Toward Automated Holistic Beat Tracking, Music Analysis and Understanding.

Conference Paper · January 2005

Source: DBLP

1 author:

Roger B Dannenberg
Carnegie Mellon University
264 PUBLICATIONS **5,198** CITATIONS

TOWARD AUTOMATED HOLISTIC BEAT TRACKING, MUSIC ANALYSIS, AND UNDERSTANDING

Roger B. Dannenberg
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
rbd@cs.cmu.edu

ABSTRACT

Most music processing attempts to focus on one particular feature or structural element such as pitch, beat location, tempo, or genre. This hierarchical approach, in which music is separated into elements that are analyzed independently, is convenient for the scientific researcher, but is at odds with intuition about music perception. Music is interconnected at many levels, and the interplay of melody, harmony, and rhythm are important in perception. As a first step toward more holistic music analysis, music structure is used to constrain a beat tracking program. With structural information, the simple beat tracker, working with audio input, shows a large improvement. The implications of this work for other music analysis problems are discussed.

Keywords: Beat tracking, tempo, analysis, music structure

1 INTRODUCTION

Music is full of multi-faceted and inter-related information. Notes of a melody fall into a rhythmic grid, rhythm is hierarchical with beats, measures, and phrases, and harmony generally changes in coordination with both meter and melody. Although some music can be successfully decomposed into separate dimensions of rhythm, harmony, melody, texture, and other features, this kind of decomposition generally loses information, making each dimension harder to understand.

In fact, it seems that musicians deliberately complicate individual dimensions to make them more interesting, knowing that listeners will use other information to fill in the gaps. Syncopation can be exaggerated when the tempo is very steady, but we hear less syncopation when tempo is more variable. Confusing rhythms are often clarified by an unmistakeable chord change on the first beat of a measure. Repetition in music often occurs in some power-of-two number of measures, providing clear metrical landmarks even where beats and tempo might be ambiguous.

It is easy to notice these interrelationships in music, but difficult to take advantage of them for automatic music analysis. If everything depends on everything else, where does one start? If perception is guided by expectations, will we fail to perceive the "truth" when it is unexpected?

Music analysis produces all kinds of data and representations. How can the analysis of one dimension of music inform the analysis of another, given the inevitable errors that will occur? These are all difficult questions and certainly will form the topic of much future research.

This paper describes a small step in this general direction. I will show how information about music structure can be used to inform a beat tracker. In all previous beat trackers known to the author, an algorithm to identify beats is applied uniformly, typically from the beginning to the end of a work. Often times, beat trackers have a tendency to be distracted by syncopation and other musical complexities, and the tracker will drift to some faster or slower tempo, perhaps beating 4 against 3 or 3 against 4.

In contrast, when musical structure is taken into account, the beat tracker can be constrained such that when a beat is predicted in one section of music, a beat is also predicted at the corresponding place in all repetitions of that section of music. In practice, these are not absolute constraints but probabilistic tendencies that must be balanced against two other goals: to align beats with sonic events and to maintain a fairly steady tempo.

It might seem that if a beat tracker can handle one section of music, it can handle any repetition of that section. If this were the case, the additional constraint of music structure would not help with the beat-tracking

Originally published as: Roger B. Dannenberg, "Toward Automated Holistic Beat Tracking, Music Analysis, and Understanding," in *ISMIR 2005 6th International Conference on Music Information Retrieval Proceedings*, London: Queen Mary, University of London, 2005, pp. 366-373.

problem. Tests with real data, however, show a dramatic improvement when music structure is utilized. How can this be? A simple answer is that the input data is audio, and the detection of likely beat events is error prone. Music structure helps the beat tracker to consolidate information from different sections of music and ultimately do a better job. This will be explained in greater detail in the discussion section.

The next section describes related work. Then, in Section 3, I explain the basic beat tracker used for experiments. In Section 4, music structure analysis is described, and the additions to the beat tracker to use structure information are described in Section 5. In Section 6, I describe tests performed and the results. Section 0 presents a discussion, which is followed by a summary and conclusions.

2 RELATED WORK

The literature has many articles on beat tracking. Gouyon and Dixon have written an excellent overview with an extensive list of references. [1] For this work, I relied especially on the HFC (high frequency content) feature [2] for detecting likely beat events, as used by Davies and Plumbley [3] and also by Jensen and Andersen [4]. The general structure of the beat tracker is related to that of Desain and Honing [5] in that the tracker relies on gradient descent. Desain and Honing adjust the times of actual beat events to fit an expected model, whereas my system adjusts a tempo estimate to fit the actual times.

