Vol.2014-DCC-8 No.11
Vol.2014-MUS-105 No.11
2014/11/21

大規模楽曲データベースに対する超高速検索に関する検討

喜田 拓也[1,a)]　高田 怜[1,b)]

概要：本稿では，大規模な楽曲データベースに対し，楽曲の一部をクエリとして，そのクエリを含む楽曲を高速に検索する手法について論じる．ここで，データベースとクエリは共に，楽曲の音声音響信号から抽出された何らかの特徴ベクトルの系列として表現されているものとする．提案する手法では，データベースに含まれる大量の高次元特徴ベクトルを，アルファベット分割の考え方を用いて複数本の文字列データとしてとらえ，任意長の部分文字列に対する厳密一致検索が可能な索引構造を用いて保存する．楽曲の検索は，クエリからサンプリングされた多数の小片を厳密一致で検索した後，その結果を統合することで行う．本手法を，音楽指紋と接尾辞配列を用いて具体的なシステムを実現し，およそ十万曲分の人工楽曲データベースに対する評価実験を行った．この実験を通して，大規模楽曲データベースに対する高速かつ精度の良い検索を実現するための課題点について検討を行った．

1. はじめに

近年，個人がコンピュータ上で取り扱うことのできる楽曲数は非常に膨大なものになっている．実際，iTunes [1] や YouTube [17]，ニコニコ動画 [25] などのインターネットサービスからユーザが取得できる楽曲の数は，数百万曲以上にも上る．こうした膨大な楽曲データの検索には，通常，楽曲のアーティスト名や作曲者名などのメタ情報が用いられる．

一方で，楽曲のメタ情報が不明な場合，楽曲の一部の音楽音響信号を手がかりに，該当する楽曲の情報を得たいという場面がある．たとえば，何かの動画の背景音楽として流れている曲名が知りたいといったことや，一般の音楽愛好家による演奏を録音したものから元の曲情報を探すといったことが挙げられる．このような例の場合，クエリとして用いられる信号データは，元の楽曲のごく一部であり，加えて録音状態が悪い，もしくは元の楽曲には存在しない音が含まれていることが想定される．したがって，クエリとなる信号データと近似的に一致する部分を含む楽曲を見つけ出す，類似楽曲検索が必要となる．

本稿では，楽曲の一部分の信号をクエリとし，巨大な楽曲データベースに対して高速な類似楽曲検索を実現する手法について議論する．提案する手法において，楽曲の音楽音響信号データは，何らかの離散的な特徴ベクトルの系列に変換されることを仮定する．近年の音楽検索手法においては，**音楽指紋**（music fingerprinting）[5] への変換がよく用いられる．音楽指紋は，元の信号に含まれる音高情報を二値のベクトルで表現したような離散的な系列データで，元の信号データと比較してデータ量が非常に小さくなるという特長がある．音楽指紋には，対象とする音響信号や用途の違いによりいくつもの変種が存在する．

音楽指紋を用いると，元の類似楽曲検索の問題は，高次元離散値ベクトル系列上での近似照合問題に帰着することになる．高次元ベクトル集合に対する最近傍探索は，多くの研究があり，大別して木構造を用いるものとハッシュを用いるものが挙げられる [12]．いずれにせよ，探索の基本は単一の特徴ベクトル（特徴点）であり，系列全体の照合に単純には適用できない．最近傍探索に基づく楽曲検索 [8,20,35] なども提案されているが，数百から数千曲レベルのデータベースに対する検索に数秒を要する．メタ情報に基づく検索同様に，インタラクティブな検索環境をユーザに提供するためには，万を超える楽曲データベースに対して 1 秒未満で検索できなければならない．そのためには，データベースサイズについて理論的に劣線形時間であるだけでなく，実際上にも高速な検索を可能とするような，データ系列に対する索引データ構造が必要となる．

文字列照合研究の分野においては，全文検索のための索引データ構造が数多く提案されている [7]．特に接尾辞木 [29] や接尾辞配列 [19] などは，任意の長さの部分文字列を検索できる優れた索引データ構造である．また，そうした索引構造を省スペース化した圧縮接尾辞配列 [6] や FM-index [3] などの簡潔索引の研究が近年盛んに行われて

[1] 北海道大学 大学院情報科学研究科
Sapporo, Hokkaido 060–0814, Japan
[a)] kida@ist.hokudai.ac.jp
[b)] takada@ist.hokudai.ac.jp

いる．ただし，索引データ構造による照合は厳密一致を想定していることが多く，近似照合への適用は容易ではない．近似照合へ拡張する研究 [16] もあるが，時間計算量のコストが増大するため，文字集合のサイズやパターン長が大きい場合にはあまり実用的ではない．

Xiao ら [30] は，Locality Sensitive Hashing (LSH) [4] のアイデアを元に，単一のハッシュでより簡潔に高速な照合を行う優れた手法を提案している．彼らの手法では，まず，ある固定長の数フレーム分（文献 [30] の実験では，1 フレーム 1.024 秒，フレームシフト 0.032 秒で 3 フレーム分）に相当する音楽指紋（SSF と呼ばれる）を接尾辞配列[*1]を用いて探索する．その結果得られるマッチ位置において，より長いフレーム分の音楽指紋 (文献中ではサブ指紋ブロックと呼ばれる) を取り出し，クエリとのビット誤り率を計算する．クエリ中の全ての SSF について上記の処理を行い，総合的にビット誤り率の低い順に上位の数曲を出力する．

