ハミング検索のための音楽データ自動時間正規化手法

小杉尚子[†] 櫻井保志[††] 森本正志[†]

本論文では，ハミング検索のための音楽データ自動時間正規化手法を提案する．ハミング検索は音楽内容検索の一種で，一般的に楽曲のハミングされる部分，長さ，テンポ，キーは事前には分からない．しかも音楽データは時間軸方向の伸縮と，音高軸方向のシフトが許される時系列データである．このようなデータに対して，伸縮の度合いや長さが事前には分からない入力データを用いて効率的な部分マッチングを行うためには，入力/参照の双方のデータを伸縮に依存しない形式に変換するか，あるいはデータの伸縮に左右されないマッチングを行うなどの対策が必要である．本論文では，データを伸縮に依存しない形式に変換する手法として，楽曲内および楽曲の一部で最頻出する発声時間や音符の長さを用いて音楽データを時間的に正規化する手法を提案する．また，その自動時間正規化手法を我々が研究開発しているハミング検索システム “SoundCompass” に実装し，定量的な評価を通して本手法が音楽データに対する有効な時間正規化手法であることを確認した．

An Automatic Time-normalization Technique for Music Data for a Query-by-Humming System

Naoko Kosugi,[†] Yasushi Sakurai[††] and Masashi Morimoto[†]

This paper proposes an automatic time-normalization technique for music data for a query-by-humming system. Query-by-humming is a kind of content-based music retrieval method that accepts as queries hummed tunes whose parts, lengths, tempos, and keys are not known a priori. Music is time-series data that allows time-scaling and tone-shifting. For efficient subsequence matching with such music data and input data whose scale and length are unknown priori, either changing both of them into a form that is independent from time-scaling or using a matching method that is independent from time-scaling is necessary. In this paper, we propose a method to time-normalize music data based on the most frequently appearing duration of utterances and tones within each piece of music or a part of it as a method to change data into a form that is independent from time-scaling. In addition, the time-normalization technique for music data proposed in this paper is implemented in a query-by-humming system “SoundCompass” that we are doing research and development work on, and quantitative evaluations confirmed that this technique is an effective time-normalization method for music data.

1. はじめに

メロディの一部を歌唱してその楽曲を検索することをハミング検索という．ハミング検索は音楽内容検索の一種で，デジタル化された音楽データの利用が急速に増加する中で，より便利な利用法を実現するための重要な研究課題である．本課題に対し，我々は “SoundCompass” というハミング検索システムの研究開発を推進してきた[1),2)]．

この「歌唱を入力して曲を検索する」という一見簡単そうな技術は，多くの難解な問題点を内包しており，データベース分野の複数の基礎技術を適切に組み合わせることで初めて実現しうる．それらは大きく分けて以下の 3 つにまとめられる．

(1) 入力データの質 → 類似検索技術
歌い間違いや音声信号処理ミスなど，入力データには様々なエラーが存在する．またどんなに歌が上手な人でも，完全に楽譜どおりに歌うことは難しい．したがって，一致検索ではなく類似検索技術が必要である．

(2) データの性質 → 時系列データハンドリング技術
音楽は，早いテンポで演奏されても遅いテンポで演奏されても同じ曲であると認識することができる．このような時系列データに対して，効果的で効率的なマッチングを行うためには，テ

† NTT サイバーソリューション研究所
NTT Cyber Solutions Laboratories
†† NTT サイバースペース研究所
NTT Cyber Space Laboratories

ンポの違いに左右されない時系列データハンドリング技術が必要である．

(3) 歌唱部分と長さ → 可変長部分マッチング技術
ハミングで曲を検索する場合，一般的にはその曲の一部が歌唱される．またその長さは一定ではない．したがって，可変長の入力データに対する部分マッチング技術が必要である．

上記の問題点について SoundCompass では様々な解決策を提案し導入してきたが，(2) の時系列データハンドリングについては，楽曲のテンポ情報を利用したデータベース構築と，ユーザにはメトロノームに合わせてハミングしてもらうことを前提にしているため，楽曲データ/ハミングデータの各々の時間正規化にはテンポ情報を利用することができた．しかし，今後は携帯電話や PDA といった，メトロノームを使用できない状況でのハミング検索を可能にする必要が見込まれる．そこで本論文の目的は，メトロノームのテンポ情報を使わないで，音楽データを時間正規化する手法を提案することである．また提案手法の有効性は，SoundCompass を用いて定量的に評価し，大規模音楽データベースを用いた実用システムとしての評価も行う．

本論文では音楽データを時間的にスケーラブル（time-scalable）で音高的にシフタブル（tone-shiftable）な時系列データと位置づける．時系列データとは「ある一定の時間間隔ごとに計測される実数の列[3]」と定義される．本論文におけるスケーリング・シフティング・時間正規化（time-normalization）の概念を**図 1** に示す．

本論文の構成は以下のとおりである．まず最初に 2 章で関連研究について述べる．次に 3 章で従来の SoundCompass について説明し，本論文で解決する課題を明らかにする．4 章では時間的にスケーラブルな時系列データのマッチング技術として最近注目されている，*Dynamic Time Warping*[4),5)]（以下 DTW）をハミング検索に適用し，定量的な評価を通してその利点や問題点について議論する．5 章では 4 章で明らかになった問題点を解決するための音楽データの自動時間正規化手法を提案する．5 章で提案した技術を導入した，新しい SoundCompass については 6 章で述べる．7 章では本論文で提案する手法を SoundCompass を用いて定量的に評価し効果を議論する．最後に 8 章で本論文をまとめる．

2. 関 連 研 究

従来，音符を数で評価（たとえば，**図 2** の楽譜例で

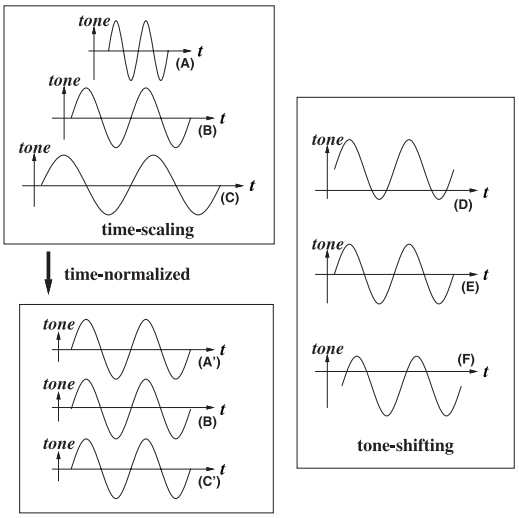

図 1 時系列データのスケーリング，シフティング，時間正規化
Fig. 1 Time-scaling, tone-shifting and time-normalization of time-series data.

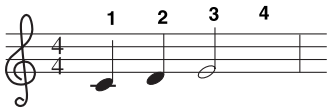

図 2 楽譜例
Fig. 2 A music score example.

は，ドが 1 つ，レが 1 つ，ミが 1 つと数える）して音楽データ処理を行う「音符ベース」のシステム[6)~8)]や，拍で評価（たとえば，図 2 の楽譜例では，ドが 1 拍，レが 1 拍，ミが 2 拍と数える）して音楽データ処理を行う「拍ベース」のシステム[1),2)]など，音符を単位にした音楽データ処理が多かった．最近は音符を単位にしない，音声信号情報を直接利用する「フレームベース」のハミング検索システムが提案されている[9)~13)]．

フレームベースの音楽データ処理の利点は，音声情報を音符に変換しないので，その際のエラーがクエリに混入しないことであると考えられている．しかしもともと入力されるハミングデータには突発的な雑音の混入などもあるので，入力データを音符に分解する際のエラーがないからといって，必ずしもフレームベース処理の方が音符ベース処理や拍ベース処理と比べて質の良いクエリを作成できるとはいい難い．また，フレームベース処理を行うと時間情報がそのまま保存されるので，マッチングの際に時間軸方向の伸縮による影響を受ける．そこで時間軸の収縮による影響を回避するために DTW が用いられている．DTW はもともと音声認識で利用されていた技術で，データの局所的な時間のずれに影響されないマッチングが可能に

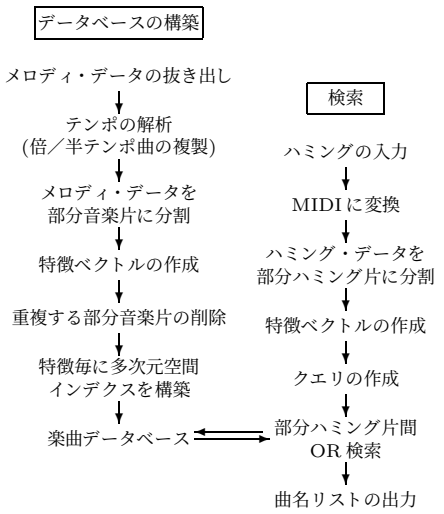

図 3 従来の SoundCompass の処理の流れ
Fig. 3 Process flow for previous SoundCompass.