This work is not unique in attempting to incorporate music structure and additional features to analyze music. In particular, Goto and Muraoka used knowledge of drum beat patterns to improve beat tracking of popular (rock) music with drums [6], and Goto used some music classification techniques to handle music with drums differently from music without drums [7].

3 THE BASIC BEAT TRACKER

In order to show that music structure can help with the beat tracking problem, I first constructed a "baseline" beat tracker to measure performance without any music structure information. This beat tracker is based on state-of-the-art designs, but it has not been carefully tuned.

As is common, the beat tracker consists of two parts. The first part computes *likely beat events* from audio. Likely beat events are time points in the audio that suggest where beats might occur. These are represented as a discrete set of (*time, weight*) pairs. The second part attempts to identify more-or-less equally spaced beats that correspond to the likely beat events. Not all likely beat events will turn out to be beats, and some beats will not coincide with a likely beat event. The baseline beat tracker attempts to balance the two criteria of steady tempo and good matches to likely beat events.

3.1 Likely beat event detection.

One might expect that beats would be simple to detect in popular music, given the typically heavy-handed rock beat. Unfortunately, the loud snare hits are not so different spectrally from rhythm guitar chords or even vocal onsets and consonants. Furthermore, much popular music exhibits heavy dynamic compression, giving the music an almost constant energy level, so looking for peaks in the amplitude envelope is unreliable for detecting beats. High frequency content (HFC) [2] and spectral flux [8] are alternatives to RMS amplitude.

I use an HFC feature to detect likely beat events. Music audio is mixed from stereo to a single channel and downsampled to 16 kHz. FFTs of size 1024 are taken using a Hanning window applied to each (possibly overlapping) segment of 512 samples to yield a sequence X_n of complex spectra[1]. The per-frame HFC feature is the sum of the magnitudes weighted by the square of the bin number [4]:

$$hfc_n = \sum_{i=1}^{512} |X_n[i]| \cdot i^2 \quad (1)$$

where $|X_n[i]|$ is the magnitude of the i^{th} bin of the n^{th} frame. Note that some authors use the square of the magnitude and others weight linearly with bin number. To detect events, some thresholding is necessary. A running average is computed as:

$$avg_n = 0.9 \cdot avg_{n-1} + 0.1 \cdot hfc_{n-1} \quad (2)$$

The ratio hfc_n/avg_n exhibits peaks at note onsets, drum hits, and other likely beat locations. Unfortunately, even after normalizing by a running average, there will be long stretches of music with no prominent peaks. This problem is solved by a second level of thresholding which works as follows:

$$r_n = hfc_n / avg_n$$
$$thr_{n+1} = \begin{cases} \min(2, \max(thr_n, r_n \cdot 0.95)) \text{ if } r_n > thr_n \\ thr_n \cdot 0.99 \text{ otherwise} \end{cases} \quad (3)$$

Thus, the nominal threshold is 2, which captures every strong peak ($r_n > 2$) that occurs. When strong peaks are not present, the threshold adapts to detect smaller peaks. Whenever the threshold thr_n is exceeded by r_n, the time is recorded along with r_n, which serves as a weight in further computation. (In the next section, these pairs of (*n/framerate,* r_n) will be referred to as (t_i, w_i), a time/weight pair.) Since some peaks are broad and span multiple samples, no further times are recorded until r_n dips below the threshold.

[1] A step size of 64, yielding a frame rate of 250 Hz, was used to minimize any time quantization effects. However, there does not appear to be any significant difference when using even the lowest frame rate tried, 31.25 Hz.

The adaptive median threshold method [9] offers an alternative method for picking peaks from hfc_n. This method essentially replaces avg_n with a local median value of hfc_n, and it does not adapt when peaks are close to the median.

