論　文

音響信号を対象としたリアルタイムビートトラッキングシステム
——コード変化検出による打楽器音を含まない音楽への対応——

後藤　真孝[†*]　　村岡　洋一[†]

A Real-Time Beat Tracking System for Audio Signals
——Application to Drumless Music by Detecting Chord Changes——

Masataka GOTO[†*] and Yoichi MURAOKA[†]

あらまし　本論文では，打楽器音を含まない音楽音響信号に対してリアルタイムにビートを認識するビートトラッキングシステムについて述べる．ビートトラッキングは，さまざまなアプリケーションにおいて有効なだけでなく，計算機による音楽認知モデルを構築する第1段階としても重要である．従来研究の多くはMIDI信号か打楽器音を含む音響信号を対象にしており，打楽器音を含まない複数の楽器による音響信号を扱うのは困難であった．本研究では，音源分離が困難な音響信号に対して音楽的判断をするために，周波数スペクトルの時間軸をビートの時刻で分割して解析することで，コード名にシンボル化しなくてもコード変化を検出できる手法を提案する．これにより，各ビートが強拍か弱拍かを判定でき，ビートのさまざまな解釈の可能性の中から適切なものを選択できる．並列計算機上に実装したシステムに対し，ポピュラー音楽のコンパクトディスクからサンプリングした多様な楽器音や歌声の含まれた音響信号を入力して実験した．その結果，40曲中34曲に対して適切なビートを得ることができ，コード変化を検出しない従来システムに比べて認識精度が大幅に向上した．

キーワード　音楽情報処理，ビートトラッキング，音楽理解，コード変化検出，マルチエージェント

1. まえがき

本研究では，人間が音楽に合わせて手拍子を打つように曲のビート（四分音符）の位置をリアルタイムに認識するビートトラッキングシステムを実現する．ビートトラッキングは，音楽に同期した画像生成や照明制御，メディア編集システムなどのさまざまなアプリケーションにおいて有効なだけでなく[1]，音響信号に対する音楽聴取過程を計算機上で実現する第1段階としても重要である．西洋音楽において，ビートは曲を構成するさまざまな時間的概念の基本単位であり，音楽を理解する上で欠かすことのできない要素の一つである．たとえすべての楽音を完全に音源分離して同定できない人でも，音楽に合わせて手拍子を打つことは比較的容易にできる．そこで我々は，まずビートの知覚を計算機上で実現した後に，より高次の音楽構造を理解する方向へと研究を進めていく．

従来のビートトラッキングに関する研究の多くは，MIDI信号や発音時刻（音が鳴り始める時刻；onset time）を対象にしていた[2]～[9]．しかし，複数種類の楽器音を含む音響信号を自動採譜する（MIDI信号に変換する）のは一般に難しく，我々が普段コンパクトディスク（CD：compact disc）を再生して聞くような音響信号を入力とすることはできなかった．独奏や少数の楽器で演奏された音響信号を対象にしたビートトラッキングの研究も報告されているが[10],[11]，自動採譜の一部として研究されているものが中心でリアルタイムに処理できなかった．一方，我々はこれまで，打楽器音を含む複数の楽器で演奏されたポピュラー音楽の音響信号を対象にした，リアルタイムビートトラッキングシステムを実現してきた[1],[12],[13]．しかし，このシステムは打楽器音（特にバスドラムとスネアドラム）が入力に含まれることを前提としていたため，打楽器音を含まない曲に対してはほとんど有効でなかった．

本論文では，打楽器音を含む音楽を対象にした従来

† 早稲田大学理工学部，東京都
School of Science and Engineering, Waseda University, Tokyo, 169-8555 Japan
* 日本学術振興会特別研究員

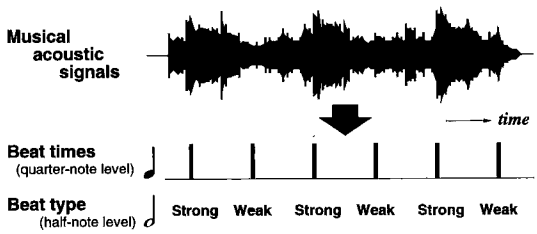


図1 ビートトラッキング問題
Fig. 1 Beat tracking problem.

システムをどのように拡張すれば，打楽器音(注1)を含まない音楽を対象にしたシステムを実現できるかについて述べる．本システムは，市販のCDなどから得た複数種類の楽器音を含む音響信号（対象テンポ：61～120 M.M.(注2)）を入力とし，四分音符に相当するビートの存在する時刻（ビート時刻；beat time）をリアルタイムに認識する．このビート時刻の系列を四分音符レベルと呼ぶ．本システムは更に高次の音楽構造として，入力曲が4/4拍子であることを前提にそのビートが強拍か弱拍か（ビートタイプ；beat type）(注3)も判断する．これは，二分音符レベルでビートを理解していることに相当する（図1）．

本システムでは，a) 各ビートのビートタイプ（強拍か弱拍か）を判定し，b) ビートのさまざまな解釈の可能性の中から適切なものを選択するために，音楽的判断を行う必要がある．そこで，音源分離が困難な音響信号に対してこのような判断をするために，周波数スペクトルからコード名にシンボル化しなくても直接コードの変化だけを検出できる手法を提案する．この検出結果を利用すれば，「コードは小節の頭でより変わりやすい」というヒューリスティックな音楽的知識により a) の判定ができ，「コードはビートの位置でより変わりやすい」という知識により b) の選択ができる．

以下，**2.** において本システムの実現上の課題について述べ，解決法としてコード変化度を利用した音楽的判断を提案する．そして **3.** で本システムの処理の中心となる周波数解析とビート予測について処理手順を説明する．**4.** では実装した本システムによる実験結果を示し，改良前の従来システムに比べ，打楽器音を含まない曲に対する認識精度が向上したことを確認する．最後に **5.** で結論と今後の課題を述べる．

