Vol.2013-MUS-100 No.7
2013/8/31

歌唱者の異なる同一楽曲の検索に適した音楽指紋

高田 怜[1,a)]　喜田 拓也[1,b)]

概要：
楽曲信号の検索には，音楽指紋が用いられることが多い．音楽指紋とは，楽曲の特徴をとらえつつ，楽曲信号を非常に小さいビット列へと変換したものである．したがって，精度良く高速に楽曲検索を行うためには，適切な音楽指紋の設計が重要となる．本稿では，異なる歌唱者によって歌われた同じ伴奏の楽曲信号からデータベース中の原曲を検索する問題について議論する．この問題のためにいくつか異なる音楽指紋を比較し，それらの検索精度への影響について調査を行う．

1. はじめに

今日，多数のデジタル音楽リソースがネットワークを介して利用できるようになった．たとえば，iTunes はおよそ 2600 万もの楽曲を販売しており[*1]，ユーザはいつでもそれらの楽曲をダウンロード購入することができる．また，Spotify[*2]のようなオン・デマンドのストリーム配信サービスも現れており，そうしたサービスにおいても数千万曲が提供されている．

一方，ユーザ生成コンテンツ (user-generated content; UGC) の成長ぶりも目覚ましい．音楽好きなアマチュアらの中には，プロの楽曲を購入するだけでなく，自らの手で楽曲を制作して YouTube[*3]やニコニコ動画[*4]等のウェブサイトで発表する者も多い．高機能で廉価な音楽生成ソフトウェアの普及はこうした状況を後押ししている．特に，YAMAHA が開発した VOCALOID[*5]とそれを基にした各種音源パッケージは多くのアマチュア作曲家の創作意欲を刺激し続けており，今後も多数の UGC 楽曲は増え続ける状況にある．

ボーカル部分を VOCALOID で合成した UGC 楽曲は，多くの場合，ボーカルを省いた楽曲データが作曲者によって公開されている．またそのデータを基に，別のアマチュアが歌い手となってボーカル部分が補われ，ミキシングされた楽曲データが 2 次創作物として公開されている．著作権に関して 2 次創作の自由度の高い UGC 楽曲では，いわゆるカラオケに自分の歌声を録音した楽曲データ（**カラオケ曲**）が多数存在している．こうしたカラオケ曲は，通常，原曲の情報と共に公開される．しかし，原曲に関するメタデータが付随していないカラオケ曲に対しては，原曲の情報を知るために信号データによる類似楽曲検索を行う必要がある．

本稿では，異なる歌い手によるカラオケ曲の信号をクエリとして，楽曲データベースからその原曲を検索する**原曲検索問題**について議論する．特に，高速な検索を実現する鍵となる**音楽指紋** (audio fingerprinting) に注目し，音楽指紋の構成の違いが原曲検索に与える影響について考察する．音楽指紋とは，楽曲データを比較的小さなサイズのビット列に変換したものである．見方を変えると，検索に必要な程度の情報を残しつつ，楽曲データに大幅な非可逆圧縮を施したものと考えることもできる．音楽指紋は，楽曲の微小な区間（フレーム）毎にビット化された特徴量が時系列に並んだものになっている．したがって，音楽指紋を用いた類似楽曲検索は，ビットベクトル列の最近傍探索問題に帰着される．

高次ベクトルの最近傍探索手法の一つに，局所性鋭敏型ハッシュ（Locality Sensitive Hashing; LSH）がある [1], [2], [3]．Xiao ら [4] は, その LSH をヒントに, Haitsma と Kalker ら [5] が提案した音楽指紋を用いて高速でノイズに強い検索手法を提案した．彼らの楽曲検索システムでは，任意の長さの楽曲信号をクエリとして用い，クエリに近い部分をもつ楽曲を高速に検索することができる．

Xiao らのシステム [4] はカラオケ曲の原曲検索に対して適用可能であり，クエリとして十分長い信号データが与えられる場合には非常に良好な結果が得られる．しかしなが