3.2 Beat tracking: initialization.

The beat tracking algorithm works from an initial beat location and tempo estimation, so the next step is to search for good initial values. This is not an on-line or real-time algorithm, so the entire song can be searched for good starting values. It is assumed that the likely beat events will be highly correlated with a "beat pattern" function shown at the top of Figure 1. This pattern represents the expected locations of quarter notes (full peaks) and eighth notes (half peaks), and is biased so that the integral is zero. The pattern is not meant to model a specific musical pattern such as a drum pattern. It merely models alternating strong and weak beats at a fixed tempo, and only this one pattern is used. The pattern is stretched in 2% increments from a beat period of 0.3s (200 bpm—beats per minute) to 1.0s (60 bpm)[1]. At each tempo, the function is shifted by 5 increments of 1/5 beat. Given a tempo and shift amount, the "goodness of fit", *gf*, to the data is given by:

$$gf(t_0, \rho, \phi) = \sum_i bp((t_i - t_0)/\rho - \phi) \cdot w_i \qquad (4)$$

where t_0 is used to center the beat pattern over some interior point in the song, ρ is the period, ϕ is the shift (in beats), *bp* is the beat pattern function (top of Figure 1), and (t_i, w_i) are the likely beat event times and weights calculated in Section 3.1.[2]

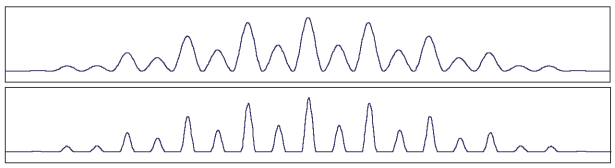

Figure 1. Beat patterns used to search for initial beat location and tempo.

Each configuration of tempo and shift is further refined using a gradient descent algorithm to find the best local fit to the data. Then the peaks of the beat pattern function are sharpened as shown in the lower half of Figure 1 to reduce the weight on outliers, and the gradient descent refinement is repeated.

[1] These are, of course, parameters that could be changed to accept a larger range of tempi. In practice, the tracker will tend to find multiples or submultiples when the "correct" tempo lies out of range.

[2] Note that we can consider the entire, continuous HFC signal simply by including every sample point r_n in the set of data points (t_i, w_i). At least on a small sample of test data, this does not improve performance.

All this estimates a tempo and offset for a general neighborhood in the song near t_0. We want to find a place where beats are strong and the data is as unambiguous as possible, so we estimate the tempo and beat offset at 5 second intervals (t_0=5, 10, 15, …) throughout the entire song. The values that maximize *gf* are used to initialize the beat tracker.

3.3 Beat tracking.

Beat tracking is accomplished by extending the idea of the beat pattern function and gradient decent. Imagine broadening the window on the beat pattern function (Figure 1) to expose more peaks and using gradient decent to align the function with increasingly many likely beat events. This is the general idea, but it must be modified to allow for slight tempo variation.

Tempo (and period) is assumed to be constant within each 4-beat measure, so a discrete array of period values serves to record the time-varying tempo. Given a vector of beat periods, *pv*, and an origin, t_0, it is not difficult to define a function from time (in seconds) to beat (a real number). Call this the "time warp" function $\tau_{pv,\, t_0}(t)$. The goodness of fit function can then be modified to incorporate this "time warping:"

$$gfw(pv, t_0) = \sum_i bp(\tau_{pv,t_0}(t_i)) \cdot w_i \qquad (5)$$

This function maps each likely beat event from time to beat, then evaluates the beat pattern at that beat. Recall that the beat pattern has peaks at integer beat and sub-beat locations.

If the only criterion was to match beats, we might see wild tempo swings to fit the data, so we add a "tempo smoothness" that penalizes tempo changes:

$$ts(pv) = \sum_i \ln(gauss(0, 0.1, 2 \cdot \frac{(pv_i - pv_{i-1})}{(pv_i + pv_{i-1})})) \qquad (6)$$

where $gauss(\mu, \sigma, x)$ is the Gaussian with mean μ and standard deviation σ, evaluated at x.

The beat tracking algorithm performs a gradient descent to fit the predicted beats to the likely beat events. The goal is to optimize the sum of *gfw* and *ts*, which represent a good fit to the beat pattern and a smooth tempo curve. Notice, however, that the beat pattern function shown in Figure 1 rapidly goes to zero, so likely beat events outside of a small window will be ignored. Although not described in detail, the beat pattern *bp* consists of a periodic beat pattern multiplied by a window function. The window function can be widened to consider more beats.

The beat tracking algorithm alternately widens the window function for the beat pattern to consider a few more beats at the left and right edges of the window. Then, gradient descent is used to make slight adjustments to the period vector (tempo curve), possibly taking into

account more likely beat events that now fall within the wider window. This alternation between widening the window and gradient descent continues until the window covers the entire song.

