Tracking Musical Beats in Real Time

Paul E. Allen and **Roger B. Dannenberg**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Abstract

Identifying the temporal location of downbeats is a fundamental musical skill. After briefly discussing the perceptual information available in beat tracking, we survey previous attempts to automate this process in both real time and non-real time. Our attempt to add flexibility to Mont-Reynaud's model [4] by parameterizing the confidence and history mechanisms failed to yield a satisfactory beat tracker. Observing that this and previous models are constrained to hold a single current notion of beat timing and placement, we find that they will fail to predict beats and not recover beyond the point at which a mistake is first made. We propose a new model that uses beam search [1] to simultaneously consider multiple interpretations of the performance. At any time, predictions of beat timing and placement can be made according to the most credible of many interpretations under consideration. Credibility is determined by a heuristic evaluation function.

Introduction

Identifying the temporal location of downbeats is a fundamental musical skill that comes naturally to most people. Even people without musical training can tap their foot with the beat when they hear a musical performance and can generally perform this task with ease and precision even in the face of unusual rhythms, musical expressiveness, and imprecise performances. (An alternate name for this task, *foot tapping*, arises from this characteristic human activity.) Musical performances contain a great deal of information that presumably facilitates the ease with which humans do beat tracking. Examples of such information are temporal information like note onsets and offsets, pitches and their interval relationships, and the relationships of multiple voices in polyphony.

A fully general, automatic beat tracker would be of great value in many applications such as human-computer improvisation, music transcription, music editing and synchronization, and (ethno-) musicological studies. It would open up new possibilities for computers in music by allowing the computer to synchronize with tempos external to it without the use of explicit synchronization information. Musicians generally would agree that it is much harder to play in an ensemble and follow a collective tempo than it is to set their own tempo and require other musicians to follow. Our goal is to transform the computer from a tempo setter to a tempo follower. A beat tracker would solve this problem and allow an outside agent (e.g. a human performer) instead of the computer to control the tempo.

Implicitly, we have been discussing beat tracking in terms of realtime reactions. Humans tap their foot in realtime and we would want a computer to improvise with a human performer in realtime. Though we could imagine that a transcription task, for example, *not* be performed in realtime since there are generally no time constraints implicit in that task. However, we feel that performing beat tracking in realtime is important since a non-realtime beat tracker would find far fewer uses in the computer music world. Thus, except for review of some previous beat tracking work, we consider only realtime beat tracking for the remainder of this paper.

Perceptual Information Available in Beat Tracking

Beat tracking is differentiated from the score following of Dannenberg [3] and Vercoe [12] primarily by the absence of a score as part of the input to the process. The minimum input needed by a beat tracker is temporal information consisting of the note onset times of a single instrument voice from which inter-onset intervals[1] may be derived. Additional input may include pitch information, and an initial tempo so that bootstrapping is avoided. A polyphonic beat tracker would use similar information derived from multiple voices. The output of a beat tracker is a prediction of where future beats will be in a musical performance. To facilitate discussion, we extend the musical concept of *beat* to include two aspects, *period* and *phase*. In a musical performance, we will say that the period or

[1]Defined as the time from one note onset to the next note onset.

duration of a beat is the time interval between two successive beats (the reciprocal of the tempo) and that the phase of a beat determines where a beat occurs with respect to performance time. A beat tracker must contain state information that, at minimum, includes the current perceived beat phase and period.

When humans perform beat tracking, they are almost certainly using knowledge and memory in addition to the the pitch and timing information coming to them via their ears. There is an amazing variety of musical knowledge and memory which a human might bring to bear on this task. Some possibilities include:

- memory of specific performances (e.g. a favorite recording of the 1812 Overture)
- memory of specific musical pieces (e.g. Beethoven's 9th Symphony)
- knowledge of specific musical forms (e.g. zydeco)
- knowledge of broad musical forms (e.g. Western classical)
- knowledge of performers' styles (e.g. Duke Ellington)
- knowledge of composers' styles (e.g. Brahms)
- knowledge of meter and musical structure (e.g. 4/4 meter and musical phrases)

A great deal of leverage could be achieved by mechanical beat trackers if such information could be appropriately distilled and encoded, but as it is, it is difficult get beat trackers to intelligently use more than a small fraction of the information available to them as input, much less use extensive stored knowledge. Stored musical knowledge and memory is difficult to use on a large scale although it can be done on a smaller scale such as the stored scores utilized in score following systems.

