Query-by-Conducting：テンポ類似性に基づく同一楽曲における多様な解釈の検索インタフェース

前澤　陽[†1]　後藤真孝[†2]　奥乃　博[†1]

本稿では与えられた楽曲の解釈をテンポの違いに基づき検索するためのインタフェース *Query-by-Conducting* とその要素技術であるテンポ推定について述べる。クラシック音楽のように任意の楽曲に対して複数の解釈が存在する場合、ユーザーの嗜好に合致する演奏を見つけるのはクラシック音楽を楽しむ上で重要な要素であり、そのような演奏の発見を支援するインタフェースは重要である。このインタフェースでは再生と連動してリアルタイムにユーザーが好みのテンポを指揮することにより、再生される演奏を動的にユーザーのテンポ入力にもっとも近いものに切り替える。楽曲再生が終わるとインタフェースはユーザーの指揮の履歴をクエリとして、それぞれの解釈におけるテンポ軌跡との類似性を算出し、類似性に基づきランク付けした結果を返す。テンポ推定手法は楽譜追従に基づく。同一の楽譜表現を演奏した複数の音源が存在することに着目し、それぞれの楽譜追従に整合性を要することによりテンポ推定精度を向上させる。室内楽とオーケストラ曲 9 曲で評価し提案手法の有効性を確認する。

Query-by-Conducting: A classical-music interpretation retrieval interface based on tempo similarity

Akira Maezawa ,[†1] Masataka Goto [†2]
and Hiroshi G. Okuno [†1]

This paper presents *Query-by-Conducting*, an interface for retrieving interpretation to a piece of classical music based on tempo similarity. Our interface allows a user to conduct along the playback of the piece, and the interface dynamically switches the playback to one that is closest to the user's conducting. At the end of playback, the system ranks each interpretation based on the similarity of user's conducting and the tempo of the interpretation. Tempo estimation method based on audio-score alignment is at the core of our interface. We improve tempo estimation by requiring the alignment for each interpretation to be consistent among one another. We evaluate the tempo estimation method using nine pieces of classical music.

1. はじめに

クラシック音楽では、同一の楽曲に対し複数の録音が存在する。例えば、 オンラインストアで「Mendelssohn Violin Concerto」の録音を検索すると 1200 件、また「Beethoven Spring Sonata」は 300 件ほどの検索結果が返ってくる (2010 年 3 月現在) 。これらは、与えられた楽譜表現を、様々な演奏者が、 固有の解釈をして演奏した録音の音響信号である。このように、膨大な同一楽曲における演奏の解釈から、ユーザーの嗜好に合致した解釈で演奏された録音を発見することは、クラシック音楽を楽しむ上で重要な要素である。同時に、このように多数の解釈を聴き比べ、全体の違いを理解した上で、自分の嗜好に合致した演奏者を見つける音楽の聴き方は、困難である。

我々の目標は、与えられた楽曲の中で、ユーザーの嗜好に合致するような解釈を、演奏解釈の多彩な要素に基づき検索することである。Content-based MIR (CBMIR)[3),9)] は我々の問題意識と近いが、音色やリズムといった音楽の要素に基づき**楽曲**を検索する CBMIR に対し、我々は**解釈**を検索する点が大きく異なる。解釈の構成要素は、テンポや演奏法など多岐に渡り、その定義は音楽家の間でも見解が異なると思われる。しかし、その中でもテンポは、音楽認知上演奏の解釈に大きな影響を与えているとされ[14)]、音楽情報処理の研究においても重要性が指摘され[5),10)]、一般的なユーザーにも比較的理解されやすい。

そこで本稿では、解釈の検索のために、同一楽曲における解釈間のテンポの違いに着目し、ユーザーが好みテンポを指揮することにより、任意の楽曲に対する解釈を検索するインタフェース *Query-by-Conducting* を開発する[⋆1]。このインタフェースは、任意の楽曲に対する楽譜表現 (標準 MIDI ファイル) とその楽譜表現を固有の解釈で演奏した複数の音響信号を読み込み、**図 1** に示すような、以下の機能を持つシステムにより楽曲の再生、テンポの可視化、解釈の検索を支援する。

(1)　テンポを可視化するための GUI
(2)　ハードウエア指揮インタフェース (*Nintendo Wii* リモコン) による、ユーザーのテ

†1 京都大学大学院情報学研究科
Kyoto University Graduate School of Informatics
†2 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
⋆1 http://www.youtube.com/QueryByConducting にインタフェースのデモビデオを公開した。

