A beat tracking system for acoustic signals of music. In *Proceedings of ACM Multimedia*, pages 365–372, San Francisco, CA, 1994. URL <http://staff.aist.go.jp/m.goto/PAPER/ACM94.300dpi.ps>.

- F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Speech and Audio Processing*, 14(5):1832–1844, 2006. doi: 10.1109/TSA.2005.858509. URL <http://www.iua.upf.es/mtg/ismir2004/contest/tempoContest/>.
- Tristan Jehan. *Creating Music by Listening*. PhD thesis, MIT Media Lab, Cambridge, MA, 2005. URL <http://web.media.mit.edu/~tristan/phd/>.
- A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3089–3092, Phoenix, AZ, 1999.
- J. Laroche. Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*, 51(4):226–233, Apr 2003.
- M. F. McKinney and D. Moelants. Audio Tempo Extraction from MIREX 2005. Web resource, retrieved July 11, 2007, from http://www.music-ir.org/mirex2005/index.php/Audio_Tempo_Extraction, 2005. URL http://www.music-ir.org/mirex2005/index.php/Audio_Tempo_Extraction.
- M. F. McKinney and D. Moelants. Audio Beat Tracking from MIREX 2006. Web resource, retrieved July 11, 2007, from http://www.music-ir.org/mirex2006/index.php/Audio_Beat_Tracking, 2006a. URL http://www.music-ir.org/mirex2006/index.php/Audio_Beat_Tracking.
- 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*, this issue, 2007.
- Martin F. McKinney and Dirk Moelants. Ambiguity in tempo perception: What draws listeners to different metrical levels? *Music Perception*, 24(2):155–166, 2006b. doi: 10.1525/mp.2006.24.2.155. URL <http://caliber.ucpress.net/doi/abs/10.1525/mp.2006.24.2.155>.

- D. Moelants and M. F. McKinney. Tempo perception and musical content: What makes a piece fast, slow, or temporally ambiguous? In S. D. Lipscomb, R. Ashley, R. O. Gjerdingen, and P. Webster, editors, *Proceedings of the 8th International Conference on Music Perception and Cognition*, pages 558–562, Evanston, IL, 2004.
- 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](https://doi.org/10.1155/2007/67215).
- R. Weiss, D. Repetto, M. Mandel, D. P. W. Ellis, V. Adan, and J. Snyder. Meapsoft: A program for rearranging music audio recordings. Web resource, retrieved July 11, 2007, from <http://www.meapsoft.org/>, 2006. URL <http://labrosa.ee.columbia.edu/meapsoft/>.