Previous Research

Much of the previous work on beat tracking has come about as a byproduct of research directed at other music understanding problems such as transcription and the perception of rhythm, meter, and melody. Longuet-Higgins [8, 9] describes what is probably the first attempt to follow the beat of an actual performance in the face of changing tempo. His simple, but fairly powerful, model is based on looking for note onsets occurring close to expected beats. If T_i is the time of an expected beat and ε is the tolerance that defines ''close to,'' then the model would assume that any note onset falling within the time interval $T_i - \varepsilon$ to $T_i + \varepsilon$ is a beat. If a beat was found at T_k and the next beat was expected at T_{k+1}, then a note onset occurring in the interval from $T_k + \varepsilon$ to $T_{k+1} - \varepsilon$ may cause a re-evaluation of the estimate of T_{k+1}. Longuet-Higgins found that a value of one to two tenths of a second for the tolerance, ε, worked well. This work is part of a non-realtime computational psychological model of the perception of Western classical melody proposed by Longuet-Higgins.

Related work by Longuet-Higgins and Lee [10] on perceiving rhythm addressed the problem of using actual performance input by basing their model of metrical inferences on relative note durations. Previous work in the same area [7, 11] avoided the issue altogether by assuming input consisted of ideal note durations.

Chafe, Mont-Reynaud, and Rush [2] use methods similar to Longuet-Higgins' 1976 and 1978 work in a non-realtime transcription oriented task. Using multiple passes over the input, their method finds anchor points to which to attach beats. Metric analysis then finishes the placement of beats while determining meter. This is an example where higher level musical analysis has been applied to beat tracking.

Desain and Honing [6] report on an iterative method that uses relaxation techniques for quantizing actual note intervals into musical note values. Their approach is not readily amenable to realtime applications, but they have a good discussion of the timing and tempo variation in performances that needs to be overcome when attacking the beat tracking problem.

As part of a realtime jazz improvisation following task, Dannenberg and Mont-Reynaud [4] describe an implementation of a method for realtime beat tracking that incorporates the concept of confidence and a history mechanism that uses a weighted average of previous perceived tempos to compute the current perceived tempo. Their model can be described by a small number of state variables and a method to update the state when new note onsets are available. The state can be represented by the tuple (*T*, *B*, *S*, *W*) where *T* is the time of the most recently seen note onset, *B* is the predicted metrical position of the most recently seen note onset, while *W* and *S* are, respectively, the weighted sum of the perceived metrical note value of the input notes and the weighted sum of the inter-onset durations. *W* and *S* are used to implement the history mechanism by facilitating the computation of a weighted beat duration. If T_{new} is the time of a new note onset, then the new state is computed as follows[2]:

$$\begin{aligned}
\delta &= W/S \\
\Delta T &= T_{new} - T \\
\Delta B &= \Delta T/\delta \\
Confidence &= 1 - 2\cdot\left|\Delta B - \lfloor \Delta B + 0.5 \rfloor\right| \\
Decay &= 0.9^{\lfloor \Delta B + 0.5 \rfloor} \\
T' &= T + \Delta T \\
B' &= B + \Delta B \\
W' &= W\cdot Decay + \Delta B\cdot Confidence \\
S' &= S\cdot Decay + \Delta T\cdot Confidence
\end{aligned}$$

There are a few points to notice about this model. First, the *Confidence* value weights the amount the new note onset is allowed to affect the new tempo. It implements the heuristic that note onsets close to expected beats should be allowed to affect the tempo more than onsets far away from expected beats. The confidence function is shown in Figure 1. Secondly, note that the *Decay* value is used to implement tempo history by decaying past tempo values by 10% every beat.

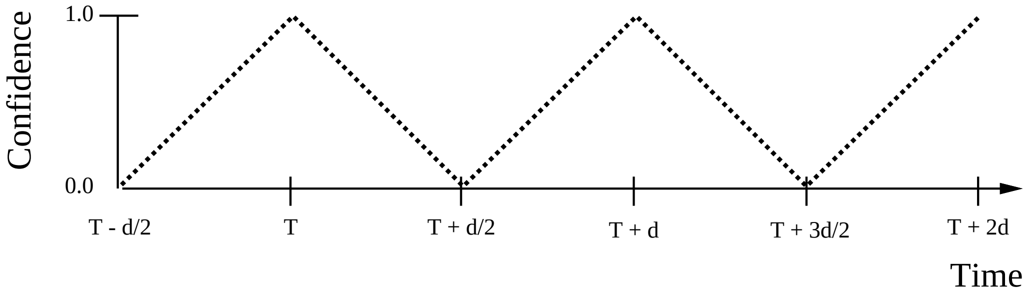

Figure 1: Confidence function with beat predicted at T with duration *d*.