2. 実現上の課題と解決法

音楽音響信号に対するビートトラッキングを実現するには，音響信号中のビートの手掛りの抽出やビートのさまざまな解釈の可能性の評価，ビートタイプの判定や解釈の選択といった音楽的な判断が課題となる[14]．従来の打楽器音を含む音楽を対象にしたビートトラッキングシステム[1],[12]～[14]では，これらの課題を以下のように解決してきた．まずビートの手掛りとしては，周波数解析により検出した発音時刻を用い，その自己相関関数・相互相関関数によりビート時刻を求めた[13]．次に，複数のビートの解釈の可能性を並列に調べるために，異なる戦略でビートの解釈を行う複数のエージェントを導入した[1],[12]．そして，音楽的判断を行うためにポピュラー音楽における典型的なドラムパターン（バスドラムとスネアドラムの発音パターン）を事前に登録し，音響信号中のドラム音の発音時刻を検出して照合することで，ビートタイプの判定等を実現した[12],[13]．

このような従来のシステムを，打楽器音を含まない音楽を対象にしたシステムへと拡張する上で最も困難な課題は，ドラムパターンを用いない音楽的判断の実現方法である．入力に打楽器音が含まれない場合には，ビートタイプを判定するための唯一の手掛りであったドラムパターンと照合することができない．更に，複数楽器音を含む音響信号を音符レベルにシンボル化するのは困難であるため，MIDI信号を対象とする研究などで使用できる音楽的知識の多くは直接適用できない．

そこで本研究では，音符レベルへのシンボル化に頼らないコード変化検出手法と，その結果に基づいてビートトラッキングのための音楽的判断を行う方法を提案する．以下，**2.1** と **2.2** においてこれらを順に説明する．

2.1 コード変化検出手法

発音時刻に基づいて求めたビート時刻を活用することで，周波数スペクトルから音符やコード名などにシンボル化せずに，直接コード変化の度合（コード変化度）を求める．これは，たとえコード名がわからない人でもコードの変化はわかる，という現象に着目した手法である．

コードの構成音の倍音も含めたすべての周波数成分

(注1)：本論文ではポピュラー音楽で多用されるバスドラムとスネアドラムを想定する．
(注2)：Mälzel's Metronome：1分間の拍数（四分音符/分）を表す．
(注3)：各小節において1拍目と3拍目を強拍，2拍目と4拍目を弱拍と定義する．

を考えると(注4)，コードが変わらない場合はこれらの周波数成分は比較的変化が少ないが，コードが変わる場合には大きく変化することが多い．複数楽器による音響信号中から，すべての周波数成分を正確に求めるのは一般に難しいが，ある一定の期間内で優勢な周波数成分はヒストグラムを計算することで推定できる．

そこで本手法では，曲中のビート時刻（四分音符の位置）の候補が仮に得られていることを前提に，それらの候補を用いて周波数解析することで，四分音符レベルでのコード変化度（quarter-note chord change possibility）と八分音符レベルでのコード変化度（eighth-note chord change possibility）という2種類のコード変化の度合を算出する．前者はその各四分音符の位置でコードがどれくらい変わった可能性があるかを表し，後者は八分音符の位置での可能性を表す．両者は，**2.2**で後述するように，音楽的判断において別々の目的に用いられる．

これらのコード変化度を求める手順を以下に示す．

（1） 周波数スペクトルの分割

まず，A–D変換された音響信号から高速フーリエ変換により周波数スペクトル（パワースペクトル）を得る（**3.1.1**）．次に，四分音符レベルでのコード変化度 Cq_n を求めるために，こうして得られた周波数スペクトルの時間軸を各ビート時刻において分割して，分割後のスペクトル Sq_n を求める．

$$Sq_n = \{p(t,f)\},\ Tq_n \leqq t < Tq_{n+1} \tag{1}$$

ここで，n 番目のビート時刻を Tq_n とし，時刻 t，周波数 f(注5)における周波数スペクトルのパワーを $p(t,f)$ とする．

一方，八分音符レベルでのコード変化度 Ce_n を求める際には，ビート時刻 Tq_n から内挿された八分音符の時刻 Te_n で分割し，分割後のスペクトル Se_n を求める．

$$Se_n = \{p(t,f)\},\ Te_n \leqq t < Te_{n+1} \tag{2}$$

$$Te_n = \begin{cases} Tq_{n/2} & (n \bmod 2 = 0) \\ (Tq_{(n-1)/2} + Tq_{(n+1)/2})/2 & (n \bmod 2 = 1) \end{cases} \tag{3}$$

（2） ヒストグラムの作成

Sq_n，Se_n 内において，時間軸方向にパワーを合計してヒストグラム $Hq_n(f)$，$He_n(f)$（以下，$Hi_n(f)$ $(i = q, e)$ のようにまとめて記述する）をそれぞれ作成する．

$$Hi_n(f) = \sum_{t=Ti_n+Gi_n}^{Ti_{n+1}-Gi_n} p(t,f) \tag{4}$$

Gi_n は発音直後のノイズ成分や非定常な音高の影響を抑えるためのマージンであり，現在の実装では $Gi_n = (Ti_{n+1} - Ti_n)/5$ とした．

（3） 支配的な音高の検出

まず，$Hi_n(f)$ の周波数軸方向のピーク $Ki_n(f)$ を式(5)により求める．但し現在の実装では，10 Hz から 1 kHz までの帯域のピークだけを考慮する．

$$Ki_n(f) = \begin{cases} Hi_n(f) \\ \quad (Hi_n(f) \geqq Hi_n(f \pm 1)) \\ 0 \quad \text{(otherwise)} \end{cases} \tag{5}$$

これらのピークは分割後のスペクトル内で支配的な音高であり，コードやメロディーの周波数成分に相当することが多い．

次に，休符等により無音区間があったときにノイズをピークと誤検出せずに，その区間は直前のコードが続いていると仮定するために，次の処理を行う．$Ki_n(f)$ 中の最大値を mi_n とし，

$$Mi_n = \max(mi_n, \Phi Mi_{n-1}) \tag{6}$$

を用いて相対的な時間変化を考慮した変換後のピーク $Qi_n(f)$ を求める．

$$Qi_n(f) = \mathrm{clip}(\Psi Ki_n(f) / Mi_n) \tag{7}$$

$$\mathrm{clip}(x) = \begin{cases} 0 & (x < 0) \\ x & (0 \leqq x \leqq 1) \\ 1 & (1 < x) \end{cases} \tag{8}$$

