Beat Tracking based on Multiple-agent Architecture — A Real-time Beat Tracking System for Audio Signals —

Masataka Goto and **Yoichi Muraoka**
School of Science and Engineering, Waseda University
3-4-1 Ohkubo Shinjuku-ku, Tokyo 169, JAPAN.
{goto, muraoka}@muraoka.info.waseda.ac.jp

Abstract

This paper presents an application of multiple-agent architecture to beat tracking for musical acoustic signals. Beat tracking is an important initial step in computer understanding of music and is useful in various multimedia applications. Most previous beat-tracking systems dealt with MIDI signals and were not based on a multiple-agent architecture. Our system can recognize, in real time, temporal positions of beats in real-world audio signals that contain sounds of various instruments. Our application of multiple-agent architecture enables the system to handle ambiguous situations in interpreting real-world input signals and to examine multiple hypotheses of beat positions in parallel. Even if some agents lose track of the beat, other agents will maintain the correct hypothesis. Each agent is able to interact with other agents to track beats cooperatively, and self-evaluate the reliability of its hypothesis on the basis of the current input situation, and adapt to the current situation in order to maintain the correct hypothesis. These agents have been implemented on different processing elements in a parallel computer. Our experimental results show that the system is robust enough to handle audio signals sampled from commercially distributed compact discs of popular songs.

Introduction

Multiple-agent architectures have recently been applied in various domains. This paper describes our application of multiple-agent architecture to beat tracking for musical acoustic signals. In our formulation, beat tracking means tracking the temporal positions of quarter notes, just as people keep time to music by hand-clapping or foot-tapping. There are various ambiguous situations that occur when a system interprets real-world input audio signals like those sampled from compact discs. Multiple-agent architecture has the advantages of interpreting those signals and tracking beats in various ways, because different agents can examine multiple hypotheses of beat positions in parallel according to different strategies. The main contribution of this paper is to show that such a multiple-agent architecture is actually useful and effective for a practical real-world application, namely, beat tracking.

Beat tracking is an important initial step in computer emulation of human music understanding, since beats are fundamental to the perception of Western music. A person who cannot completely segregate and identify every sound component can nevertheless track musical beats. It is almost impossible to understand music without perceiving beats, since the beat is the basic unit of the temporal structure of music. Moreover, musical beat tracking is itself useful in various applications, such as video editing, audio editing, stage lighting control, and music-synchronized CG animation (Goto & Muraoka 1994). We therefore first build a computational model of beat perception and then extend the model, just as a person recognizes higher-level musical events on the basis of beats.

Various beat-tracking related systems have been undertaken in recent years (Dannenberg & Mont-Reynaud 1987; Desain & Honing 1989; Allen & Dannenberg 1990; Driesse 1991; Rosenthal 1992; Desain & Honing 1994; Vercoe 1994; Large 1995). Some previous systems (Allen & Dannenberg 1990; Rosenthal 1992) have maintained multiple hypotheses to track beats, and an earlier paper (Rosenthal, Goto, & Muraoka 1994) has presented the advantages of the strategy of pursuing multiple hypotheses. Most of the systems maintaining multiple hypotheses, however, were not based on a multiple-agent architecture. The one described in (Allen & Dannenberg 1990) examined two or three hypotheses by beam search and tracked beats in real time. It dealt only with MIDI signals as its input and was not able to deal with audio signals played on several musical instruments, however. Another MIDI-based system (Rosenthal 1992) maintained a number of hypotheses that were periodically ranked and selected. Those hypotheses were examined sequentially and the system did not work in real time.

We built a beat tracking system that processes real-world audio signals that contain the sounds of various instruments and that recognizes the temporal positions of beats in real time. Our system is based on multiple-agent architecture in which multiple hypotheses are maintained by programmatic agents using different strategies for beat-tracking. Because the input signals are examined from the viewpoints of these various agents, various hypotheses can emerge. Agents that pay attention to different frequency ranges, for example, may track different beat positions. This multiple-agent architecture enables the system to cope with difficult beat-tracking situations: even if some agents lose track of beats, the system will track beats correctly as long as other agents maintain the correct hypothesis.

Each agent is capable of interaction, self-evaluation, and adaptation. In making a hypothesis, the agent interacts with other agents to track beats cooperatively. Each agent then

evaluates the reliability of its own hypothesis on the basis of the current input situation, and the most reliable hypothesis is considered the final output. If the reliability of a hypothesis becomes high enough, the agent tries to adapt to the current situation by adjusting a parameter that controls its strategy in order to maintain the correct hypothesis.

To perform this computationally intensive task in real time, the system has been implemented on a parallel computer, the Fujitsu AP1000. Each agent and frequency-analysis module has been implemented on a different processing element. In our experiment with audio signals sampled from compact discs, the system correctly tracked beats in 34 out of 40 popular songs that did not include drum-sounds. This result shows that our beat-tracking model based on multiple-agent architecture is robust enough to handle real-world audio signals.

Multiple-agent Architecture for Beat Tracking

In this section we specify the beat tracking problem that we are dealing with and present the main difficulties of tracking beats: ambiguity of interpretation and the need for context-dependent decisions, difficulties which are common to other real-world perceptual problems. We then describe the multiple-agent architecture to address the beat tracking problem, defining our agents and outlining their interaction.

