SoundCompass：ハミングによる音楽検索システム

小杉尚子[†] 櫻井保志[†]
山室雅司[††] 串間和彦[†††]

本論文では，我々が研究開発しているハミングを用いた音楽検索システム “SoundCompass” を，カラオケの選曲システムとして実用化するために考案した技術について述べ，それらの有効性を定量的に評価する．このシステムを実用化するにあたって，一般ユーザの使用により，様々なハミングに柔軟に対応することが必要となった．データベース・サイズの増大に対応できるように，蓄積コストや検索速度の改善も必要となった．本論文では様々なハミングに対応するために，これまでの手法では検索できなかったハミングについてその原因を明らかにし，それらを解決するために部分特徴ベクトルの導入，部分ハミング片間 OR 検索方式の導入，半テンポ曲/倍テンポ曲の重複登録の導入を提案する．一方，検索対象曲数の増加にともなう音楽データベースの大規模化に対して，曲内での繰返しによる特徴ベクトルの冗長性を手がかりにしたデータベース・サイズの縮小を提案する．これらの技術を導入することで，新しい SoundCompass は実用範囲内の検索速度を保ちつつ，20,000 曲を超えるデータベースに対して 84.2%の検索精度を達成し，従来の我々のシステムと比較して，約 20%の向上が可能となった．

SoundCompass: A Music Retrieval System with Humming

Naoko Kosugi,[†] Yasushi Sakurai,[†] Masashi Yamamuro[††]
and Kazuhiko Kushima[†††]

This paper describes techniques incorporated into SoundCompass, a query-by-humming system, to enable it to be put to practical use as a karaoke song selection system. Quantitative evaluations of the techniques are also provided. Solutions for variations in hummed tunes by general users are required to make the system practical. Moreover, improvements in both the cost of storing a large number of songs in a database and retrieval efficiency are also required to deal with database size expansion. In this paper, we analyze hummed tunes which cannot be retrieved by traditional techniques. According to the analysis, a partial feature vector, an OR retrieval among query keys, and double registration of songs whose tempo is doubled/halved are proposed to deal with the diversity among hummed tunes. We also propose a method to reduce database size based on the repetitive structure of songs to solve the problem of increasing database size as the number of songs stored in the database becomes larger. The new SoundCompass system achieves 84.2% retrieval accuracy, which is about 20% more than that of the previous system, for a database that stores over 20,000 songs, while maintaining the applicable retrieval time required for practical use.

1. はじめに

マルチメディア・データの使用は広く一般ユーザに普及し，これにともなって大規模で効率的なマルチメディア・データベースの構築，またそのようなデータベースに対する効率的な内容検索方式の確立は重要な課題となっている[1)~4)]．

我々は，音楽データベースに対する内容検索システムの 1 つとして，ハミング検索システム “SoundCompass” を研究開発中である．SoundCompass は，人の歌唱（ハミング）を検索キーとして，そのハミングに類似する部分を持つ曲を，類似する順に出力することができる．メトロノームに合わせて「タタタ」で数小節歌うと，数秒で検索結果を出力する．楽曲データには約 2 万曲の MIDI データを使用している．検索には多次元特徴ベクトルに基づくインデクスを利用した HyperMatch エンジン[5)]を使用している．

SoundCompass は，様々な改良の末，2000 年 12 月

† NTT サイバースペース研究所
NTT Cyber Space Laboratories
†† NTT サイバーソリューション研究所
NTT Cyber Solutions Laboratories
††† NTT ドコモマルチメディア研究所
NTT Docomo Multimedia Laboratories

に世界で初めて商用のハミング検索システムとして実用化された[6]．この実用化にあたって，検索精度向上を達成し，ハミング特有の“検索キーとしての曖昧さ”に柔軟に対応することが可能となった．さらにデータベース・サイズの増大に対応するために，蓄積コストと検索速度を改良した．本論文では，検索精度を向上させるための新しい技術を述べ，その有効性について評価する．さらに，改良した蓄積コストと検索速度についても定量的に評価する．最後に，ユーザ・インタフェースについて述べる．