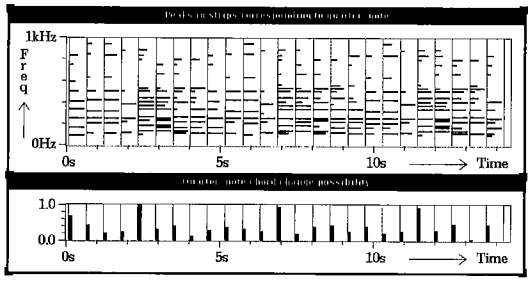
式中の定数は $\Phi = 0.99$，$\Psi = 5$ とした．最後に，ピークの合計値が大きく減少する無音に相当する区間で，直前のピークが連続しているとみなすように，式(9)により分割後のスペクトル内における最終的なピーク $Pi_n(f)$ を求める．

$$Pi_n(f) = \begin{cases} Qi_n(f) & \left(\sum_f Qi_n(f) \right. \\ & \left. \quad \geqq \Theta \sum_f Pi_{n-1}(f)\right) \\ Pi_{n-1}(f) & \text{(otherwise)} \end{cases} \tag{9}$$

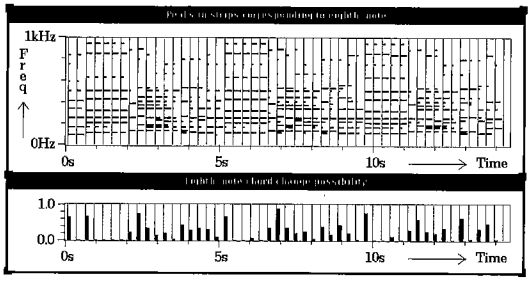
式中の定数は $\Theta = 0.1$ とした．これにより，一時的な

(注4)：実際の曲中では，メロディー等他の音の周波数成分も含めて考える．これらはコードと調和する音高であることが多い．

(注5)：t，f は整数値をとり，$1t$，$1f$ はそれぞれフレームシフト時間，周波数分解能に相当する．



(a) 四分音符レベルでのコード変化度



(b) 八分音符レベルでのコード変化度

図2 分割された周波数スペクトル内のピークとコード変化度の例

Fig. 2 Examples of peaks in sliced frequency spectrum and of chord change possibilities.

無音区間の後に常にコード変化度が大きくなってしまう状況を回避できる．

（4） 隣接ピークの比較

時間的に隣接するピーク $Pi_{n-1}(f)$，$Pi_n(f)$ を比較することで，コード変化度を求める．両者の分割時刻 Ti_n でコードが変化すると，$Pi_{n-1}(f)$ に比べ $Pi_n(f)$ では異なるピークが多く生じる可能性が高い．そこで，ピークの増加量 ci_n の相対的な時間変化から，四分音符レベルでのコード変化度 Cq_n と八分音符レベルでのコード変化度 Ce_n を式(10)により求める．

$$Ci_n = ci_n / di_n \tag{10}$$

$$ci_n = \sum_f \mathrm{clip}(Pi_n(f) - Pi_{n-1}(f)) \tag{11}$$

$$di_n = \max(ci_n, \Phi\, di_{n-1}) \tag{12}$$

実際に得られたコード変化度の例を図2に示す．細い縦線はビートの解釈に基づいてスペクトルを分割した時刻（Tq_n あるいは Te_n）である．本例では，(a)では一番左から四つごとに小節の先頭，(b)では一番左から二つごとにビート時刻（四分音符の時刻）となっている．(a)と(b)のそれぞれにおいて，上段の個々の横線の長さが分割後のスペクトル内におけるピーク $Pi_n(f)$ の大きさを表し，下段の太い縦線の高さがコード変化度 Ci_n を表す．

2.2 コード変化に基づく音楽的判断

2種類のコード変化度を用いて，a)各ビートのビートタイプ（強拍か弱拍か）を判定し，b)複数のエージェントによるビートのさまざまな解釈の中から適切なものを選択する．そのために，コードがどのようなときに変化しやすいかを表す，次の2種類のヒューリスティックな音楽的知識を導入する．

（1） コードは他の位置よりも小節の頭で変わりやすい．つまり，四分音符レベルでのコード変化度は，強拍（小節の頭）の位置の方が弱拍の位置よりも大きい傾向がある．

（2） コードはビートからずれた位置よりもビートの位置で変わりやすい．つまり，八分音符レベルでのコード変化度は，ビートの位置の方が八分音符ずれた位置よりも大きい傾向がある．

a)のビートタイプの判定は知識（1）を利用して行う．まず，四分音符レベルでのコード変化度 Cq_n の一つおきの過去の傾向

$$Uq_n = \lambda_1 Uq_{n-2} + \lambda_2 Cq_n \tag{13}$$

を求める．式中の定数は $\lambda_1 = 0.99, \lambda_2 = 0.2$ とした．知識（1）のように，一つおきに出現する強拍で Cq_n が大きい傾向にあれば，その過去の値を集計している Uq_n も大きくなる．そこで，$Uq_n - Uq_{n-1} > \mu_q$ のとき強拍の位置を Tq_n と判断し，その判断の信頼度を

$$Lq_n = \mathrm{clip}(Uq_n - Uq_{n-1}) \tag{14}$$

と定義する．但し，しきい値は $\mu_q = 0.3$ とした．そして，この強拍の位置を基準に，例えば Tq_n が強拍のときは Tq_{n+1} が弱拍であるといったように強拍と弱拍が交互に現れる前提を利用して，予測したビート時刻のビートタイプを判定する．

b)の解釈の選択は知識（2）を利用して行う．**3.2**で述べるように，本システムでは各エージェントによる解釈の確信度に基づいて適切な解釈を選択する．そこで，八分音符レベルでのコード変化度に基づいて，$Tq_n(= Te_{2n})$ がビートの位置であると判断する信頼度を

$$Le_n = \lambda_1 Le_{n-1} + \lambda_2 (Ce_{2n} - Ce_{2n+1}) \tag{15}$$

と定義する．知識（2）より Le_n の値が高ければそのコード変化度を計算した際のビート時刻が適切であっ

たとみなせる．そこで，Le_n が高いほど解釈の確信度が高くなるように評価することで，適切なビート時刻をもつ解釈が選択されやすくなる．具体的な確信度の評価は他の要素も影響するため **3.2.1**（4）で後述する．

