Vol.2009-MUS-80 No.6
2009/5/21

複数F0候補データベースによる歌声楽曲検索の検討

小杉 優[†1] 伊藤 仁[†1]
伊藤彰則[†1] 牧野正三[†1]

本稿では，複数 F0 候補を持つデータベースを用いた歌声楽曲検索について検討する．ここでは，CD 等の音響信号から自動的にメロディーラインの基本周波数 (F0) 候補を複数選択し，データベースとして保持する．上記データベースに対し，楽曲検索を行う手法について検討した．その結果，F0 候補を複数個使用することによって，単独 F0 を用いた場合の 1 位正解率を，29.2%から 41.7%，10 位以内正解率を 58.3%から 70.8%まで改善することができ，F0 候補を複数個持つことの優位性が確かめられた．

Music Information Retrieval using database with multiple F0 candidates

Yu Kosugi,[†1] Masashi Ito,[†1] Akinori Ito [†1]
and Shozo Makino[†1]

In this paper, we propose a melody-based music information retrieval that uses a database with multiple F0 candidates. This database contains multiple F0 candidates of melody lines extracted from acoustic signals such as CDs, automatically. We conducted an experiment of music retrieval using the proposed database. The experimental result showed that the retrieval accurary of the top candidate was improved from 29.2% to 41.7%, and that of the top 10 candidates was improved from 58.3% to 70.8%. This result proved the effectiveness of the proposed method.

†1 東北大学大学院工学研究科
Graduate School of Engineering, Tohoku University

1. はじめに

近年，音楽情報処理の産業的な利用が急激に増加している．それに伴い，楽曲検索の必要性が高まっている．従来の音楽情報検索は,「楽曲のタイトル」,「アーティスト名」等といった，テキストベースの入力によるものに限られていた．そこで，メロディーラインをユーザーがハミング歌唱することにより検索を実現するハミング楽曲検索 (Query-by-Humming,QBH) の手法に関する研究が進められている[1)–3)]．

ハミング楽曲検索は，入力音声から特徴量を抽出し，楽曲データベースとのマッチングを行うことにより実現される．従来の楽曲検索にまつわる研究では，MIDI 等によるメロディーラインの音符情報が存在していると仮定している．しかし，実際には事前に音符情報が与えられず，音響信号しか存在しない場合も多い．音響信号のみのデータをデータベースに追加する場合，手動で採譜を行う必要があるが，これには手間やコストがかかる．また，自動でデータベース構築を行う場合，音響信号からのメロディーラインの F0 推定が必要となるが，現状での複数音源が存在する音響信号からのメロディーラインの F0 推定を完全に行うことは困難である．そこで，記号化されたメロディーラインによる MIDI 等の形式とは異なるアプローチによって，データベースを構築し，それを用いることで生の音響信号からの検索を可能とする必要がある．

そこで本研究では，メロディーラインの基本周波数 (F0) を一意に推定しない状態でデータベースを構築し，F0 推定が不完全な状態でも音響信号から自動で歌声による楽曲検索を行うことを目標とする．

2. データベース構築

2.1 データベース構築の流れ

従来のハミング検索におけるデータベースは，各楽曲に対し，MIDI 形式のようにメロディーラインの基本周波数 (F0) の時間的な変化を記述したものが一般的である．このようなデータベースの形態の場合，音響信号のみのデータから自動でデータベースを追加構築する場合，音響信号からのメロディーラインの F0 推定が必要となる．しかし，複数音源が存在する音響信号からのメロディーラインの F0 推定を完全に行うことは困難である．

そこで，本研究では，F0 を一意に決めずに複数個の候補を F0 候補としてデータベースに用意することで，上述の問題に対処する．このときの，データベース構築の一連の流れは次のようになる．