3.4 Beat tracking performance.

As might be expected, this algorithm performs well when beats are clear and there is a good correspondence between likely beat events and the "true" beat. In practice, however, many popular songs are full of high frequency content from drums, guitars, and vocals, and so there are many detected events that do not correspond to the beat pattern. This causes beat tracking problems. In particular, it is fairly common for the tempo to converge to some integer ratio times the correct tempo, e.g. 4/3 or 5/4. This allows the beat pattern to pick up some off-beat accents as well as a number of actual downbeat and upbeat events.

One might hope that the more-or-less complete search of tempi and offsets used to initialize the beat tracker might be used to "force a reset" when the tempo drifts off course. Unfortunately, while the best match overall usually provides a good set of initial values, the best match in the neighbourhood of any given time point is not so reliable. Often, it is better *not* to reset the beat tracker when it disagrees with local beat information.

Human listeners can use harmonic changes and other structural information to reject these otherwise plausible tempi, and we would like to use structural information to improve automatic beat tracking, perhaps in the same way. The next two sections look at ways of obtaining structure and using structure to guide beat tracking.

4 STRUCTURAL ANALYSIS

Previous work on structural analysis identified several approaches to music analysis. [10] This work aimed to find "explanations" of songs, primarily in the form of repetition, e.g. a standard song form is AABA. For this study, I use the chroma vector representation [11], which is generally effective for the identification of harmony and melody. [12] The chroma vector is a projection of the discrete Fourier transform magnitude onto a 12-element vector representing energy at the 12 chromatic pitch classes. [13]

A self-similarity matrix is constructed from chroma vectors and a distance function: every chroma frame is compared to every other chroma frame. Within this matrix, if music at time a is repeated at time b, there will be roughly diagonal paths of values starting at locations (a, b) and (b, a), representing sequences of highly similar chroma vectors and extending for the duration of the repetition. (See Figure 2.)

In many cases, it is possible to determine a good "explanation" that covers the entire song, e.g. ABABCA. One can imagine inferring the length of sections, e.g. 8 or 16 measures, and this could be extremely helpful for beat tracking. However, not all songs have such a clear structure, and we cannot make such strong assumptions. For this study, only the paths in the similarity matrix are used, but even this small amount of structural information can be used to make large improvements in beat-tracking performance.

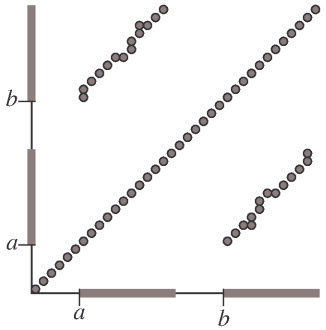

Figure 2. Paths of high similarity in the similarity matrix. Sections starting at a and b in the music are similar.

5 BEAT TRACKING WITH STRUCTURE

When two sections of music are similar, we expect them to have a similar beat structure. This information can be combined with the two previous heuristics: that beats should coincide with likely beat events and tempo changes should be smooth.

The structure analysis finds similar sections of music and an alignment, as shown in Figure 2. The alignment path could be viewed as a direct mapping from one segment to the other, but an even better mapping can be obtained by interpolating over multiple frames. Therefore, to map from time t in one segment to another, a least-squares linear regression to the nearest 5 points in the alignment path is first computed. Then, the time is mapped according to this line.

But how do we use this mapping? Note that if beat structures correspond, then mapping from one segment to another and advancing several beats should give the same result as advancing several beats and then mapping to the other segment.[1] The formalization of this "structural consistency" is now described.

5.1 Computing Structural Consistency.

The "structural consistency" function is illustrated in Figure 3 and will be stated as Equation 9. The roughly diagonal line in the figure represents an alignment path between two sections of music starting at a and b. (Note

[1] We could state further that every beat in one segment should map directly to a beat in a corresponding segment, but since alignment may suffer from quantization and other errors, this constraint is not enforced. Future work should test whether this more direct constraint is effective.