[1] 北海道大学 大学院情報科学研究科
Sapporo, Hokkaido 060–0814, Japan
[a)] takada@ist.hokudai.ac.jp
[b)] kida@ist.hokudai.ac.jp
[*1] 2013 年 5 月現在．http://www.apple.com/jp/itunes/より．
[*2] http://www.spotify.com/int/
[*3] http://www.youtube.com/
[*4] http://www.nicovideo.jp/
[*5] http://www.vocaloid.com/en/

ら，我々の予備実験では，原曲が VOCALOID の楽曲で，かつクエリがサビ部分の数秒間しか与えられない場合に著しい性能低下がみられた．これは，ボーカルの異なりが音楽指紋に大きく影響しているためであると考えられる．今回，フレームと周波数帯域の取り方を変えた 4 種類の音楽指紋について，原曲検索の正答率の変化を調査した．その結果，原曲が VOCALOID 楽曲の場合には，Haitsma と Kalker ら [5] の音楽指紋で選択されている 300Hz から 2000Hz の帯域よりも，より下方の 39Hz から 247Hz の帯域を選択した音楽指紋のほうが優れていることが分かった．

2. 音楽指紋

本節では，音楽指紋の生成方法と，その検索方法についての概要を述べる．

本稿で述べる音楽指紋は，楽曲の音響信号データから作られるビット列のことで，その楽曲の特徴を数キロビット程度の比較的小さいデータ量で表現したものである．音楽指紋から元の音響信号に復元することはできないが，楽曲の同一性を軽量に判定するために用いられる．音楽指紋を用いると，元の楽曲信号をそのままデータベースにした場合に比べて格段にデータ量が小さくなる．

2.1 音楽指紋の生成方法

本節では，Haitsma と Kalker ら [5] が提案した音楽指紋をもとに，本稿で扱う音楽指紋の生成方法の概要を述べる．

まず，楽曲の先頭から順にある長さ L の区間を区切り出しながら，各区間の周波数解析を行う．このとき，切り出す区間のずらし幅を L より小さくとると，区間同士は重複することになる．先頭から i 番目の区間を，i 番目の**フレーム**と呼ぶことにする．

取り出した各フレームについて，指定した範囲の周波数帯を M 個の重複しない領域に分割し，各領域の強度を測る．この分割は対数目盛りで行われる．ここで，i 番目のフレームの下から j 番目の周波数帯域の強度を $E(i,j)$ と書くことにする．また，$E(i) = \langle E(i,1), E(i,2), \cdots, E(i,M)\rangle^T$ とする．楽曲の全体にわたって上記の手順を行うと，M 次元のベクトル $E(i)$ が並んだデータが得られる．

いま，楽曲全体から取り出されるフレームの総数を N とすると，i, j はそれぞれ，$[1,N], [1,M]$ の範囲の整数を取る．次に，$E(i)$ の隣接する成分同士の差分を求め，$M-1$ 次元のベクトル $E'(i)$ を求める．すなわち，$E'(i) = \langle E(i,1) - E(i,2), E(i,2) - E(i,3), \cdots, E(i,M-1) - E(i,M)\rangle$ とする．さらに，$E'(i)$ から隣接するフレーム同士の差分を求め，$ED(i)$ とする．すなわち，$ED(i) = E'(i) - E'(i-1)$ である．ここで，便宜上，$E'(0)$ は零ベクトルとする．最終的に，$ED(i)$ の各成分の正負で二値化したベクトル $F(i)$ の並びを楽曲の音楽指紋とする．すなわち，

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

である．ただし，ここで $F(i,j), ED(i,j)$ は，それぞれ $F(i), ED(i)$ の j 番目（$1 \leq j \leq M-1$）の成分である．

各ベクトル $F(i)$ は，**サブ指紋**と呼ばれる．上述した構成方法から明らかなように，サブ指紋の一つ一つは $M-1$ ビットのベクトルであり，単一のサブ指紋は楽曲信号の同定を行えるだけの十分な情報を持っていない．言い換えると，単一のサブ指紋だけで検索しても，ほとんど検索を絞り込めない．そこで，検索する際は連続する複数のサブ指紋をまとめて取り扱うことが多い．