2. 関 連 研 究

ハミングを入力して曲を検索する，いわゆるハミング検索技術の研究は 1990 年代に本格的に始まった[1),7)~9)]．2000 年には The International Conferences on Music Information Retrieval and Related Activities（ISMIR）という，音楽検索に関する専門の国際会議も発足し，ハミング検索は重要な研究課題の 1 つとなっている[10)~12)]．

MUSART[11)]の大きな特徴は，テーマのインデクスを自動的に作成できる点と，検索手法が複数用意されている点である．楽曲の繰返し構造などが分析され，テーマが自動的に抽出されるので，技術の有効性はハミング検索にとどまらない．また，マルコフモデリングや，メロディライン，音声ストリームまたは歌詞など，複数の情報を利用した検索が可能なので，効率の良い絞り込みが可能だが，入力情報に間違いが多ければ，検索精度への悪影響も大きいと思われる．SoundCompass は，ハミング検索のための音楽の特徴を効果的に表現する手法と，それを用いて効率的に類似メロディを検索するための手法の確立を目指しているので，複数の入力情報を用いてマルチモーダルに楽曲を検索する MUSART とは立場が異なる．

Nishimura ら[12)]によるシステムは，SoundCompass を含む従来のハミング検索システム[13),14)]と違って，データベースに WAV 形式の音楽データを用いている点や，DP マッチングに様々な改良を加えてハミング検索に適用している点に特徴がある．特に文献 12) では，ハミングの最初の音が正しいという仮定の下で，検索時間を 1/40 に削減することに成功している．

3. 従来の SoundCompass と検討課題

3.1 従来の SoundCompass システム

文献 15) では，データベースには約 10,000 曲を収録しており，検索精度は人が聞いて曲の一部であると判断できるレベル☆のハミングの約 70%に対して，正解を 5 位以内に出力することができた．

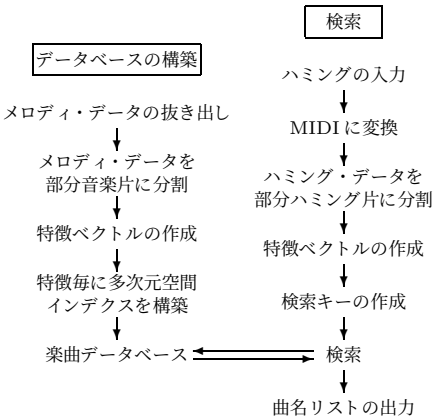

図 1 SoundCompass[15)]の処理の流れ
Fig. 1 Processes for SoundCompass[15)].

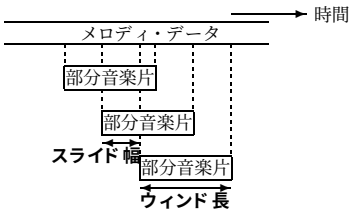

図 2 スライディング・ウィンド方式によるメロディ・データの部分音楽片への分割
Fig. 2 Split of melody data into subdata by a sliding-window method.

SoundCompass[15)]の処理の流れを**図 1** に示す．データベースを構築する（図 1 左側）ために，まず，MIDI データからメロディ・データを抜き出し，スライディング・ウィンド方式で，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）を切り出す（**図 2** 参照）．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，特徴ベクトルに変換した後，それらを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．特徴ベクトルとしては，音高の時系列を表現する「音高推移特徴ベクトル」と，隣り合う音符の音高差の分布を表現する「音高差分布特徴ベクトル」を使用する．音高推移特徴ベクトルにおいて，隣り合う要素間の拍数（$= b$）を決める値を拍粒度 r といい，$r = 1/b$ である．音高の時系列は，拍粒度ご

☆ 4.1 節の判定方法と同じ．

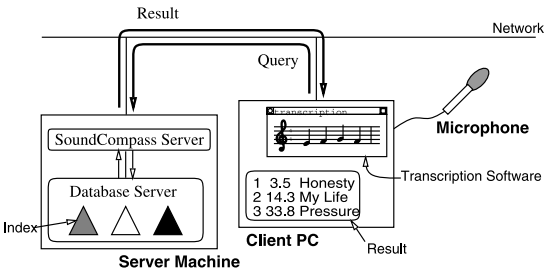

図 3 SoundCompass システム構成
Fig. 3 SoundCompass system architecture.