なる．たしかに，一般ユーザは一定のリズムを保ってハミングするのが難しいので，DTW は音楽データのマッチングに有効であると考えることもできる．しかし，マッチングする部分を曲の出だしに限定したり[9]，マッチングする部分を特定する手法の記述がない[11]など，適用方法は限定的で，ハミング検索に利用するうえでどのような利点や問題があるのかは十分に明らかにはなっていない．

3. 従来の SoundCompass と検討課題

3.1 従来の SoundCompass

SoundCompass の最大の特徴は，「拍ベース」の音楽データ処理の導入により，時間的な伸縮に影響されないデータに変換したうえで，音符の音長情報もクエリに活用することで，高速マッチングと高い分解能の両方を実現していることである．拍ベースの処理では，「拍」を単位に音楽をスケールの合った時系列データに変換するので，単純な距離計算での類似判定が可能となる．また音高と音長の両方を特徴としてマッチングに利用できるので「音符ベース」処理によるシステムに比べて分解能が高い．結果的に，約 2 万曲の楽曲データベースに対して，約 3 秒以内に検索結果を出力することができ，5 位以内の正解率は 84.2%であった[2]．

図 3 に従来の SoundCompass [2]の処理の流れを示す．データベースの構築を図 3 の左側に示す．まず，MIDI データからメロディデータを抜き出しテンポを解析する．メロディの存在する部分のテンポが閾値より大きい/小さい場合は，そのテンポの半分/倍のテンポの曲を合成し，オリジナルとともに両方ともデータベースに登録する．これは非常に大きい/小さいテンポの曲は，ユーザが正しいテンポの半分/倍でハミングするという傾向があり，これが起きると 1 拍あたりのデータ量が倍/半分になるので，スケールの合った正しい比較演算を高速に行えなくなるのである．次にスライディング・ウィンド方式[14]で，楽曲データを先頭から一定の長さ（スライド幅）ごとに一定の長さ（ウィンド長）に切り出す．これを部分音楽片と呼び，それらを用いてデータベースを作成することにより，ユーザは曲の任意の部分を歌って検索することができる．

次に各部分音楽片から特徴量を抽出し特徴ベクトルを作成する．特徴ベクトルとしては「音高推移全体特徴ベクトル」,「音高推移部分特徴ベクトル」と「音高差分布特徴ベクトル」を使用する．「音高推移全体特徴ベクトル」と「音高推移部分特徴ベクトル」は，ある一定の拍ごとの音高値を並べた音高値の時系列データで，メロディの類似度を判定するに効果的な特徴ベクトルである．特に「音高推移部分特徴ベクトル」は，スライディング・ウィンド方式で機械的に分割された部分音楽片に対して，特徴ベクトルの先頭を部分音楽片の先頭とは独立に選択することを可能にするので，列データのマッチングで致命的なマッチするデータどうしの先頭の不一致の問題を緩和することができるのが特徴である．「音高差分布特徴ベクトル」は隣り合う音符の音高差の分布を示す特徴ベクトルである．これだけでは高い検索精度を出せないが，音符数や発声回数の違いを表現することができるので，補助的な特徴ベクトルとして有効である．

特徴ベクトルを抽出し終わったら，重複する部分音楽片の排除を行う．曲の中では繰り返し使用されるフレーズなどもあるので[2),15]同じ特徴ベクトルに変換される部分音楽片どうしは最初の 1 つだけをデータベースに登録することで，データベースのサイズを縮小すると同時に検索精度の劣化も防止するのである．残った特徴ベクトルは多次元空間インデクスに格納して楽曲データベースが完成する．

検索の流れを図 3 の右側に示す．ユーザにはメトロノームに合わせて「タ」でメロディを歌唱してもらう．メトロノームは，歌いやすい早さ（=テンポ）に調節してよい．マイクを通して入力されたハミングとメトロノームのテンポ値を用いて，市販の採譜ソフトによってハミングデータを MIDI 形式に変換し，データベースの構築における部分音楽片作成時と同様のスライド幅とウィンド長でハミングデータを切り出す．これを部分ハミング片と呼ぶ．部分ハミング片からも

データベースと同様の特徴ベクトルを作成し，それを用いて楽曲データベースから類似するベクトルを持つ楽曲を検索し，類似している順に整列して曲名リストを出力する．

3.2 課題と前提

SoundCompass では，「拍」を単位に音楽データのスケールを統一したが，そのためにユーザにはメトロノームに合わせて歌ってもらう必要があった．すなわちハミングデータの時間正規化には，ハミングそのもののほかに外部から入力されるメトロノームのテンポ情報が必要であった．しかし携帯電話や PDA の普及などで，今後は必ずしもそのようなインタフェースを用意できない状況も予想されるので，入力されるハミングだけで今までと同様に音楽データのスケールを統一することが必要となった．そこで本論文では，ハミング検索のための音楽データに対するメトロノームのテンポ情報を必要としない自動時間正規化手法を考案する．この技術を考案するうえで重要な知見は，「音楽は時間的にスケーラブルな時系列データであるが，スケールは 1 つの楽曲内においておおよそ統一的である」ということである．

4. DTW を用いたハミング検索

本章では DTW をハミング検索に適用し，定量的な評価をベースにその利点や問題点について議論する．

4.1 データベースとクエリ

本論文ではデータ解析と精度評価に，以下の条件を満足する楽曲データと**表 1** のハミングセットを使用する．本論文の目的はメトロノームを用いずに音楽データを時間正規化する手法の提案なので，ほとんどの実験と評価にはセット A とセット C を用いるが，7 章での比較評価のためにセット B とセット D も用意した．

- 楽曲データ
  - MIDI 形式の 21,804 曲．
  - 四分音符は 480 ティック☆で表される．
  - ほとんどの曲は日本の歌謡曲であるが，約 6,000 曲は洋楽，童謡，民謡などである．
- ハミングセット
  - 34 人（男性 26 名，女性 8 名）による 591 個のハミング（表 1）．
  - 上記の人の多くは歌唱訓練を受けた経験はない．
  - この 591 のハミングデータは，複数の人によって十分に良質であることが確認されたものである．具体的には，収録した全ハミングデータを，以下の 3 段階にレベル分けしたとき，過半数の人によって A または B と評価されたハミングである．
    A ··· 検索できて当然
    B ··· 検索されてほしい
    C ··· 検索できなくてもやむをえない
  - 楽曲の出だしを歌ったハミング（表 1 のセット A とセット B）と，任意の部分を歌ったハミング（表 1 のセット C とセット D）がある．
  - メトロノームを使用しなかったハミング（表 1 のセット A とセット C）と，メトロノームを使用したハミング（表 1 のセット B とセット D）がある．

☆ MIDI データの分解能を表すための最小単位．MIDI データの分解能は，データの時間管理を行うために必要な情報である[16]．

表 1 ハミングセット
Table 1 Sets of hummed tunes.

	メトロノームなし	メトロノームあり	合計
出だし部分	151（セット A）	184（セット B）	335
任意の部分	99（セット C）	157（セット D）	256
合計	250	341	591

楽曲データをスライディング・ウィンド方式で部分データに分割し，単位時間ごとの音高値を要素とするシーケンスを作成してデータベースに登録する．そのデータベースに対して DTW による距離計算を行い，ハミング検索の精度評価実験を行った．各変数は以下のとおりである．

- ut（単位時間）··· 0.25 秒
- sl（スライド幅）··· 2.0 秒
- wl（ウィンド長）··· 13.75 秒（= 55 要素）

これらの値は，以下の楽曲データの分析によって確定した．**図 4** に全楽曲データのテンポの分布を示す．横軸はテンポ，縦軸は曲数である．120 前後または 80 前後のテンポの曲がほぼ同数で多いことが分かる．テンポ 120 は標準的に人間が歌いやすいテンポの 1 つである．さらに**図 5** に音価の出現分布を示す．各音価はオンセット情報処理（5.1 節参照）で評価する．横軸はティックを，縦軸は音符数を表す．この図から最も多く出現する音価は八分音符（240 ティック）であることが分かる．テンポ 120 における八分音符の実音長は 0.25 秒であるので ut を 0.25 秒とした．次に**表 2** に拍子の使用頻度を示す．4/4 拍子が最も多く使われていることが分かる．そこで，4/4 拍子における 1 小節，すなわち八分音符 8 個分をずらし幅とした．したがって sl は 2 秒とした．最後にハミング長の平均値

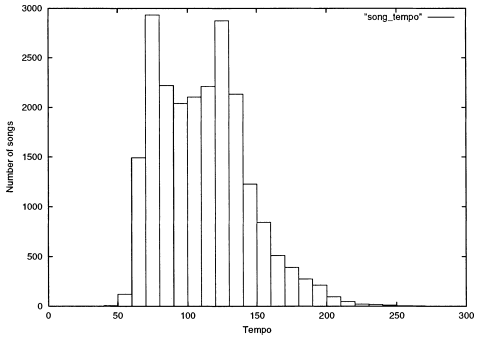

図 4 楽曲のテンポの分布
Fig. 4 Tempo distribution of songs.