that the origins of the time axes are not zero, but close to a and b, respectively, to make the figure more compact.) The time t_1 is the time of the first measure beginning after a. This is mapped via the alignment path to a corresponding moment in the music u_1. Next, we advance 4 beats beyond t_1. To accomplish this, we use the time warp function: $\tau_{pv,t_0}(t_1)$, add 4 beats, and then map back to time using the inverse function:

$$t_2 = \tau^{-1}_{pv,t_0}(\tau_{pv,t_0}(t_1) + 4) \quad (7)$$

Then, t_2 is mapped via the alignment path to u_2 as shown by dashed lines. The resulting time should be consistent with u_1 plus 4 beats, which is computed in the same way as t_2:

$$u_2 = \tau^{-1}_{pv,t_0}(\tau_{pv,t_0}(u_1) + 4) \quad (8)$$

In practice, there will be some discrepancy between u_2 and the mapping of t_2. This is illustrated and labeled "error" in Figure 3.

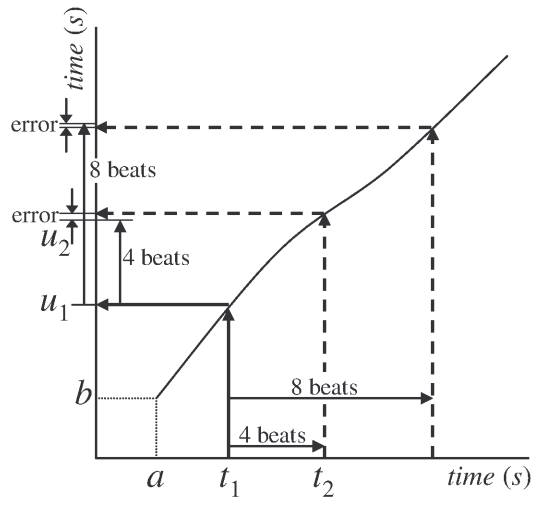

Figure 3. If beat locations are consistent with structure, then advancing 4 or 8 beats in one section of music and mapping to the corresponding point in another section will be equivalent to mapping to the corresponding point (u_1) first, and then advancing 4 or 8 beats.

Having computed an error value for a 4-beat offset, a similar procedure is used to compute the error at 8 beats and every other measure that falls within the alignment path. There may be multiple alignment paths, so all errors for these alignment paths are also computed. The overall "structural consistency" function is then:

$$sc_w = \sum_{p \in P_w} \sum_{e \in E_{p,w}} gauss(0, 0.2, e) \quad (9)$$

where w indicates a range of the song (a "window") over which the function is computed, P_w is the set of alignment paths that overlap the window w, and $E_{p,w}$ is the set of error values computed for alignment path p within window w. Although not mentioned explicitly, sc_w also depends upon the period vector pv as implied by Equations 7 and 8.

5.2 Beat Tracking With Structure Algorithm.

Now we have three functions to guide our beat tracker: *gfw* is the "goodness of fit with time warping" function that evaluates how well the likely beat events line up with predicted beats, given a period vector that maps real time to beats. *ts* is the "tempo smoothness" function that evaluates how well the period vector meets our expectation for steady tempo. *sc* is the structural consistency function that measures the consistency of beats and tempo across similar sections of music. These three functions are simply summed to form an overall objective function. Recall that *sc* is parameterized by a window (a starting and ending time); this is set to match the window of the beat pattern function used in *gfw*.

It remains to describe an algorithm that performs beat tracking utilizing these three functions. The algorithm is similar to the beat tracking algorithm of Section 3.3 (among other things, using a similar algorithm will help us to isolate and assess the impact of structural consistency). We begin with a small window around the same t_0 found in Section 3.2 and, as before, alternately widen the window and perform a gradient descent optimization of the period vector *pv*.

What is different now is that the existence of music structure will force us to "jump" to other locations in the song to evaluate the structural consistency function. These other sections will need a well-defined period vector, and because of the coupling between similar sections of music, all similar sections will need to be considered when attempting to use gradient descent to optimize the objective function.

The new algorithm uses the concept of "islands," which are simply regions of the song that are relevant to the computation. Each island has an associated period vector and time offset. The "time warp" function, τ, is defined on a per-island basis.

Initially, there is one island centered on t_0, and the period vector is only defined within the "shores" of the island. When this initial island grows to overlap an alignment path (or if the island already overlaps an alignment path when it is initialized), the structural consistency function will need to examine some other place in the song, quite possibly "off the island." When this happens (see Figure 4), a new island is created. It is initialized with a small window using an offset and period vector that makes it consistent with the initial island.