Dannenberg reports that the results of the beat tracking portion of the jazz improvisation following were somewhat disappointing because the system was either unreliable or unresponsive to tempo change, depending upon the rate of decay.

A More Flexible Beat Tracking Model

The beat tracking models of Longuet-Higgins, and Chafe, Mont-Reynaud, and Rush combine a bottom-up and top-down approach to beat tracking where low level beat tracking analysis is aided by higher level rhythmic analysis. The approaches of Desain and Honing, and Dannenberg and Mont-Reynaud are entirely bottom-up approaches dealing with no significant higher level analysis. We decided to pursue a purely bottom-up approach to beat tracking for two reasons. First, we wanted to discover how well the beat tracking task could be done without higher level musical reasoning such as rhythmic analysis. Secondly, a bottom-up approach seemed best suited for realtime implementation, an attribute we consider important in beat tracking. So, of the models we have reviewed here, we decided to use Dannenberg and Mont-Reynaud's model as a starting point since that fit our goals best.

---

[2] Note that this presentation contains a few corrections to the originally published description of this model.

In an attempt to overcome the sluggishness and insensitivity of this model reported by Dannenberg, we parameterized the history and confidence mechanisms with the expectation of being able to find a set of values for those parameters that worked well for a given style of music or performance. The original history mechanism used an exponential decay for the weighting of previous tempos where the amount of decay of each of the previous tempos was 10% per beat. This decay rate has a direct affect on the responsiveness and stability of the beat tracker. At one extreme, 100% decay, the tracker will ignore all history and respond only with respect to the latest note data, allowing rapid tempo fluctuations, but making the tracker very unstable. At the other extreme, 0% decay, the tracker will consider each bit of history equally with the present state so that response is slow, if present at all. We made the decay rate a parameter to the method so we could experiment with various decay values for different musical and performance styles and try to find a better compromise between responsiveness and stability.

The second parameterization we applied to the method was to the confidence calculation. Confidence serves to indicate the certainty with which the beat tracker believes a given note onset occurred on a downbeat and thus, indirectly, the amount the tempo should change in response to the onset. We took the original confidence calculation and raised it to a power, γ, which is provided as a parameter to the tracker. This parameter allows us to regulate the impact a note onset will have on the new tempo as a function of the distance of the onset from the predicted downbeat. The effect on the confidence function for various ranges of γ is shown in Figure 2. When γ is 1.0, we attain the same confidence calculation as the original method. When γ nears 0.0, note onsets fairly far away from the predicted beat are allowed to change the tempo almost as much as note onsets close to the predicted beat. Conversely, when γ grows very large, only those note onsets very near the predicted beat can affect the tempo significantly. With the new confidence function we are again searching for compromise -- this time, balance between tolerance for imprecise downbeat notes and correctly played non-downbeat notes.

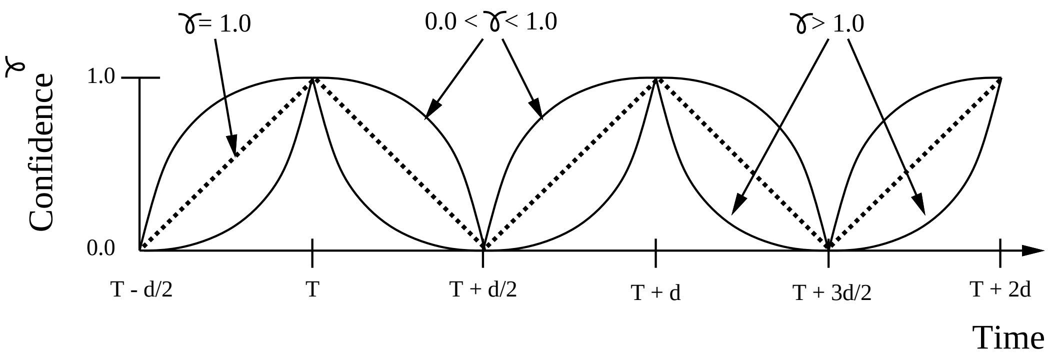

Figure 2: Examples of the new confidence function for various ranges of γ.