3. 処理モデル

本システムの処理の流れを図 3 に示す．まず，音響信号の入力（A–D conversion）で A–D 変換された音響信号に対して周波数解析（frequency analysis）を行う．発音時刻の検出器（onset-time finders）が各周波数帯域ごとの発音時刻を求め，発音時刻のベクトル化器（onset-time vectorizers）がそれらをベクトル化して発音時刻ベクトル（onset-time vectors）にまとめる．次に，ビート予測（beat prediction）の各エージェントが，過去に得られた発音時刻ベクトルからビートの間隔（inter-beat interval）を求め，次のビート時刻の予測とビートタイプの判定を行う．この際，コード変化検出器（chord change checkers）から得た情報を用いて音楽的に判断する．そして，全エージェントの解釈を統合してビート情報（ビート時刻，ビートタイプ，現在のテンポからなる）を生成する．最後にビート情報の出力（beat information transmission）が，ネットワークを通じて他のアプリケーションプログラムへとビート情報を送信する．

以下では，主要な処理である周波数解析とビート予測について述べる．

3.1 周波数解析

周波数解析により，入力音響信号から周波数スペクトルと N 次元の発音時刻ベクトル（現在の実装では 7 次元）の系列を求める（図 4）．発音時刻ベクトルの各次元は，分割された各周波数帯域での発音時刻に対応する．このようにベクトル表現することで，すべての帯域を同時に考慮できる．

3.1.1 高速フーリエ変換（FFT）

A–D 変換された音響信号に対してハニング窓を用いた FFT を行い，周波数スペクトルを得る．FFT は，観測区間（WindowSIZE）を時間軸方向に単位時間（ShiftSIZE）ずつずらしながら適用する．

現在の実装では，音響信号を標本化周波数 22.05 kHz，量子化ビット数 16 bit，モノラルで A–D 変換し，2 種類の FFT を計算する．一方は発音時刻を検出するための FFT で，WindowSIZE は 1024 点，ShiftSIZE は 256 点である．従って周波数分解能は 21.53 Hz，フレームシフト時間（1 フレーム時間(注 6)：1 ft）は 11.61 ms となる．もう一方はコード変化を検出するための FFT

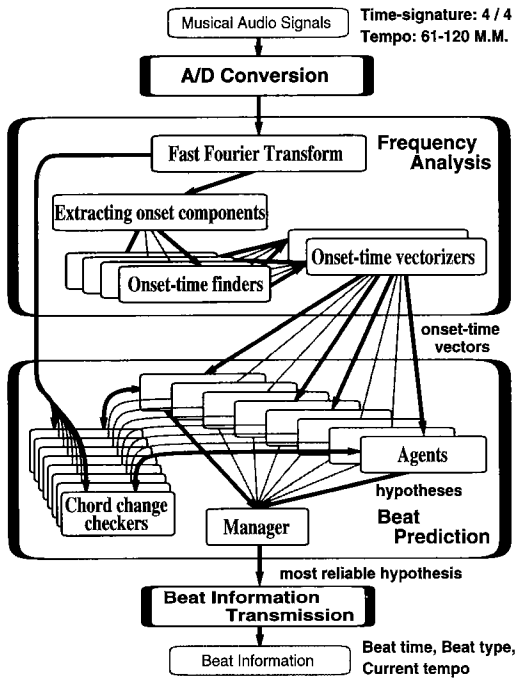


図 3 処理の流れ
Fig. 3 Overview of our beat tracking system.

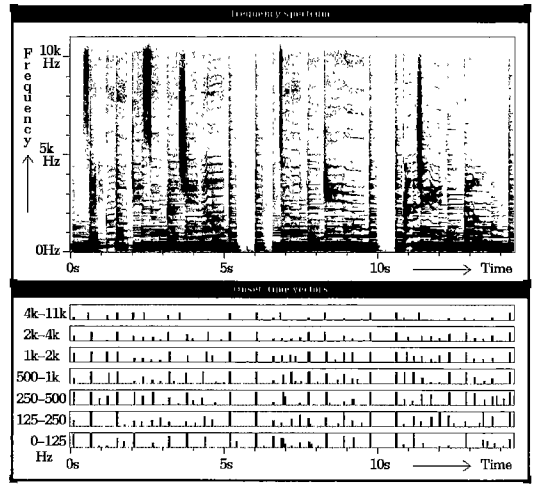


図 4 周波数スペクトルと発音時刻ベクトル系列の例
Fig. 4 Example of a frequency spectrum and an onset-time vector sequence.

(注 6)：フレーム時間を本システムのすべての処理の時間単位とする．

で，11.025 kHz にダウンサンプリングした信号に対し WindowSIZE は 1024 点，ShiftSIZE は 128 点で計算する．この場合，周波数分解能は 10.77 Hz，フレームシフト時間は 1 ft となる．

3.1.2 立上り成分の抽出

式 (16) を満たす周波数成分 $p(t,f)$ を立上り成分として抽出し，その立上りの度合（パワーがどれくらい急激に増加しているか）$d(t,f)$ を式 (18) により求める．

$$\min(p(t,f),p(t+1,f)) > pp \tag{16}$$

$$pp = \max(p(t-1,f),p(t-1,f\pm 1)) \tag{17}$$

$$d(t,f) = \begin{cases} \max(p(t,f),p(t+1,f)) - pp \\ (\text{if condition (16) is fulfilled}) \\ 0 \qquad\qquad (\text{otherwise}) \end{cases} \tag{18}$$

3.1.3 発音時刻の検出器

複数の発音時刻の検出器（現在の実装では 7 個）が，それぞれ異なる帯域（0〜125 Hz，125〜250 Hz，250〜500 Hz，500 Hz〜1 kHz，1〜2 kHz，2〜4 kHz，4〜11 kHz）の発音時刻を求める．まず各時刻における立上りの度合の合計 $D(t)=\sum_f d(t,f)$ を求め（$\sum_f$ の f を対象とする帯域に制限する），次にその時間変化のピーク時刻とピーク値を，平滑化微分を用いたピーク検出により求める．そして，こうして得られたピーク時刻を発音時刻とする．