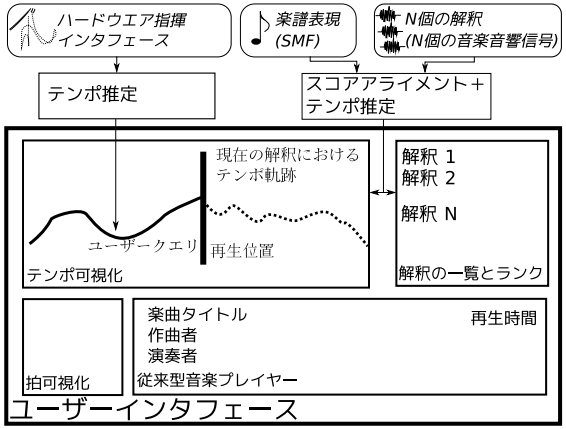


図 1 *Query-by-Conducting* のシステム図。

ンポクエリの入力支援

(3) ユーザーが入力したテンポクエリと、各解釈のテンポの類似性に基づく解釈のランク付け

想定されるシステムの大まかな流れとしては、まず、ユーザーは本インタフェースを、従来の楽曲再生インタフェースのように用いて楽曲を鑑賞する。途中で、ユーザーが、現在再生されている解釈のテンポに不満を持つと、GUI で可視化されているテンポ情報に基づき、ユーザーが聴きたいテンポを指揮する。システムは、ユーザーの指揮にもっとも近い解釈で演奏される録音を検索し、再生中の録音を、それに切り替える。楽曲を通して視聴したあとは、ユーザーが、どの地点で、どのような指揮を行ったかを、そのユーザーのテンポにおける嗜好と見做す。システムは、それぞれの解釈が、全体を通したユーザーの嗜好にどれだけ合致しているかに基づき、解釈を順序付け、ユーザーに提示する。

本インタフェースは、通常の指揮インタフェース[1),7),12),13)] と、三つの点で異なる。第一に、本インタフェースでは、指揮棒を、従来のように楽曲全体のテンポを指定するのではなく、楽曲の解釈を切り替えるために部分部分で用いる。ユーザーは、本インタフェースを、楽曲を鑑賞するためにも、任意の解釈を検索するためにも用いることができる。第二に、本インタフェースでは、大まかなテンポを受け付けるが、ダイナミクスや、拍の間における局所的なテンポの揺らぎといった、指揮や音楽の素養がある程度必要な要素は受け付けない。このように、ユーザーの操作できる幅を意図的に制限することが、初心者が手軽にインタ

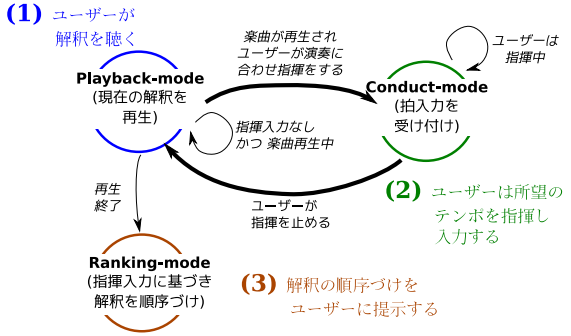


図 2 システムレベルでの状態遷移図。 インタフェースは楽曲の終わりまで状態 (1) と (2) を交互に繰り返す。

フェースを使うために必要であると提唱されている[2)]。最後に、本インタフェースは、解釈の検索が目的であり、指揮インタフェースのように、楽曲をユーザーが指定するテンポで再生することを意図して設計されていない。

本インタフェースには、高精度なテンポ推定が求まれるが、それぞれの解釈における楽譜追従結果を比較することにより、テンポ推定精度を向上させることが、可能であると考えられる。テンポ推定の中枢にある、楽譜追従の先行研究では、任意の楽譜追従はその音響信号と楽譜表現を用いて生成されている[4),6),8),10)]。この場合、仮に楽譜追従結果が間違えていても、比較対象がないが故にそれを知る術はない。しかし、今回の問題設定では複数の解釈の楽譜追従結果を用いる。そのため、楽譜追従結果同士の、時間軸における対応付けの整合性を調べることができる。任意の解釈二つに対して、音響信号同士の時間的対応付けと、両者の楽譜追従結果から算出されるお互いの時間的対応付けを比較することにより、整合性が取れている箇所とそうでない箇所を推定することができるためである。このように、追従結果の整合性に基づいた、楽譜追従の改善手法を提案し評価する。

2. インタフェースデザイン

本インタフェースは通常の音楽再生インタフェースで提供される再生や巻き戻しといった機能に加えて、(a) テンポ可視化、(b) 指揮ハードウエアインタフェースを用いたリアルタイムでのテンポクエリの入力、(c) テンポクエリに基づいた解釈の検索、といった三つの機能を提供する。**図 2** に示すように、インタフェースは現在の解釈を再生するモード