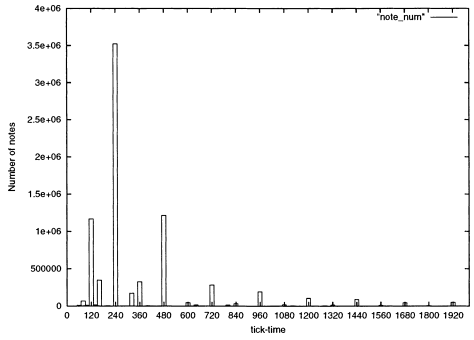

図 5 音価の分布
Fig. 5 Distribution of phonetic values.

表 2 拍子の使用頻度
Table 2 Distribution of meters.

拍子	4/4	3/4	6/8	2/4	others
曲数	21139	414	102	95	54
割合（%）	97.0	1.9	0.5	0.4	0.2

は 13.7 秒であった．そこで wl を 13.75 秒（=55 要素）とした．

ハミングからも FFT を用いて 0.25 秒ごとに音高値を抽出し，それを並べてクエリとして使用した．ハミングデータは分割せずに全体を 1 つのクエリとしたので，クエリはすべて可変長である．キー（音高）の違いを吸収するために，楽曲データもハミングデータも平均音高値を基準にシフティングした．

4.2 実験結果

図 6 に検索精度実験結果を示す．横軸は順位を表している．縦軸は検索精度を表しており，ある順位以内に正解を検索したハミングの割合を示している．“DTW_full_A_C” は，全曲の全部分データを登録したデータベースに対して表 1 のハミングセット A と C を用いて検索した場合の検索精度で，“DTW_head_A” は全曲の先頭の部分データのみを登録したデータベースに対して表 1 のハミングセット A のみを用いて検索した場合の検索精度を示す．全部分データのデータ数は 2,035,602 で，先頭の部分データのデータ数は楽曲数と同じ 21,804 である．

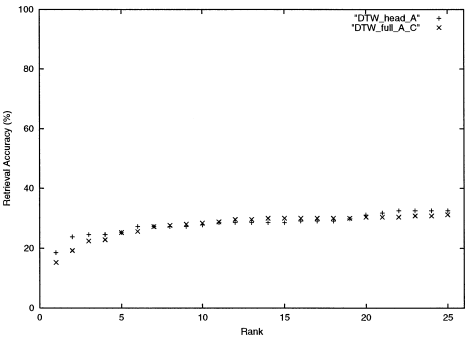

図 6 DTW を用いたハミング検索の精度
Fig. 6 Retrieval accuracy with DTW.

4.3 DTW による検索実験に対する議論

DTW を用いた検索では，25 位以内に検索された件数が少ないが，図 7 から図 10 のシーケンス例を用いて議論する．各図では横軸は時間で縦軸は音高値である．また楽曲データを点線で，ハミングデータを実線で示す．ここで，楽曲データはスライディング・ウィンド方式により，機械的に切り出された楽曲の一部であり，必ずしもまとまり感のあるフレーズという単位ではないという前提で，各図がハミング検索という観点では，どのような状況を表しているのかも合わせて説明する．

まず，25 位以内に検索されたハミングは，**図 7** に示すように全体の長さが異なっていてもメロディという観点からは情報量が等しいものや，**図 8** に示すようにマッチすべき部分がおおむね合っていて，先頭または最後は同じ音高が続いているものだった．図 7 のケースは，ハミング検索という観点では，切り出された楽曲の一部とハミングされた部分が合致しているが検索対象の楽曲のテンポと入力されたハミングのテンポが異なる場合に相当する．また図 8 のケースは，ユーザは最後の音符を正しい時間分（拍数分）伸ばして発声したが，楽曲データの方は途中で切り取られていた場合などに相当する．これは DTW では音符の長さが考慮されないので，最初または最後の音符が長く演奏されても短く演奏されてもマッチングの結果が変わらないことに起因している．またここでは図示していないが，正解との距離が非常に大きいものもあった．これはその部分データによるメロディが稀で，ハミングが正しいメロディからかなりずれていても検索されやすかったということを意味している．

これに対し 25 位以内に検索されなかったハミング

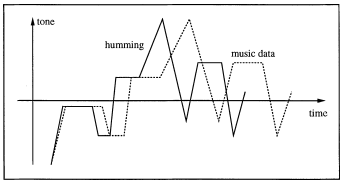

図 7 メロディという観点で情報量が等しい 2 つのシーケンス
Fig. 7 Two sequences that hold the same quantity of information in terms of melody.

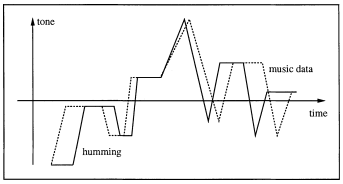

図 8 マッチする部分は合っていて，最初と最後に同じ値が連続するシーケンス
Fig. 8 A sequence that partially matches another one and has the same values at the beginning and the end.

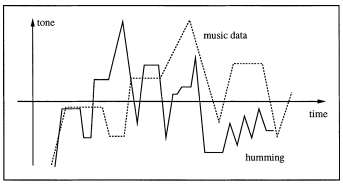

図 9 メロディという観点で情報量が等しくない 2 つのシーケンス
Fig. 9 Two sequences that do not hold the same quantity of information in terms of melody.

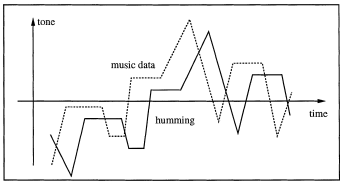

図 10 全体的にずれている 2 つのシーケンス
Fig. 10 Two sequences that are mismatched on the whole.

は，**図 9** に示すようにメロディという観点から情報量が等しくないものや，**図 10** に示すようにマッチすべき部分が全体的にずれていたり，音高のシフティングに失敗していたりしたものだった．図 9 のケースは，ハミング検索という観点では，ユーザは楽曲データと同じ時間分のハミングを入力したが，楽曲のテンポとハミングのテンポが異なっていたために，結果的にユーザが余計な部分までハミングしてしまった場合などに相当する．また，図 10 のケースは，ユーザは楽曲データと同じテンポで同じ時間分のハミングを入力したが，そもそも楽曲から切り取られた部分データとハミングした部分が合致していなかった場合などに相当する．情報量やマッチすべき部分が合わなければ，合っていない部分の距離がどんどん大きくなり，平均音高も変化するのでシフティングにも失敗する．

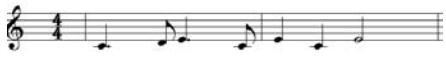

図 11 ドレミの歌の出だしのメロディ
Fig. 11 Starting melody of the song “Do-Re-Mi”.

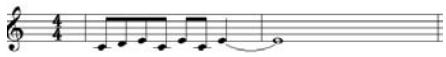

図 12 ドレミの歌の出だしにおいて，各音符の音高は変えずに音価を八分音符に変えたメロディ
Fig. 12 A melody that has the same tones as “Do-Re-Mi” but with different phonetic values.

マッチすべき部分が合っていないということは，すなわちマッチングする列データどうしの始点と終点が合っていないということを意味する．図 7 や図 8 のようなシーケンスどうしがマッチしていることから，DTW は時間の伸縮に柔軟に対応できることが分かるが，図 10 のように比較するデータどうしが全体的にずれていて，始点と終点が合っていない場合はその効果が発揮されないということが分かる．

また図 8 のようなシーケンスがマッチするということは，そもそも DTW は音楽データのマッチングには向かない可能性がある．DTW の特性として，時間軸方向の伸縮に左右されないマッチングが可能であることが知られている．したがって，図 8 のように 1 つの音高値でも 2 つ以上その音高値が連続して存在しても距離には影響を与えない．これは音声認識などには都合がよいが，音楽データのマッチングには良い面も悪い面もある．良い面は，多少のリズムの狂いに影響されないマッチングができることである．悪い面は，音楽の最大の特徴の 1 つである音長情報をマッチングに利用できないことによる分解能の低下である．たとえば**図 11** と**図 12** のような 2 つの楽曲を考えてみる．図 11 はドレミの歌の出だしで，図 12 はドレミの歌の出だしにおいて，各音符の音高は変えずに音価を 8 分音符などに変えたメロディである．どちらの楽譜もテンポは 120 とし，0.25 秒ごとの音高を MIDI の音高番号を使って時系列データ化すると，(60, 60, 60, 62, 64, 64, 64, 60, 64, 64, 60, 60, 64, 64, 64, 64) と (60, 62, 64, 60, 64, 60, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64) となる．これを DTW でマッチングすると距離はゼロとなる．すなわち DTW ではそれぞれを異なるメ