3.1.4 発音時刻のベクトル化器

発音時刻のベクトル化器が，すべての発音時刻の検出器の結果をまとめて発音時刻ベクトルの系列へと変換する．現在の実装では，7 個の発音時刻の検出器の結果を，3 種類の発音時刻のベクトル化器が周波数軸方向にそれぞれ異なった重み付け（全帯域・低域中心・中域中心の 3 種類(注7)）をしてベクトル化する．この重み付けの種類を注目周波数帯域（frequency focus type）と呼ぶ．低域中心は主にベースやピアノの低音部などに注目するために，中域中心は主旋律などに注目するために用意した．こうして得られた 3 種類の 7 次元発音時刻ベクトルの系列は，ビート予測のエージェントへと送られる．

3.2 ビート予測

複数のエージェントが，発音時刻ベクトルの系列をそれぞれ異なった戦略により解釈し，次のビート時刻を予測する．各エージェントが出力する解釈の結果は，次のビート時刻，そのビートタイプ，ビートの間隔の三つからなり（図 5），その解釈がどれくらい適切であるかを解釈の確信度として自己評価する．この結果は解釈の統合へと集められ，最も確信度の高い解釈に基づいて出力が決定される．

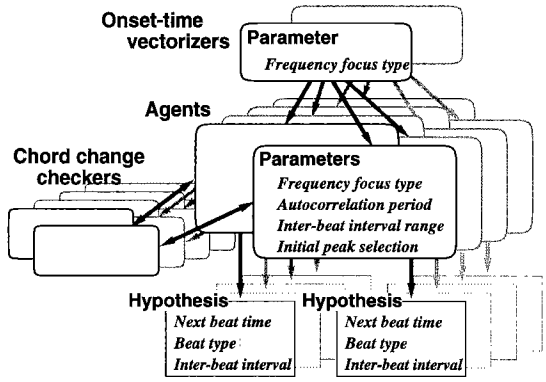


図 5 発音時刻のベクトル化器とエージェント，コード変化検出器の関係
Fig. 5 Relations between onset-time vectorizers, agents, and chord change checkers.

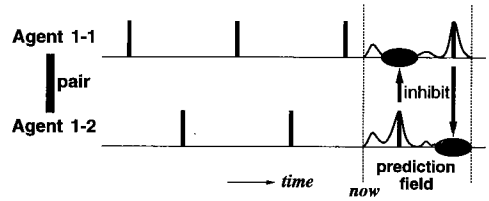


図 6 エージェント間の予測場を介した相互作用
Fig. 6 Agent interaction through a prediction field.

すべてのエージェントは二つずつの組に分けられる（現在の実装では，エージェントの数は 12 個 6 組である）．同じ組の二つのエージェントは，予測場（prediction field）を介して相互作用しながら，同じビートの間隔で互いにビートの間隔の 1/2 ずれた時刻を予測する（図 6）．予測場とは次のビートが来ることが期待される度合の分布であり，予測場中の各ピークは次のビート時刻の候補とみなされる．両エージェントは相手の予測場において，自分が解釈したビート時刻に対応する位置の周辺を抑制し合う（ビートの間隔の 3 分の 1 の範囲を 0 にする）ことで相互作用する．

エージェントは，ビートを解釈する際の戦略を決定

(注 7)：具体的に 0〜125 Hz から 4〜11 kHz まで低い帯域から順に重み係数を 7 個列挙すると，全帯域 [1.0, 1.0, 0.9, 0.8, 0.7, 0.6, 0.5]，低域中心 [1.0, 1.0, 0.6, 0.1, 0.1, 0.1, 0.1]，中域中心 [0.1, 0.1, 1.0, 1.0, 1.0, 0.9, 0.1] である．

表1　戦略を決定するパラメータの初期設定値
Table 1　Initial settings of the strategy parameters.

組-エージェント	注目周波数帯域	自己相関期間	ビート間隔検出範囲	初期ピーク選択
1-1	全帯域	500 ft	43-85 ft	primary
1-2	全帯域	500 ft	43-85 ft	secondary
2-1	全帯域	1000 ft	43-85 ft	primary
2-2	全帯域	1000 ft	43-85 ft	secondary
3-1	低域	500 ft	43-85 ft	primary
3-2	低域	500 ft	43-85 ft	secondary
4-1	低域	1000 ft	43-85 ft	primary
4-2	低域	1000 ft	43-85 ft	secondary
5-1	中域	500 ft	43-85 ft	primary
5-2	中域	500 ft	43-85 ft	secondary
6-1	中域	1000 ft	43-85 ft	primary
6-2	中域	1000 ft	43-85 ft	secondary

する以下の四つのパラメータをもつ（図5）．現在の実装でのパラメータの初期設定値を表1に示す．

（1）　注目周波数帯域

どの発音時刻のベクトル化器から結果を受け取るかを定める．この値はそれぞれベクトル化での重み付けの種類に対応する．これはその帯域に選択的に注意を向けて音楽を聞くことに相当する．

（2）　自己相関期間

後述する発音時刻ベクトル系列の自己相関関数を計算する際の，窓の大きさを定める．この値が大きいほど，より古い発音時刻まで考慮される．

（3）　ビート間隔検出範囲

ビートの間隔がとり得る値の範囲を表す．自己相関関数の結果からピークを選ぶ際に，この範囲内のみに限定する．

（4）　初期ピーク選択

primary か secondary をとり，予測場からのピークの選び方を定める．このパラメータにより，異なる解釈が得られやすくなる．

以下では各エージェントの動作（**3.2.1**），エージェントが音楽的判断をするための手掛りを提供するコード変化検出器（**3.2.2**），全エージェントの解釈の統合（**3.2.3**）について順に述べる．

3.2.1　エージェントの動作

各エージェントは，以下の四つの処理によってビートの解釈を生成し，結果をコード変化検出器と解釈の統合へ送る．

（1）　ビートの間隔の決定

発音時刻ベクトルの自己相関関数を計算することで，各周波数帯域の発音時刻の間隔を同時に考慮しながらビートの間隔を求める．ベクトル化した窓付き正規化

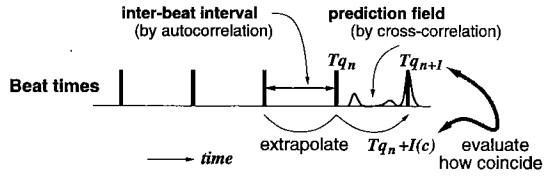


図7　次のビート時刻の予測
Fig. 7　Predicting the next beat.