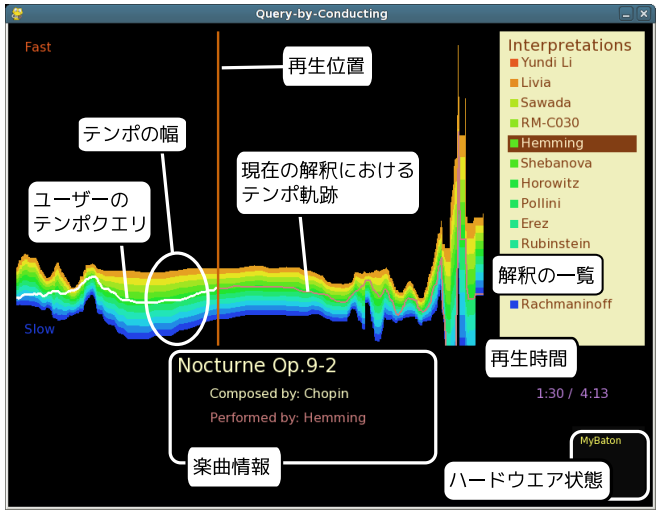


図 3 playback-mode 時の GUI は、再生に合わせテンポを表示する。

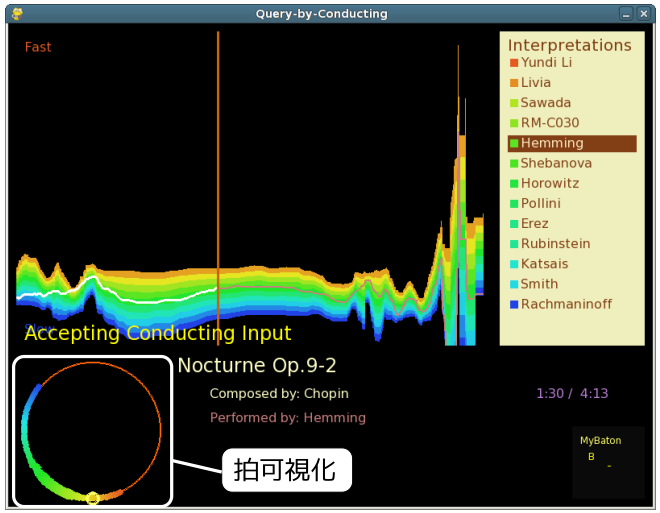


図 4 conduct-mode 時の GUI は、ユーザーの指揮入力を受け付け、 再生される解釈を切り替える。 拍可視化 GUI は指揮入力を支援すると考えられる。

(“playback-mode”) と、再生されている録音を、ユーザーのテンポクエリに適合する解釈へと切り替えるモード (“conduct-mode”) を交互に繰り返す。楽曲の再生が終わると、システムは、各解釈をテンポクエリの類似性に基づきランク付ける (“ranking-mode”)。

図 3 に再生中 (“playback-mode”) 時の GUI を示す。GUI の下部に、タイトル、演奏者、再生時間を、既存の音楽再生 GUI と同様に表示する。上部に楽曲のテンポを可視化し、GUI の右上に、解釈の一覧を、現在再生されている時点でのテンポの降順で表示する。

楽曲の再生時に、ユーザーが現在のテンポに不満を持った場合、**図 4** に示すような、所望のテンポをリアルタイムに入力できるモード (“conduct-mode”) に移ることが可能である。conduct-mode では、指揮ハードウエアのテンポ入力をインタフェースが受理する。また、画面左下に指揮を支援するための拍可視化 GUI を表示する。入力されたテンポと、再生されている時点における各解釈のテンポを算出し、最もテンポが入力に近い解釈で演奏する録音に、再生されている録音を切り替える。常にユーザーの指揮に合致する解釈に切り替えることにより、このモードでは、能動的な音楽鑑賞体験をユーザーに提供する。 楽曲の再生が終了すると、インタフェースは ranking-mode で、全ての解釈を、それぞれのテンポ軌跡とユーザーの指揮入力の履歴の類似性に基づき順序付け、ユーザーに提示する。

2.1 テンポ可視化 GUI

図 3 の上部には、現在再生されている解釈のテンポ軌跡、すべての解釈のテンポの幅、ユーザーが指揮したテンポの履歴の三点を表示する。**図 5** にその詳細を示す。テンポ可視化 GUI は、曲長 t_l の楽曲において、各 tick (楽譜表現上での時間軸) t でのテンポを表示するが、再生位置 t_c の近傍はレンズのように詳細に、それ以外は鳥瞰図のように簡略に表示させたい。そこで、以下の非線形伸縮を適用し 0 から 1 の間で正規化された時間軸 $x(t)$ を表示する。

$$x(t) = \frac{t}{2t_l} + \frac{1}{2}\frac{\exp\left(\frac{t-t_c}{T}\right)}{\exp\left(\frac{t-t_c}{T}\right)+1} \tag{1}$$

このような伸縮を行うことにより、再生位置の T-近傍を詳細に表示する。今回、T を四分音符 4 つ分と設定する。

現在の再生位置を鉛直線によって示す。再生位置より左の線分はユーザの入力したテンポの履歴であり、右の線分は現在の解釈のテンポ軌跡である。再生位置の左側にユーザーが今まで入力したテンポクエリ、右側にテンポを入力しない場合のクエリを、それぞれ表示して

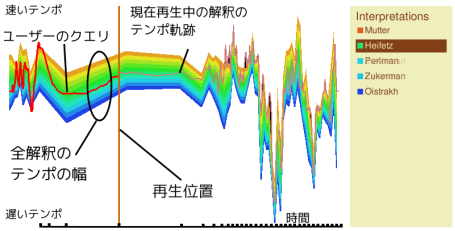


図 5 解釈可視化 GUI は再生されている解釈のテンポ軌跡、ユーザーのテンポ入力の履歴とテンポの幅を表示する。

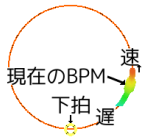


図 6 拍可視化 GUI は、拍と連動して回転する帯を表示することにより拍の位置を示す。

いると見做せる。

解釈の幅を表示するために、全ての解釈が取りうるテンポの幅を、グラデーションがかかった帯として表示する。色彩は早いテンポを低い色相 (オレンジ) 、遅いテンポを高い色相 (青) として表示するように、色相を徐々に変化させる。Tick t における、最も遅いテンポと速いテンポが、それぞれ $\tau_{\min}(t)$ と $\tau_{\max}(t)$ ならば、その範囲内の任意のテンポ $\tau(t)$ を、以下の色相 $\mathrm{hue}(t)$ で描画する。

$$\mathrm{hue}(t) = 240^\circ - \frac{\tau(t) - \tau_{\min}(t)}{\tau_{\max}(t) - \tau_{\min}(t)} 230^\circ \tag{2}$$