Haitsma と Kalker ら [5] が提案した音楽指紋では，300Hz から 2000Hz の帯域からサブ指紋を抽出し，フレームのずらし幅をフレーム長の $\frac{31}{32}$ とすると述べられている．Xiao らのシステムで用いられているものは，[5] と同じ帯域で，フレームの長さを 1.024 秒とし，フレームのずらし幅を 32 ミリ秒としている．

歌唱者が異なる原曲を検索するという目的からは，なるべく歌声部分の影響を抑えつつも，楽曲を十分に判別できる帯域からサブ指紋を抽出するほうが望ましい．理由は後述するが，我々はそのために 39Hz から 247Hz というかなり低い帯域から音楽指紋を生成する方法を比較対象として調査を行った．

また，フレームの重複の度合によって，データベースの総量にかなりの差がでる．当然，フレームをなるべく重複させないほうがよりデータの総量は少なくなる．逆に，重複を少なくとると，元の楽曲の情報がより多く捨てられてしまうという弊害がある．

2.2 音楽指紋の検索方法

前述したように，サブ指紋単体では十分な情報量を持っていないので，連続する m 個（$m > 1$）のサブ指紋を並べた短いサブ指紋の系列を考える．[4] では，この短いサブ指紋の系列を**サブ指紋系列** (Sequence of sub-fingerprint: SSF) と呼んでいる．すなわち，楽曲データベース中の全曲から得られた音楽指紋を $FP = (FP_1, FP_2, \cdots, FP_n)$ とすると，この楽曲データベースは $n-m+1$ 個の SSF

$$\begin{aligned} SSF_1 &= (FP_1, FP_2, \cdots, FP_m) \\ SSF_2 &= (FP_2, FP_3, \cdots, FP_{m+1}) \\ &\vdots \\ SSF_{n-m+1} &= (FP_{n-m+1}, FP_{n-m+2}, \cdots, FP_n) \end{aligned}$$

が並んでいると考えることができる．音楽指紋の検索は，この SSF を要素の単位として行う．本稿では，[4] にならって，$m = 3$ で考える．すなわち，以降，各 SSF は 3 つの連続したサブ指紋からなるものとする．

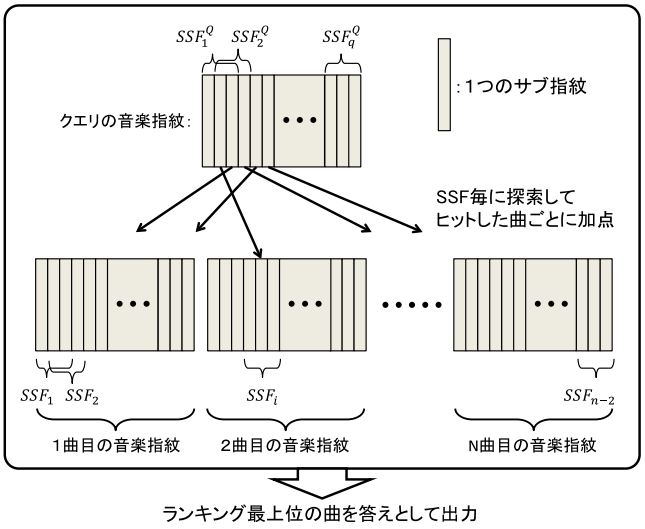

図 1 SSF による類似楽曲の検索方式.
Fig. 1 Search Method by SSF.

類似楽曲の検索の方式は様々だが，本稿では次のようなランキング方式で行う．まず，クエリの音楽指紋の各 SSF に対し，データベース中の SSF を探索して最も類似する SSF を見つける．このとき，二つの $SSF^A = (FP_1^A, FP_2^A, FP_3^A)$ と $SSF^B = (FP_1^B, FP_2^B, FP_3^B)$ の類似度は，ハミング距離

$$D_H(SSF^A, SSF^B) = \sum_{k=1}^{3} W_H(FP_k^A \oplus FP_k^B)$$

を用いる．ここで，$\oplus$ は排他的論理和 (XOR) の論理演算子，$W_H(x)$ はビットベクトル x のハミング重みである．次に，見つかった SSF を含むデータベース楽曲に点数を加点する．このとき，該当する楽曲が複数あった場合には，すべての楽曲に加点する．ただし，同一の楽曲内でハミング距離 D_H が最小となる SSF が複数見つかった場合は，その曲への加点は一度のみとする．この手順をクエリの全 SSF について行い，最終的に最も点数の高かった楽曲をクエリと同一の楽曲として出力する．