With these two parameterizations, we thought we had added enough flexibility for the beat tracker to be able to track large classes of performances well. For a given set of parameter values the algorithm could run in realtime, although searching for parameter values that performed well for a given set of data was a non-realtime task. The search was a brute force search of a grid of points in the parameter space defined by the legal values for the decay rate (0.0 to 1.0) and γ (0.0 to ∞). For each grid point in the parameter space, the algorithm was run on a set of between one to twenty sample performances of similar style. A rough measure of error was determined by counting the average number of beats missed and finding the average relative error calculated by averaging the differences between the timing of actual beats and predicted beats divided by the duration of the beat at that point in the performance.

While we were often able to find large ranges of values for the decay rate and γ that were fairly effective for tracking the training performances, those same parameter value ranges gave rather haphazard results for performances of similar style, expressiveness, and complexity that were not in the training set. These results seemed to indicate that this model was not powerful enough to perform beat tracking with the generality we had hoped.

A Second Approach To Beat Tracking

One of the difficulties with the methods previously discussed is that the beat tracker is required to decide upon a single new state that represents its belief of the current beat period and phase. If it makes a mistake at some point it most likely will never be able to recover and will fail to predict beats beyond the point at which the mistake was made.

We have implemented a method that uses realtime beam search [1] to allow the beat tracker to consider several possible states at once. The method uses a history mechanism similar to the previous method but no longer relies solely on a confidence measure to guide it in making decisions. Instead, a credibility measure is used so that at any given time there is a set of active states that represent the most credible interpretations for the portion of the performance encountered so far. Each state has a credibility value determined by a heuristic evaluation function [1] that measures the credibility of the particular interpretation that the state represents. The state that describes the most credible interpretation is used as the basis for making beat phase and period predictions until a new note onset is processed and a new most credible state is found. When a new note onset is encountered in the performance, each active state is expanded into several new states that represent a subset of the multiple possible interpretations that arise from the note onset and the interpretation described by the active state (see Figure 3). Those interpretations that do not make musical sense are discarded immediately. The set of states that remain become the new active states for the next expansion driven by the next note onset. Repeated application of this expansion process yields a search tree in which the root is the very first state ever expanded, the leaves are the current set of active states, and the nonleaf nodes are states that have been expanded. States that have ungone pruning, explained below, are not considered part of the search tree.

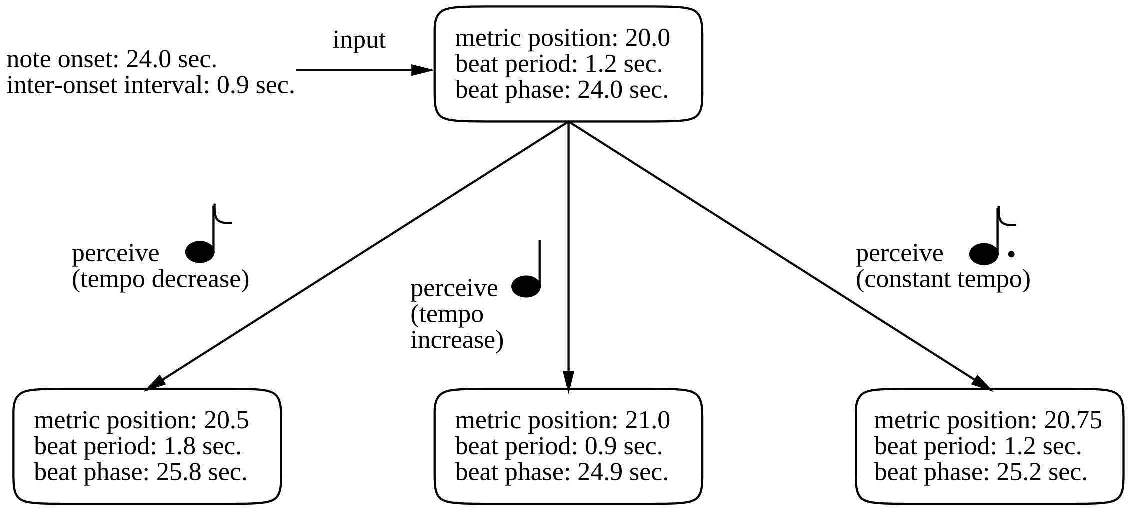

Figure 3: Simplified example of expansion of a single state.