Beat Tracking Problem

In our formulation, beat tracking is defined as a process that organizes music into almost regularly spaced beats corresponding to quarter notes. Our beat tracking problem is thus to obtain an appropriate sequence of *beat times* (temporal positions of beats) that corresponds to input musical audio signals (Figure 1). This sequence of beat times is called the quarter-note level. We also address the higher-level beat tracking problem of determining whether a beat is strong or weak (*beat type*)[1] under the assumption that the time-signature of an input song is 4/4. This is the problem of tracking beats at the half-note level.

There are various difficulties in tracking the beats in real-world musical acoustic signals. The simple technique of peak-finding with a threshold is not sufficient since there are many energy peaks that are not directly related to beats. Multiple interpretations of beats are possible at any given point because there is not necessarily a single specific sound that directly indicates the beat position; the beat is a perceptual concept that a human feels in music. There are various ambiguous situations, such as ones where several events obtained by frequency analysis may correspond to a beat and where different *inter-beat intervals* (the temporal difference between two successive beats) seem plausible. In addition, higher-level processing using musical knowledge is necessary for making context-dependent decisions, such as determining whether a beat is strong or weak and evaluating which is the best interpretation in an ambiguous situation.

[1] In this paper, a *strong beat* is either the first or third quarter note in a measure; a *weak beat* is the second or fourth.

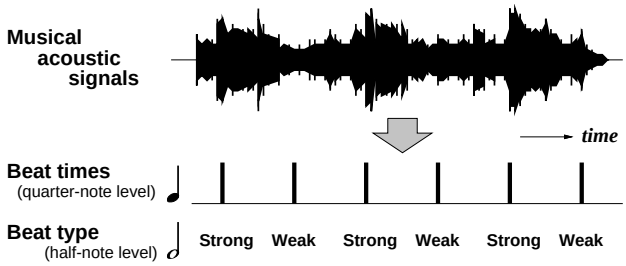


Figure 1: Beat tracking problem.

Our solution to the problem of handling ambiguous situations is to maintain multiple hypotheses, each of which corresponds to a provisional or hypothetical interpretation of the input. A real-time system using only a single hypothesis is subject to garden-path errors. A multiple-hypothesis system can pursue several paths simultaneously, and later decide which one was correct. In other words, in real-time beat tracking these hypotheses represent the results of predicting the next beat in different ways and it is impossible to know in advance which one will be correct (because the future events are not available).

Multiple-agent Architecture

To examine multiple hypotheses in parallel, we use a multiple-agent architecture in which agents with different strategies interact through cooperation and competition to track beats (Figure 2). Several definitions of the term *agent* have been proposed (Minsky 1986; Maes 1990; Shoham 1993; CACM 1994; Nakatani, Okuno, & Kawabata 1994; ICMAS 1995); in our terminology, the term *agent* means a software component that satisfies the following three requirements:

1. the agent interacts with other agents to perform a given task.
2. the agent evaluates its own behavior on the basis of the input.
3. the agent adapts to the input by adjusting its own behavior.

Each agent maintains a beat-position hypothesis, which consists of a predicted next-beat time, its beat type (strong

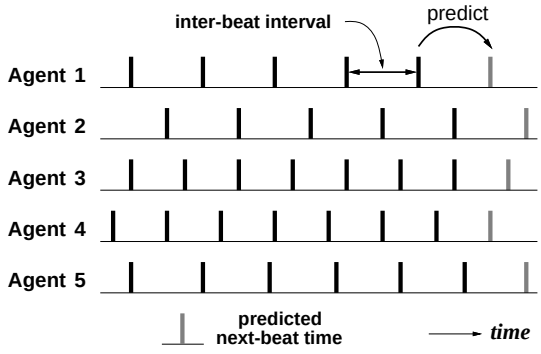


Figure 2: Multiple hypotheses maintained by multiple agents.

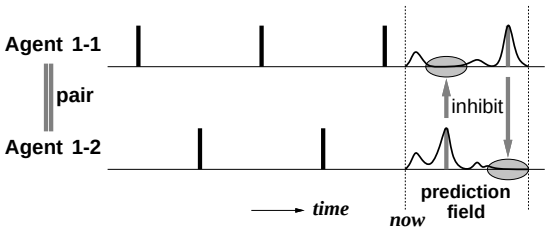


Figure 3: Interaction between agents through a prediction field.

or weak), and the current inter-beat interval. In making the hypothesis, the agent interacts with other agents to perform the beat-tracking task (the first requirement). All agents are grouped into pairs that have different strategies for beat tracking. Each agent in the pair examines the same inter-beat interval using the same frequency-analysis results. To predict the next beat times cooperatively, one agent interacts with the other agent in the same pair through a *prediction field*. The prediction field is an expectancy curve[2] that represents when the next beat is expected to occur (Figure 3). The height of each local peak in the prediction field can be interpreted as the next beat-position possibility. The two agents interact with each other by inhibiting the prediction field in the other agent. The beat time of each hypothesis inhibits the temporally corresponding neighborhood in the other's field (Figure 3). This enables one agent to track the correct beats even if the other agent tracks the middle of the two successive correct beats (which compensates for one of the typical tracking errors).