解釈可視化 GUI の右半分には、解釈の一覧を、現在の再生時間におけるテンポの降順で表示し、それぞれの解釈の隣には、式 (2) で表す色相を持ったボックスを表示し、現在再生されている解釈を反転表示する。

2.2 ハードウエア指揮インタフェース

ハードウエア指揮インタフェースは、ハードウエアコントローラに搭載されている加速度センサーの出力から拍を検出し、テンポに変換する。また、拍の入力を支援するためにインタフェースに拍を可視化する。

2.2.1 拍 検 出

拍の検出には、ゲームコントローラ (*Nintendo Wii* リモコン) に搭載されている加速度センサーを用いる。鉛直軸の加速度が、一定の閾値を越えた地点を検出することにより、拍を検出する。誤検出を減らすため、拍の検出後 200msec は入力を無視する。つまり、本インタフェースは、300BPM までのテンポを入力として受け付ける。

2.2.2 拍入力からテンポへの変換

予備実験で、再生されている音響信号と違うテンポを、ユーザーが指揮する場合、ユーザーは目標とするテンポを指揮するのではなく、再生中楽曲の下拍から下拍のタイミングが、目標とするテンポとの違いに応じて前後するような指揮を行うことが、観察された。例えば、速いテンポを指定する場合には、再生中のテンポで指揮をするが、下拍を実際の下拍よりやや前に置き、また、遅いテンポを指定する場合には下拍をやや後ろにずらす傾向が見られる。

このような知見に基づき、本インタフェースでは、実際にユーザーが指揮したテンポに基づき入力テンポを算出するのではなく、下拍のタイミングのずれに基づき、テンポを算出する。テンポが BPM_0 で再生中の楽曲において、ユーザーが Δ_t 秒下拍を遅れて指定したとする。すると、ユーザーが意図した入力テンポ BPM を以下のようにモデル化する。

$$BPM = BPM_0 \frac{1}{\frac{\Delta_t}{60/BPM_0} + 1} \tag{3}$$

クエリとして、ユーザーが過去 4 拍で指揮したテンポの軌跡を用い、また適合する解釈は、後述する解釈の類似性が最も小さいものを選ぶ。

2.2.3 拍可視化 GUI

予備実験で、ユーザーは必ずしもリズムの感覚があるとは限らず、下拍がどこにあるのかを理解するのが困難であることが観察された。特に、弦四重奏のように、アタックが弱い楽器で構成される楽曲において、顕著に表れた。

そこで、テンポ入力を支援するために、図 3 の左下 (詳細は**図 6**) に示すような、拍可視化 GUI を実装する。この GUI は、色相が変化するグラデーションを持つ帯が、楽曲のテンポと連動し回転する。また、下拍に合わせ、帯が GUI 下部の円を通過するように設定されている。帯の弧長は現在のテンポの幅に対応し、色相は式 (2) に基づき算出される。ユーザーが拍を把握できない場合、帯が GUI 下部の円を通過するタイミングを見計らい、指揮棒を振れば良い。

2.3 解釈の検索

楽曲を通して聞いた後、システムは ranking-mode に入り、全ての解釈をユーザーのテンポ入力との類似性に基づきランク付けし、掲示する。ここでは、再生された音響信号のテンポを、クエリとして見做す。例えば、ユーザーが曲長が 3 分の楽曲のうち、解釈 x を最初の 1 分だけ聞いて、インタフェースで指揮をした結果、残りの 2 分では解釈 y が再生されたとする。すると、解釈 x のテンポ軌跡の最初の 1 分と y の残り 2 分のテンポ軌跡が、クエリとして用いられる。ここで、解釈 i の拍 t でのテンポを $\tau_i(t)$ とし、ユーザーのクエリ

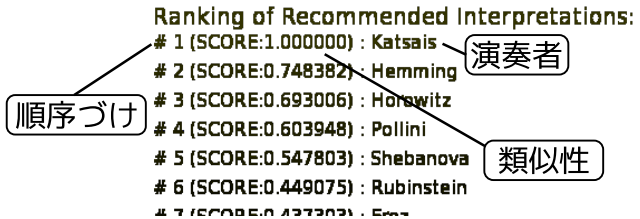


図 7 ranking-mode ではそれぞれの解釈の類似性を評価し、類似性の降順で解釈を提示する。

を $\tau_q(t)$ としたとき、解釈 i とクエリの非類似性 r_i を以下のように定義する。

$$r_i = \frac{1}{T}\int_0^T \left(\frac{\tau_i(t) - \tau_q(t)}{\tau_q(t)}\right)^2 dt \tag{4}$$