Because this is a realtime method and the number of interpretations is endless, we must liberally apply pruning techniques to decrease the magnitude of the search (see Figure 4). The first pruning method was already mentioned -- we discard those states whose interpretations make no musical sense. This constitutes the highest level of musical reasoning performed by the beat tracker. The following rules are used to judge ‘‘musicality’’:

- Quarter notes must start on the downbeat or upbeat to be considered musical.
- Dotted eighths must start on the downbeat or 1/4 of the way through a beat to be considered musical.
- Eighths are musical only if they begin on an even multiple of sixteenths. If they begin 1/4 of the way through a beat, they must be preceded by a sixteenth.

- Sixteenths are musical only of they begin on an even multiple of sixteenths.
- Quarter and eighth triplets are musical only if they start on an even multiple of eighth triplets. If an eighth triplet starts 2/3 of the way through a beat, it must be preceded by a quarter triplet or another eighth triplet.
- Sixteenth triplets are musical only if they start on an even multiple of sixteenth triplets.
- Notes smaller than triplet quarters crossing a downbeat are not musical.

All rules except the last apply for tied notes also so that, for example, a dotted eighth tied to a quarter must start on the downbeat or 1/4 of the way through a beat to be considered musical. Even though derived in an ad-hoc manner, these rules seem to provide a reasonable amount of pruning power while still being conservative by not pruning possibly correct expansions. Of course, not all music fits these rules, so we designed the system so that rules can be selectively disabled. Although not intentionally derived as such, these rules form a context sensitive grammar for allowable meter similar in spirit to the context free grammars for meter used by Longuet-Higgins [8, 9, 10].

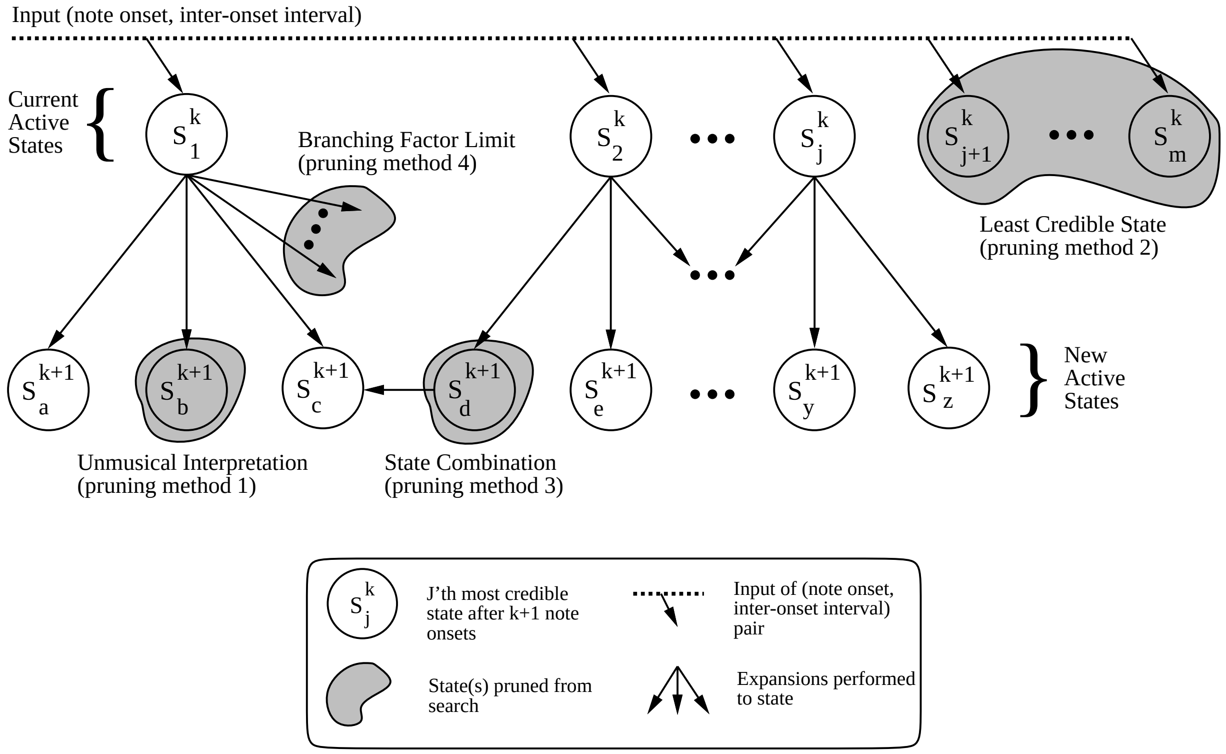

Figure 4: One step of the beam search with example pruning applications.