Vol.2009-MUS-80 No.6
2009/5/21

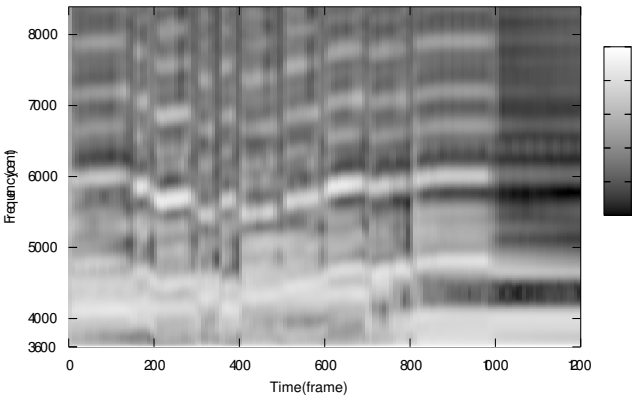

図 1 $p_{F0}(F)$ の分布例
Fig. 1 Example of $p_{F0}(F)$ distribution.

(1) 音響信号からの特徴量の抽出
(2) 基本周波数 (F0) の存在確率の推定
(3) 基本周波数 (F0) 候補の選択
(4) データベースの作成

2.2 基本周波数 (F0) の存在確率の推定

F0 推定には，後藤によって提案されている「PreFEst(Predominant F0 Estimation)」[4] の PreFEst-core 部を用いる．この PreFEst では，モノラルの音楽音響信号に対し，その信号中のメロディーラインの推定を実現している．このシステムにより，ある時刻 t における周波数 $F^{\star 1}$に基本周波数 (F0) が存在する確率密度関数 $p_{F0}(F)$ を推定することが可能となる．この操作をすべての時刻 t に対し行うことにより，楽曲の確率密度関数が求まる (**図 1**).

2.3 F0 候補の選択

本研究では，PreFEst により求まった確率密度関数から，$p_{F0}(F)$ の値の高いピークをいくつか選択し，それを F0 の候補とする．このとき，楽曲の時刻 t における i 番目の F 候補の周波数を $db_i(t)$ とする．このようにして，複数個の F0 候補を楽曲のデータとし，データベースの構築が実現できる (**図 2**).

⋆1 対数周波数表現 cent を用いる．

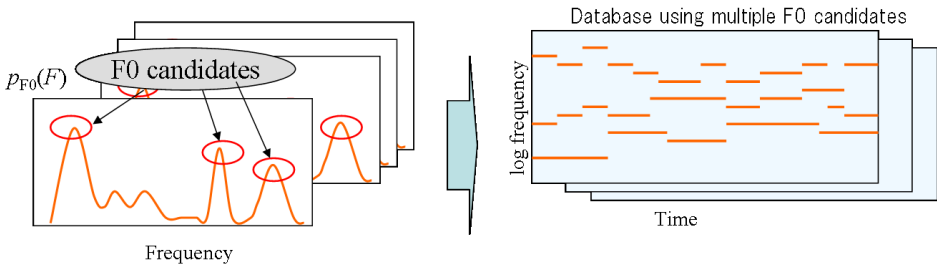

図 2 データベースの構築方法
Fig. 2 How to make database.

表 1 既存手法の解決方法
Table 1 How to solve the issue.

	既存手法
テンポの違い	IOI(Inter Onset Interval) を用いる
キーの違い	音程特徴量を用いることで解消
歌唱部分の違い	検索クエリを時間軸上でシフトさせてマッチング

3. 楽曲検索手法

3.1 問 題 点

楽曲検索を行う際に，入力された検索クエリとデータベースとのマッチングを行うにあたり，以下のような問題点があげられる．

- 歌唱者によるテンポの違い
- 歌唱者の歌唱キーの違い
- 楽曲の歌唱部分の違い

3.2 問題解決手法

市川らの研究[5] では，上記の問題に対し，**表 1** のように問題を解決していた． しかしながら，今回提案する複数 F0 候補を持つデータベースでは，テンポの違いについては IOI が算出できないという理由で，キーの違いについては音程の候補が複数存在するという理由で，上述の解決方法をそのまま使うことが出来ない．