とに最も長く演奏される音高の列で構成され，拍粒度値が r，ウィンド長が w 拍のとき，$w \times r$ 次元の特徴ベクトルとなる．なお，音高推移特徴ベクトルでは，「基準音」を基準とする相対音高値を用いて音高列を表現する．

検索（図 1 右側）の際は，マイクから入力されたハミングを MIDI に変換し，データベースの構築における部分音楽片の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．楽曲データもハミングも「楽譜に記載されたテンポ」と「メトロノームの打拍」を用いて時間正規化されているので，ユーザは任意のテンポで歌って検索することができる．また，特徴ベクトルには相対音高値を用いているので，ユーザは任意の高さのキーでハミングしてかまわない．

図 3 にシステム構成を示す．ネットワーク経由でサーバマシンとクライアント PC がつながっている．サーバマシン側には，SoundCompass サーバ（SoundCompass Server）と，データベース・サーバが動いている．データベース・サーバは内部に特徴量ごとの多次元空間インデクスを保持している．クライアント PC にはマイクがつながっていて，ハミング検索用 GUI（図中では検索結果例が表示されている）と採譜ソフトが動作している．マイクから入力された歌声は，採譜ソフトで MIDI に変換されて SoundCompass サーバに送られる．SoundCompass サーバではハミングから問合せを作成し，データベース・サーバに送信する．データベース・サーバは検索終了後，結果を SoundCompass サーバに返送する．SoundCompass サーバは検索結果を整形加工して，クライアント PC に返信する．

3.2 検討課題

SoundCompass [15] をカラオケの選曲システムとして実用化するにあたって，登録曲数を増やし，被験者の層を広げて実験を行った結果，蓄積コストや検索速度の改良，および様々なハミングに柔軟に対応することが必要となった．要求事項を整理すると以下のようになる．

(1) ハミングの多様化
今まで想定されていなかった新たな検索エラーが発生した．エラーを起こしたハミング・データを詳しく分析し，それに基づいて新しい特徴量や検索方法を検討する必要がある．

(2) データベース・サイズの増大
データベースへの登録曲数の増加によってデータベースのサイズもリニアに増加した．これにより検索速度の向上とデータの蓄積コスト削減が必要である．

このほか，一般ユーザにサービスするにあたって，使いやすい GUI が必要である．7 章で，実際にカラオケボックスで使用しているものを紹介する．

4. データ分析/調査

本章では，まず 4.1 節でハミング・データの分析を行い，SoundCompass [15] について，検索精度に関する解決すべき問題点を明らかにする．次に 4.2 節では楽曲データの調査を行い，データベースのサイズに関する問題を解決するための知見を得る．

4.1 ハミング・データの分析

分析対象のハミング・データとして，以下の条件を満たす，39 人の被験者による 240 件を選出した．

- 人が聞いて曲の一部であると判断できること．具体的には，ハミングされた曲をよく知る 3 人以上の人が，3 段階（A，B，C）でハミングを評価したとき，過半数の人が A または B をつけていること．
  - A ··· 検索できて当然
  - B ··· 検索されて欲しい
  - C ··· 検索できなくてもやむをえない
- 各被験者について，ハミング曲は重複しないこと（同じ人が同じ曲を何度もハミングしない）

SoundCompass [15] は，この 240 件中 84 件について，5 位以内に正解を検索することができなかった．主な原因は 3 つあることが分かり，4.1.1 項から 4.1.3 項でそれぞれについて詳しく述べるが，検索できなかった 84 件の原因の内訳を**表 1** に示す．

表 1 84 件の検索されなかったのハミングの誤りの原因の内訳
Table 1 Reasons behind 84 non-retrievable cases.