Each agent is able to evaluate its own hypothesis, using musical knowledge, according to the input acoustic signals (the second requirement). We call the quantitative result of this self-evaluation the reliability of the hypothesis. The final beat-tracking result is determined on the basis of the most reliable hypothesis that is selected from the hypotheses of all agents.

Each agent also adapts to the current input by adjusting its own strategy parameter (the third requirement). If the reliability of a hypothesis becomes high enough, the agent tunes a parameter to narrow the range of possible inter-beat intervals so that it examines only a neighborhood of the current appropriate one. This enables the agent to maintain the hypothesis that has the inter-beat interval appropriate to the current input.

System Description

The system for musical audio signals without drum-sounds[3] assumes that the time-signature of an input song is 4/4 and that its tempo is constrained to be between 61 M.M.

[2]Other systems (Desain 1992; Desain & Honing 1994; Vercoe 1994) have used a similar concept of expectancy curve for predicting future events, but not as a means for managing interaction among agents.

[3]A detailed description of our beat-tracking system for audio signals that include drum-sounds is presented in (Goto & Muraoka 1995a; 1995b).

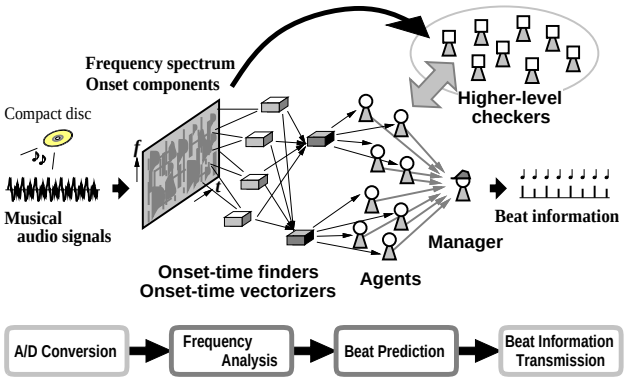


Figure 4: Processing model.

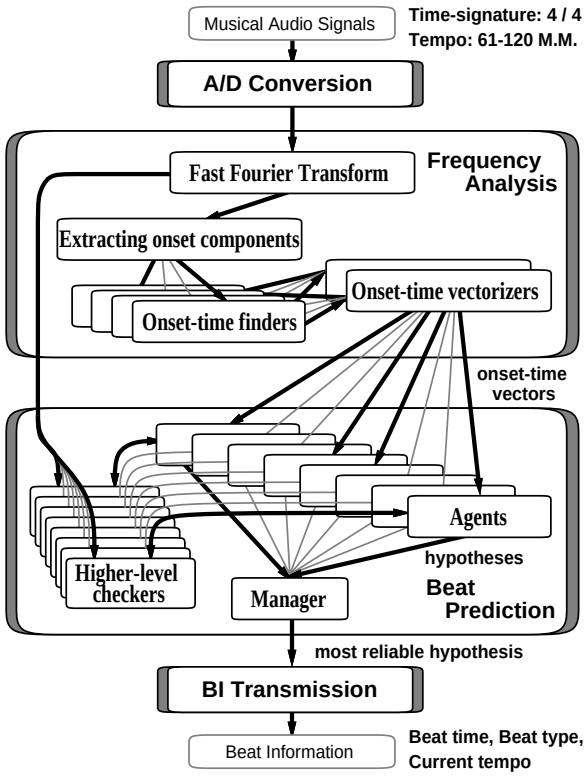


Figure 5: Overview of our beat tracking system.

(Mälzel's Metronome: the number of quarter notes per minute) and 120 M.M., and is roughly constant. The emphasis in our system is on finding the temporal positions of quarter notes in audio signals rather than on tracking tempo changes. The system maintains, as the real-time output, a description called *beat information* (*BI*) that consists of the beat time, its beat type, and the current tempo.

Figure 4 is a sketch of the processing model of our beat tracking system, and Figure 5 shows an overview of the system. The two main stages of processing are *Frequency Analysis*, in which several cues used by agents are detected, and *Beat Prediction*, in which multiple hypotheses of beat positions are examined by multiple agents. Since accurate onset

times are indispensable for tracking beats, in the *Frequency Analysis* stage, the system uses multiple onset-time finders that detect onset times in several different frequency ranges. Those results are transformed into vectorial representation (called *onset-time vectors*) by several *onset-time vectorizers*. In the *Beat Prediction* stage, the system manages multiple agents that, according to different strategies, make parallel hypotheses based on these onset-time vectors. Each agent first calculates the inter-beat interval and predicts the next beat time; it then infers the beat type by communicating with a *higher-level checker* (described later), and evaluates the reliability of its own hypothesis. The manager gathers all hypotheses and then determines the final output on the basis of the most reliable one. Finally, the system transmits BI to other application programs via a computer network.

The following describe the main stages of Frequency Analysis and Beat Prediction.

Frequency Analysis

In the *Frequency Analysis* stage, the frequency spectrum and several sequences of n-dimensional onset-time vectors are obtained for later processing (Figure 6). The full frequency band is split into several frequency ranges, and each dimension of the onset-time vectors corresponds to a different frequency range. This representation makes it possible to consider onset times of all the frequency ranges at the same time. Each sequence of onset-time vectors is obtained using a different set of weights for frequency ranges. One sequence, for example, focuses on middle frequency ranges, and another sequence focuses on low frequency ranges.