そこで，既存手法とは異なるアプローチから問題点を解決する必要がある．上述の問題を解決するために，テンポの違いにはテンポ非依存の時間軸を導入し，キーの違いには周波数

軸上でクエリをシフトさせることによりマッチングを行った．

3.3 テンポに依存しない時間軸の導入

ハミング検索を行う場合，歌唱者によりその歌唱するテンポには個人差が現れる．そこで，構築されているデータベース楽曲の時間軸については，実時間軸ではなく，64 分音符単位を 1 フレームとするテンポ非依存な時間軸を導入した．これにより，クエリのテンポが異なる場合も検索が可能となる．

3.4 マッチング

データベース db とのマッチングについては，入力されたクエリ $in(t)$ を db 上でシフトすることにより，最もスコアの高い位置を見つけ検索することにより実現される．なお $in(t)$ は時刻 t における歌唱入力の F0 の値とする．

つまり，入力クエリとデータベース楽曲とを 1 フレームごとに比較し，スコアを算出し，入力クエリを時間軸，周波数軸上でシフトすることで，最もスコアの大きくなる箇所を見つけ，そのときのスコアを楽曲自体のスコアとする．以上の動作を全ての楽曲に行う．なお，周波数軸をシフトすることで，検索クエリのキーの差を吸収することが出来る．

入力音声の時刻 t における F0 の周波数を $in(t)$，データベースの時刻 t における i 番目の F0 候補の周波数を $db_i(t)$ とすると，楽曲のスコア S は以下のように表わされる．

$$S = \underset{0\le\tau\le t_{db}, 0\le F\le f_r - f_{in}}{\operatorname{argmax}} \left(\sum_{t=0}^{t_{in}} m\left(in(t) + F, db_1(t+\tau), ..., db_n(t+\tau)\right) \right) \tag{1}$$

ただし，t_{in},t_{db} はそれぞれ入力および楽曲の長さ，f_{in},f_r は $in(t)$,db の周波数帯域の幅を示す．また，$m(in, db_1, ..., db_n)$ はマッチング関数であり，以下で定義する．

3.5 スコアリング

今回の報告では，以下の 5 パターンのスコアリング算出を導入した．

(1) F0 候補を 1 個使用し，入力 F0 との差が 50 cent 以内という正解判定によりスコアをつける場合．

$$m(in, db_1, ..., db_n) = \delta_{50}(in, db_1) \tag{2}$$

$$\delta_{50}(x, y) = \begin{cases} 1 & |x - y| < 50 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

(2) F0 候補の上位 3 個を使用し，(1) と同様に正解判定を順に行う場合．

$$m(in, db_1, ..., db_n) = \max_{1\le i\le 3} \delta_{50}(in, db_i) \tag{4}$$

(3) F0 候補の上位 3 個を使用し，(1) と同様に正解判定を順に行う場合．この際，1 位から 3 位 F0 候補の間で重み付け (1 位:2 位:3 位 =3:2:1 の比率) を行う．

$$m(in, db_1, ..., db_n) = \max_{1\le i\le 3} \delta_{50}(in, db_i)(4 - i) \tag{5}$$