本稿では，上述のアイデアを踏まえ，より簡潔で高速な近似照合の枠組みを提案する．我々の基本アイデアは，クエリの小片がより数多く一致する楽曲を答えとするものである．ここで，各小片は，文字列の全文検索で用いられる索引データ構造を用いて厳密一致する限り長くとられる．また，クエリから小片をサンプリングして探索を行い，その結果をランキングすることで類似楽曲を求める．同様の考えに基づく先行研究としては，永野ら [24,33] によるものが挙げられるが，系列に対する索引データ構造を用いて，特徴量を点ではなく短い系列の一致に基づいて候補となる楽曲を高速に絞り込んでいる点が異なる．

本稿ではまた，提案手法を，接尾辞配列と文献 [39] の音楽指紋を用いて実装したシステムについて議論する．10 万曲分の擬似データを含む楽曲データベースに対して，原曲とは歌唱者の異なる楽曲検索を本システムを用いて行った結果から，大規模楽曲データベースに対する類似楽曲検索の課題点について検討を行う．ここで述べるシステムとは，著者らが文献 [40] にて既に発表したものであり，本稿で提案する類似楽曲検索の枠組みは，本システムをより一般的な視点からとらえ直したものである．

1.1 関連研究

楽曲の音楽音響信号データを，離散的な特徴ベクトルの系列に変換した上で類似楽曲検索を行う手法には，Xiao ら [30] の手法の他に次のようなものがある．Müller と Kurth ら [15,23] は，和音特徴量であるクロマベクトルを用いた特徴付けを音楽検索に利用する手法を提案している．特に，文献 [15] では，テンポの微少な変動なども考慮したロバスト性を高めた CENS 特徴量（Chroma Energy Normalized Statistics feature）と呼ばれる特徴量を定義し，転置ファイルに基づいた近似照合のシステムを提案している．Barrington ら [2] は，意味的概念モデル（semantic concept model）を用いて楽曲の信号データから概念の集合へと写像し，Bag-of-feature 的な扱いでもって楽曲検索を行う手法を提案している．

またこの他，コンテンツベースでの楽曲検索の研究としては，特定のワードをクエリとした検索 [32] [36] や，ボーカルの声の質の類似さに基づく検索 [34]，ユーザの好みをジャンルで学習した上での検索 [9]，楽曲にタグ付けを行う手法 [10] などがある．

2. 準備

2.1 音楽指紋

音楽指紋（audio fingerprinting）とは，楽曲の音響信号データから得られる二値ベクトルの特徴量であり，楽曲が持つ特徴を数キロビット程度の比較的小さなデータで表現することができる．音楽指紋から元の音響信号へ復元することはできないが，楽曲を特定できる程度には情報が残っている．すなわち，見方を変えると，楽曲の音響信号を極度に非可逆圧縮したものといえる．**音響指紋**（acoustic fingerprinting）と呼ばれていることが多いが，本稿では，「楽曲を検索するために用いる指紋」という意味合いを強調して音楽指紋と呼ぶことにする．音響指紋の歴史的背景と応用技術については，文献 [37] に詳しい．

Haitsma と Kalker ら [8] の手法にしたがって，音楽指紋の生成方法を説明すると次のようになる．まず，音響信号を先頭から順に，ある短い長さの区間で区切りながら周波数解析を行う．この時の区間をフレームと呼ぶことにする．フレーム長よりもフレーム間のずれ幅が小さい場合，フレームどうしは重複部分を持つことになる．各フレームでは，解析された周波数スペクトルを対数スケールで分割し，領域ごとに累計する．このとき，通常，各領域がピアノの鍵盤上の音（十二平均律での音高）に対応するよう分割を行う．フレームの数を $N+1$，区切られる周波数の領域数を $M+1$ とおくと，$M+1$ 次元のベクトルが $N+1$ 個並んだデータが得られる．このデータを行列 E とおく．E の要素 $E(i,j)$ には，i 番目のフレームの，j 番目の高さの音の強さが格納されている．i,j はそれぞれ $[0,N],[0,M]$ の範囲をとる．

次に，E の隣接する成分同士の差分を求め，大きさが (M,N) となるような行列 E' を計算する．具体的には，

$$E'(i,j) = E(i,j) - E(i,j+1) - (E(i-1,j) - E(i-1,j+1)) \quad (1)$$

という計算を行う．この E' の値を用いて，次のような行列 F を生成する．

[*1] 文献 [30] では，「SSF をフレーム単位でソートした配列」と説明している．これは，部分文字列の長さを定数長に制約した接尾辞配列と見ることができる．

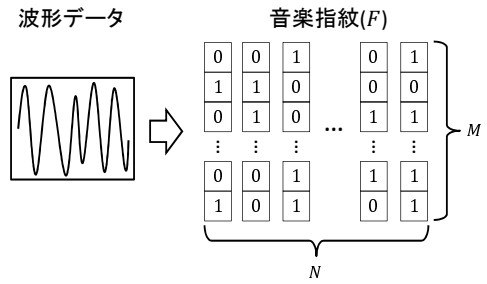

図 1 音楽音響信号から音楽指紋への変換

$$F(i,j) = \begin{cases} 1 & E'(i,j) > 0 \text{ のとき}, \\ 0 & E'(i,j) \leq 0 \text{ のとき}. \end{cases} \quad (2)$$

この行列 F を音楽指紋とよぶ．F は各要素が 0 または 1 であり，二値の行列となっていることがわかる（図 1）．

上記の手順により生成された音楽指紋では，フレームのずれが無ければ，同じ音源からは全く同じものが生成される．逆に言うと，音楽音響信号どうしの差異は，音楽指紋どうしのビット異なり数（ビット誤り率）として現れる．例えば，同一の楽曲でも，演奏や録音の状況が異なれば生成される音楽指紋は幾分異なるものとなる．どの程度異なるものになるかは，音楽指紋を生成する際に選択する周波数帯や分割数，フレーム長，フレームの重複の有無などによって変化する．