ロディとして区別することはできない．しかし，これらのメロディは同じとはいえないし，似ているともいえない．それにもかかわらず距離がゼロとなるのは，マッチングの際に音長情報を使用していないので分解能が十分ではないことが原因であると思われる．楽曲データベースが大きくなると，このようなケースは増えると考えられる．

以上の議論から，マッチングの際には，

- 比較すべき双方のデータの始点と終点を合わせなければならないこと，
- 分解能を向上させるためには，音長情報を活用すべきであること，

が分かる．DTW において始点と終点を合わせることはよりいっそうコストがかかる．したがって，ハミング検索のように入力されるデータの伸縮度合いと長さが一定ではない音楽内容検索に DTW を使用するのは，コストと精度の観点から現実的ではない．

5. 音楽データの自動時間正規化手法

音長情報を利用して，かつ効率的なマッチングを行うためにはリニアマッチングが有効だと考えられる．本章では，音長情報を利用した音楽データの効率的なマッチングのための音楽データの時間正規化を自動で行う手法を提案する．

マッチすべきデータどうしの始点と終点が一致しているということは，双方のデータの始点と情報量が等しいということを意味する．しかし，音楽データは時間的にスケーラブルなデータなので，「時間」を用いて始点の位置や情報量を定義して利用するのは難しい．そこで，まず 5.2 節では，音楽データを時間の伸縮に左右されないデータに変換したり情報量を定義したりするための基底となる情報量を定義する．これを基底単位長と呼ぶ．5.4 節では基底単位長を基に音楽データを等しい情報量を持つ部分シーケンスに分割する．この部分シーケンスを子/孫シーケンスと呼ぶ．最後に，5.5 節で子/孫シーケンスから始点の合いやすい特徴ベクトルを作成する手法を述べる．

5.1 オンセット情報処理

本論文では，ハミングについては発声開始から次の発声開始までを 1 回の「発声時間」とし，楽曲データについては 1 つの音符の再生開始から次の音符の再生開始までを，その音符の「再生時間」とする．これをオンセット情報処理[7]と呼ぶ．

歌唱における発声開始のタイミングは，楽曲における音符の再生開始のタイミングを再現している可能性がきわめて高い．なぜなら発声のタイミングが意図しているものから外れるとリズムが狂い，そのハミングが表現している楽曲を認知する/されるのが難しくなるからである．したがって人は歌唱する場合は発声の音高と同様に発声開始のタイミングには非常に注意をはらう．これに対し発声終了のタイミングはその定義が曖昧であり，ハミング検索のためのクエリを作成するという観点からあまり重要な情報ではない[10]ので，本論文では利用しない．

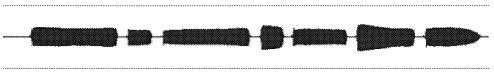

図 13 “タ” で歌った場合の音声信号
Fig. 13 Voice signal waves sung with “ta.”

この発声開始のタイミングは，音声信号の振幅の変化率や，FFT によるフレームごとのパワーの変化率から比較的正確に推定することができる．**図 13** にドレミの歌の出だしを「タ」で歌った場合の音声信号の波形データを示す．図 13 から分かるように，「タ」のような無声破裂音から始まる音で発声された場合は，発声の直前にストップギャップと呼ばれる無音区間ができる[17]ため，かなり正確に発声開始のタイミングを推定することができる．

5.2 基底単位長

本論文では音楽データを時間正規化し発声時間や再生時間を相対値で表現するための基底となる情報量を「基底単位長」と呼ぶことにする．基底単位長は，ハミングについてはハミング内で最頻出する発声時間とする．これは，ハミングデータにおいて最頻出する発声時間を基底に時間正規化するのが最も安定に時間正規化を行うことができるからである．これに従って，楽曲データについては部分音楽片（本論文では，子/孫シーケンスと呼ぶ．5.4 節参照）内で最頻出する再生時間を基底単位長とする．なお，最頻出の発声時間/再生時間が複数あった場合は，それぞれを基底単位長として定義する．

ハミング中の各発声時間を L_{h_i}，楽曲データ中の各再生時間を L_{s_j} としたとき，それぞれの基底単位長を U_h，U_s とすると，それぞれは以下の式を用いて時間正規化される．

$$L'_{h_i} = L_{h_i}/U_h \qquad L'_{s_j} = L_{s_j}/U_s \qquad (1)$$

この手法によって時間正規化された音楽データは，我々が以前から提案し，その効果を実証している拍ベース[1),2)]で処理された音楽データに似ており，各発声時間/再生時間を反映した特徴ベクトルの作成など，拍ベースの音楽データを利用する際の利点を利用することができる．しかも，従来の拍ベース処理のような

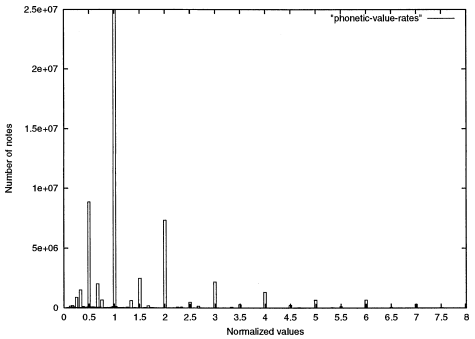

図 14 時間正規化された再生時間の分布
Fig. 14 Distribution of time-normalized tone-duration.

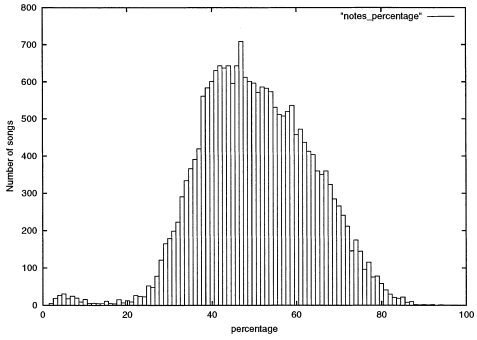

図 15 最頻出する再生時間（1 種）がその曲内で占める割合
Fig. 15 The percentage of the most frequently appearing tone-duration covered in each song.

テンポ情報を用いた音楽データの時間正規化と違って，ハミングや部分音楽データそのものから抽出できる情報を基底単位長とするので，より実情にあった時間正規化を行うことが可能となる．

5.3 音価レート表現

音価の種類は，一般的な四分音符や八分音符に加えて，付点つきのものや飾り音符まで入れると相当なバリエーションがあることは想像に難くないが，本論文で使用する MIDI データにおいては，さらにたくさんのバリエーションが存在する．たとえば標準的な四分音符は 480 ティックで表現されるが，本論文で使用する楽曲データにおいては，460，470，または 490，500 ティックなどで表現されることも多い．これは MIDI データの作者ができるだけ実際の演奏を再現するように努力しているからである．しかしこれらのわずかな音長の違いをユーザが正確に使い分けて発声することはほとんどない．同様に，時間正規化された発声時間（L'_{h_i}）におけるバリエーションも，ユーザが意図したものではなく単なる歌い間違いであることが多い．したがってこの微妙なバリエーションをそのままマッチング用のデータに反映させるのは意味がないし，ハミング検索システムにとってむしろ危険である．そこでいくつかの代表値を用いて限定された値にマップする手法を提案する．この代表値を「音価レート値」として以下の 8 種と定義し，値の単位を「音価レート」と呼ぶことにする．

$$0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 5.0, 6.0 \quad (2)$$

ハミングも楽曲データも，基底単位長により正規化された後は，この 8 種の値だけで長さの情報が表現される．

これらの音価レート値は時間正規化された再生時間の出現頻度と事前実験から決定した．**図 14** に時間正規化された再生時間の分布を示す．この図は各部分シーケンス（本論文では，子/孫シーケンスと呼ぶ．5.4 節参照）を正基底単位長（5.4 節参照）で時間正規化した際の実値の分布である．ただし 8.0 以上になるものはきわめて少ないのでこの図に表示していない．図 14 より，1.0，0.5，2.0 の値が特に多く，1.5，3.0，4.0，5.0，6.0 が比較的多いことが分かる．また，事前実験によりユーザは音符の長さが長くなればなるほど正しく伸ばしきれないことが分かった．さらに再生時間はオンセット情報処理で評価しているので，時間正規化された再生時間と実際の音価とは，値が大きくなるほど一致しない．そこで音価レート値としては 6.0 を最大値として，それ以上長いものもすべて 6.0 にマップすることとする．このように時間正規化された再生時間を限定された代表値で表現することで，多少のテンポのずれなどには頑健な時間正規化データを作成することができる．

5.4 子シーケンスと孫シーケンス

ハミングはその中で最頻出の発声時間で時間正規化される．そしてハミングされるのは楽曲の一部である．したがって，楽曲データも部分データごとにその部分の最頻出再生時間で時間正規化されていなければ，ハミングと共通のスケールで情報量を評価することはできない．しかし楽曲内のどの部分をとってもその部分で最頻出の再生時間が同じであるとは限らない．**図 15** は 1 つの楽曲内で最頻出する再生時間（1 種）がその楽曲内で占める割合を表す．横軸がその割合で縦軸が曲数を表している．多くの楽曲において最頻出する再生時間が占める割合は約 40%から 60%である．したがって，楽曲のどの部分も最頻出の再生時間で時間正規化されている状態にするためには，楽曲全体を通しての最頻出再生時間だけで時間正規化したのでは不十分である．しかし，楽曲のどの部分をとっても最頻出の再生時間で時間正規化されている状態にするためには，まず「部分」を定義しなければならないが，音楽

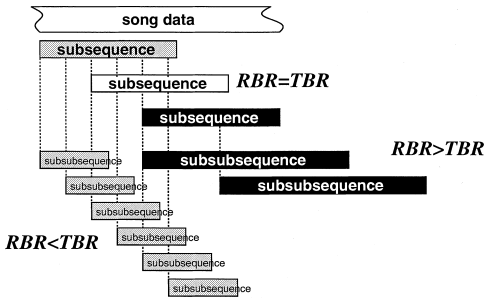

図 16 子/孫シーケンスの作成
Fig. 16 Generation of sub/subsubsequences.