図 1 は，本検索方式を図解したものである．ちなみに，Xiao らのシステムでは，SSF の探索に接尾辞配列を用いた二分探索を行っている[*6]．

3. 実験

3.1 実験データ

データベースとして，総数 7849 曲の楽曲データを用いた．この中には，一般的な J-POP 曲の他に，ニコニコ動画や YouTube などで公開されている VOCALOID オリジナル楽曲（以下，VOCALOID 曲）などが含まれる．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（伴奏のみ）の楽曲 22 曲の計 200 曲を用意した．これらクエリ楽曲は，元となった原曲がデータベース中に含まれるが歌唱者が異なっているものを選択した．また，オフボーカルのクエリは，歌唱付きの原曲がデータベース中に含まれているものを選んだ．

データの収集は，主にニコニコ動画や CD 音源などを用いて行った．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．この時，J-POP 曲と VOCALOID 曲は，原曲からキーとアレンジを変えずに歌唱されているものを選んだ．このうち J-POP 曲については，著作権の問題から，原曲とはかなり録音状態が異なるものになっている．一方，オフボーカルの楽曲は，データベースにある楽曲と同一の 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良い条件のものである．

さらに，クエリとして用意した楽曲は，楽曲全体のものとサビ部分を抽出したものの 2 種類を用意した．サビ部分を用意した理由としては，イントロや間奏，アウトロなど，ボーカルのない部分が検索の助けとなってしまうことを考慮した．このクエリデータでは，クエリ全体にボーカルのあるデータとなる必要があるが，楽曲によってはサビ部分の認識が難しい物もあったため，その場合にはボーカルのある部分を一定量抽出した．また，オフボーカルの楽曲のサビ部分とは，もともとの楽曲でサビである部分とした．これらクエリの抽出は全て人力で行った．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 秒程度であった．

音楽指紋を生成する際，周波数帯域の解析には Muller らが公開している chroma toolbox[6] を用いた．

3.2 実験方法

本実験において比較した音楽指紋は，次の 4 つである．

- 帯域は 300Hz から 2000Hz，フレームを重複してサブ指紋を取り出す（Haitsma と Kalker ら [5] が提案した音楽指紋）．
- 帯域は 300Hz から 2000Hz，フレームは重複させずにサブ指紋を取り出す．
- 帯域は 29Hz から 247Hz，フレームを重複してサブ指紋を取り出す．
- 帯域は 29Hz から 247Hz，フレームは重複させずにサブ指紋を取り出す．

便宜上，音域を 300Hz から 2000Hz とした音楽指紋を，[5] の著者の頭文字をとって **HK** と呼ぶことにする．同様に，音域を 29Hz から 247Hz とした音楽指紋を，我々の頭文字をとって **TK** と呼ぶことにする．

サブ指紋の数はずらし幅によって増減する．すなわち，フレームに重複がなければ SSF の総数も少なく，結果，検索にかかる時間や保持するデータサイズも小さくなる．今回，フレームを重複させる音楽指紋では，フレーム幅を 0.96 秒，フレームのずらし幅を 0.03 秒とした．また，フレームを重複させない音楽指紋では，フレーム幅とずらし

[*6] ただし，本稿のようなランキング方式ではない．

表 1 フレームを重複させた音楽指紋における正答率.
Table 1 Accuracy for audio-fingerprints with overlapped frames.

	全体		サビのみ	
	HK	TK	HK	TK
オフボーカル	100%	100%	100%	100%
VOCALOID 曲	96%	96%	71%	94%
J-POP 曲	79.5%	70.5%	67.9%	62.8%

表 2 フレームを重複させない音楽指紋における正答率.
Table 2 Accuracy for audio-fingerprints without overlap.