Fast Fourier Transform (FFT) The frequency spectrum (the power spectrum) is calculated with the FFT using the Hanning window. Each time the FFT is applied to the digitized audio signal, the window is shifted to the next frame.

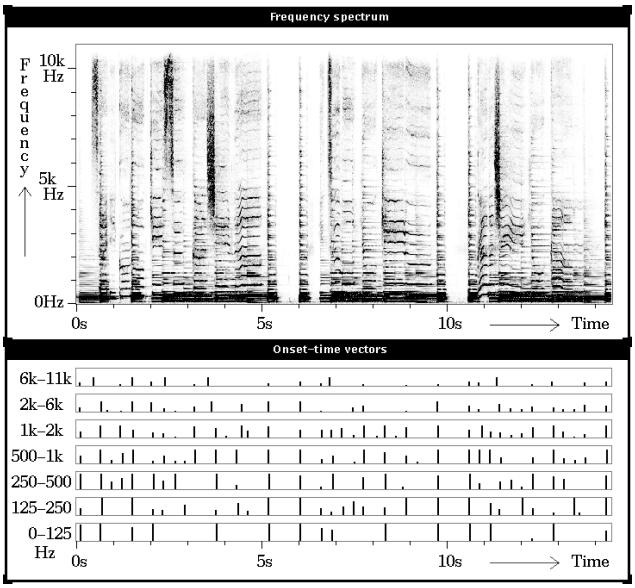


Figure 6: An example of frequency spectrum and an onset-time vector sequence.

In our current implementation, the input signal is digitized at 16bit/22.05kHz, and two kinds of FFT are calculated. One FFT, for extracting onset components in the Frequency Analysis stage, is calculated with a window size of 1024 samples (46.44 ms), and the window is shifted by 256 samples (11.61 ms). The frequency resolution is consequently 21.53 Hz and the time resolution (1 *frame-time*) is 11.61 ms. The frame-time is the unit of time used in our system, and the term *time* in this paper is defined as the time measured in units of the frame-time. The other FFT, for examining chord changes in the Beat Prediction stage, is simultaneously calculated in audio down-sampled at 16bit/11.025kHz with a window size of 1024 samples (92.88 ms), and the window is shifted by 128 samples (11.61 ms). The frequency and time resolution are consequently 10.77 Hz and 1 frame-time.

Extracting Onset Components Frequency components whose power has been rapidly increasing are extracted as onset components. The onset components and their degree of onset (rapidity of increase in power) are obtained from the frequency spectrum by a process that takes into account the power present in nearby time-frequency regions. More details on the method of extracting onset components can be found in (Goto & Muraoka 1995a).

Onset-time Finders Multiple onset-time finders (seven in our current implementation) detect onset times in several different frequency ranges (0-125 Hz, 125-250 Hz, 250-500 Hz, 500 Hz-1 kHz, 1-2 kHz, 2-6 kHz, and 6-11 kHz). Each onset time is given by the peak time found by peak-picking in $D(t)$ along the time axis, where $D(t) = \sum_f d(t, f)$, and $d(t, f)$ is the degree of onset of frequency f at time t. Limiting the range of frequency for the summation of $D(t)$ makes it possible to find onset times in the different frequency ranges.

Onset-time Vectorizers Each onset-time vectorizer transforms the results of all onset-time finders into sequences of onset-time vectors: the same onset times in all the frequency ranges are put together into a vector. In the current system, three vectorizers transform onset times from seven finders into three sequences of seven-dimensional onset-time vectors with the different sets of frequency weights (focusing on all/low/middle frequency ranges). These results are sent to agents in the *Beat Prediction* stage.

Beat Prediction

Multiple agents interpret the sequences of onset-time vectors according to different strategies and maintain their own hypotheses. Musical knowledge is necessary to determine the beat type (strong or weak) and to evaluate which hypothesis is best. For the audio signals without drum-sounds, the system utilizes the following musical knowledge:

1. Sounds are likely to occur on beats. In other words, the correct beat times tend to coincide with onset times.
2. Chords are more likely to change at the beginning of measures than at other positions.
3. Chords are more likely to change on beats (quarter-notes) than on other positions between two successive correct beats.

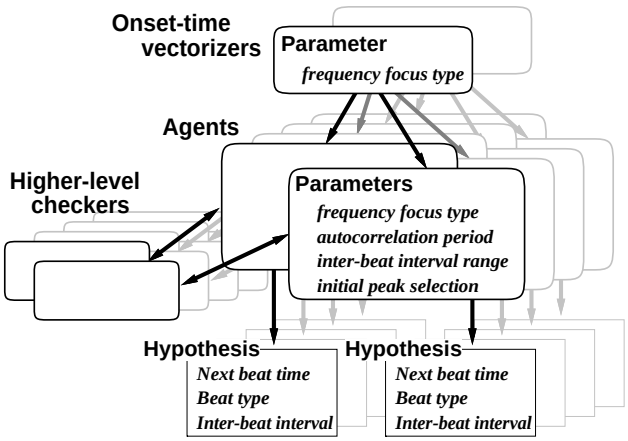


Figure 7: Relations between onset-time vectorizers, agents, and higher-level checkers.