2.2 接尾辞配列

文字列（String）とは，アルファベット Σ 上の記号の列である．長さ n の文字列 T を，$T = x_0 x_1 \dots x_{n-1}$ と表す（$x_i \in \Sigma, i = 0, 1, ..., n-1$）．長さが 0 の文字列を特に**空語**という．また，文字列 T の i 番目から j 番目の文字（ただし $0 \leq i \leq j \leq n-1$）を並べてできる文字列を**部分文字列**といい，$T[i,j]$ と表す．つまり，$T = \texttt{abcabb}$ のとき，$T[1,4] = \texttt{bcab}$ である．文字列 T の i 番目の文字から最後までを取り出した部分文字列 S_i は，T の**接尾辞**と呼ばれる．すなわち，$0 \leq i \leq n-1$ に対して，$S_i = T[i, n-1]$ である．接尾辞は，長さ n の文字列に対して n 個存在する[*2]．例えば，先の例の文字列 T の場合，その接尾辞 S_i は次のようになる．

$$S_0 = \texttt{abcabb},$$
$$S_1 = \texttt{bcabb},$$
$$S_2 = \texttt{cabb},$$
$$S_3 = \texttt{abb},$$
$$S_4 = \texttt{bb},$$
$$S_5 = \texttt{b}.$$

[*2] テキストアルゴリズムの分野では，アルゴリズムや理論の記述の簡便さから，空語も接尾辞に含めることが多い．ここでは，理解の容易さを優先して，もとより接尾辞に空語を含めない定義にしている．

また，文字列 T の先頭から i 番目までの部分文字列 $T[0,i]$ は，**接頭辞**（Prefix）と呼ばれる．文字列 T 中の任意の部分文字列は，ある接尾辞 S_i の接頭辞であることに注意する．

接尾辞配列（Suffix Array）とは，1990 年に Manber と Myers によって提案された，文字列を高速に検索するための索引データ構造である [19]．文字列 T に対する接尾辞配列は，T の全ての接尾辞 S_i を辞書順にソートした時，各 S_i の開始位置である i をそのソート順に並べた配列である．例えば，文字列 $T = \texttt{abcabb}$ に対する接尾辞配列 SA_T は，

$$SA_T = [3, 0, 5, 4, 1, 2]$$

となる．接尾辞配列の長さは，元のデータ長と等しいことに注意する．

この接尾辞配列 SA_T を用いると，文字列 T 中に存在する任意の部分文字列を，配列 SA_T 上を二分探索することで高速に検索することができる．なぜならば，等しい部分文字列の T 上での出現位置は，接尾辞配列 SA_T 上で連続して格納されているからである．例えば，先の例の文字列 $T = \texttt{abcabb}$ において，部分文字列 ab の出現位置は $0, 3$ であり，それぞれ $T[SA_T[0]], T[SA_T[1]]$ から始まる位置に出現している．ASCII テキストなど，一文字が 1 Byte で，長さが 4 GByte に収まる文字列の場合，その接尾辞配列は元のテキスト長 n に対して $4n$ Byte となる．全文検索のためには元のテキストも必要なので，全部で $5n$ Byte を必要とする．

3. 検索システムの枠組み

提案する類似楽曲検索手法の基本アイデアは，特徴ベクトル列どうしの類似度を正確に計算することをあきらめ，クエリの多数の小片が完全一致する楽曲を類似楽曲と判断することである（図 2）．この考えは，Ukkonen [28] の近似文字列照合のアイデアに基づいている．提案する手法の特長を概説すると，以下のとおりである．

- 各フレームの特徴ベクトルを分割し，低次元ベクトル系列の集合としてデータベースをとらえる．
- クエリのすべてではなくサンプリングした地点から探索を行う．
- 任意長のフレームを厳密一致探索することで候補位置を絞り込む．
- ビット誤り率を計算することなく，投票によってランキングを行う．

以下では，まず，データベースに対する索引データ構造の構成について述べる．

3.1 索引データ構造

クエリ，およびデータベース中の各楽曲の音響信号データは，フレーム毎にある離散的な M 次元の二値特徴ベクトルに変換されていると仮定する．二値ベクトルでない場

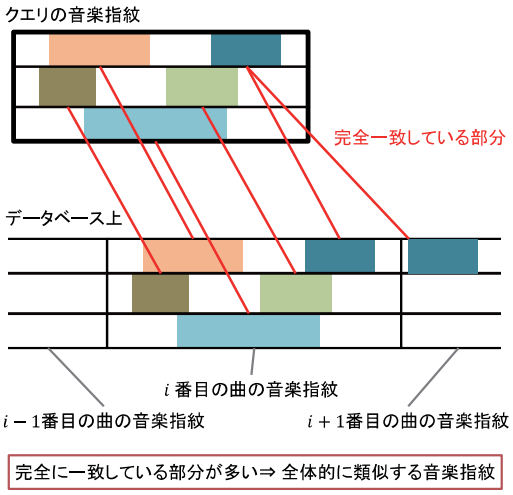

図 2 小片の厳密一致検索による類似楽曲検索のアイデア

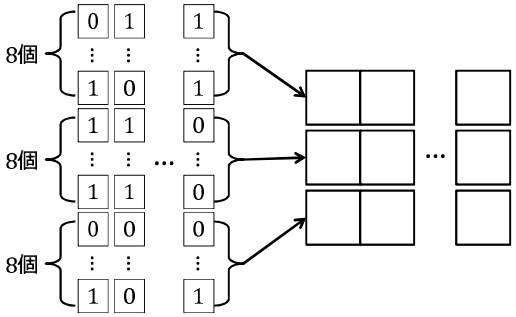

図 3 フレームの文字列への分解の仕方

合は，一個のベクトルが M 次元の二値ベクトルになるよう適当な符号化を行う．議論の簡便のため，以下では M は 8 の倍数であると仮定する．すなわち，一つの楽曲は，フレーム毎に $m = M/8$ バイトの二値ベクトルが並んだ系列データで表されているものとする．