自己相関関数 $Ac(\tau)$ を式(19)により求める．

$$Ac(\tau)=\frac{\sum_{t=c-W}^{c}\mathrm{win}(c-t,W)\,(\vec{o}(t)\cdot\vec{o}(t-\tau))}{\sum_{t=c-W}^{c}\mathrm{win}(c-t,W)\,(\vec{o}(t)\cdot\vec{o}(t))} \tag{19}$$

ここで，$\vec{o}(t)$ は時刻 t における N 次元発音時刻ベクトルであり，c は現在時刻，W は自己相関期間（戦略パラメータ）とする．窓幅 s の窓関数 $\mathrm{win}(t,s)$ は式(20)で与えられる．

$$\mathrm{win}(t,s)=1.0-0.5\,\frac{t}{s} \tag{20}$$

ビートの間隔は，$Ac(\tau)$ においてビート間隔検出範囲（戦略パラメータ）内でピーク値をとる τ として得られる．但し，この結果は一時的に不安定になることがあるため，こうして得られたビートの間隔のヒストリを作る．ヒストリ中に最もよく現れる間隔を用いることで，正しい間隔が安定して得られるようになる．また確信度が高いとき（（4）で述べる確信度が $R>0.55$ のとき）には，ビート間隔検出範囲を現在の間隔の近傍に制限し，現在の適切な間隔を維持する．

（2）　次のビート時刻の予測

次のビート時刻を予測するために予測場を形成する（図7）．予測場は，各時刻における $\vec{o}(t)$ の全次元の要素の合計値 $O(t)$ と，（1）で得られたビートの間隔をもつ仮のビート時刻の系列 $B_m(t)$[注8] との窓付き相互相関関数 $Cc(\tau)$ の結果として式(21)により求める．

$$Cc(\tau)=\sum_{t=c-V}^{c}\Big(\mathrm{win}(c-t,V)\,O(t)\sum_{m=1}^{\Omega}\delta(t-B_m(c+\tau))\Big) \tag{21}$$

(注8)：m が一つ増すごとにビートの間隔分ずつ過去にさかのぼっていく．

$$B_m(t) = \begin{cases} t - I(t) & (m = 1) \\ B_{m-1}(t) - I(B_{m-1}(t)) & (m > 1) \end{cases} \tag{22}$$

$$\delta(x) = \begin{cases} 1 & (x = 0) \\ 0 & (x \neq 0) \end{cases} \tag{23}$$

但し，時刻 t におけるビートの間隔を $I(t)$ とし，相互相関期間を $V = \Omega I(c)$ とする．式中の定数は $\Omega = 12$ とした．こうして，τ が 0 から $I(c)-1$ までの $Cc(\tau)$ の値として予測場を求める．

得られた予測場が同じ組のエージェントによって前述したように抑制された後，エージェントは予測場のピークの中から次のビート時刻を選択する．その際，解釈の確信度が低いとき（(4) で述べる評価値が $R_o < 0.2$ のとき）には，初期ピーク選択（戦略パラメータ）の値に従ってピークを選ぶ．primary のときには最大ピークを，secondary のときには 2 番目に大きいピークを次のビート時刻とする．その後は，前回のビート時刻にビートの間隔を加えた時刻に最も近いピークを次のビート時刻とする．

(3) ビートタイプの判定

コード変化検出器から受け取った四分音符レベルでのコード変化度に基づいて，**2.2** で述べたようにビートタイプを判定する．

(4) 解釈の確信度の自己評価

以上のように得られた解釈に対して，確信度 R を式 (24) により評価する．

$$R = \mathrm{clip}(\lambda_o R_o + \lambda_e R_e + \lambda_q R_q) \tag{24}$$

ここで，R_o は発音時刻に基づいて適切であるかを判断した評価値，R_e は八分音符レベルでのコード変化に基づいて適切であるかを判断した評価値，R_q は四分音符レベルでのコード変化に基づいて適切であるかを判断した評価値である．式中の定数は $\lambda_o = 0.5, \lambda_e = 0.4, \lambda_q = 0.1$ とした．

各評価値は以下のように求める．まず R_o は，発音時刻に基づいて自己相関・相互相関により予測されたビート時刻 Tq_{n+1} と，過去のビート時刻 Tq_n から外挿された時刻 $Tq_n + I(c)$ との一致度 r_o に応じて前回の評価値 R'_o を増減することで求める（図 7）．

$$R_o = \mathrm{clip}(R'_o + \Lambda r_o) \tag{25}$$

$$r_o = \eta\left(\left|\frac{Tq_{n+1} - (Tq_n + I(c))}{I(c)}\right|\right) \tag{26}$$

$$\eta(x) = \begin{cases} 1 - \mathrm{Z}_1 \dfrac{x}{\mathrm{Z}_2} & (0 \leqq x < \mathrm{Z}_2) \\ -1 & (\mathrm{Z}_2 \leqq x) \end{cases} \tag{27}$$

式中の定数は $\Lambda = 0.04, \mathrm{Z}_1 = 0.8, \mathrm{Z}_2 = 0.125$ とした．次に R_e は，**2.2** で述べたように Le_n を用いて式 (28) により求める．

$$R_e = \begin{cases} -1 & (Le_n < -1) \\ Le_n & (-1 \leqq Le_n \leqq 1) \\ 1 & (1 < Le_n) \end{cases} \tag{28}$$

最後に R_q は，Lq_n から式 (29) により求める．

$$R_q = \mathrm{clip}\left(\frac{Lq_n - \mathrm{Y}}{1 - \mathrm{Y}}\right) \tag{29}$$

式中の定数は $\mathrm{Y} = 0.5$ とした．

3.2.2 コード変化検出器

コード変化検出器は **2.1** で述べた処理を行う．1 対 1 に対応するエージェントから受け取ったビートの解釈に基づいて周波数スペクトルを分析し，コード変化度を同じエージェントへ送り返す（図 5）．

3.2.3 解釈の統合

全エージェントから集めた解釈の結果を，ビートの間隔が同じでビート時刻が近いもの同士グルーピングする．そして各グループにおいてグループ内の全解釈の確信度を合計し，最も合計値の高い支配的なグループを選ぶ．その際，ビート時刻の一時的な変動などにより，本来支配的であるべきグループがばらばらになり，誤ったグループが選択される可能性がある．そこで，同じグループにするためのビート時刻の近さの基準を徐々に狭めながら 3 段階グルーピングと選択を繰り返すことで，適切な解釈が選択されるようにする．最終的に，こうして得られた最も支配的なグループの解釈に基づいて出力を決定する．

最終的な出力のビートタイプは，その信頼度 Lq_n を用いてここで再決定される．個々のエージェントが判定したビートタイプは，コード変化度の誤検出や一時的に不規則なコード変化などにより，適切でないことがある．そこで，選択された解釈において Lq_n が高いときだけそのビートタイプを信頼し，低いときは強拍と弱拍が交互に現れるように再判定する．Lq_n が高いか否かを判断するしきい値は動的に変動し，時間が経つと減衰する．現在のしきい値より高い Lq_n が現れるとしきい値をその値に設定し，以後しきい値より低い Lq_n のビートタイプは無視する．

表2　コード変化検出手法の実験結果
Table 2　Results of testing chord change detection.