Table 1: Initial settings of the strategy parameters.

pair -agent	frequency focus type	auto-correlation period	inter-beat interval range	initial peak selection
1-1	type-all	500 f.t.	43-85 f.t.	primary
1-2	type-all	500 f.t.	43-85 f.t.	secondary
2-1	type-all	1000 f.t.	43-85 f.t.	primary
2-2	type-all	1000 f.t.	43-85 f.t.	secondary
3-1	type-low	500 f.t.	43-85 f.t.	primary
3-2	type-low	500 f.t.	43-85 f.t.	secondary
4-1	type-low	1000 f.t.	43-85 f.t.	primary
4-2	type-low	1000 f.t.	43-85 f.t.	secondary
5-1	type-middle	500 f.t.	43-85 f.t.	primary
5-2	type-middle	500 f.t.	43-85 f.t.	secondary
6-1	type-middle	1000 f.t.	43-85 f.t.	primary
6-2	type-middle	1000 f.t.	43-85 f.t.	secondary

* "f.t." is the abbreviation of frame-time (11.61 ms).

To utilize the second and third kinds of knowledge, each agent communicates with a corresponding *higher-level checker*, which is a module to provide higher-level information, such as the results of examining the possibility of chord changes according to the current hypothesis (Figure 7). The agent utilizes this information to determine the beat type and to evaluate the reliability of the hypothesis.

Each agent has four parameters that determine its strategy for making the hypothesis (Figure 7), and the settings of these parameters vary from agent to agent. The first parameter, *frequency focus type*, determines which vectorizer the agent receives onset-time vectors from. This value is chosen from among *type-all*, *type-low*, and *type-middle*, respectively corresponding to vectorizers focusing on all frequency ranges, low frequency ranges, and middle frequency ranges. The second parameter, *autocorrelation period*, determines the window size for calculating vectorial autocorrelation of the sequence of onset-time vectors to determine the inter-beat interval. The greater this value, the older the onset-time information considered. The third parameter, *inter-beat interval range*, controls the range of possible inter-beat intervals. As described later, this limits the range of selecting a peak in the result of the vectorial autocorrelation. The fourth parameter, *initial peak selection*, takes a value of either *primary* or *secondary*. When the value is *primary*, the largest peak in the prediction field is initially selected, and the peak is considered as the next beat time; when the value is *secondary*, the second largest peak is selected. This helps to obtain a variety of hypotheses.

In our current implementation there are twelve agents grouped into six agent-pairs, and twelve higher-level checkers corresponding to these agents. Initial settings of the strategy parameters are listed in Table 1. As explained in Section *Multiple-agent Architecture*, the parameter *inter-beat interval range* is adjusted as the processing goes on.

The following sections describe the formation and management of hypotheses. First, each agent determines the inter-beat interval using autocorrelation; it then interacts with its paired agent through the prediction field that is formed using cross-correlation, and predicts the next beat time. Second, the agent communicates with the higher-level checker to infer the beat type and evaluates its own reliability. The checker examines possibilities of chord changes by analyzing the frequency spectrum on the basis of the current hypothesis received from the agent. Finally, the manager gathers all the hypotheses, and the most reliable one is considered as the output.

Beat-predicting Agents In our formulation, beats are characterized by two properties: period (inter-beat interval) and phase. The phase of a beat is the beat position relative to a reference point, usually the previous beat time. We measure phase in radians; for a quarter-note beat, for example, an eighth-note displacement corresponds to a phase-shift of π radians.

Each agent first determines the current inter-beat interval (period) (Figure 8). The agent receives the sequence of onset-time vectors and calculates their vectorial autocorrelation[4]. The windowed and normalized vectorial autocorrelation function $Ac(\tau)$ is defined as

$$Ac(\tau) = \frac{\sum_{t=c-W}^{c} (\vec{o}(t) \cdot \vec{o}(t-\tau))\, w(c-t)}{\sum_{t=c-W}^{c} (\vec{o}(t) \cdot \vec{o}(t))\, w(c-t)}, \qquad (1)$$

where $\vec{o}(t)$ is the n-dimensional onset-time vector at time t, c is the current time, and W is the strategy parameter *autocorrelation period*. The window function $w(t)$ is given by

$$w(t) = 1.0 - 0.5\frac{t}{W}. \qquad (2)$$

The inter-beat interval is given by the τ with the maximum height in $Ac(\tau)$ within the range limited by the parameter *inter-beat interval range*.

To determine the beat phase, the agent then forms the prediction field (Figure 8). The prediction field is the result of calculating the cross-correlation function between the sequence of the onset-time vectors and the sequence of beat

[4]The paper (Vercoe 1994) also proposed using a variant of autocorrelation for rhythmic analysis.