それぞれの解釈は、$1/r_i$ の降順で並び替えられ、**図 7** に示すように掲示する。また、$1/r_i$ の値が最大となるものは不透明、最小のものは透明になるように透明度を関連付けて描画する。さらに、この値を 0-1 の間でシフトし正規化したものを、類似性のスコアとして表示する。

3. テンポ推定手法

テンポ推定は、楽譜表現と音響信号の時間軸同士の写像を求める楽譜追従 (オーディオ-スコアアライメント) 手法に基づく。そのため、正確なアライメントは、正確なテンポ推定に必須である。よりテンポの推定精度を向上させるため、我々は、一つの楽譜表現に対して複数の音響信号が存在するという点を活用した、楽譜追従精度向上手法を提案する。

3.1 初期アライメントの生成

まず、楽譜追従の初期値として、Dynamic Time Warping (DTW) に基づく既存手法[4),6),11)] を用い、アライメントを生成する。ここでは、特徴量としてクロマベクトル、類似性の尺度としてコサイン距離を用いる。$c_k^{(t)}$ を解釈 k のフレーム t で算出した 12 次元のクロマベクトルとし、$c_S^{(t)}$ を楽譜表現の tick t で算出されたクロマベクトルとする。ここで、楽譜表現から解釈 k のアライメント $M_{k\leftarrow s}$ や、解釈 i から解釈 j のアライメント $M_{j\leftarrow i}$ を生成することを考える。$M_{k\leftarrow s}$ を生成するために、まず k と s の各時刻における特徴量の類似性を示す類似性行列 $R_{k\leftarrow s}$ を計算する。N_s と N_k がそれぞれ楽譜表現と解釈 k のデータ長さ (フレームの総数や楽曲を通した tick の総数) とすると、$R_{k\leftarrow s}$ は $\mathcal{R}^{N_s\times N_k}$ であり、

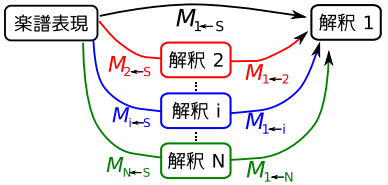


図 8 二つのアライメントを組み合わせると、楽譜表現からある解釈までの対応付けの方法は複数存在する。

要素 (i, j) は以下のように求められる。

$$R_{k\leftarrow s}(i,j) = 1 - \frac{c_s^{(i)} \cdot c_k^{(j)}}{\|c_s^{(i)}\| \cdot \|c_k^{(j)}\|} \tag{5}$$

次に、アライメント (時間的な対応付け) を DTW により求めるために、以下のような、$R_{k\leftarrow s}$ と同じ大きさのコスト行列を定義する。

$$C_{k\leftarrow s}(i,j) = R_{k\leftarrow s}(i,j) + \min\{C_{k\leftarrow s}(i-1,j), C_{k\leftarrow s}(i,j-1), C_{k\leftarrow s}(i-1,j-1)\} \tag{6}$$

ただし、全ての t に対し、$C_{k\leftarrow s}(t,-1) = C_{k\leftarrow s}(-1,t) = 0$ である。次に、これをバックトラックすることにより対応付け $M_{k\leftarrow s}^{(t)}$ を求める。初期値として、$M_{k\leftarrow s}^{(0)}$ を (N_s, N_k) に設定し、$M_{k\leftarrow s}^{(t)} = (0,0)$ になるまで、以下のように更新する。

$$M_{k\leftarrow s}^{(t+1)} := \underset{(I,J)\in S(t)}{\operatorname{argmin}} C_{k\leftarrow s}(I,J) \tag{7}$$
$$S(t) \ : \ \{(i-1,j),(i,j-1),(i-1,j-1)|(i,j) = M_{k\leftarrow s}^{(t)}\}$$

また、解釈 i と解釈 j 同士のアライメント $M_{j\leftarrow i}$ も、類似性行列を求めるときに、それぞれ c_i と c_j を用いることにより同様に算出する。

3.2 アライメントの改善手法

我々は、複数の解釈における楽譜追従結果の整合性を考慮することにより、その精度を向上させる。まず、楽譜表現とそれを録音した N の解釈を考える。すると、楽譜から解釈 i の楽譜追従の生成方法は、**図 8** に示すように N 通り存在する。すなわち、直接楽譜表現の時間軸と音響信号 i の時間軸の写像 $M_{i\leftarrow s}$ を算出するだけでなく、写像 $M_{j\leftarrow i}, j \neq i$ に写像 $M_{s\leftarrow j}$ を射影した対応付けも算出できる。理想的には、これらの N 個のアライメントは同一であるはずだが、それぞれの写像に推定誤差が生じるため、これらは一般的には一致しない。

楽譜表現から、解釈 j を介した、解釈 i への写像 $M_{i\leftarrow j}\circ M_{j\leftarrow S}$ を生成するには、$M_{j\leftarrow S}$ と $M_{i\leftarrow j}$ が連続かつ一対一である必要がある。しかし、前項で生成した初期アライメントはバックトラックでの性質上これを満たさない。そこで、初期アライメントの上に、一対一であり連続である写像を「辿る」ことにより、上記の性質を満たす写像を生成する。全ての時系列 s と k (解釈ないし楽譜表現) の組み合わせに対し、以下のアルゴリズムを適用する。