The second pruning technique we use involves the credibility of the state and the order in which expansions are made. Since we want to perform expansions only until new input is ready or until we expand some predetermined maximum number of states (called the maximum beam width), we expand states in order according to their credibility, those with highest credibility being expanded first. Thus during expansion, if we find that new input is ready to be processed, we can immediately stop expanding the active states knowing that we have already expanded the most credible ones. We require that some minimum number of states (called the minimum beam width) be expanded even if new input is waiting to be processed. This guarantees that we always process some of the most credible states.

The third pruning method employed combines multiple states that represent similar interpretations into a single state representing a single interpretation. Two states are judged to be the same for purposes of combination if their beat period and phase predictions are the same down to a few decimal places. The state with the best score is kept while the other state is removed from further consideration by the algorithm.

The fourth and final technique for pruning the search limits the branching factor by bounding the number of new interpretations that may arise from the expansion of a single active state. Some pruning already occurs at this point from the first pruning method, but since there are an infinite number of interpretations possible given a state and a new note onset, the number of expansions derived from a single state is limited by an arbitrary static cutoff. The set of specific basic note values available for expansion is a separate parameter to the algorithm. This allows us to limit expansions to notes we know are possible to be in the score of a performance. For example, if we know the music is fairly simple and does not contain triplets we may limit the basic note values to sixteenths, eighths, dotted eighths, and quarter notes. From these basic values, longer combinations and tied notes such as half notes and dotted quarter may be derived. If the score for a performance contains triplets, we add the various basic triplet note values such as triplet sixteenth, triplet eighth, and triplet quarter to the ones already mentioned. If the branching factor limit is *k*, then expansions for a state are chosen from the basic set of note values (and their combinations) such that the resulting *k* interpretations are those that have the smallest change in tempo.

This concept of tempo coherence -- that tempo does not change drasticly from note to note -- is also used in the heuristic evaluation function. The heuristic credibility value of a state is the sum of five parameters so that numerically small credibility values correspond to favorable states while large values correspond to less favorable states. The five parameters are as follows:

- The heuristic value of the state's parent. This serves to give continuity to the credibility values by favoring states whose parents had good credibility.
- The magnitude of the tempo change between the state and its parent's state normalized by the metrical length of the last note encountered. This component favors small tempo changes over large changes.
- Three static penalties that are parameters to the beat tracker encapsulate very rudimentary higher level musical reasoning. Each penalty is applied if the state represents an interpretation where the last note processed was assigned a note value that meets certain conditions. These conditions are as follows:
    - The note is some sort of triplet. Triplets are not usually the most abundant type of note even in music that contains a lot of triplets so this serves to favor interpretations with few triplets. We call this the triplet penalty.
    - The note spans a downbeat but is not a ''reasonable'' note like a half, whole, dotted quarter, dotted half, quarter, or a tied combination of them. This serves to penalize interpretations that contain the plausible but not extremely common situation of having strangely tied notes like, for example, a triplet sixteenth tied to a quarter. This is called the ''across beat'' penalty.
    - The note is a dotted eighth or larger but does not begin on a downbeat. This attempts to encapsulate the observation that longer notes tend to start on downbeats and therefore are probably a good clue to where downbeats fall. We call this the ''short note'' penalty.

Inherent in this model is the ability to recover candidate transcriptions of the performance from the interpretations represented by the leaf states. The transcription corresponding to a certain state can be found by finding the path from the root of the search tree to the state. Since each state corresponds to assigning a metrical value to a single note interval, we may traverse the path from the root to the leaf state extracting note values as we do, yielding the transcription. Note that these transcriptions contain no higher level musical organization such as measure bars. They are simply sequences of metrical note values.

An issue that must be addressed before this method is usable is that of how to establish the initial tempo. Dannenberg and Mont-Reynaud [4] describe a method of estimating the initial tempo, but currently, since we are most interested in the main beat tracking algorithm, we avoid the problem by requiring that our method be given the metrical value of the first note of the performance. From this single metrical note value and the actual timing of the first inter-onset interval an initial tempo estimate is found. This first note becomes the state that is the root of the search tree from which actual beat tracking begins and expansions are made. Although this does avoid the problem of estimating the initial tempo, we do admit a potential problem by creating a dependency of the entire tracking system on this single initial note which may or may not be representative of the actual tempo. This depends on the precision with which is it played with respect to the intended tempo of the remainder of the performance. In practice, though, we do not find this dependency to be a problem since our model incorporates a great deal of flexibility.