	全体		サビのみ	
	HK	TK	HK	TK
オフボーカル	100%	100%	100%	100%
VOCALOID 曲	97%	99%	78%	96%
J-POP 曲	75.6%	70.5%	56.4%	62.8%

幅を同じ 0.1 秒とした．このパラメータでは，重複させる音楽指紋は，重複させない音楽指紋のおよそ 3 倍のデータ量となる．

以上の音楽指紋に対し，クエリとして楽曲全体を用いた場合とサビ部分のみをクエリとした場合，それぞれの検索の正答率を調査した．

3.3 実験結果

フレームを重複させて生成した音楽指紋における検索の正答率を**表 1** に，フレームを重複させずに生成した音楽指紋における検索の正答率を**表 2** に示す．

3.4 考察

まず，オフボーカルの正答率は全ての環境において 100%となった．これは，クエリのデータとデータベース中の正解データが同じ CD から抽出されており，照合する条件としては最も良いためである．このような場合には，フレームを重複しない音楽指紋であっても十分に高い正答率が得られることが分かった．

J-POP 曲と VOCALOID 曲で共通する結果としては，HK，TK ともに，楽曲全体での検索の正答率に対し，サビのみでの検索の正答率が低くなるという結果となった．この結果は，歌唱者の違いが検索にマイナスの影響を与えるという予想を裏付けるものである．ただし，TK の下がり幅は HK よりも小さい．特に，VOCALOID 曲においては顕著であり，HK はクエリをサビのみにすると 80%未満まで落ち込むが，TK は 90%以上の正答率を保っている．

一方，フレームの重複の有無に関しては，J-POP に対する HK を除くすべての場合について，当初の予想に反してフレームを重複させないほうが正答率が高い結果となった．

4. 音楽指紋の帯域ごとのビット誤り率

本節では，音楽指紋の帯域ごとのビット誤り率を計測し，その結果について考察を行う．

新たに 27.5Hz から 4180Hz の音域を用いた音楽指紋を生成し，それを用いて前節と同様の検索実験を行う．ここで，音域の区切り方は対数目盛とし，分割数は 88 とする．すなわち，分割されるそれぞれの領域は MIDI データのノートナンバーと一対一で対応する．最も低い部分が A0，最も高い部分が C8 である．したがって，生成されるサブ指紋のサイズはひとつあたり 87 ビットとなる．この 87 ビットの各ビットについて，最も低い帯域に対応するものから順番に番号付けを行い，以降ではこれを**帯域番号**と呼ぶことにする．実験の便宜上，今回はフレームの重複をさせないでサブ指紋を抽出した．

上述の音楽指紋を用いてデータベースとクエリを作成し，楽曲の検索を行う．それとと同時に，一致した SSF のハミング誤りがどの帯域で起こっているのかを調べた．その手順は以下のとおりである．

(1) SSF による楽曲の検索
2 節で述べた SSF による探索を行い，ランキングによってクエリと一致する楽曲がデータベース中のどの楽曲であるかを特定する．

(2) 検索の成否判定
手順 (1) で特定された楽曲が，クエリの楽曲と本当に一致しているかを判定する．

(3) サブ指紋における各ビットの誤り率の算出
手順 (2) によって，手順 (1) で選ばれた楽曲（楽曲 A とする）がクエリと一致していると判定されたとき，楽曲 A の SSF と一致したクエリの SSF について，一致した SSF 同士でどの帯域のビットでハミング誤りが発生しているかを調べる．この手順を，楽曲 A の SSF と一致したクエリ中の SSF 全てについて行い，帯域番号ごとに誤りの発生率の平均をとる．

上記の操作により，正解の楽曲データを発見している SSF において，どの帯域で異なっているのかを確認することができる．

この結果をグラフでまとめたものが**図 2**，**図 3**，**図 4**，**図 5**，**図 6**，**図 7**，である．

これらの図において，縦軸はハミング誤りの発生率 (%)，横軸は音の高さごとに区切った区間であり，左側が低く右側が高い音域となっている．この横軸の数値は帯域のインデックスである．前節の実験で使用した 2 種類の音楽指紋は，HK が帯域番号 42 から 73 の部分を抽出したものに対応し，TK が帯域番号 7 から 38 の部分を抽出したものに対応している．