(4) PreFEst-core で出力された確率密度 $p_{F0}(F)$ を，スコアとして足し合わせた結果を用いる場合．

$$m(in, db_1, ..., db_n) = p_{F0}(in) \tag{6}$$

(5) PreFEst-core で出力された確率密度 $p_{F0}(F)$ を対数変換した値を，スコアとして足し合わせた結果を用いる場合．

$$m(in, db_1, ..., db_n) = \log p_{F0}(in) \tag{7}$$

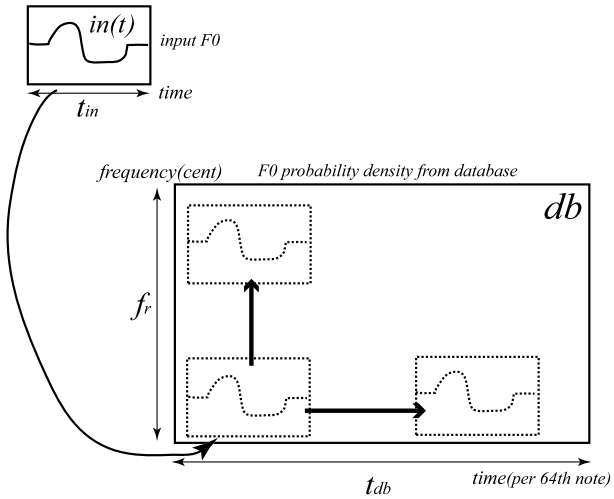

図 3 マッチング
Fig. 3 How to match.

4. マッチングシステムの実装と検証

4.1 実験条件

生成されたデータベースに対しマッチングを行った．実験条件を**表 2** に示した．データベース楽曲には，RWC 研究用音楽データベース[6] のポピュラー音楽のなかから 50 曲を使用した．また入力する検索クエリとしては，同データベースの 24 曲を選択し，楽曲のサビ

表 2 データベース生成について
Table 2 About database generation.

データベース楽曲	RWC 研究用音楽データベース ポピュラー音楽 (RWC-MDB-P-2001) から 50 曲 (表 3)
楽曲サンプリング周波数	44.1kHz
ファイル形式	モノラル raw ファイル
F0 推定周波数帯域	3600-8400 cent(130-2090 Hz)
周波数の刻み	10 cent (半音の 1/10)
入力クエリ	RWC 研究用音楽データベースポピュラー音楽 (RWC-MDB-P-2001) のメロディーラインを手動でラベル付けしたもの 24 曲分 (表 3)

表 3 マッチング使用楽曲
Table 3 Used music in matching experiment.

	RWC ポピュラー音楽データベース (RWC-MDB-P-2001)
入力クエリ	24 曲 no.1-3,5-13,15-17,19,20,28,32-34,40,47,48
データベース	no.1-50 の 50 曲分

表 4 各手法の検索結果 (%)
Table 4 Experimental results.

入力楽曲	手法 (1)	手法 (2)	手法 (3)	手法 (4)	手法 (5)
	1 個候補	3 個候補	3 候補 (重み付)	$p_{F0}(F)$	$\log p_{F0}(F)$
1 位検索	29.2	41.7	41.7	20.8	29.2
3 位以内検索	45.8	41.7	50.0	25.0	45.8
5 位以内検索	58.3	62.5	62.5	33.3	50.0
10 位以内検索	58.3	62.5	70.8	45.8	58.3
20 位以内検索	66.7	70.8	70.8	62.5	66.7
30 位以内検索	87.5	87.5	91.6	75.0	87.5

の一部分のメロディーラインの F0 を採譜し手動でラベル付けしたものを入力した (**表 3**). 今回入力する検索クエリは，正確なメロディーラインの F0(ただし，高さのみ相対化) であると仮定している．

4.2 結　　果

結果を**表 4** に示す．

全体的に，F0 単独候補を用いた場合の結果 (手法 (1)) に比べ，複数個 F0 候補を用いた手法 (2)，(3) のほうが検索結果が上昇している．逆に，確率密度関数を用いた手法 (4)，(5) は複数候補を用いた場合よりも，正解率が低い結果となってしまっている．

マッチングがうまくいった場合の，マッチング例と楽曲スコアを**図 4**，**図 5** に掲載する．

図のように 1 位 F0 候補にあうように検索クエリをシフトし，マッチングさせることに成功している．また，楽曲スコアもサビの区間で最もスコアが高くなるというように，正しくサビの区間にシフトすることが出来た．