One other practical aspect of beat tracking is that of making sure that the beat tracker is given only those notes that were intended and not notes that were mistakenly generated. An example of such a unintended note is when a pianist depresses the key next to the desired key for an instant until her finger is placed wholly on the correct key. This has the effect of generating a very short note before the intended note. We deal with this problem simply by filtering the input note stream so that it is free of notes with length below a certain static threshold. If a short note is encountered, its onset is recorded and used as the onset for the next note, thus combining the timing information of the incorrect note with the longer note following it. With a threshold of about 20 ms to 50 ms we find this method works well even for the rather noisy note streams generated by acoustic-to-MIDI interfaces. Of course this same filtering also may be used to remove from the input stream some types of ornamentations such as grace notes.

Results

We have implemented this new beat tracking method in C running on a Commodore Amiga using the Carnegie Mellon MIDI Toolkit [5] for support. It accepts realtime input in the form of MIDI note on/off messages from either a MIDI instrument external to the Amiga or from playback of MIDI sequences stored on the computer. During operation, the beat tracker process sends beat phase and period predictions to a concurrent timer process that controls an external MIDI percussion instrument so that predicted beats are audible along with the input musical performance.

Figure 5 shows an example of the significant plausible interpretations that our method found in an actual performance. In the graph, the major diagonal line indicates the interpretation which was most credible at the end of the performance. The branches from the main diagonal represent alternate interpretations that were maintained for short periods until they were pruned from the search. The plus symbols denote that beat predictions were derived from that interpretation at that point in the performance. Note that a characteristic of our method is that it is possible that consecutive predictions may be not be derived from related interpretations. An example of this is found in the rather long branch above the diagonal near the middle of the performance. Here the two alternate interpretations active at that time shared the generation of predictions because their relative credibility fluctuated making one the most credible interpretation at certain times and the other the most credible at other times. In general, this gives rise to discontinuities in the beat phase and duration predictions that result in unmusical sounding beat tracking performance.

It is usually difficult to quantitatively measure the performance of beat trackers and this case is no different. It is valuable for the purposes of performance evaluation to differentiate two classes of musical scores -- those with very few or no triplets and those with a more significant number of triplets. We will concentrate on the former first.

For performances of musical pieces that fall into the first category, our method has given good performance. It can properly follow large continuous tempo changes in expressive performances. The input can be of the quality produced by an amateur keyboard player with very little training so precision of performance is not a significant factor in the beat tracking results. We have even tested it with input derived from acoustic instruments attached to an IVL Pitch Rider, a commercial pitch follower, and found that the beat tracker handles the glitches and imprecision of the resulting input very well. Typical parameter values used for performances of this class are as follows: decay rate of 60%, triplet penalty of 0.1, across beat penalty of 0.1, short note penalty of 0.0, branching factor of 2 or 3,