図 2，図 4，図 6 は，それぞれの種類のクエリにおける

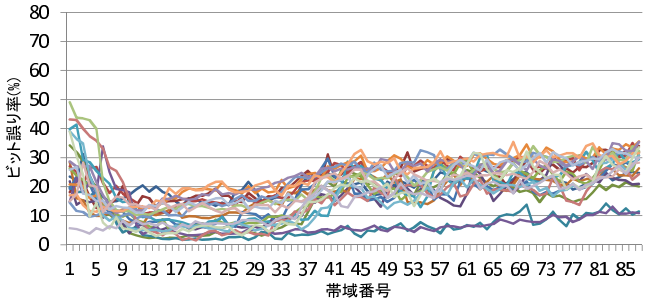

図 2 オフボーカルの帯域ごとのビット誤り率.
Fig. 2 Bit error rate (off vocal).

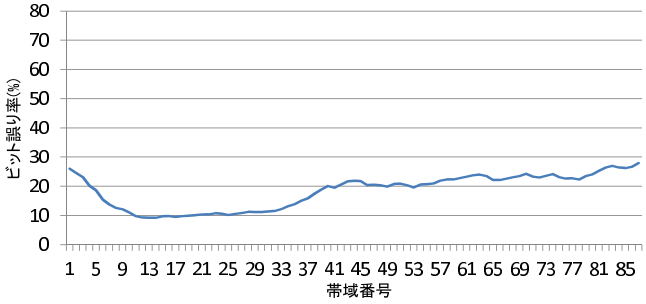

図 3 オフボーカルの帯域ごとの平均のビット誤り率.
Fig. 3 Average of bit error rate (off vocal).

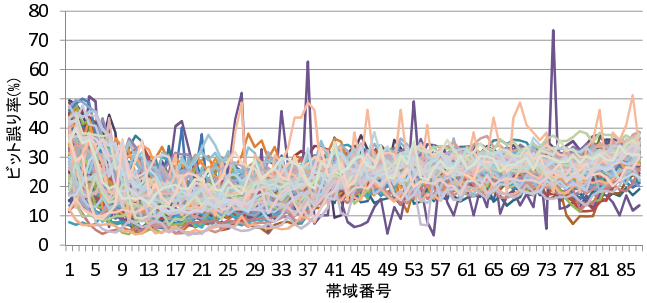

図 4 VOCALOID 曲の帯域ごとのビット誤り率.
Fig. 4 Bit error rate (VOCALOID songs).

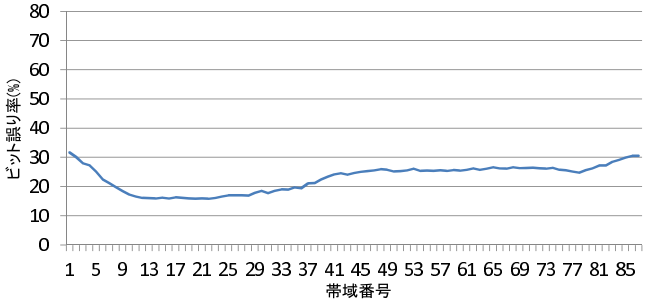

図 5 VOCALOID 曲の帯域ごとの平均のビット誤り率.
Fig. 5 Average of bit error rate (VOCALOID songs).

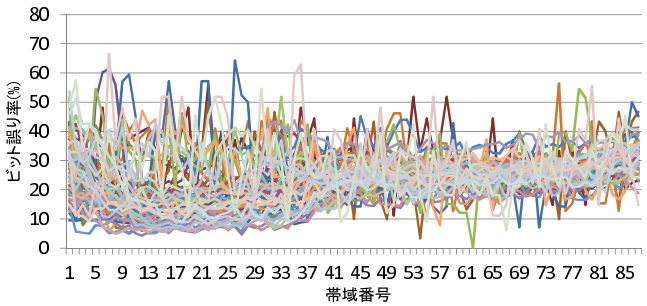

図 6 J-POP 曲の帯域ごとの平均のビット誤り率.
Fig. 6 Bit error rate (J-POP songs).