データには時間伸縮性があるので，時間を基準に部分データを定義するのは効果的ではない．そこで我々は「部分」を定義するための時間正規化（仮時間正規化）と，その「部分」に対して，その中での最頻出の再生時間を抽出し，その再生時間を基に部分データの情報量を調整するための時間正規化（正時間正規化）という，2 段階の時間正規化手法を提案する．仮時間正規化によって作成される部分データを子シーケンス，正時間正規化によって作成される部分データを孫シーケンスと呼ぶ．

まず楽曲データ全体を通して最頻出する再生時間を「仮基底単位長（TBR）」とする．そして仮基底単位長を用いて楽曲データをいったん仮時間正規化し，音価レート値に変換してから一定のずらし幅ごとに固定長のデータを切り出す．これを子シーケンスと呼ぶ．次にこの子シーケンスごとに最頻出する再生時間を調べ，仮基底単位長が最頻出ではなかった場合は，最頻出の再生時間を正基底単位長（RBR）としてスライド幅とウィンド長を再定義し部分シーケンスを作り直す．これを孫シーケンスと呼ぶ．**図 16** に子/孫シーケンスの作成を示す．

- RBR > TBR のとき
  RBR で計る子シーケンスの情報量がウィンド長に満たないのでその分の情報量を付け足して孫シーケンスを作る．
- RBR < TBR のとき
  RBR で計測る子シーケンスの情報量がウィンド長を超えるので，超過した分は切り捨てて孫シーケンスを作る．

5.5 特徴ベクトル

本論文では以下の 2 パターン 3 種類の特徴ベクトルを使用する．ベクトルの略称を（）内に示す．

- 音高値の時系列特徴ベクトル
  - 固定スライド幅の特徴ベクトル（CSTPV）
  - 可変スライド幅の特徴ベクトル（VSTPV）

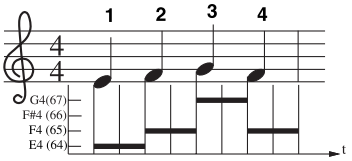

図 17 音高値の時系列特徴ベクトルの作成を説明するためのメロディ例
Fig. 17 Example melody to describe how to make tone values' time-series feature vector.

- 音高差分布特徴ベクトル[2]（DTD）

5.5.1 音高値の時系列特徴ベクトル

音楽鑑賞において大きな楽しみの 1 つは音の高さと長さの時間的な変化に耳を傾けることであり，これこそが音楽の最も重要な特徴の 1 つであるといえる．そこで，それを表現する特徴ベクトルとして 1 章の時系列データの定義に従って，1 音価レートごとの音高値をベクトルの要素とする「音高値の時系列特徴ベクトル」を作成する．データベース内の楽曲とは異なるキーのハミングでも検索可能にするために，ベクトル内の各要素は，ある音高を基準としてシフティングする．この基準となる音高を「基準音」と呼ぶ．

音高値の時系列特徴ベクトルの作成方法を**図 17** を用いて説明する．E4，F4，G4 は音名☆で，その右の数字は MIDI の音高番号である．楽譜内の 1 から 4 の数字は音符の番号である．このメロディから作られる音高値の時系列特徴ベクトルは，このメロディを含む子/孫シーケンスの基底単位長に依存する．たとえば，このメロディ内で最も高い音（この場合は 67）を基準音とすると，四分音符が基底単位長の場合は，4 つの各音符に 1 音価レートが割り当てられ，(−3, −2, 0, −2) という 4 次元分の音高値の時系列特徴ベクトルの一部ができる．また八分音符が基底単位長の場合は，4 つの各音符に 2 音価レートが割り当てられ，(−3, −3, −2, −2, 0, 0, −2, −2) という 8 次元分の音高値の時系列特徴ベクトルの一部ができる．

5.5.2 固定スライド幅の特徴ベクトルと可変スライド幅の特徴ベクトル

5.2 節の手法で楽曲データもハミングデータも同じスケールの音楽データに変換され，5.4 節の手法で保持する情報量が等しい子/孫シーケンスに分割された．本項では，その子/孫シーケンスから，始点を合わせやすい音高値の時系列特徴ベクトルを作成するための手法について述べる．

☆ 音名とは正しくは「楽音の基礎になる 7 つの音の名称」[18]であるが，本論文では便利のためそれに高さ（オクターブ単位）を表す数字を合わせた表記を音名と表現した．

楽曲データもハミングデータもスライディング・ウィンド方式で分割されるので，ハミングの先頭と子/孫シーケンスの先頭は，最大でスライド幅の半分ずれている可能性がある．我々は効果的にベクトルの先頭を合わせる手法として，ベクトルの先頭を部分シーケンスの先頭とは独立に決定する手法を提案してきた[2)]．本論文でもベクトルの先頭を子/孫シーケンスの先頭とは独立に決定する手法を導入する．結果的に可変スライド幅で作成された子/孫シーケンスから特徴ベクトルを作成していると考えることができるので，この手法で作成する特徴ベクトルを「可変スライド幅の特徴ベクトル」と呼ぶ．この可変スライド幅の特徴ベクトルに対し，ベクトルの先頭を子/孫シーケンスの先頭と一致させる特徴ベクトルを「固定スライド幅の特徴ベクトル」と呼ぶ．

スライド幅を sl 音価レート，ウィンド長を wl 音価レートとしたとき，ある子/孫シーケンス s_i から，wl 次元の固定スライド幅の特徴ベクトル $CSTPV_i$ と，$(wl - sl)$ 次元の可変スライド幅の特徴ベクトル $VSTPV_i$ を作成する．$VSTPV_i$ は，$CSTPV_i$ の先頭から sl 次元内で最初に出現する最も特徴的な音高の存在する次元を先頭とするベクトルである．たとえば，最も特徴的な音を「最も高い音高」とし，ベクトルの各要素を $e_i(k)$ と表すと，各ベクトルは以下の (3) から (6) のように表現することができる．

$$CSTPV_i = \{e_i(1), e_i(2), e_i(3), \cdots, e_i(wl)\} \quad (3)$$

$$VSTPV_i = \{e_i(j), e_i(j+1), \cdots, e_i(j+v-1)\} \quad (4)$$

$$v = wl - sl \quad (5)$$

$$j = \arg\max_k e_i(k) \quad \{k | 1 \leq k \leq sl\} \quad (6)$$

固定スライド幅の特徴ベクトルと，可変スライド幅の特徴ベクトルの関係を**図 18** に示す．この図から分かるように，可変スライド幅の特徴ベクトルは固定スライド幅の特徴ベクトルの一部からなるベクトルとなる．なお，最も特徴的な音高として，「最も高い音高」，「最も低い音高」，「最も長い音符の音高」，「最も多く出現する音高」の 4 種について予備実験を行った結果，最も高い音高を用いた場合が最も高い精度で検索できることを確認した．