(1) $t=0$ とし、アライメントの初期点 $\tilde{M}^{(t)}_{k\leftarrow s}$ を $(0,0)$ に設定する。

(2) $\epsilon<\theta<\frac{\pi}{2}-\epsilon$ に対し以下のコストを計算する。

$$c(\theta)=E_{q\sim\exp(-3q/Q)}[\min_n d^{(t)}_n(q,\theta)] \tag{8}$$

$$d^{(t)}_n(q,\theta)=\|\tilde{M}^{(t)}_{k\leftarrow s}+q\,(\cos(\theta),\sin(\theta))-M^{(n)}_{k\leftarrow s}\| \tag{9}$$

$Q=20$ フレーム、 $\epsilon=\pi/20$ ラジアンとする。θ を $\pi/20$ ラジアン毎に評価する。

(3) $\tilde{M}_{k\leftarrow s}$ を任意の $\Delta r\in(0,1]$ を設定し次のように更新する。

$$\tilde{M}^{(t+1)}_{k\leftarrow s}:=\tilde{M}^{(t)}_{k\leftarrow s}+\Delta r\left(\cos(\hat{\theta}),\sin(\hat{\theta})\right) \tag{10}$$

$$\hat{\theta}\;=\;\operatorname*{argmin}_{\theta} c(\theta)$$

$\Delta r=1$ とした。

(4) もし $\tilde{M}^{(t)}_{k\leftarrow s}\cdot(1,0)\geq N_s$ か $\tilde{M}^{(t)}_{k\leftarrow s}\cdot(0,1)\geq N_k$ なら終了。

(5) $t:=t+1$ としステップ 2 に戻る。

次に、それぞれのアライメントには、独立に、次のような形状パラメータ b と位置パラメータ $\hat{M}_{i\leftarrow S}(t)$ を持つ、ラプラシアンノイズが混在していると考える。

$$p(t)=\exp\left(-\|M_{i\leftarrow S}(t)-\hat{M}_{i\leftarrow S}(t)\|/b\right)/2b \tag{11}$$

また、$M_{i\leftarrow j}\circ M_{j\leftarrow S}, j\neq i$ でも同様だが独立したノイズが混在すると考える。つまり、全ての j に対して $M_{i\leftarrow j}\circ M_{j\leftarrow S}$ は真のアライメント $\hat{M}_{i\leftarrow S}$ から生成され、独立したラプラスノイズが混在していると考える。すると、アライメントの推定は、楽譜表現と解釈 k 同士の、N 通りのアライメントを与えられた時に、$\hat{M}_{i\leftarrow S}$ を推定する問題となる。この推定値は、全てのアライメントのメジアンであるため、以下のように $M_{i\leftarrow S}$ を更新する。

$$M_{i\leftarrow S}(t):=\mathrm{median}(\{M_{i\leftarrow j}\circ M_{j\leftarrow S}(t)\}) \tag{12}$$

任意の i に対してこの更新則を適用すると、全ての $k\neq i$ における $\hat{M}_{k\leftarrow S}$ の推定値も変化しうる。そのため、全ての i に対してこの更新則を反復的に適用することにより、推定値 $\hat{M}_{i\leftarrow S}$ は反復する毎に更新されていくと考えられる。後述するが、多くの場合この更新則を反復することにより、精度が向上する。

3.3 テンポ推定

テンポの推定は、各時刻における楽譜追従の、楽譜表現の時間軸に対する音響信号の傾きを求めることにより求まる。Tick t のテンポを、任意の $T>0$ を与えたとき $t-T$ から $t+T$ のオンセット時刻でのアライメント情報を用いて算出する。オンセット以外のアライメント結果は意味を持たないため破棄する。T は最低でも 2 拍かつ 20 個のノートオンセット時刻を考慮するよう動的に決められる。$(s(p),a(p))$ を、$M_{i\leftarrow S}$ が tick t 時に $t-T$ から $t+T$ に含むオンセットでの追従結果とする。Tick t での BPM $\tau(t)$ は、$(s(p),a(p))$ の傾き $m(t)$ を線形回帰を用いて求め、以下のように重み付けすることにより算出する。

$$\tau(t)=\frac{1}{m(t)}\frac{\text{audio frame-per-minute}}{\text{ticks-per-beat}} \tag{13}$$

4. 評価実験

テンポ推定手法と、推定されたテンポに基づく解釈の検索性能を評価する。評価にあたっては、様々な楽器編成の、平均再生時間が 5 分ほどのクラシック音楽 9 曲を分析した。そのうち、オーケストラ曲を *orch-1* から *orch-3*、小規模なアンサンブルを *duo-1* から *duo-3*、ソロ曲を *solo-1* から *solo-3* と呼ぶ。それぞれの楽曲に対して、4 から 13 の録音を用意し、計 4 時間 40 分ほどのデータの、テンポの正解データを用意した。