	Cq_n		Ce_n	
	変化	無変化	表拍（変化，無変化）	裏拍
平均	0.73	0.01	0.56 (0.81, 0.30)	0.03
標準偏差	0.22	0.02	0.29 (0.18, 0.08)	0.05
最大	1.00	0.10	1.00 (1.00, 0.48)	0.21
最小	0.28	0.00	0.12 (0.37, 0.12)	0.00

4. 実験結果

以上の処理モデルを富士通の分散メモリ型並列計算機 AP1000 [15] 上に実装し，コード変化検出手法が有効に機能していることを確認する実験（**4.1**），改良前の従来システムと比較して打楽器音を含まない音響信号に対する認識精度が向上したことを確認する実験（**4.2**），2種類のコード変化度がビート時刻の認識やビートタイプの判定に効果があることを確認する実験（**4.3**）を行った．

4.1 コード変化検出手法に関する実験

コード変化検出手法の基本的な能力を確認するために，ランダムに推移するコード進行を用いて実験した．入力にはコード変化が100回起きる101個のコードの進行を用い，その個々のコードは，ルートの音名がA, A♯, B, C, C♯, D, D♯, E, F, F♯, G, G♯の12種類，それぞれの構成がMajor Triad, Minor Triad (m), Dominant 7th Chord (7), Minor 7th Chord (m7), Major 7th Chord (M7) の5種類の計60種類から，同じものが連続しないようにランダムに決定した．シンセサイザによるピアノ音を用い，コードのルートの音高の基本周波数は110 Hzから208 Hzの範囲で，すべて基本形で演奏した．また，コードが変化しないときの結果も確認するために，同じコードを2回ずつ四分音符（テンポ100 M.M. で600 ms）で演奏した．

実験した結果，適切なビート時刻が与えられたときの，各条件における四分音符レベルと八分音符レベルでのコード変化度 Cq_n, Ce_n の統計値を表2に示す．Cq_n の変化，無変化は，それぞれコードが変化した場合と変化しなかった場合を表す．この結果から，コードが変化した場合に Cq_n が高く，適切に検出できていることがわかる．一方，Ce_n の表拍，裏拍は，それぞれ四分音符の頭（ビート時刻）の場合（$n \bmod 2 = 0$）と八分音符ずれた時刻の場合（$n \bmod 2 = 1$）を表す．但し，表拍の場合にはコードが変化する場合としない場合が交互に起こるので，それらを別々に分析した結果も（ ）内に示した．この結果から，Ce_n でもコード変化を適切に検出できていることがわかる．表拍のコード無変化時が裏拍の場合より高い傾向にあるのは，表拍ではピアノが打鍵されるのに対し，裏拍では常に表拍からの持続音になっているためと考えられる．

4.2 改良前の従来システムとの比較実験

本システムの認識精度と，文献[13]の時点での改良前のシステムの認識精度とを比較する実験を行った．実験には，市販のCDからサンプリングしたモノラルの音響信号を用いた．入力の前提を満たす，4/4拍子で打楽器音が含まれない(注9)テンポがほぼ一定なポピュラー音楽40曲の，最初の1〜2分間を入力して実験した．これらの曲はテンポが62〜116 M.M. と広い範囲から選び，28アーティストによって演奏されたものである．

実験の結果，従来システムでは40曲中2曲(5%)(注10)において正しいビートを認識したのに対し，本システムでは40曲中34曲（85%）において正しいビートをリアルタイムに出力できた．本実験では，適切なビート時刻とビートタイプをともに出力した曲を正しいと判定した．正しかったどの曲も最初はビート時刻を認識できてもビートタイプは判定できないが，演奏が始まって10数小節以内にはビートタイプも正しく出力していた．以上から，従来システムでは適切に扱うことが困難であった打楽器音を含まない音響信号に対して，認識精度が向上したことが確認された．

本システムが誤認識した曲の多くでは，音数が非常に少なかったり，テンポが局所的に変動したりして，ビート時刻が正しく得られていなかった．そのためコード変化も適切に検出できず，正しい解釈が出力されていなかった．また，ビート時刻は正しいがビートタイプだけを誤る曲もあった．

4.3 コード変化度の効果を確認する実験

四分音符レベルでのコード変化度 Cq_n が実際にビートタイプの判定に効果があったか，八分音符レベルでのコード変化度 Ce_n が実際にビート時刻の認識に効果があったか，を確認する実験を行った．Cq_n と Ce_n をともに用いる場合（通常の状態），一方だけ用いる

(注9)：あるいは，曲中で打楽器音が演奏されていない部分．例えば，曲の後半になって初めて打楽器音が鳴り始める曲の前半など．
(注10)：継続的に正しいビートはとれないが数秒間は正しくとれた曲も含めると40曲中15曲（37.5%）となる．しかし，本システムの認識精度では，このような一時的にしか正しい出力をしなかった曲は誤りと判定し，一度正しいビートを得た後は最後まで継続して正しい出力をしたものだけを正解と判定した．

表3 コード変化度の効果確認実験の結果
Table 3 Results of testing the effects of chord change possibilities.