逆に，マッチングがうまくいかなかった楽曲の，マッチング例と楽曲スコアを**図 6**，**図 7**

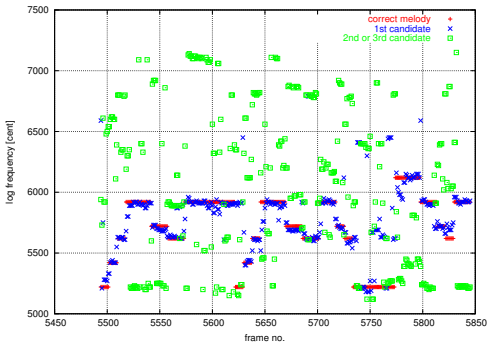

図 4 F0 候補とクエリのマッチング例 (no.07)
Fig. 4 Matching with query and F0 caindidates (no.07).

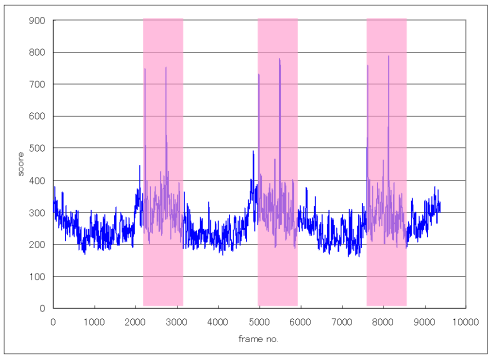

図 5 マッチングスコア (no.07, 網掛部はサビの区間)
Fig. 5 Matching score(no.07, the part painted out is Sabi).

Vol.2009-MUS-80 No.6
2009/5/21

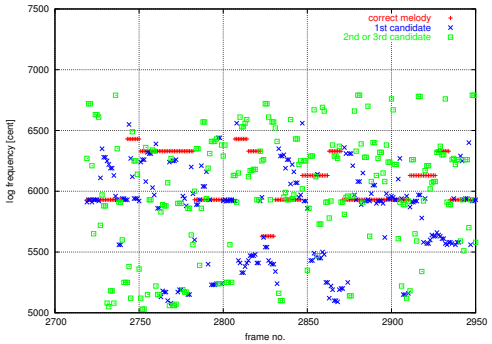

図 6 F0 候補とクエリのマッチング例 (no.06)
Fig. 6 Matching with query and F0 caindidates (no.06).

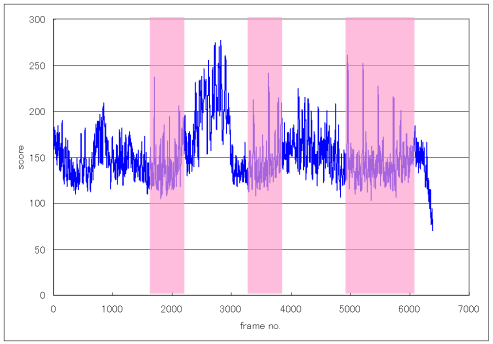

図 7 マッチングスコア (no.06, 網掛部はサビの区間)
Fig. 7 Matching score(no.6, the part painted out is Sabi)

に掲載する．

この図のように，先ほどの楽曲に比べると，F0 候補と入力クエリがあまりマッチングしていないといえる．楽曲スコアも，サビの部分以外の箇所のスコアが高くなってしまっているので，正しくマッチング出来ていないといえる．

4.3 考　　察

今回，正解精度が悪くなる原因として，特定の楽曲 (例えば no.06) については，5 つの手法全てにおいて正解精度が低くなってしまうことが考えられる．

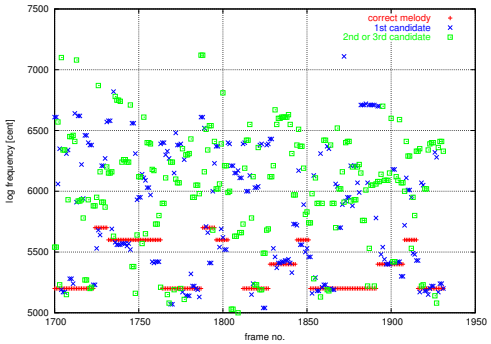

図 8 クエリ no.06 を入力したときのマッチング結果 (no.06 サビ区間)
Fig. 8 Matching result of inputted query no.06(no.06 Sabi)