4.1 テンポ推定の評価

$\tau_g(t)$ を正解データのテンポ軌跡とする。推定された軌跡 $\hat{\tau}(t)$ が与えられた時の誤差を、以下に記す、正規化された二乗誤差 (MSE) を用いて評価する。

$$\mathrm{MSE}=\frac{1}{T}\int_0^T\left(\frac{\tau_g(t)-\hat{\tau}(t)}{\tau_g(t)}\right)^2 dt \tag{14}$$

これは正解データと推定されたテンポ軌跡同士の非類似性と見做すこともできる。

表 1 に、全ての解釈の平均 MSE を全ての曲に対して、更新式 Equation (12)) を複数回適用したときの結果を記す。まず、本手法が MSE を軽減することが分かる。特に、初期アライメント (更新を 0 回) の MSE が大きい場合に、大きな改善が見られる。次に、多くの場合、更新式を複数回適用することにより、MSE が低減することが分かる。MSE が増える場合は、楽譜追従の誤差が存在する箇所は、解釈の間で独立に起こるという仮定が、破られた時だと考える。例えば、楽譜表現に記載されていないカデンツァが含まれる *solo-3* では、正しくないアライメントが常にカデンツァの位置で起こるため、N 個のアライメントのメジアンには、意味の無いデータが含まれる。

表 1 式 (12) を複数回繰り替えしたときの平均 MSE ($\times 10^{-3}$)。

楽曲 (解釈の数)	0 回	1 回	2 回	10 回
solo-1 (13)	8.9	8.6	8.4	8.4
solo-2 (6)	17.3	15.0	13.0	12.7
solo-3 (5)	266.7	73.1	85.4	98.8
duo-1 (5)	4.5	3.9	3.7	3.8
duo-2 (4)	34.8	22.1	20.6	20.4
duo-3 (4)	185.4	12.5	10.2	10.2
orch-1 (5)	646.8	54.4	47.2	44.9
orch-2 (5)	231.5	14.7	13.3	13.2
orch-3 (5)	3941.6	1091.4	1038.3	833.2

4.2 検索性能の評価

次に、指揮の誤差やテンポの推定エラーが、楽曲の検索性能に及ぼす影響について評価する。ユーザーが解釈 i に合わせて指揮した場合、ユーザーのテンポクエリにもっとも適合する解釈として、システムは i を返すべきである。これが実現されない場合には、ユーザーが十分な精度で楽曲を指揮できず、意図された楽曲が返ってこないか、テンポ推定精度が、意図された楽曲を返すのには不十分であると考えられる。これらの場合、i は最も適合はしてない、適合する解釈上位 M 位に入ると考えられる。

まず、正解データに平滑化されたノイズを付与することにより、人間の指揮エラーをシミュレートしたクエリを、用意する。各解釈 i に対して、次のような分散 s を持つ雑音が混在したテンポ軌跡を用意する。

$$\tau_{\mathrm{query,i}}(t;s) = \tau_{g,i}(t) \cdot 2^{\sqrt{\frac{s}{L}}\sum_{l=0}^{L-1} n(t-l)} | L = 10 \qquad (15)$$
$$n(t;s) \sim \mathcal{N}(0,1)$$

次に、生成されたクエリに適合する上位 M 位の解釈を検索し、検索性能を F 値として評価する。n を上位 M 位に意図された解釈が返ってきた数、N を解釈の総数とすると、適合率を $P = n/(N \times M)$、再現率を $R = n/N$ とし、F 値を $F = 2PR/(P+R)$ とする。

solo-1、*duo-1*、*orch-1* の結果を、それぞれ**図 9** (a) 、 (b) 、 (c) に図示する。おおよそ $s = 0.15$、すなわち原曲のテンポの 0.7 倍から 1.3 倍 (3σ) までの指揮の誤差を許容することが、図 9 (a) 、 (b) から分かる。しかし図 9(c) から分かるように、*orch-1* の結果は他の二つよりも著しく低い。これは、*orch-1* のテンポ推定精度がもともと劣悪であるだけでなく、解釈同士の違いも、それほど顕著に現れないためだと考えられる。異なる *orch-1* の解釈間における、最小の非類似度は 2×10^{-3} ほどである。対して、*solo-1* のそれは 20×10^{-3}、*duo-1* のそれは 23×10^{-3} である。この結果は、室内楽のように奏者が自由にテンポを変えられる場合、曲中のテンポの変化が大きいため非類似度が一般的に大きいのに対して、*orch-1* では曲を通してテンポが一定であるため、全体の解釈の差がテンポに反映されづらいことを示唆する。

これらを踏まえると、本システムではテンポの推定精度が正確で、かつ解釈間の非類似度が大きければ、想定された解釈を検索できると示唆される。また、テンポの推定精度は、アンサンブルの規模とカデンツァなどの、楽譜に表記されていない箇所の存在に、依存すると考えられる。