Computation proceeds in a round-robin fashion, looking at each island in turn. The island's window is widened and gradient descent is used to optimize the island's period vector. Then the next island is considered.

At some point, islands will begin to overlap. Overlapping islands are merged by consolidating their period vectors. Ideally, islands will meet on an exact measure boundary, but this does not always happen in practice. To avoid large discontinuities, one of the vectors is shifted by some integer number of beats so that the vectors are maximally consistent at their meeting point. When the vectors are merged, beat times are preserved and it is assumed that gradient descent will fix any remaining inconsistencies.

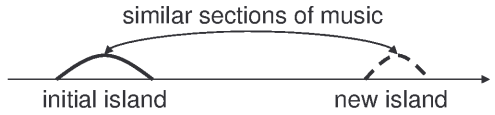

Figure 4. New islands are created when parts of an existing island are similar to music elsewhere in the song. This allows for the computation and evaluation of structural consistency as part of the beat-tracking process.

Since islands never grow smaller, the algorithm eventually terminates with one island covering the entire song. At this point, all beat times are determined from the single remaining period vector and time origin t_0.

5.3 Implementation.

The HFC feature extraction is implemented in Nyquist [14], and the structure analysis is implemented in Matlab, while the beat tracking algorithms are implemented in C++. Nyquist is then used to synthesize "tap" sounds and combine these with the original songs for evaluation. The total CPU time to process a typical popular song is on the order of a few minutes. Using a compiled language, C++, for the gradient-descent beat tracking algorithms is important for speed, but other language choices were just for convenience.

The beat tracking program logs the current period vector and other information so that when the computation completes, the user can display a plot of the warped and windowed beat pattern(s) against the expected beat events. The user can then visualize the iterative search and optimization by stepping forward or backward in time, and by zooming in or out of various regions of the song. This feature proved invaluable for debugging and verifying the behaviour of the program.

6 EVALUATION

Since beats are a perceptual construct, there is no absolutely objective way to determine where beats occur. Some listeners may perceive the tempo to be twice or half the rate of other listeners. Furthermore, if the tempo is slightly fast or slow, it will appear to be correct almost half the time, as estimated beats go in and out of phase with "true" beats.

For this study, the goal is to compare beat tracking performance with and without the use of structural consistency. To evaluate beat tracking, the beat-tracker output is used to synthesize audio "taps," which are mixed with the original song. The audio mix is then auditioned and subjective judgements are made as to when the beat tracker is following the beat and when it is not. Tapping on the "upbeat" and/or tapping at twice or half the preferred rate are considered to be acceptable; however, tapping at a slightly incorrect tempo, causing beats to drift in and out of phase (which is a common mode of failure) is not acceptable even though many predicted beats will be very close to actual (perceived) beats. Beat tracking is rated according to the percentage of the song that was correctly tracked, and percentages from a number of songs are averaged to obtain an overall performance score. Although human judgement is involved in this evaluation, the determination of whether the beat tracker is actually tracking or not seems to be quite unambiguous, so the results are believed to be highly repeatable.

Sixteen (16) popular songs were tested. Using the basic beat tracking algorithm without structural consistency, results ranged from perfect tracking through the entire song to total failure. The average percentage of the song correctly tracked was 30%. With structural consistency, results also ranged from perfect to total failure, but the number of almost perfectly tracked songs (> 95% correct) doubled from 2 to 4, the number of songs with at least 85% correctly tracked increased from 2 to 6, and the overall average increased from 30% to 59% ($p < 0.0034$). (See Table 1.)

Table 1. Performance of basic beat tracker and beat tracker using music structure information.

	Basic Tracker	**Tracker Using Music Structure**
Percentage tracked	30	59
Number tracked at least 95% correct	2	4
Number tracked at least 85% correct	2	6

7 DISCUSSION

The results are quite convincing that structural consistency gives the beat tracker a substantial improvement. One might expect that similar music would cause the beat tracker to behave consistently anyway, so it is surprising that the structural consistency information has such a large impact on performance. However, one of the main problems with beat tracking in audio is to locate the "likely beat events" that guide the beat tracker. Real data is full of sonic events that are not on actual beats and tend to distract the beat tracker. By imposing structural consistency rules, perhaps "random" events are averaged

out, essentially bringing the law of large numbers into play: structural consistency considers more information and ultimately allows for better decisions.