この可変スライド幅の特徴ベクトルを導入することの利点は，楽曲データの子/孫シーケンスの先頭とハミングデータの子シーケンスの先頭が一致していなくても，それぞれから先頭を同じくする音高値の時系列特徴ベクトルを作成できる可能性があることである．たとえば，データベースのある子シーケンスの先頭が図 17 の 1 番目の音符から始まるメロディだったとする．またユーザは図 17 の音符列の 3 番目の音符からハミングしたとする．このときハミングから作成される子シーケンスの先頭は 3 番目の音符になる．双方の子シーケンスから固定スライド幅の特徴ベクトルを作成すると，それぞれ 1 番目の音符を先頭とする固定スライド幅の特徴ベクトルと，3 番目の音符を先頭とする固定スライド幅の特徴ベクトルが作られて，マッチすべき双方の固定スライド幅の特徴ベクトルの先頭は一致しない．これに対し，可変スライド幅の特徴ベクトルを導入し，ベクトルの先頭をスライド幅内で最も高い音高が最初に現れる次元とし，基底単位長は四分音符でスライド幅は 4 音価レートとすると，楽曲データの子シーケンスから生成される可変スライド幅の特徴ベクトルの先頭は 3 番目の音符の位置，すなわち固定スライド幅の特徴ベクトルの 3 次元目になる．ハミングについては，3 番目の音符からハミングしていることから，図 17 には 2 音価レート分しか現れていないので，もし残りの後続の 2 音価レート中に 67 を超える音高の音符がなければ，やはり可変スライド幅の特徴ベクトルの先頭は 3 番目の音符の位置，すなわちハミングから作成される固定スライド幅の特徴ベクトルの 1 次元目となり，楽曲データとハミングデータの双方の可変スライド幅の特徴ベクトルの先頭は一致する．もちろんハミングの後続の 2 音価レート中に 67 を超える音高の音符が存在すれば，その音符が可変スライド幅の特徴ベクトルの先頭になるので，図 17 の音符列を先頭とする楽曲データの子シーケンスから作られる可変スライド幅の特徴ベクトルとは先頭が一致しない．しかし楽曲データの図 17 の音符列の次の子シーケンスから作られる可変スライド幅の特徴ベクトルと先頭が一致する可能性もある．このように，可変スライド幅の特徴ベクトルを導入することによって子/孫シーケンスの先頭とは独立して，ある共通ルー

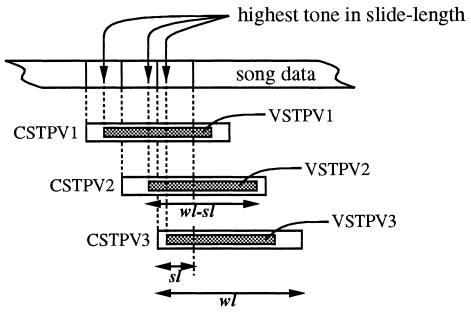

図 18 固定スライド幅の特徴ベクトル（CSTPV）と，可変スライド幅の特徴ベクトル（VSTPV）
Fig. 18 Constant slide-length feature vector and variable slide-length feature vector.

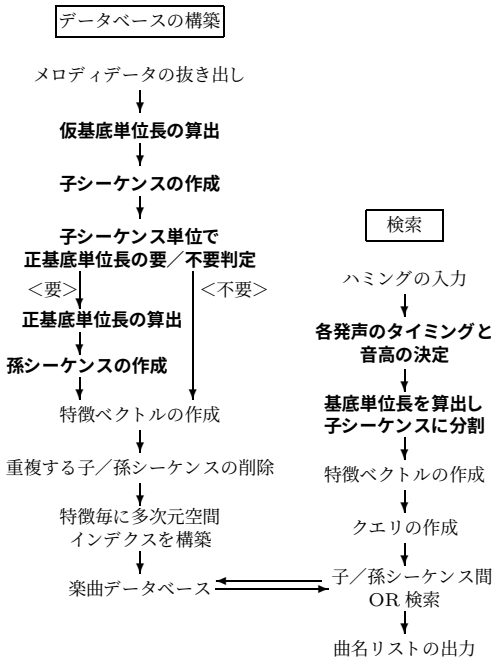

図 19 新しい SoundCompass の処理の流れ
Fig. 19 Process flow for new SoundCompass.

ルに従って特徴ベクトルの先頭を決めることができるので，双方の特徴ベクトルの先頭を一致させられる可能性が増す．

6. SoundCompass の改良

本章では，5 章で提案した技術を導入した新しい SoundCompass について述べる．**図 19** に処理の流れを示す．図 3 の従来の SoundCompass と異なる処理を太字で示す．

6.1 データベースの構築

図 19 の左側にデータベースの構築処理の流れを示す．メロディデータを抜き出した後，楽曲全体を通して最頻出する再生時間を仮基底単位長として楽曲全体を仮時間正規化し，各再生時間は音価レート値に整形する．次に仮基底単位長を基に楽曲をスライディング・ウィンド方式で子シーケンスに分割する．スライド幅とウィンド長をそれぞれ 8 音価レート，32 音価レートとした．なぜなら図 5 から最頻出の音価は八分音符であることが分かるので，多くの曲において 1 音価レートが八分音符に相当していると考えられる．また表 2 から全曲の 97%が 4/4 拍子であることが分かる．フレーズをまとまり感のある短いメロディとするならば，4/4 拍子の曲の多くのフレーズは 4 小節程度であり，4/4 拍子の 4 小節は 16 拍，すなわち八分音符 32 個分である．そこで子/孫シーケンスの長さを 32 音価レートとし，ずらし幅は 1 小節分の 8 音価レートとした．子シーケンスに分割した後，仮基底単位長がその子シーケンス内で最頻出の再生時間ではなかった場合，正基底単位長を算出して孫シーケンスを作成する．孫シーケンス内の各再生時間を正基底単位長によって時間正規化し，音価レート値に変換する．特徴ベクトルは各子/孫シーケンスから作成し，重複排除を行った後，特徴量ごとにデータベースに登録する．

6.2 ハミングデータの処理

図 19 の右側にハミングデータの処理を示す．音声はマイクを通して，11 kHz/8 bit/monoral で収録する．新しい SoundCompass では，ハミングを市販の採譜ソフトで MIDI データ化するのではなく，FFT を用いた音声信号処理を行って，発声開始のタイミングと発声の音高を抽出する．できるだけ効果的な信号処理をするために，ユーザには「タ」を使ってハミングしてもらう．発声開始のタイミングは，主に波形データの振幅の変化率から検出する．各フレームの発声音高は，256 点ずつずらしながら 512 点の音声信号を FFT を用いて算出する．本論文では，1 回の発声で 1 つの音高のみを発声すると仮定し，各フレームの音高と発声開始フレームの情報から，1 回の発声の音高と発声時間を決定する．歌詞のある多くのメロディでは，1 つの音符に 1 つの音節（日本語の場合は 1 文字）が割り当てられているので，1 回の発声で 1 つの音高が発声されるという仮定は自然である．次に発声フレーム数の分布からハミングデータを時間正規化するための基底単位長を決定する．その基底単位長を用いて各発声時間を時間正規化し，音価レート値に変換して，子シーケンスを作成する．各子シーケンスからは，データベース作成時と同じ特徴量の特徴ベクトルを作成し，それを用いてクエリを作成する．

6.3 検　　索

類似検索では，ハミングから作成された特徴ベクトルに似ている特徴ベクトルを楽曲データベースから見つけ出す．最終的には，ハミングに類似している部分を持つ曲が類似度順に並べられた，曲名のランキングリストとして出力する．類似度は特徴ベクトル間の距離で表現し，類似しているものほど距離が短いとする．距離関数には City-block 距離を使用する．

ウィンド長より長いハミングが入力された場合は，複数の子シーケンスを作成することができるので，個々の子シーケンスの検索結果を OR 方式で統合して最終結果を作成する[2]．ハミング h から m 個の子シーケンスができ，曲 s から n 個の子/孫シーケンスができ，曲 s の i 番目の子/孫シーケンスとハミングの j

番目の子シーケンスとの距離を $d(s_i, h_j)$ としたとき，曲 s とハミング h との距離 $D(s,h)$ を，

$$D(s,h) = \min_{1 \le i \le n, 1 \le j \le m} \{d(s_i, h_j)\} \qquad (7)$$

とする．この手法を子シーケンス間 OR 検索と呼ぶ．

一般的に，複数の特徴量を用いて検索したほうがより正確に正解を検索できる．文献 2) と 5.5 節では複数の特徴ベクトルを考案したが，検索に使用する場合は組み合わせ方に注意が必要である．楽曲の子/孫シーケンスの先頭とハミングの子シーケンスとの先頭が合わない場合の解決策として可変スライド幅の特徴ベクトル（VSTPV）を導入したが，事前実験から固定スライド幅の特徴ベクトル（CSTPV）ではベクトルの先頭が合わずに可変スライド幅の特徴ベクトルによって先頭が一致したものもあれば，逆に固定スライド幅の特徴ベクトルでは先頭が合うのに，可変スライド幅の特徴ベクトルでは先頭がずれてしまうものがあることが分かった．したがって，検索にはこの 2 つの特徴量は別々に使用し，それぞれの検索結果を独立にマージする OR 方式での利用が効果的だと考えられる．

また，音高値の時系列特徴ベクトルでは，同じ音高の連続した 8 分音符 8 つと全音符 1 つを区別できないので，音符数/発声回数を表現する音高差分布特徴ベクトル（DTD）を合わせて利用するのが効果的だと考えられる．音高差分布特徴ベクトルは子/孫シーケンスの先頭の違いにはそれほど左右されないと考えられるので，音高値の時系列特徴ベクトルとは AND 方式で使用する．

まとめると，本論文では 7 章での SoundCompass の検索精度評価実験には，これら 3 種類の特徴ベクトルを “(VSTPV && DTD) || (CSTPV && DTD)” という組合せで使用する．

7. 評　　価

7.1 実 験 環 境

本評価実験における実験システム構成を**図 20** に示す．ネットワーク経由で検索エンジン/データベースと SoundCompass サーバ（ハミング検索サーバ）とクライアント PC がつながっている．クライアント PC にはマイクがつながっている．入力された歌声は SoundCompass サーバに送られる．SoundCompass サーバはハミングを 6.2 節に記載した方法で処理し，クエリを検索エンジンに送信する．検索エンジンは類似検索を行い結果を SoundCompass サーバに返送する．SoundCompass サーバは検索結果を整形してクライアント PC に返信する．本章の評価実験で使用する楽曲データとハミングデータは，4 章で使用したものと同じである．

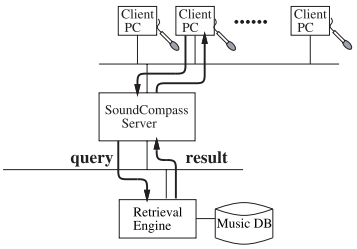

図 20 実験システム
Fig. 20 Experimental system.