次にこれを，図 3 のように，各フレームを 8 ビット毎（バイト単位）に分割し，横方向に連結する（図 3 の例では，$M = 24$）．分割された各ビット列を 1 バイトでコード化された**文字**とみなすと，一つの楽曲は，m 本の**文字列**が段に積みあがったものと見ることができる．このとき，文字の集合 Σ の大きさは $|\Sigma| = 256$ である[*3]．図 3 の例では，右側の四角形一つが一個の文字に対応している．さらに，図 4 のように，データベース全体で同じ高さの文字列を順に連結することで，m 本の長大な文字列が得られる．

こうして得られた m 本の長い文字列に対して，それぞれ全文検索用の索引構造（接尾辞配列など）を構築する．このような分割を行う利点は二つある．第一に，クエリに多少のノイズが含まれており，その特徴ベクトル系列がデータベース上のものと異なっていたとしても，適切な符号化の下では各段の横方向（文字列の方向）がより長く厳密一

[*3] 後段で使用する索引データ構造が ASCII テキストを仮定している場合は，フレームを 7 ビット毎に分割し，それぞれ 1 ビットを頭に足して 8 ビットで符号化するとよい．その場合は，$|\Sigma| = 128$ となる．

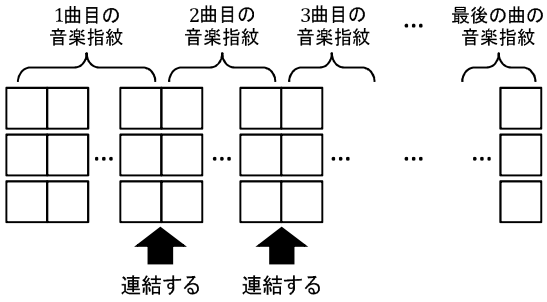

図 4 データベース全体から 3 本の文字列を構成するイメージ

致しやすくなるため，クエリ小片による絞り込みの効果が高まる．第二に，全文検索のための索引構造を実装したライブラリの多くは，バイト単位の文字列もしくは ASCII コード文字列を入力として仮定するため，そうした高速なライブラリの利用が容易になる．

3.2 類似楽曲検索の手順

上述した索引構造を用いて，類似楽曲検索を行う手順は次のとおりである．

(1) まず，クエリの m 本の文字列から，小片の開始位置を一つ選ぶ．ここで，クエリの長さ（フレーム数）を N とし，クエリの i 段目の文字列を $X_i = x_{i,0}x_{i,1}\cdots x_{i,N}$ $(x_{i,j} \in \Sigma)$ とする．以下では，選択された開始位置を i 段目の j 文字目とする．ただし，同じ開始位置を 2 回以上繰り返して用いることがないように注意する．

(2) i 段目の索引構造上で，j 文字目から始まるクエリの小片（部分文字列）$x_{i,j}x_{i,j+1}\cdots$ を検索する．検索する部分文字列は，一致する箇所の数がある閾値 Δ 以下になるまで延長する．例外処理として，あまりに長い小片が多数の箇所と一致する場合，その小片は候補を絞るには不適切として処理を打ち切り，次の小片の選択を行う．

(3) クエリの小片と一致した部分を含む楽曲に，投票を行う．簡単には，一致箇所毎に 1 点を加点する．すなわち，同じ曲に小片との一致箇所が多数含まれていれば，その数だけ加点を行うことになる．

(4) 上記の手順を，ある一定回数（S 回）繰り返す．

(5) 最終的に，最も得票数の多かった上位数曲を，クエリの楽曲と類似した楽曲として出力する．検索精度をより高めたい場合には，上位の曲に対してより厳密な検査を行って候補を篩にかける．

得票数でランキングを取るという特性上，S を小さくすると検索にかかる全体の時間は小さくなるが，候補の絞り込みが不十分になり精度が悪化する．逆に S を大きくすると，ランキング上位に上がる楽曲が安定するようになるが，検索に時間がかかってしまう．また，Δ についても，用いる特徴量やその符号化との兼ね合いから適切に設定する必要がある．

Vol.2014-DCC-8 No.11
Vol.2014-MUS-105 No.11
2014/11/21

4. 音楽指紋と接尾辞配列による実装

4.1 実装方法

ここでは，筆者らの先行研究 [39] で提案した音楽指紋を特徴量とし，接尾辞配列を全文検索用の索引構造とした場合の実装について述べる．文献 [39] で提案した音楽指紋は，Haitsma と Kalker ら [8] が提案した音楽指紋を元にしており，歌唱者が異なるクエリに対する類似楽曲の検索により適したものとなっている．

文献 [39] の音楽指紋の生成方法は 2.1 節で述べたものと同じだが，フレーム長を 0.1 秒とし，各フレームは重複させないものとしている．また，周波数の高さの区切り数 M を $M = 24$（すなわち段数は $m = 3$）とし，使用する周波数帯は 41.2Hz〜164.8Hz とする．つまり，フレーム毎に，楽曲の音楽音響信号から 41.2Hz〜164.8Hz の部分を取り出し，その対数領域を 25 分割する．そして，式 (1) と (2) によって長さ 24 のビット列を生成する．この処理により，例えば長さ 200 秒の楽曲からは，縦 24 ビット，横 2000 フレーム分の音楽指紋が生成される．