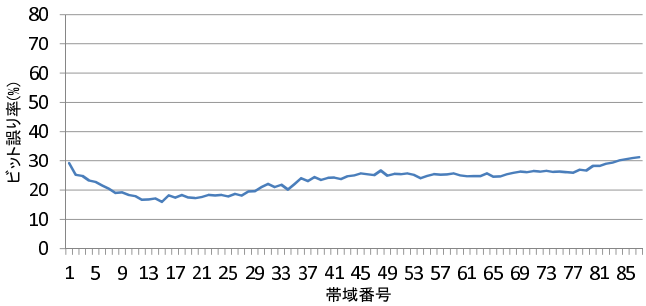

図 7 J-POP 曲の帯域ごとの平均のビット誤り率.
Fig. 7 Average of bit error rate (J-POP songs).

SSF の帯域番号ごとのハミング誤りの発生率である．グラフの一本一本がそれぞれひとつの楽曲に対応する．このグラフが高ければ高いほど，そのビットでのハミング誤りが多く発生していたということである．また，図 3，図 5，図 7 は，それぞれ図 2，図 4，図 6 において，帯域番号毎に平均をとったものである．

まず，オフボーカル曲と VOCALOID 曲では全体的に似た傾向が見られた．どちらも，最も低い帯域と最も高い帯域の両方でビット誤り率が高い．それ以外の部分では，比較的低い帯域でのビット誤り率が低い傾向がある．これは，前節の実験の VOCALOID 曲の場合において，HK よりも TK のほうが正答率が高いという結果を裏付けている．

J-POP 曲については，他の 2 種類のクエリと比べて異なる点が見られた．平均をとった曲線について，オフボーカルや VOCALOID 曲の場合と傾向が共通している．しかしながら，低い音域では，クエリによってビット誤り率の分散が大きい．このことが，J-POP の場合おいて，今回のようなランキング方式で安定的な検索が行えないことの原因になっていると考えられる．

5. おわりに

本稿では，異なる歌い手によるカラオケ曲をクエリとしてその原曲を検索する問題について議論し，音楽指紋の違いによって検索の正答率に影響があることを確かめた．

その結果，VOCALOID 曲やオフボーカル曲をクエリとしたノイズの少ない場合には，既存の音域（300Hz から 2000Hz）を抽出した音楽指紋よりも，本稿で提案した低音域（29Hz から 247Hz）を抽出した音楽指紋のほうが優れていることが分かった．特に，ボーカルが含まれているサビ部分のみをクエリとした際の結果では，前者が正答率を大きく落とすのに対して後者は高い正答率を保った．また，音楽指紋を生成する際にフレームの重複の有無につい

ては，今回用いたランキング方式の検索では，ほぼ影響がないか場合によっては正答率が向上する結果となった．

今後の課題は，クエリが短くノイズの多い場合でも頑健な検索が行える方式を開発することである．そのためには，音楽指紋と検索方式両方のバランス良い設計が課題となる．また本稿では，検索速度については議論しなかったが，今後ますます増加する楽曲データの効率よい管理のためには，より高速な検索システムの開発が必須である．

参考文献

[1] 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 '99, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., pp. 518–529 (online), available from ⟨http://dl.acm.org/citation.cfm?id=645925.671516⟩ (1999).

[2] Kulis, B. and Grauman, K.: Kernelized locality-sensitive hashing for scalable image search, *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2130 – 2137 (online), DOI: 10.1109/ICCV.2009.5459466 (2009).

[3] Kulis, B. and Darrell, T.: Learning to hash with binary reconstructive embeddings, *Proc. NIPS, 2009*, pp. 1042–1050 (2009).

[4] Xiao, Q., Suzuki, M. and Kita, K.: Fast Hamming Space Search for Audio Fingerprinting Systems, *ISMIR*, pp. 133–138 (2011).

[5] Haitsma, J. and Kalker, T.: A Highly Robust Audio Fingerprinting System, *ISMIR 2002* (2002).

[6] Muller, M. and Ewert, S.: Chroma Toolbox: Matlab Implementations for Extracting Variants of Chroma-Based Audio Features, *ISMIR*, pp. 215–220 (2011).