Another advantage of music structure is that by propagating good tempo information to new "islands," the beat tracker can more successfully approach regions of uncertainty between the islands. Looked at another way, regions that are difficult to track do not have as many opportunities to "throw off" the beat tracker to the extent that it cannot recover the correct tempo later in the song. To further isolate this factor, one could use the islands to determine the order in which beat tracking is performed, but ignore the structural consistency function *sc* when optimizing the period vectors.

7.1 Absolute Quality of Beat Tracker

One possible criticism of this work is that if the basic beat tracker had better performance, structural consistency might not be so useful. Are we seeing great tracking improvement because the basic tracker is entirely inadequate? The basic beat tracker is based on recent published work that claims to be successful. Readers should recognize that correlating the beat pattern function with beat events is closely related to autocorrelation and wavelet techniques used by other beat induction programs [1] to detect periodicity. My method of widening the beat pattern window and then optimizing the beat period vector is closely related to other methods of entrainment for beat tracking. While we do not have shared standards for measuring beat-tracking performance, it seems likely that any technique that can *substantially* improve the basic beat tracker will offer some improvement to most others.

For comparison, Scheirer's beat tracker [15] was used to identify beats in the same test set of songs. The results are difficult to interpret because Scheirer's program does not actually fit a single smooth tempo map to the data. Instead, there are multiple competing internal tempo hypotheses that can switch on or off at any time. As a result, the output beats are often correct even when there is no underlying consistent tempo. In many cases, however, it seems that a little post-processing could easily recover a steady tempo. Giving the output this subjective benefit of the doubt, Scheirer's tracker correctly tracked about 60% of the songs. This is significantly better than my baseline tracker, and essentially the same as my tracker using music structure.

This may indicate that the baseline tracker could be improved through tuning. It may also indicate that searching for periodicity independently in different frequency bands (as in the Scheirer tracker) is advantageous. A third possibility is that using continuous features rather than discrete peaks may be important; however, modifying the baseline tracker to use continuous *hfc* values appears not to make any significant difference. Much more investigation is needed to understand the many factors that affect beat tracker performance in general. This investigation was designed to explore only one factor, the use of music structure, while keeping other factors the same.

7.2 The Non-Causal Nature

This algorithm is non-causal. It searches for a strong beat pattern as a starting point and expands from there. When music structure is considered, the algorithm jumps to similar musical passages before considering the rest of the music. Certainly, human listeners do not need to perform multiple passes over the music or jump from one location to another. However, musical memory and familiarization are part of the listening process, and composers use repetition for good reasons. Although inspired by intuitions about music listening, this work is not intended to model any more than a few interesting aspects of music cognition.

7.3 Other Comments

Because the goal of this work was to explore the use of structure in beat tracking, I did not try the system on jazz or classical music, where the repetitions required for structure detection are less common. Most of the test set is music with drums. Further work will be needed to expand these ideas to work with different types of music and to evaluate the results.

The main goal of this work is to show that music structure and other high-level analysis of music can contribute to better detection of low-level features. Ultimately, there should be a bi-directional exchange of information, where low-level features help with high-level recognition and vice-versa. For example, beat and tempo information can help to segment music, and music segmentation [16-20] can in turn help to identify metrical structure. Metrical structure interacts closely with beat detection. One of the fascinating aspects of music analysis is the many levels of interconnected features and structures. Future automatic music analysis systems will need to consider these interconnections to improve performance. This work offers a first step in that direction.

8 SUMMARY AND CONCLUSIONS

Two beat-tracking algorithms were presented. Both use high frequency content to identify likely beat events in audio data. The first is a basic algorithm that begins by searching for a good fit between the likely beat event data and a windowed periodic "beat pattern" function. After establishing an initial tempo and phase, the beat pattern window is gradually widened as gradient descent is used

to find a smoothly varying tempo function that maps likely beat events to predicted beat locations.

A second algorithm is based on the first, but adds the additional constraint that similar segments of music should have corresponding beats and tempo variation. The beat tracking algorithm is modified to incorporate this heuristic, and testing shows a significant performance improvement from an average of 30% to an average of 59% correctly tracked.