接尾辞配列を用いた音楽指紋の検索は, 以下のような手順で行われる．データベースの各段 i ($0 \leq i < 3$) に対応する接尾辞配列を SA_i とする．まず，初めに選択された小片の開始位置を i 段目の j 文字目とする．すなわち，最初は $x_{i,j}$ を接頭辞とする接尾辞を接尾辞配列 SA_i を用いて二分探索する．このとき，該当する接尾辞は複数あることが考えられるので，その数 R_0 を求める．接尾辞配列の生成方法から，ここで探索される接尾辞は SA_i 上で連続していることに注意する．次に R_0 と閾値 Δ を比較し，$R_0 < \Delta$ ならば，その接尾辞の開始位置を含む楽曲に加点する．そうでなければ，続いて $x_{i,j+1}$ を取り出し，$X_i[j, j+1] = x_{i,j}x_{i,j+1}$ を SA_i 上で探索する．このとき，$X_i[j, j+1]$ を接頭辞とする接尾辞の数 R_1 は $R_1 \leq R_0$ となっている．このように，徐々に探索する接頭辞 $x_{i,j}x_{i,j+1}\cdots x_{i,j+\ell}$ ($\ell = 0, 1, 2\cdots$) を延長しながら上記の探索を $R_\ell < \Delta$ となるまで繰り返す．この探索の様子を図 5 に示す．

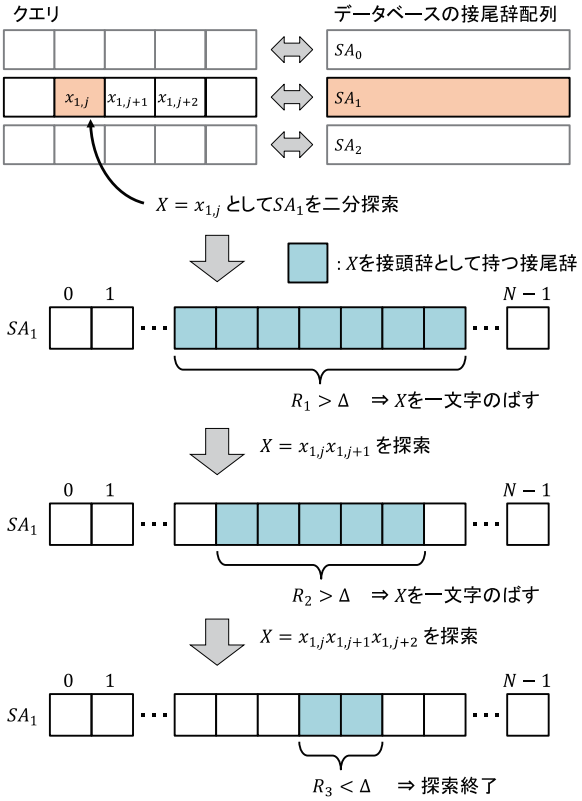

図 5 接尾辞配列を用いてクエリの小片を探索する様子

4.2 評価実験

巨大なデータベースに対して，上述した方法で実装したシステムのパフォーマンスを評価するための実験を行った．クエリとして，楽曲の音響信号全体を用いる場合と，楽曲の一部分（サビとなる部分）のみを用いた場合とで，類似楽曲検索の精度と検索速度の変化を調べた．また，楽曲データベースとしては，実際の楽曲 7849 曲を用いたもの（実験１）と，それに人工的に生成した音楽指紋を 10 万曲を追加したもの（実験２）を用いた．

4.2.1 実験環境

実験１で用いたデータベースは，一般的な J-POP や VOCALOID による楽曲（VOCALOID 曲）などの実際の楽曲 7849 曲から構成されている．これらのデータの収集は，主にニコニコ動画や YouTube，CD 音源などを用いて行った．クエリとしては，J-POP 曲から 78 曲，VOCALOID 曲から 100 曲，オフボーカル曲（伴奏のみの曲）から 22 曲の計 200 曲を用意した．クエリの J-POP 曲と VOCALOID 曲はニコニコ動画から，オフボーカルの楽曲は CD から取得した．データベース側の楽曲の長さは平均で 3 分 49 秒，クエリのデータは楽曲全体のもので平均 4 分 19 秒，サビ部分を抽出したもので平均 27 秒であった．実験２で用いたデータベースは，上記のデータベースに，乱数によって生成した 10 万曲分の疑似的な音楽指紋を追加したものである．元のデータベースの 1 曲あたりの平均的な長さが 3 分 49 秒であったため，疑似的に生成した音楽指紋は 1 曲あたりのフレーム数を 2300 とした．

クエリのうち，オフボーカル曲については，いわゆるカラオケバージョン（Instrumental バージョン）をデータベース中の楽曲と同一の CD から入手したものであるため，クエリとして用いるデータの中では最も条件の良いものである．一方で，J-POP 曲，VOCALOID 曲に対するクエリは，同一の楽曲がデータベース中に存在するが，原曲とは歌唱者が異なるものを選択した．ただし，どちらの場合においても，原曲からキーを変えずに歌唱されているものを選んだ．特に J-POP 曲については，著作権の問題から，音質が悪いものやバンドのアレンジが異なっているなど，原曲とはかなり録音状態が異なるものになっている．したがっ

表 1 実際の楽曲 7849 曲に対する検索の正答率と一曲あたりの検索時間.

	全体	サビのみ
オフボーカル	100.0%	100.0%
VOCALOID 曲	95.3%	86.0%
J-POP 曲	66.0%	57.7%
合計	84.4%	76.5%
一曲あたりの検索時間	0.018 秒	0.006 秒

表 2 人工データを加えた 107849 曲に対する検索の正答率と一曲あたりの検索時間.