楽曲検索の正解精度が低くなる原因の一つとして，一つに F0 候補選択がうまくいっていないことが挙げられる．no.06 の F0 候補選択結果 (**図 8**) では，F0 候補の周波数変動が大きくメロディーラインの軌跡を描けていない．このため，スコアが低くなってしまったと考えられる．この対策として，F0 候補の周波数変動に対し何らかの制約条件を与える必要がある．

精度低下のもう一つの原因は，他の楽曲のメロディーライン，フレーズと似ていることである．no.06 の楽曲のクエリ自身よりもスコアが高くなった no.13 の楽曲のマッチング結果 (**図 9**) を見ると，no.06 よりもクエリに対してマッチしているように見受けられる．このように，別の楽曲のスコアが高くなってしまったために，相対的に正解の楽曲のスコアが低くなり正解精度が低くなったと考えられる．

今回は比較的単純な方法でスコアリングを行ったので，今後は，スコアリングの値を調整し，以下の問題を解決するような値を導く必要がある．

5. ま と め

本研究では，F0 を一意に推定しない複数 F0 候補を用いたデータベースを自動構築し，上記データベースを用いた楽曲検索手法について検討した．F0 候補を複数 (3 個) 利用することにより，1 位正解率を 29.2%から 41.7%，10 位以内正解率を 58.3%から 70.8%まで改善することが出来た．

今後は，F0 候補推定の精度の向上を目指すとともに，マッチングにおけるスコアリング

Vol.2009-MUS-80 No.6
2009/5/21

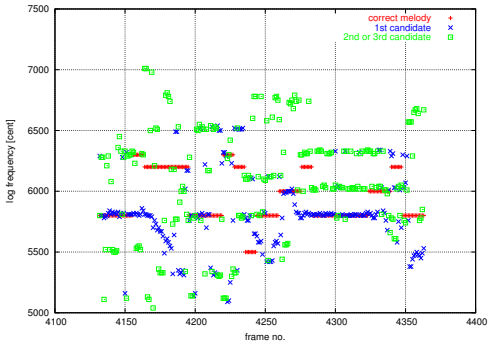

図 9 クエリ no.06 を入力したときのマッチング結果 (no.13 サビ区間)
Fig. 9 Matching result of inputted query no.06(no.13 Sabi)

の手法について検討する．また，実際の歌声入力に対しての検索についても比較検討を行う必要がある．

参 考 文 献

1) 後藤真孝，平田圭二：音楽情報処理の最近の研究，日本音響学会誌， Vol.60, No.11, pp.675–681 (2004).
2) 後藤真孝， 齋藤毅，中野倫靖，藤原弘将：歌声情報処理の最近の研究，日本音響学会誌， Vol.66, No.10, pp.616–623 (2008).
3) Demopoulos, R. and J.Katchabaw, M.: Music Information Retrieval: A Survey of Issues and Approaches, Technical Report 677, Department of Computer Science The University of Western Ontario (2007).
4) Goto, M.: A Real-time Music Scene Description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-world Audio Signals, *Speech Communication*, Vol.43, No.4, pp.311–329 (2004).
5) 市川拓人，鈴木基之，伊藤彰則，牧野正三：複数の音程特徴量によるハミング入力楽曲検索システムの高精度化，情処研資 2008-MUS-074，pp.7–12 (2008).
6) 後藤真孝，橋口博樹，西村拓一，岡 隆一：RWC 研究用音楽データベース: ポピュラー音楽データベースと著作権切れ音楽データベース，日本音響学会 2002 年春季研究発表会 講演論文集 2-6-7，pp.705–706 (2002).