表 3 時間正規化と子/孫シーケンス作成の精度
Table 3 Accuracy of time-normalization and sub/subsubsequence generation.

	基底単位長数		
	1	2	合計
子シーケンス	496 (83.93%)	17 (2.88%)	513 (86.81%)
孫シーケンス	59 (9.98%)	2 (0.33%)	61 (10.31%)
合計	555 (93.91%)	19 (3.21%)	574 (96.15%)

7.2 時間正規化手法の効果

5 章の手法で作成された子/孫シーケンスは 2,236,962 個で，4 章で用いた DTW のデータベースのデータ数とほぼ同じとなった．全ハミングデータについて，ハミングと同じように時間正規化された子/孫シーケンスが存在したハミングの個数を集計し**表 3** に示す．「同じように時間正規化されている」とは，「同じ曲のある音符について，ハミングデータにも楽曲データにも同じ音価レート値が割り当てられている」という意味である．表 3 では，1 つのハミングデータから，最頻出する発声時間のみを基底単位長として時間正規化した場合と，それに加えて，2 番目に頻出する発声時間も基底単位長にして時間正規化した場合について，それぞれハミングと同じように時間正規化された子/孫シーケンスがデータベースに存在したハミングの数を示す．

表 3 に示すように基底単位長を 1 つだけしか使用しない場合は，ハミングの 93.91%について同じように正規化された子/孫シーケンスがデータベースに存在し，基底単位長を 2 つまで使用すると，96.15%のハミングについて，同じように時間正規化された子/孫シーケンスがデータベースに存在することが分かる．またハミングの 10.31%が，子シーケンスではなく孫シーケンスの方で同じように時間正規化された楽曲データを見つけられるということが分かるので，本論文で提案する子/孫シーケンスの作成が有効であることが確

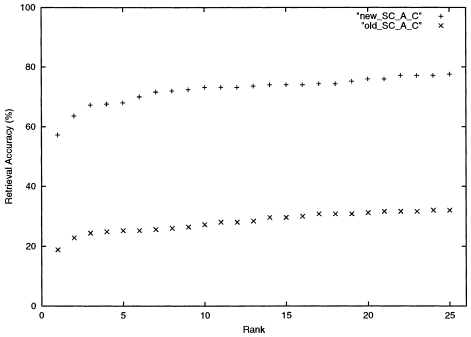

図 21 ハミングセット A と C に対する検索精度
Fig. 21 Retrieval accuracy for humming set A and C.

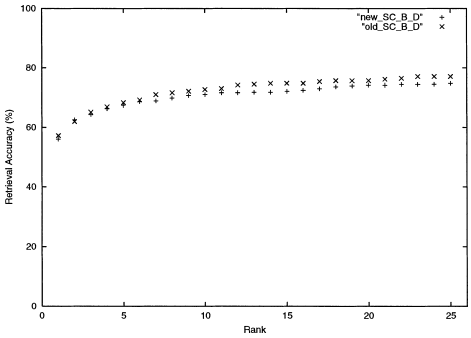

図 22 ハミングセット B と D に対する検索精度
Fig. 22 Retrieval accuracy for humming set B and D.

認できる．

7.3 検索精度評価

本節では，5 章で提案した手法の検索精度に対する効果を定量的に評価する．本論文における検索精度とは，検索結果の特定の順位以内に，正解が検索されたハミングの割合であると定義する．

7.3.1 従来の SoundCompass との検索精度比較

従来の SoundCompass では，検索にはハミング時に設定したメトロノームのテンポ値が必要なので，ハミングセット B と D についてはそのときのメトロノームのテンポ値を用いて，ハミングセット A と C についてはテンポ値が分からないので 120 を用いて検索を行った．実験結果を**図 21** と**図 22** に示す．図 6 と同様に横軸は順位を示す．縦軸は検索精度を示しており，その順位以内に検索されたハミングの割合を示している．図中の記号で，“new_SC” がついている方が音楽データの自動時間正規化を導入した SoundCompass による検索結果で，“old_SC” がついている方は従来の SoundCompass による検索結果である．

図 21 よりメトロノームを用いずに歌ったハミングについては，従来の SoundCompass では事実上検索不可能だったが，音楽データの自動時間正規化を導入した SoundCompass を用いることで，検索精度を大きく向上させられることが分かる．また図 22 よりメトロノームに合わせて歌ったハミングについても，音楽データの自動時間正規化を導入した SoundCompass の検索精度は従来の SoundCompass の検索精度を下回らないことが分かる．よって，本論文で提案する音楽データの自動時間正規化手法が，伸縮度合いも長さも事前には分からない音楽データどうしの部分マッチングに対して有効であることが分かる．

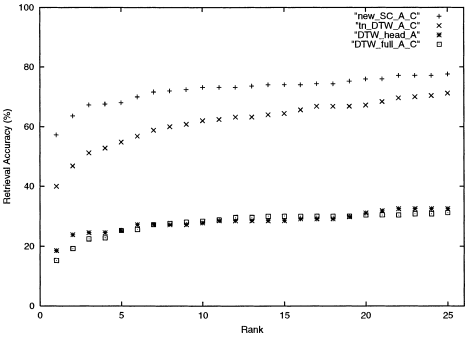

図 23 4 つの手法の検索精度実験結果
Fig. 23 Retrieval accuracies of four techniques.

7.3.2 音楽データの自動時間正規化を導入した DTW によるハミング検索と SoundCompass の検索精度の比較

本項では自動時間正規化した音楽データに対して，

- DTW による検索
- City-block 距離による距離計算と複数の特徴ベクトルを用いた検索（SoundCompass）

の検索精度を比較する．**図 23** に実験結果を示す．見方は図 21 や図 22 と同様である．図中の “tn_DTW_A_C” は DTW による検索を行った場合の検索結果を示す．“new_SC_A_C” は図 21 と同じで City-block 距離による距離計算と複数の特徴ベクトルを用いた場合の検索結果を示している．“DTW_head_A” と “DTW_full_A_C” はどちらも図 6 に示した結果と同じで，時間正規化していない音楽データに対して DTW を用いて検索した場合の検索結果である．

図 23 より “DTW_full_A_C” と “tn_DTW_A_C” を比較すると，“tn_DTW_A_C” の方が検索精度が高いことが分かる．これは楽曲データ/ハミングデータが時間正規化されて，スライディング・ウィンド方式を用いた分割により，等しい情報量を持つデータどうしの公正なマッチングを行えたからである．本結果からも本論文で提案する音楽データの自動時間正規化手法が有効であることが分かる．

また “new_SC_A_C” の方が “tn_DTW_A_C” よりも

検索精度が高いのは，可変スライド幅の特徴ベクトルの導入によって，マッチすべき双方の特徴ベクトルの先頭が合っているケースが増えたことと，複数の特徴量を使用することで分解能が向上したことがあげられる．特に可変スライド幅の特徴ベクトルと固定スライド幅の特徴ベクトルを OR 方式で検索に使用したことで，両方のベクトルの特性を有効に活用できたことが精度向上につながったと考えられる．

一方，4.3 節で DTW をハミング検索に利用した場合の懸案事項として分解能の問題を指摘した．しかしシフティングの際に平均音高値を基準音にすることにより，この問題は若干緩和することができる．たとえば DTW では (1, 1, 1, 2, 3) と (1, 2, 3, 3, 3) の距離は 0 であるが，データの平均値（実数）を用いてシフティングするとそれぞれは，(−0.6, −0.6, −0.6, 0.4, 1.4) と (−1.4, −0.4, 0.6, 0.6, 0.6) となり距離が 0 にはならない．ところが最も大きい値（この場合は 3）を 0 としてシフティングするとそれぞれは，(−2, −2, −2, −1, 0) と (−2, −1, 0, 0, 0) となり，やはり距離は 0 になってしまう．実際に予備実験として，四捨五入して整数値化した平均音高値を用いてシフティングしたところ，検索精度は全体的に約 10%低下し，さらに最も高い音高値を用いてシフティングしたところ約 15%検索精度が低下した．このことからも，音楽データのマッチングには音長情報も利用すべきであることが分かる．