	全体	サビのみ
オフボーカル	100.0%	100.0%
VOCALOID 曲	94.5%	88.0%
J-POP 曲	66.3%	56.4%
合計	84.1%	77.0%
一曲あたりの検索時間	0.035 秒	0.010 秒

て，オフボーカル曲のクエリは，同一の音楽指紋を持つ楽曲の検索に相当し，他方，J-POP 曲と VOCALOID 曲のクエリは，歌唱の録音状況や歌唱者が異なる類似楽曲の検索に相当する．

音楽指紋の生成の際, 音響データの解析には Muller らが公開している chroma toolbox [22] を用いた．また，接尾辞配列の生成には Yuta Mori が公開しているプログラム [21] を用いた．実験に使用した計算機は，Intel Core i5 (2.5GHz)，メモリサイズ 8GB，OS は Ubuntu12.04 (64bit 版) である．

4.2.2 実験 1

実験では，得票数の多い順に上位 10 曲を出力するものとし，その中にクエリと同一の楽曲が含まれているならば検索成功と判断する．また，検索時間は，クエリの音楽指紋を与えてからランキングの答えが出力されるまでとする．ここで，小片を選択する回数 S は 2000，探索を中断するための閾値 Δ は 15 とした．これらのパラメータは，予備実験によって検索時間と精度のバランスを考慮して決定した．楽曲全体およびサビのみのクエリ双方で，各クエリセットについて 20 回の検索を行い，その正答率と一曲あたりの検索時間の平均を求めた．表 1 にその結果を示す．

まず，検索精度について考察する．オフボーカル曲は，クエリの音源が CD から得られているため，クエリが楽曲全体，サビのみのどちらでも検索精度は非常に良い結果となった．VOCALOID 曲と J-POP 曲については，サビのみとした時の正答率が，楽曲全体で検索した時と比べて下がる結果となった．これは，ボーカルの違いが音楽指紋の違いに影響し，それが検索精度に影響を与えるという予想を裏付けているといえる．また，特に J-POP 曲での正答率は，全体，サビのみのどちらにおいても低い結果になった．これは，クエリとして用意した音源の録音状況が悪いことが大きく影響していると考えられる．同様に VOCALOID 曲は，J-POP 曲ほどではないが，データベース中の原曲とは録音状況が異なっているため，正答率低下にある程度の影響がみられた．以上のように，検索精度は，クエリの音源データの質そのものが結果に大きく影響しており，検索手法の性能というよりは音楽指紋の設計の仕方により大きく依存していると考えられる．

次に，検索時間について考察する．7849 曲のデータベースに対し，楽曲全体での検索にかかる時間は 0.02 秒未満となった．また，サビのみでは 0.01 秒未満である．サビのみでの検索が全体での検索よりも時間が小さい理由は，サビのみのクエリは全体の文字数がサンプル数 $S = 2000$ よりも少ないものが多く，開始位置を重複して選択することを排除しているために小片の検索が 2000 回未満で打ち切られるからである．

参考までに，Xiao ら [30] の実験では，Intel Core i7 (1.73GHz)，メモリサイズ 4GB の計算機が用いられ，楽曲の検索に要した時間は 8740 曲のデータベースに対し 1 曲あたりおよそ 0.4〜0.6 秒程度とのことであった．同じ計算機，実験データを用いた比較が行えなかったのは，著者らから提供を受けた検索システムが，クエリの信号解析から検索までを一度に全て行う仕様となっており，検索部分のみの時間計測を行えなかったためである．

4.2.3 実験 2

実験 1 と同様に，得票数の多い順に上位 10 曲を出力するものとし，その中にクエリと同一の楽曲が含まれているならば検索成功と判断する．各クエリセットについて 20 回の検索を行い，その正答率と計算時間の平均を求めた．表 2 にその結果を示す．

楽曲全体での検索で 1 曲あたり 0.035 秒，サビのみでの検索で 1 曲あたり 0.01 秒という結果となった．7849 曲のデータベースに対しての検索時間は 0.018 秒であり，およそ 14 倍の大きさのデータベースに対して検索時間は 2 倍程度で抑えられるという結果となった．これは，接尾辞配列上を二分探索するために，検索がデータベースサイズの対数に比例した時間で抑えられていることが理由として挙げられる．また，精度についても，7849 曲のデータベースでの検索と大きな差はないという結果となった．データベースを大きくした上で精度が上昇したものがあったのは，ランキングの阻害となる偽陽性の照合結果がならされ，正解データが目立ちやすくなったためだと考えられる．

5. 今後の課題と展望

本稿では，楽曲の高速な類似検索を実現する新しい枠組みを提案した．また，その枠組みに基づいて音楽指紋と接尾辞配列による楽曲検索システムの実装を行い，10 万曲分を超える擬似的な楽曲データベースに対する性能実験を行った．その結果，クエリとなる音楽指紋が与えられてから該当する楽曲を検索するまでにかかる時間は，現在普及するスペックの PC 上で 0.04 秒未満であった．その際，

ノイズの無いクエリに対しては 10 割，歌唱者が人である VOCALOID 曲をクエリとしたときの原曲検索は約 9 割の正答率であった．よって，今回実装したシステムは，10 万曲程度の楽曲数の検索に関して一定の成果を収めたと言える．

以下に，現状の課題点と展望について述べる．

まず，今回実装したシステムの主たる欠点の一つは，その使用メモリ量である．実験で使用した疑似的な音楽指紋 10 万曲分のデータ量は，2300 フレーム×3 Byte×100,000 曲 ≃ 658 MByte となる．必要とする接尾辞配列はその 4 倍となることから，索引データ構造だけでも総計で約 3.2 GByte を必要とする．オペレーティングシステムや他のプロセスが使用するメモリ量を考慮すると，搭載メモリ量が 8 GB のマシンでは 10 万曲程度が限界である．