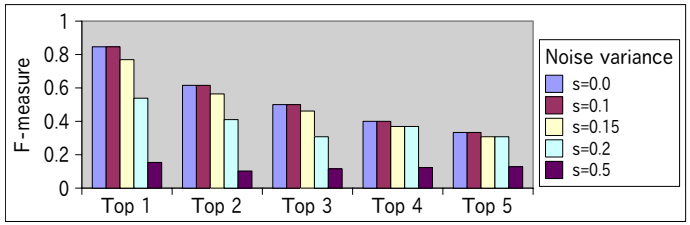
5. ま と め

本稿では、任意の楽曲における、解釈を検索するためのインタフェース *Query-by-Conducting* を開発した。このインタフェースでは、ユーザーがインタラクティブに指揮をし、時々刻々とユーザーの指揮に適合する解釈に再生を切り替えつつ、大局的な指揮の傾向に適合する解釈を検索する機能を実現した。また、ユーザーの指揮との類似性に基づきそれぞれの解釈の適合度を提示することにより、ユーザー好みのテンポで奏でる演奏者の検索を支援する。中枢的な要素技術であるテンポ推定精度は、その根本となる楽譜追従手法を、複数の解釈に対して同時に考慮することにより、劇的に改善した。今後の課題として、テンポ推定精度を向上させ、拍入力からテンポへの変換を行うモデルの妥当性や拍可視化 GUI の有用性をはじめとするユーザースタディを行った上で、よりユーザーにクラシック音楽を分かりやすく提示するための可視化とインタラクション手法について考えていきたい。

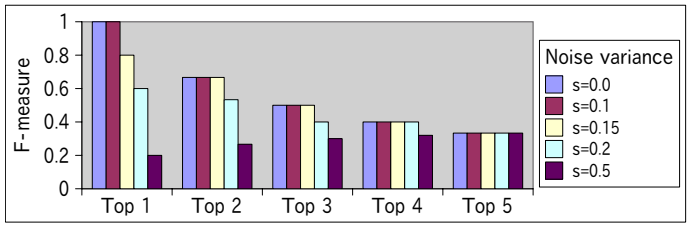
謝辞: 本研究は科研費 (S) と JST CrestMuse の支援を受けた。

参 考 文 献

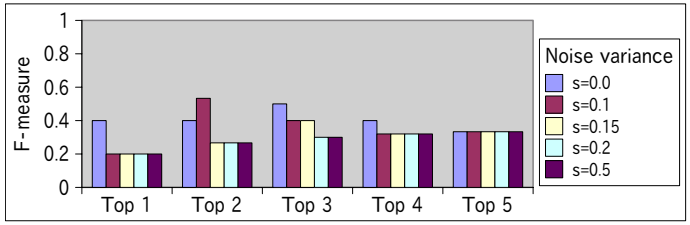
1) Bevilacqua, F., Guédy, F., Schnell, N., Fléty, E. and Leroy, N.: Wireless sensor interface and gesture-follower for music pedagogy, *NIME '07*, pp.124–129 (2007).
2) Dixon, S., Goebl, W. and Widmer, G.: The "Air Worm": An Interface for Real-Time Manipulation of Expressive Music Performance, *ICMC '05*, pp. 614–617 (2005).
3) Dixon, S., Guyon, F. and Widmer, G.: Towards characterization of music via rhythmic patterns, *ISMIR '04*, pp.509–516 (2004).
4) Dixon, S. and Widmer, G.: MATCH: Music Alignment Tool Chest, *ISMIR'05*, pp.



(a) *solo-1*



(b) *duo-1*



(c) *orch-1*

図 9 分散 s のノイズを正解データに混在させたものをクエリとして用い、それにもっとも適合する上位 $M \leq 5$ 位までを検索結果として考慮した場合の F 値。

11–15 (2005).
5) Honing, H.: From Time to Time: The Representation of Timing and Tempo, *Comp. Music J.*, Vol.25, No.3, pp.50–61 (2001).
6) Hu, N., Dannenberg, R. and Tzanetakis, G.: Polyphonic audio matching and alignment for music retrieval, *WASPAA '03*, pp.185–188 (2003).
7) Katayose, H. and Okudaira, K.: Using an expressive performance template in a music conducting interface, *NIME '04*, pp.124–129 (2004).
8) Kurth, F., Müller, M., Damm, D., Fremerey, C., Ribbrock, A. and Clausen, M.: SyncPlayer - An Advanced System for Multimodal Music Access, *ISMIR '05*, pp. 381–388 (2005).
9) Michael, C., Veltkamp, R., Goto, M., Leman, M., Rhodes, C. and Slaney, M.: Content-Based Music Information Retrieval: Current Directions and Future Challenges, Vol.96, No.4, pp.668–696 (2008).
10) Müller, M., Konz, V., Scharfstein, A., Ewert, S. and Clausen, M.: Towards Automated Extraction of Tempo Parameters from Expressive Music Recordings, *ISMIR '09*, pp.69–74 (2009).
11) Müller, M., Kurth, F. and Röder, T.: Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization, *ISMIR'04*, pp.365–372 (2004).
12) Schertenleib, S., Gutiérrez, M., Vexo, F. and Thalman, D.: Conducting a Virtual Orchestra, *IEEE MultiMedia*, Vol.11, No.3, pp.40–49 (2004).
13) Segen, J., Kumar, S. and Gluckman, J.: Visual Interface for Conducting Virtual Orchestra, *ICPR '00*, p.1276 (2000).
14) Timmers, R.: Predicting the similarity between expressive performances of music from measurements of tempo and dynamics, *JASA*, Vol.117, No.1, pp.391–399 (2005).