コード変化度	条件（用いる：○，用いない：×）			
四分音符レベル（Cq_n）	×	○	×	○
八分音符レベル（Ce_n）	×	×	○	○
ビート時刻のみの認識精度	12曲（30.0%）	17曲（42.5%）	31曲（77.5%）	35曲（87.5%）
ビートタイプも含めた認識精度	0曲（0.0%）	17曲（42.5%）	0曲（0.0%）	34曲（85.0%）

（全40曲）

場合，全く用いない場合において，ビート時刻のみの認識精度とビートタイプも含めた認識精度（ビート時刻とビートタイプがともに正しい曲）の両者を求めた結果を表3に示す．但し，入力曲には **4.2** と同じ40曲を用い，コード変化度を用いない場合には常に $Ci_n = 0$ となるように設定した．

表3の結果から，コード変化度を全く用いない場合と比較して，四分音符レベルでのコード変化度を用いることによりビートタイプも含めた認識精度が向上し，八分音符レベルでのコード変化度を用いることによりビート時刻の認識精度が向上したことがわかる．これにより，両者が実際に効果があったことが確認された．但し，八分音符レベルでのコード変化度を用いない場合にも，確信度 R を構成する他の評価値（R_o と R_q）があるために，場合によっては適切な解釈が選択されて正しいビート時刻が得られていた．また，ビート時刻とビートタイプの両者の認識精度を上げるためには，四分音符レベルと八分音符レベルのコード変化度がともに必要であることも確認された．

5. むすび

本論文では，音響信号を対象としたビートトラッキングシステムにおいて，打楽器音が含まれない音楽を扱う際の課題と解決法を述べ，実際のシステムの実現方法について処理モデルを中心に述べた．実験の結果，打楽器音を除く多様な楽器音や歌声の含まれたポピュラー音楽の市販CDによる音響信号に対し，40曲中34曲において四分音符と二分音符のレベルでビートをリアルタイムに理解できることを確認した．

本研究では，周波数スペクトルの時間軸を仮に得られたビート時刻で分割して解析することで，シンボル化せずにコード変化を検出する手法を提案した．これは人間が必ずしも音楽をシンボルとして聞いていないことに着目した手法である．これにより，打楽器音が含まれない音響信号に対しても，各ビートが強拍か弱拍かを判定し，複数のエージェントによるビートの解釈の中から適切なものを選択することが可能になった．

今後は，他のジャンルの音楽への対応やテンポ変化への追従など入力に対する制約を減らしていく．今回提案したコード変化検出手法は，小節レベルでのビート理解などにも有効である．そこで，今後はより高次の音楽構造も理解できるようシステムを改良していく予定である．

謝辞 AP1000の実行環境を提供して頂いた富士通研究所並列処理研究センターに感謝する．また，本論文に関し有益な御意見を頂いた査読者の方々に感謝する．

文献

[1] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," Proc. the Second ACM Intl. Conf. on Multimedia, pp.365–372, 1994.
[2] R.B. Dannenberg and B. Mont-Reynaud, "Following an improvisation in real time," Proc. Intl. Computer Music Conf., pp.241–248, 1987.
[3] P. Desain and H. Honing, "The quantization of musical time: A connectionist approach," Computer Music Journal, vol.13, no.3, pp.56–66, 1989.
[4] P.E. Allen and R.B. Dannenberg, "Tracking musical beats in real time," Proc. Intl. Computer Music Conf., pp.140–143, 1990.
[5] A. Driesse, "Real-time tempo tracking using rules to analyze rhythmic qualities," Proc. Intl. Computer Music Conf., pp.578–581, 1991.
[6] D. Rosenthal, "Emulation of human rhythm perception," Computer Music Journal, vol.16, no.1, pp.64–76, 1992.
[7] D. Rosenthal, "Machine Rhythm: Computer Emulation of Human Rhythm Perception," Ph.D. Thesis, Massachusetts Institute of Technology, 1992.
[8] P. Desain and H. Honing, "Advanced issues in beat induction modeling: Syncopation, tempo and timing," Proc. Intl. Computer Music Conf., pp.92–94, 1994.
[9] E.W. Large, "Beat tracking with a nonlinear oscillator," Working Notes of the IJCAI-95 Workshop on Artificial Intelligence and Music, pp.24–31, 1995.
[10] W.A. Schloss, "On The Automatic Transcription of Percussive Music—From Acoustic Signal to High-Level Analysis," Ph.D. Thesis, CCRMA, Stanford University, 1985.
[11] H. Katayose, H. Kato, M. Imai, and S. Inokuchi, "An

approach to an artificial music expert," Proc. Intl. Computer Music Conf., pp.139–146, 1989.
[12] M. Goto and Y. Muraoka, "Music understanding at the beat level—Real-time beat tracking for audio signals," Working Notes of the IJCAI-95 Workshop on Computational Auditory Scene Analysis, pp.68–75, 1995.
[13] M. Goto and Y. Muraoka, "A real-time beat tracking system for audio signals," Proc. Intl. Computer Music Conf., pp.171–174, 1995.
[14] 後藤真孝，村岡洋一，"ビートトラッキングシステムの並列計算機への実装—AP1000によるリアルタイム音楽情報処理，" 情処学論，vol.37, no.7, pp.1460–1468, July 1996.
[15] 堀江健志，石畑宏明，清水俊幸，池坂守夫，"高並列計算機 AP1000 のアーキテクチャと性能評価，" 信学技報，CPSY91-26, 1991.

（平成9年2月21日受付，6月16日再受付）

後藤　真孝　（学生員）

1993早大・理工・電子通信卒．現在，同大大学院博士後期課程在学中．日本学術振興会特別研究員．音楽情報処理，並列処理，インタラクティブシステムなどに興味をもつ．1992 jus設立10周年記念UNIX国際シンポジウム論文賞受賞．1993 NICOGRAPH '93 CG教育シンポジウム最優秀賞受賞．情報処理学会，人工知能学会，日本ソフトウェア科学会，日本音楽知覚認知学会，日本神経回路学会，ICMA各会員．

村岡　洋一　（正員）

1965早大・理工・電気通信卒．1971イリノイ大学電子計算機学科博士課程了．Ph.D.この間，Illiac-IVプロジェクトで並列処理ソフトウェアの研究に従事．同学科助手の後，日本電信電話公社（現NTT）電気通信研究所に入所．1985より早稲田大学理工学部教授．現在同大学情報科学研究教育センター所長，図書館副館長．並列処理，マン・マシンインタフェースなどに興味をもつ．「コンピュータアーキテクチャ」（近代科学社）など著書多数．