もう一つの欠点は，接尾辞配列が，新しい楽曲のデータベースへの追加に対して柔軟性に乏しいことである．なぜならば，新しいデータがデータベースに追加された場合，その接尾辞配列の構築にはデータベース全体に係る整列処理が伴うからである．したがって，新しく生み出される楽曲を逐次にシステムへと追加するには，運用上での工夫が必要となる．

このような課題点に対する，解消方法の一つは，索引データ構築や検索処理を負荷分散する分散型システムを構築することであろう．もう一つ解消方法としては，接尾辞配列に代わる高度な索引データ構造を用いることである．接尾辞配列よりもよりコンパクトで，なおかつ逐次のデータ追加に対して柔軟に更新できる索引データ構造が望ましい．期待できる関連研究として，FM-index を動的に構築する手法 [18, 26] や，複数の圧縮接尾辞配列を動的に統合する手法 [27] がある．こうした高度な索引データ構造を用いたシステムを構築し，その性能実験を行うことが至急の課題である．

次に検索精度に関して考察する．今回実装したシステムでは，クエリが短かったり，ノイズが多く含まれていたりすると十全な結果が得られない．また，そもそもの演奏形態が異なるようなクエリに対しても，類似楽曲の検索は困難である．その主たる原因は，使用している特徴量にある．本楽曲検索の枠組みを，例えば，路上で録音した信号や鼻歌をクエリとする検索に適用するためには，よりロバストな特徴量を選択しなければならない．また，上に述べた検索データ構造のサイズの問題にも関連するが，よりコンパクトに楽曲を特徴付けられる特徴量が望ましい．

したがって，音楽指紋をよりロバストなものへと直接的に改善する手法 [38] の他に，メロディ推定 [31] や MIDI への写像 [11] を利用した，（ノイズを含む）音楽音響信号からの離散的特徴量への変換に関する研究との組み合わせも，非常に有望である．また，近年，Khemiri ら [13, 14] は，対象とするデータベースから音要素とも呼べるものを隠れマルコフモデル（HMM）で学習して，その学習した HMM を用いて楽曲を離散的なデータ系列に変換し検索に用いる手法を提案している．このような，機械学習によるデータ駆動型の音楽指紋も，検索精度の向上に有効であると考えられる．

本稿の実験では，文献 [39] の音楽指紋のみを用いたが，提案する枠組みにおいては複数の異なる特徴量を組み合わせた検索システムの構築が容易である．クエリが極端に短い場合には，数種類の特徴量を組み合わせる必要があるだろう．今後の課題の一つに，膨大な音響信号データベースに対して，ジングル[*4]や効果音といった短い音響信号をクエリとした高速な検索が挙げられる．

謝辞

本研究を進めるにあたって，文献 [30] のシステムを快くご提供くださいました，徳島大学の北 研二先生に深く感謝いたします．

参考文献

[1] Apple.com: iTunes Store. http://www.apple.com/jp/itunes/.
[2] Barrington, L., Chan, A., Turnbull, D. and Lanckriet, G.: Audio information retrieval using semantic similarity, *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2007*, pp. II–725–II–728 (2007).
[3] Ferragina, P. and Manzini, G.: Indexing Compressed Text, *Journal of ACM*, Vol. 52, No. 4, pp. 552–581 (2005).
[4] Gionis, A., Indyk, P. and Motwani, R.: Similarity Search in High Dimensions via Hashing, *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999*, pp. 518–529 (1999).
[5] Gold, B., Morgan, N. and Ellis, D.: *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, Wiley-Interscience (2011).
[6] Grossi, R. and Vitter, J. S.: Compressed Suffix Arrays and Suffix Trees with Applications to Text Indexing and String Matching, *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, ACM, pp. 397–406 (2000).
[7] Gusfield, D.: *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press (1997).
[8] Haitsma, J. and Kalker, T.: A Highly Robust Audio Fingerprinting System, *Proceedings of the 3rd International Society on Music Information Retrieval, ISMIR 2002*, pp. 107–115 (2002).
[9] Hoashi, K., Matsumoto, K. and Inoue, N.: Personalization of User Profiles for Content-based Music Retrieval Based on Relevance Feedback, *Proceedings of the Eleventh ACM International Conference on Multimedia*, pp. 110–119 (2003).
[10] Hoffman, M. D., Blei, D. M. and Cook, P. R.: Easy as CBA: A Simple Probabilistic Model for Tagging Music, *Proceedings of the 10th International Society on Mu-*

[*4] ラジオやテレビにおいて，番組の節目に挿入される短い音楽．

sic Information Retrieval, ISMIR 2009, pp. 369–374 (2009).

[11] Hu, N., Dannenberg, R. B. and Tzanetakis, G.: Polyphonic Audio Matching and Alignment for Music Retrieval, *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2003*, pp. 185–188 (2003).

[12] Iwamura, M., Sato, T. and Kise, K.: What is the Most EfficientWay to Select Nearest Neighbor Candidates for Fast Approximate Nearest Neighbor Search?, *IEEE International Conference on Computer Vision (ICCV)*, pp. 3535–3542 (2013).

[13] Khemiri, H., Petrovska-Delacrétaz, D. and Chollet, G.: Speaker diarization using data-driven audio sequencing, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7736–7740 (2013).

[14] Khemiri, H., Petrovska-Delacrétaz, D. and Chollet, G.: ALISP-Based Data Compression for Generic Audio Indexing, *Data Compression Conference (DCC), 2014*, pp. 273–282 (2014).