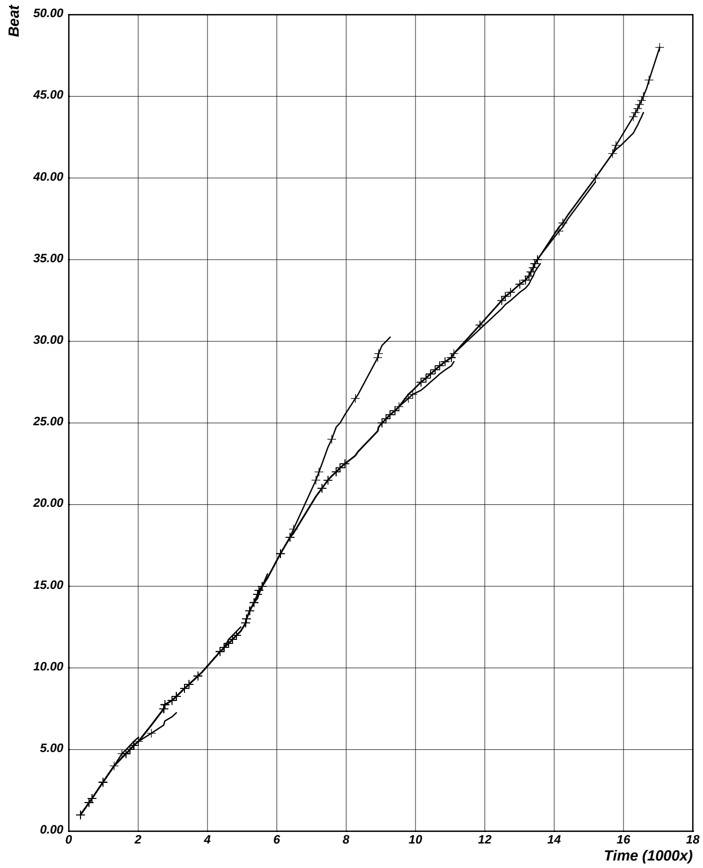

Figure 5: A plot of multiple interpretations generated by our method for an actual performance.

both minimum and maximum beam width of 2 or 3, and a basic note set containing triplets is used. We were surprised by the low values required for the beam widths. There is enough time to do an order of magnitude more processing on each input, but these numbers prove to be usable most of the time. A specific example of a performance from this class where this model performs well may be found in [8] as Figure 9a. Though this example does not contain any tempo changes it does contain a few triplets and is an example of the precision (or lack thereof) of actual recorded input. Another specific example of a short performance of this class that our method handles well is given in [6] as Figure 6. In this example, the flexibility inherent in the model to deal with imprecisely played note values is exhibited.

For performances of musical pieces that fall into the second category of containing more than a few triplets, our method is far less satisfactory than for performances from the first category. The method often has trouble following constant tempo performances of pieces from this class. What often results is unmusical beat tracking performance as shown previously in Figure 5. Even with these problems, the model is often still usable when applied to performances from the second category. This is shown by the ability to perform well for the example given in [8] as Figure 10a. This is an example of a constant tempo piece with a very significant amount of triplets. The parameter values for the model used for this example are the same as previously mentioned except that the triplet penalty is 0.0 and the short note penalty is 0.1.

Despite the difficulties this method has in successfully processing performances from the second category of musical pieces, we believe that this method would make an excellent preprocessor for a more complex realtime system performing higher level musical reasoning such as rhythmic and metric analysis. At any instant of a performance our method can provide a set of plausible interpretations of the performance, mapping low level timing information into symbolic note level information. Thus a higher level reasoning system could query the low level beat tracker for a set of the most plausible interpretations and perform analysis on those interpretations saving those that are plausible with respect to the higher level analysis and disregarding those that prove to be implausible under the higher level musical scrutiny.

References

[1] Barr, Avron and Feigenbaum, Edward A. (editors).
The Handbook of Artificial Intelligence.
Addison-Wesley, Menlo Park,CA, 1981.

[2] Chafe, Chris; Mont-Reynaud, Bernard; and Rush, Loren.
Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs.
Computer Music Journal 6(1):30-41, 1982.

[3] Dannenberg, Roger B.
An On-Line Algorithm for Real-Time Accompaniment.
In *Proceedings of the 1984 International Computer Music Conference*, pages 193-198. 1984.

[4] Dannenberg, Roger B. and Mont-Reynaud, Bernard.
Following an Improvisation in Real Time.
In *Proceedings of the 1987 International Computer Music Conference*, pages 241-248. 1987.

[5] Dannenberg, Roger B.
The CMU MIDI Toolkit
Carnegie Mellon University, Pittsburgh, PA, 1989.
Version 3.0.

[6] Desain, Peter and Honing, Henkjan.
The Quantization of Musical Time: A Connectionist Approach.
Computer Music Journal 13(3):56-66, 1989.

[7] Longuet-Higgins, H. C. and Steedman, M. J.
On Interpreting Bach.
In Meltzer, B. and Michie, D. (editors), *Machine Intelligence 6*, pages 221-241. Edinburgh University Press, Edinburgh, 1971.

[8] Longuet-Higgins, H. C.
Perception of Melodies.
Nature 263:646-653, 1976.

[9] Longuet-Higgins, H. C.
The Perception of Music.
Interdisciplinary Science Reviews 3(2):148-156, 1978.

[10] Longuet-Higgins, H. C. and Lee, C. S.
The Perception of Musical Rhythms.
Perception 11:115-128, 1982.

[11] Steedman, Mark J.
The Perception of Musical Rhythm and Metre.
Perception 6:555-569, 1977.

[12] Vercoe, Barry and Puckette, Miller.
Synthetic Rehearsal: Training the Synthetic Performer.
In *Proceedings of the 1985 International Computer Music Conference*, pages 275-278. 1985.

List of Figures