7.4 検索時間

本節では，音楽データの時間正規化を導入した SoumdCompass の検索実行時間について述べる．実験には，Dell PowerEdge 2600（Xeon(R) 3.2 GHz × 2，主記憶 4 GB）1 台を使用した．検索時間として，SoundCompass サーバがクエリを発行して検索エンジンから検索結果を受け取るまでの時間を測定したところ，21,804 曲のデータベースに対して約 2.7 秒だった．

7.5 音楽データの自動時間正規化を導入した SoundCompass での検索ミスに関する議論

本論文で提案した手法が効果的であることは定量的に確認できた．しかし，音楽データの自動時間正規化と子/孫シーケンスの作成が正しくできることは表 3 の結果が示しているが，図 23 などが示す音楽データの自動時間正規化を導入した SoundCompass の検索精度評価結果はそれとつり合っているようには見えない．その理由の 1 つは音声信号処理のミスによる影響であり，もう 1 つはシフティングの失敗である．

まず音声信号処理によるミスについて整理する．発声開始フレームの検出率を，「プラスマイナス 1 の範囲で実際の発声回数と検出された発声回数が合致しているハミングの割合」と定義するならば，本論文では 591 個のハミングに対して検出率は 92.22%であった．さらに音高決定の過程でもミスが発生すると考えると，現状の SoundCompass の正解率の上限は，約 80〜85%程度ではないかと考えられる．ハミング検索システムにとっては，信号処理の精度とマッチングの精度を切り放して論じることはできないのである．また，ハミング被験者の数が増すと，信号処理によるミスは増える．これは被験者が増えるたびに声質の種類や歌いかたの癖の種類が増え，その時点の処理では十分に対応しきれない音声信号が含まれている可能性が増すからである．したがって被験者が増すたびに，検索できなかったハミングの原因を究明して，原因が音声信号処理のミスにあった場合は，そのアルゴリズムや実際の音声信号処理で使用している閾値などを改良していかなければならない．ある程度たくさんの被験者による実験を重ねて，統計的な知見を得て音声信号処理方法を見直していくことが必要である．

また音高のシフティングについてであるが，基準音として本実験では子/孫シーケンスごとに音高の平均値を算出し，その平均音高が 0 になるように楽曲データもハミングデータもシフティングした．この方法はおそらく最も妥当であると考えられるが，弱点は基準音が子/孫シーケンス全体に依存することである．たとえば，子/孫シーケンスの中で比較的長く発声される音が誤った音高で発声されていた場合などは，平均音高の算出に悪影響を与え，適切な音高シフティングを行えなくなってしまう．予備実験では子/孫シーケンス内の最も高い音高を基準音とした場合の検索精度も定量的に評価した．この方法の良いところは，基準音が子/孫シーケンス全体に依存しないことであるが，欠点は選んだ値が悪いとやはり適切なシフテイングが行えないことである．本実験では，基準音に最も高い音高を用いた場合も平均音高を用いた場合も，音楽データの自動時間正規化を導入した SoundCompass の検索精度はあまり変わらなかった．7.3.2 項において，DTW の場合は音高のシフティングによって検索精度に大きな差が出たことを考えると，リニアマッチングはシフティングに対して安定的だといえるが，今後もさらに効果的なシフティングの方法について検討が必要である．

8. まとめ

本論文では，ハミング検索のための音楽データの自動時間正規化手法を提案した．音楽データを「時間ス

ケーリングと音高シフティングが許される時系列データ」と定義し，楽曲データとハミングデータを時間正規化することで，スケールの合った時系列データに変換し，マッチングには音長情報も利用可能にした．結果的に，高速検索と高い分解能の両方を実現するハミング検索システムを構築することができた．また時間正規化された個々の音符や発声の時間を 8 種類の代表値で整形する，音価レート表現の導入によって，ロバストな音楽データに変換することができた．さらに，ハミング検索のための効果的な特徴量とそれらを効果的に組み合わせて検索に使用する手法を提案した．これらの各手法の有効性は定量的に評価しその効果を確認した．これらの新しい手法は，DTW をハミング検索に適用し，その問題点と利点の分析を通して得られた知見から考案したものである．今後はこの時間正規化手法をベースに，効果的な音高シフティングの手法やハミング検索に特化したマッチング手法などを検討していく予定である．

参 考 文 献

1) 小杉尚子，小島 明，片岡良治，串間和彦：大規模音楽データベースのハミング検索システム，情報処理学会論文誌，Vol.43, No.2, pp.287–298 (2002).
2) 小杉尚子，櫻井保志，山室雅司，串間和彦：SoundCompass: ハミングによる音楽検索システム，情報処理学会論文誌，Vol.45, No.1, pp.333–345 (2004).
3) Goldin, D.Q. and Kanellakis, P.C.: On Similarity Queries for Time-Series Data: Constraint Specification and Implementation, *Proc. 1st International Conference on the Principles and Practice of Constraint Programming*, pp.137–153 (1995).
4) Rabiner, L. and Juang, B.-H.: *FUNDAMENTALS OF SPEECH RECOGNITION*, PTR Prentice-Hall, Inc (1993).
5) Keogh, E.: Exact Indexing of Dynamic Time Warping, *Proc. 28th VLDB Conference* (2002).
6) Ghias, A., Logan, J. and Chamberlin, D.: Query By Humming, *Proc. ACM Multimedia 95*, pp.231–236 (1995).
7) McNab, R.: INTERACTIVE APPLICATIONS OF MUSIC TRANSCRIPTION, Master's thesis, Computer Science at the University of Waikato (1996).
8) Uitdenbogerd, A. and Zobel, J.: Melodic Matching Techniques for Large Music Database, *Proc. ACM Multimedia 99*, pp.57–66 (1999).
9) Jang, J.-S.R. and Lee, H.-R.: Hierarchical Filtering Method for Content-based Music Retrieval via Acoustic Input, *Proc. 9th ACM International Conference on Multimedia*, pp.401–410 (2001).
10) Mazzoni, D. and Dannenberg, R.B.: Melody Matching Directly From Audio, *2nd Annual International Symposium on Music Information Retrieval* (2001).
11) Zhu, Y. and Shasha, D.: Warping Indexes with Envelope Transforms for Query by Humming, *Proc. ACM SIGMOD, International Conference on management of Data*, pp.181–192 (2003).
12) Nishimura, T., Hashiguchi, H., Takita, J., Zhang, J.X., Goto, M. and Oka, R.: Music Signal Spotting retrieval by a Humming Query Using Start Frame Feature Dependent Continuous Dynamic Programming, *2nd International Conference on Music Information Retrieval* (2001).
13) Birmingham, W.P., Dannenberg, R.B., Wakefield, G.H., Bartsch, M., Bykowski, D., Mazzoni, D., Meek, C., Mellody, M. and Rand, W.: MUSART: Music Retrieval Via Aural Queries, *2nd International Conference on Music Information Retrieval* (2001).
14) Faloutsos, C., Ranganathan, M. and Manolopoulos, Y.: Fast Subsequence Matching in Time-Series Database, *Proc.ACM SIGMOD, International Conference on management of Data*, pp.419–429 (1994).
15) Foote, J.: Visualizing Music and Audio using Self-Similarity, *Proc. ACM Multimedia 99*, pp.77–80 (1999).
16) 鈴木直美：[明解] インターネット時代の標準ファイルフォーマット事典，インプレス (1998).
17) 鹿野清宏，中村 哲，伊勢史郎：音声・音情報のディジタル信号処理（ディジタル信号処理シリーズ），昭晃堂 (1997).
18) 真篠 将：音楽通論，全音楽譜出版社 (1956).

(平成 15 年 12 月 20 日受付)
(平成 16 年 2 月 13 日採録)

（担当編集委員 原嶋 秀次）

小杉　尚子

1993 年慶應義塾大学理工学部電気工学科卒業．1995 年同大学院理工学研究科計算機科学専攻修士課程修了．同年日本電信電話（株）入社．以来，リアルタイム OS の研究を経て，現在は音楽検索システムの研究開発に従事．

櫻井　保志（正会員）

1991 年同志社大学工学部電気工学科卒業．同年日本電信電話（株）入社．1996 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．1999 年同大学院博士後期課程修了．工学博士．現在，NTT サイバースペース研究所に所属．索引技術，情報検索に関する研究開発に従事．

森本　正志（正会員）

1986 年京都大学工学部情報工学科卒業．1988 年同大学院工学研究科情報工学専攻修士課程修了．同年日本電信電話（株）入社．1996 年米国スタンフォード大学客員研究員．2002 年京都大学情報学研究科知能情報学専攻博士後期課程在籍．現在，NTT サイバースペース研究所において，画像・楽音・映像等のメディア・ハンドリング技術の研究開発に従事．電子情報通信学会，映像情報メディア学会各会員．