This work is based on the idea that human listeners use many sources of information to track beats or tap their feet to music. Of course, low-level periodic audio features are of key importance, but also high-level structure, repetition, harmonic changes, texture, and other musical elements provide important "musical landmarks" that guide the listener. This work is a first step toward a more holistic approach to music analysis and in particular, beat tracking. I have shown that musical structure can offer significant performance improvements to a fairly conventional beat tracking algorithm. It is hoped that this work will inspire others to pursue the integration of high-level information with low-level signal processing and analysis to build more complete and effective systems for automatic music understanding.

9 ACKNOWLEDGEMENTS

The author would like to thank the Carnegie Mellon School of Computer Science where this work was performed.

REFERENCES

[1] Gouyon, F. and Dixon, S. "A Review of Automatic Rhythm Description Systems", Computer Music Journal, 29, 1, (Spring 2005), 34-54.

[2] Masri, P. and Bateman, A. "Improved Modeling of Attack Transients in Music Analysis-Resynthesis", Proceedings of the 1996 International Computer Music Conference, Hong Kong, 1996, 100-103.

[3] Davies, M.E.P. and Plumbley, M.D. "Causal Tempo Tracking of Audio", ISMIR 2004 Fifth International Conference on Music Information Retrieval Proceedings, Barcelona, 2004, 164-169.

[4] Jensen, K. and Andersen, T.H. "Beat Estimation on the Beat", 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Palz, NY, 2003, 87-90.

[5] Desain, P. and Honing, H. "The Quantization of Musical Time: A Connectionist Approach", Computer Music Journal, 13, 3, (Fall 1989), 55-66.

[6] Goto, M. and Muraoka, Y. "Music Understanding at the Beat Level: Real-Time Beat Tracking of Audio Signals", in Rosenthal, D. and Okuno, H. eds. Computational Auditory Scene Analysis, Lawrence Erlbaum Associates, New Jersey, 1998.

[7] Goto, M. "An Audio-Based Real-Time Beat Tracking System for Music with or without Drums", Journal of New Music Research, 30, 2, (2001), 159-171.

[8] Alonso, M., David, B. and Richard, G. "Tempo and Beat Estimation of Musical Signals", ISMIR 2004 Fifth International Conference on Music Information Retrieval Proceedings, Barcelona, 2004, 158-163.

[9] Bello, J.P., Duxbury, C., Davies, M. and Sandler, M. "On the Use of Phase and Energy for Musical Onset Detection in the Complex Domain", IEEE Signal Processing Letters, 11, 6, (June 2004), 553-556.

[10] Dannenberg, R.B. and Hu, N. "Pattern Discovery Techniques for Music Audio", Journal of New Music Research, 32, 2, (June 2003), 153-164.

[11] Bartsch, M. and Wakefield, G.H. "Audio Thumbnailing of Popular Music Using Chroma-based Representations", IEEE Transactions on Multimedia, 7, 1, (Feb. 2005), 96-104.

[12] Hu, N., Dannenberg, R.B. and Tzanetakis, G. "Polyphonic Audio Matching and Alignment for Music Retrieval", 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Palz, NY, 2003, 185-188.

[13] Wakefield, G.H. "Mathematical Representation of Joint Time-Chroma Distributions", International Symposium on Optical Science, Engineering, and Instrumentation, SPIE'99, Denver, 1999.

[14] Dannenberg, R.B. "Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis", Computer Music Journal, 21, 3, (Fall 1997), 50-60.

[15] Scheirer, E. "Tempo and Beat Analysis of Acoustic Music Signals", Journal of the Acoustical Society of America, 104, (January 1998), 588-601.

[16] Tzanetakis, G. and Cook, P. "Multifeature Audio Segmentation for Browsing and Annotation", Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, 1999.

[17] Logan, B. and Chu, S. "Music Summarization Using Key Phrases", Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings (ICASSP 2000), Istanbul, Turkey, 2000, II-749-752.

[18] Foote, J. "Automatic Audio Segmentation Using a Measure of Audio Novelty", Proceedings of the International Conference on Multimedia and Expo (ICME 2000), 2000, 452-455.

[19] Aucouturier, J.-J. and Sandler, M. "Segmentation of Musical Signals Using Hidden Markov Models", Proceedings of the 110th Convention of the Audio Engineering Society, Amsterdam, The Netherlands, 2001.

[20] Peeters, G., Burthe, A.L. and Rodet, X. "Toward Automatic Audio Summary Generation from Signal Analysis", ISMIR 2002 Conference Proceedings, Paris, France, 2002, 94-100.