[15] Kurth, F. and Müller, M.: Efficient Index-Based Audio Matching, *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 16, No. 2, pp. 382–395 (2008).

[16] leung Chan, H., wah Lam, T., kin Sung, W., lung Tam, S. and seong Wong, S.: Compressed indexes for approximate string matching, *Proceedings of the European Symposium on Algorithms*, pp. 208–219 (2006).

[17] LLC, Y.: YouTube. http://www.youtube.com/.

[18] Mäkinen, V. and Navarro, G.: Dynamic Entropy-compressed Sequences and Full-text Indexes, *ACM Trans. Algorithms*, Vol. 4, No. 3, pp. 32:1–32:38 (2008).

[19] Manber, U. and Myers, G.: Suffix Arrays: A New Method for On-line String Searches, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 319–327 (1990).

[20] Miller, M. L., Rodriguez, M. A. and Cox, I. J.: Audio fingerprinting: nearest neighbor search in high dimensional binary spaces, *IEEE Workshop on Multimedia Signal Processing*, pp. 182–185 (2002).

[21] Mori, Y.: White page. http://homepage3.nifty.com/wpage/.

[22] Müller, M. and Ewert, S.: Chroma Toolbox: Matlab Implementations for Extracting Variants of Chroma-Based Audio Features, *Proceedings of the 12th International Society on Music Information Retrieval, ISMIR 2011*, pp. 215–220 (2011).

[23] Müller, M., Kurth, F. and Clausen, M.: Audio Matching via Choroma-based Statistical Features, *Proceedings of the 6th International Society on Music Information Retrieval, ISMIR 2006*, pp. 288–295 (2005).

[24] Nagano, H., Kashino, K. and Murase, H.: Fast music retrieval using polyphonic binary feature vectors, *Proceedings of 2002 IEEE International Conference on Multimedia and Expo, ICME 2002* (2002).

[25] niwango: ニコニコ動画. http://www.niconico.jp/.

[26] Salson, M., Lecroq, T., Lëonard, M. and Mouchard, L.: Dynamic extended suffix arrays, *Journal of Discrete Algorithms*, Vol. 8, No. 2, pp. 241 – 257 (2010). Selected papers from the 3rd Algorithms and Complexity in Durham Workshop (ACiD) 2007.

[27] Sirén, J.: Compressed Suffix Arrays for Massive Data, *Proceedings of the 16th International Symposium on String Processing and Information Retrieval*, SPIRE '09, Berlin, Heidelberg, Springer-Verlag, pp. 63–74 (2009).

[28] Ukkonen, E.: Approximate string-matching with q-grams and maximal matches, *Theoretical Computer Science*, Vol. 92, No. 1, pp. 191 – 211 (1992).

[29] Weiner, P.: Linear Pattern Matching Algorithms, *The 14th Annual Symposium on Switching and Automata Theory (Swat 1973)*, SWAT '73, IEEE Computer Society, pp. 1–11 (1973).

[30] Xiao, Q., Suzuki, M. and Kita, K.: Fast Hamming Space Search for Audio Fingerprinting Systems, *Proceedings of the 12th International Society on Music Information Retrieval, ISMIR 2011*, pp. 133–138 (2011).

[31] 角尾衣未留，井上　晃，西口正之：鼻歌検索システムのための楽曲からのボーカルメロディ推定，情報処理学会研究報告．音楽情報科学研究会報告，Vol. 2013, No. 18, pp. 1–6 (2013).

[32] 辻　康博，星　守，大森　匡：曲の局所パターン特徴量を用いた類似曲検索・感性語による検索，電子情報通信学会技術研究報告. SP, 音声， Vol. 96, No. 565, pp. 17–24 (1997).

[33] 永野秀尚，柏野邦夫，村瀬　洋：多数の小領域スペクトログラムの探索に基づく背景音楽の高速探索法 (音楽情報処理)，電子情報通信学会論文誌. D-II, 情報・システム, II-パターン処理， Vol. 87, No. 5, pp. 1179–1188 (2004).

[34] 藤原弘将，後藤真孝：VocalFinder: 声質の類似度に基づく楽曲検索システム，情報処理学会研究報告．音楽情報科学研究会報告，Vol. 2007, No. 81, pp. 27–32 (2007).

[35] 柏野邦夫，スミスガビン，村瀬　洋：ヒストグラム特徴を用いた音響信号の高速探索法: 時系列アクティブ探索法，電子情報通信学会論文誌. D-II, 情報・システム, II-パターン処理， Vol. 82, No. 9, pp. 1365–1373 (1999).

[36] 福村純也，川越恭二：特徴空間とメロディ空間を用いた楽曲の類似検索方法，情報処理学会研究報告. データベース・システム研究会報告， Vol. 2003, No. 5, pp. 17–24 (2003).

[37] 柏野邦夫：音響指紋技術とその応用，日本音響学会誌，Vol. 66, No. 2, pp. 71–76 (2010).

[38] 黒住隆行，木村昭悟，永野秀尚，柏野邦夫：幾何変換パラメータを特定する縮退生成探索法 (テーマセッション (4), パターン認識・メディア理解のための学習理論とその応用)，電子情報通信学会技術研究報告. PRMU, パターン認識・メディア理解， Vol. 106, No. 429, pp. 1–6 (2006).

[39] 高田　怜，喜田拓也：歌唱者の異なる同一楽曲の検索に適した音楽指紋，情報処理学会研究報告．音楽情報科学研究会報告，Vol. 2013, No. 7, pp. 1–6 (2013).

[40] 高田　怜，喜田拓也：大規模音楽音響信号データベースに対する超高速部分一致近似検索手法，日本データベース学会和文論文誌，Vol. 13-J, No. 1 (2014). (to appear).