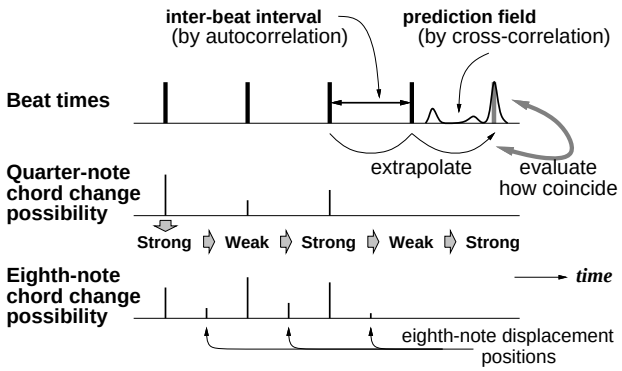


Figure 8: Predicting the next beat.

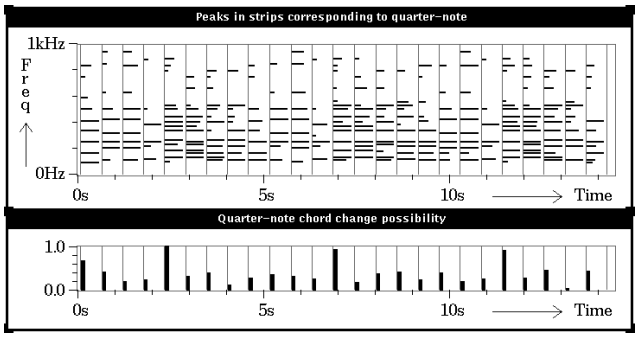
times whose interval is the inter-beat interval. As mentioned in Section *Multiple-agent Architecture*, the two agents in the same pair interact with each other by inhibiting the prediction field in the other agent. Each local peak in the prediction field is considered as a possible beat phase. When the reliability of a hypothesis is low, the agent initially selects the peak in the prediction field according to the parameter *initial peak selection*, and then tries to pursue the peak equivalent to the previously selected one. This calculation corresponds to evaluating all possibilities of the beat phase under the current inter-beat interval. The next beat time is thus predicted on the basis of the inter-beat interval and the current beat phase.

The agent receives the two kinds of possibilities of chord changes, at the quarter-note level and at the eighth-note level, by communicating with the higher-level checker. We call the former the *quarter-note chord change possibility* and the latter the *eighth-note chord change possibility*. The quarter-note (eighth-note) chord change possibility represents how a chord is likely to change on each quarter-note (eighth-note) position under the current hypothesis.

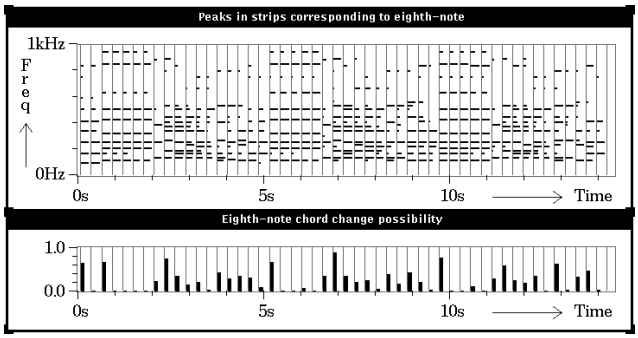
To infer the beat type, we use the second kind of musical knowledge, which means that the quarter-note chord change possibility is higher on a strong beat than on a weak beat. If the quarter-note chord change possibility is high enough, its time is considered to indicate the position of the strong beat. The following beat type is then determined under the assumption that strong and weak beats alternate (Figure 8).

The agent finally evaluates the reliability of its own hypothesis by using the first and third kinds of musical knowledge. According to the first kind, the reliability is determined according to how the next beat time predicted on the basis of the onset times coincides with the time extrapolated from the past two beat times (Figure 8). If they coincide, the reliability is increased; otherwise, the reliability is decreased. According to the third kind of knowledge, if the eighth-note chord change possibility is higher on beats than on eighth-note displacement positions, the reliability is increased; otherwise, the reliability is decreased.

Higher-level Checkers For the audio signals without drum-sounds, each higher-level checker examines two kinds of chord change possibilities according to the hypotheses received from the corresponding agent.



(a) Examining quarter-note chord change possibility



(b) Examining eighth-note chord change possibility

Figure 9: Examples of peaks in sliced frequency spectrum and chord change possibility.

The checker first slices the frequency spectrum into strips at the quarter-note times (beat times) for examining the quarter-note chord change possibility, and slices at the eighth-note times interpolated from beat times for examining the eighth-note chord change possibility (Figure 9). The checker then finds peaks along the frequency axis in a histogram summed up along the time axis in each strip. These peaks can be considered as the dominant tones' pitches in each strip. Some peaks may be components of a chord, and others may be components of a melody. Our current implementation considers only peaks whose frequency is less than 1 kHz.

The checker evaluates the chord change possibilities by comparing these peaks between adjacent strips. The more and the louder the peaks occur compared with the previous strip, the higher the chord change possibilities. For the quarter-note (eighth-note) chord change possibility, the checker compares the strips whose period corresponds to the quarter-note (eighth-note) duration under the current hypothesis.

Figure 9 shows examples of two kinds of chord change possibilities. The horizontal lines above represent peaks in each strip's histogram. The thick vertical lines below represent the chord change possibility. The beginning of measure comes at every four quarter-notes from the extreme left in (a), and the beat comes at every two eighth-notes from the extreme left in (b).

Hypotheses Manager The manager classifies all agent-generated hypotheses into groups according to beat time and inter-beat interval. Each group has an overall reliability given by the sum of the reliabilities of the group's hypotheses. The manager then selects the dominant group that has the highest reliability. Since a wrong group could be selected if temporarily unstable beat times split the appropriate dominant group, the manager repeats grouping and selecting three times while narrowing the allowable margin of beat times for becoming the same group. The reliable hypothesis in the most dominant group is thus selected as the output and sent to the BI Transmission stage.

The manager updates the beat type in the output using only the beat type that was labeled when the quarter-note chord change possibility was high compared with the recent maximum possibility. When the possibility was not high enough, the updated beat type is determined from the previous reliable beat type based on the alternation of strong and weak beats. This enables the system to disregard an incorrect beat type that is caused by a local irregularity of chord changes.

Implementation on a Parallel Computer

Parallel processing provides a practical and feasible solution to the problem of performing a computationally intensive task, such as processing and understanding complex audio signals, in real time. Our system has been implemented on a distributed-memory parallel computer, the Fujitsu AP1000 that consists of 64 processing elements(Ishihata *et al.* 1991). A different element or group of elements is assigned to each module, such as FFT, the onset-time finder, the onset-time vectorizer, the agent, the higher-level checker, and the manager. These modules run concurrently and communicate with others by passing messages between processing elements.

We use four kinds of parallelizing techniques in order to execute the heterogeneous processes simultaneously (Goto & Muraoka 1996). The processes are first pipelined, and then each stage of the pipeline is implemented with data/control parallel processing, pipeline processing, and distributed cooperative processing. This implementation makes it possible to analyze audio signals in various ways and to manage multiple agents in real time.

Experiments and Results

We tested the system for audio without drum-sounds on 40 songs performed by 28 artists. The initial one or two minutes of those songs were used as the inputs. The inputs were monaural audio signals sampled from commercial compact discs of the popular music genre. Their tempi ranged from 62 M.M. to 116 M.M. and were roughly constant. It is usually more difficult to track beats in songs without drum-sounds than in songs with drum-sounds, because they tend to have fewer sounds which fall on the beat and musical knowledge is difficult to apply in general.

In our experiment, the system correctly tracked beats (i.e., obtained the beat time and type) in 34 out of the 40 songs in real time[5]. In each song where the beat was eventually determined correctly, the system initially had trouble determining the beat type, even though the beat time was correct. Within at most fifteen measures of the beginning of the song, however, both the beat time and type had been determined correctly. In most of the mistaken songs, beat times were not obtained correctly since onset times were very few or the tempo fluctuated temporarily. In other songs, the beat type was not determined correctly because of irregularity of chord changes.

These results show that the system is robust enough to deal with real-world musical signals. We have also developed an application with the system that displays a computer graphics dancer whose motion changes with musical beats in real time (Goto & Muraoka 1994). This application has shown that our system is also useful in multimedia applications in which human-like hearing ability is desirable.

Discussion

Our goal is to build a system that can understand musical audio signals in a human-like fashion. We believe that an important initial step is to build a system which, even in its preliminary implementation, can deal with real-world acoustic signals like those sampled from compact discs. Most previous beat tracking systems had great difficulty working in real-world acoustic environments, however. Most of these systems (Dannenberg & Mont-Reynaud 1987; Desain & Honing 1989; Allen & Dannenberg 1990; Driesse 1991; Rosenthal 1992) have dealt with MIDI signals as their input. Since it is quite difficult to obtain complete MIDI representations from audio data, MIDI-based systems are limited in their application. Although some systems (Schloss 1985; Katayose *et al.* 1989) dealt with audio signals, they had difficulty processing music played on ensembles containing a variety of instruments and did not work in real time.

Our strategy of first building a real-time system that works in real-world complex environments and then upgrading the ability of the system is related to the scaling-up problem (Kitano 1993) in the domain of artificial intelligence (Figure 10). As Hiroaki Kitano stated:

> experiences in expert systems, machine translation systems, and other knowledge-based systems indicate that

[5]Our other beat-tracking system for audio signals that include drum-sounds, which is based on a similar multiple-agent architecture, correctly tracked beats in 42 out of the 44 songs that included drum-sounds(Goto & Muraoka 1995b).

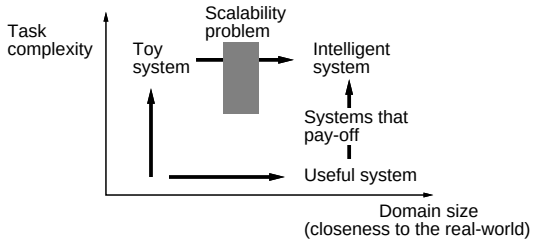


Figure 10: Scaling-up problem (Kitano 1993).

> scaling up is extremely difficult for many of the prototypes. (Kitano 1993)

In other words, it is hard to scale-up a system whose preliminary implementation works only in laboratory environments. We think that our strategy addresses this issue and that the application of multiple-agent architecture makes the system robust enough to work in real-world environments.

Some researchers might regard as *agents* several modules in our system, such as the onset-time finders, the onset-time vectorizers, the higher-level checkers, and the manager. In our terminology, however, we define the term *agent* as a software component of distributed artificial intelligence that satisfies the three requirements presented in Section *Multiple-agent Architecture*. We therefore do not consider those modules agents: they are simply concurrent objects.

Conclusion

We have presented a multiple-agent architecture for beat tracking and have described the configuration and implementation of our real-time beat tracking system. Our system tracks beats in audio signals containing sounds of various instruments and reports beat information in time to input music. The experimental results show that the system is robust enough to handle real-world audio signals sampled from compact discs of popular music.

The system manages multiple agents that track beats according to different strategies in order to examine multiple hypotheses in parallel. This enables the system to follow beats without losing track of them, even if some hypotheses become wrong. Each agent can interact with other agents to track beats cooperatively and can evaluate its own hypothesis according to musical knowledge. Each agent also can adapt to the current input by adjusting its own strategy. These abilities make it possible for the system to handle ambiguous situations by maintaining various hypotheses, and they make the system robust and stable.

We plan to upgrade the system to make use of other higher-level musical structure, and to generalize to other musical genres. Future work will include application of the multiple-agent architecture to other perceptual problems and will also include a study of more sophisticated interaction among agents and more dynamic multiple-agent architecture in which the total number of agents is not fixed.

Acknowledgments

We thank David Rosenthal for his helpful comments on earlier drafts of this paper. We also thank Fujitsu Laboratories Ltd. for use of the AP1000.

References

Allen, P. E., and Dannenberg, R. B. 1990. Tracking musical beats in real time. In *Proc. of the 1990 Intl. Computer Music Conf.*, 140−143.

CACM. 1994. Special issue on intelligent agents. *Communications of the ACM* 37(7):18−147.

Dannenberg, R. B., and Mont-Reynaud, B. 1987. Following an improvisation in real time. In *Proc. of the 1987 Intl. Computer Music Conf.*, 241−248.

Desain, P., and Honing, H. 1989. The quantization of musical time: A connectionist approach. *Computer Music Journal* 13(3):56−66.

Desain, P., and Honing, H. 1994. Advanced issues in beat induction modeling: syncopation, tempo and timing. In *Proc. of the 1994 Intl. Computer Music Conf.*, 92−94.

Desain, P. 1992. Can computer music benefit from cognitive models of rhythm perception? In *Proc. of the 1992 Intl. Computer Music Conf.*, 42−45.

Driesse, A. 1991. Real-time tempo tracking using rules to analyze rhythmic qualities. In *Proc. of the 1991 Intl. Computer Music Conf.*, 578−581.

Goto, M., and Muraoka, Y. 1994. A beat tracking system for acoustic signals of music. In *Proc. of the Second ACM Intl. Conf. on Multimedia*, 365−372.

Goto, M., and Muraoka, Y. 1995a. Music understanding at the beat level − real-time beat tracking for audio signals −. In *Working Notes of the IJCAI-95 Workshop on Computational Auditory Scene Analysis*, 68−75.

Goto, M., and Muraoka, Y. 1995b. A real-time beat tracking system for audio signals. In *Proc. of the 1995 Intl. Computer Music Conf.*, 171−174.

Goto, M., and Muraoka, Y. 1996. Parallel implementation of a beat tracking system − real-time musical information processing on AP1000 − (*in Japanese*). *Transactions of Information Processing Society of Japan* 37(7):1460−1468.

ICMAS. 1995. *Proc., First Intl. Conf. on Multi-Agent Systems*, The AAAI Press / The MIT Press.

Ishihata, H.; Horie, T.; Inano, S.; Shimizu, T.; and Kato, S. 1991. An architecture of highly parallel computer AP1000. In *IEEE Pacific Rim Conf. on Communications, Computers, Signal Processing*, 13−16.

Katayose, H.; Kato, H.; Imai, M.; and Inokuchi, S. 1989. An approach to an artificial music expert. In *Proc. of the 1989 Intl. Computer Music Conf.*, 139−146.

Kitano, H. 1993. Challenges of massive parallelism. In *Proc. of IJCAI-93*, 813−834.

Large, E. W. 1995. Beat tracking with a nonlinear oscillator. In *Working Notes of the IJCAI-95 Workshop on Artificial Intelligence and Music*, 24−31.

Maes, P., ed. 1990. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. The MIT Press.

Minsky, M. 1986. *The Society of Mind*. Simon & Schuster, Inc.

Nakatani, T.; Okuno, H. G.; and Kawabata, T. 1994. Auditory stream segregation in auditory scene analysis. In *Proc. of AAAI-94*, 100−107.

Rosenthal, D.; Goto, M.; and Muraoka, Y. 1994. Rhythm tracking using multiple hypotheses. In *Proc. of the 1994 Intl. Computer Music Conf.*, 85−87.

Rosenthal, D. 1992. *Machine Rhythm: Computer Emulation of Human Rhythm Perception*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Schloss, W. A. 1985. *On The Automatic Transcription of Percussive Music − From Acoustic Signal to High-Level Analysis*. Ph.D. Dissertation, CCRMA, Stanford University.

Shoham, Y. 1993. Agent-oriented programming. *Artificial Intelligence* 60(1):51−92.

Vercoe, B. 1994. Perceptually-based music pattern recognition and response. In *Proc. of the Third Intl. Conf. for the Perception and Cognition of Music*, 59−60.