要因	件数	要因	件数
A	4	A&B	2
B	19	A&C	3
C	21	B&C	25
D	8	A&B&C	2

A：半テンポミス/倍テンポミス（4.1.1 項）
B：部分ハミング片の選択エラー（4.1.2 項）
C：特徴ベクトルの先頭の不一致（4.1.3 項）
D：その他

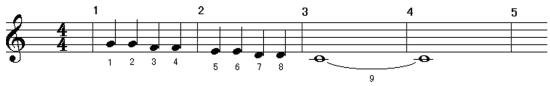

図 4 曲 A の楽譜．テンポは 180
Fig. 4 Segment of song A. The tempo is 180.

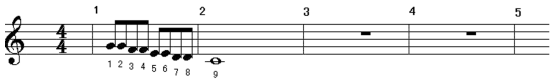

図 5 曲 B の楽譜．テンポは 90
Fig. 5 Segment of song B. The tempo is 90.

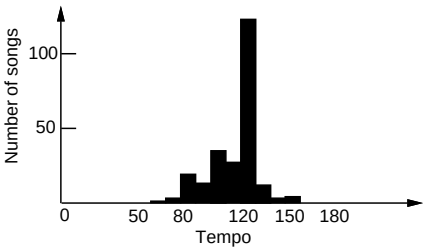

図 6 ハミングのテンポの分布
Fig. 6 Humming tempo distribution.

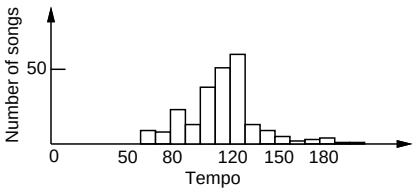

図 7 ハミングされた旋律に対応する実際の楽曲部分のテンポの分布
Fig. 7 Tempo distribution of hummed parts of songs.

4.1.1 半テンポミス/倍テンポミス

四分音符が八分音符や二分音符に間違えられたハミングは 11 件あった．楽譜に四分音符で記載されている音符を八分音符として歌うということは，半分のテンポで歌うということである．逆は 2 倍のテンポで歌うということである．実験中の観察から，楽譜に記載されているテンポが速い場合は，四分音符を八分音符に間違える人が多く，楽譜に記載されているテンポが遅い場合は，四分音符を二分音符に間違える人が多いということが分かった．本論文では，このように四分音符を八分音符に間違えて歌うことを半テンポミス，逆に四分音符を二分音符に間違えて歌うことを倍テンポミスと呼ぶ．本項では，最初にこの現象とその影響について説明し，次にこの問題を起こしたハミング・データの分析結果を報告する．

半テンポミスを例を用いて説明する．曲 A（**図 4**）のテンポは 180 で，曲 B（**図 5**）のテンポの 2 倍であり，各々の楽譜上の見た目は大きく異なるが，演奏は同じである．なぜならテンポ 90 における八分音符はテンポ 180 における四分音符と実演奏時間は等しいからである．

SoundCompass ではメロディ・データやハミング・データに対して，楽譜に記載されたテンポまたはメトロノームの打拍を基準にしたデータの時間正規化を行っているので，ユーザが楽譜に記載されているテンポより少々遅い/速いテンポでハミングしても検索精度に影響しない．しかし楽譜に記載されているテンポより極端に遅い/速いテンポで歌うと，半テンポミス/倍テンポミスを起こす確率が高くなる．これらのテンポミスを起こすと，1 拍あたりの情報量が 2 倍または半分になるので，特徴ベクトルとしてまったく別のものになり，ハミング・データとメロディ・データの正しいマッチングができなくなる．これはどんなに上手にハミングしても，また何度やり直しても決して正解の曲を検索することができないので，深刻な問題である．

図 6 にハミング・データのテンポの分布を示す．これらは，ハミングする際にユーザ自身が選んだものである．SoundCompass のユーザ・インタフェースでは，録音の際のメトロノームの初期値は 120 である．**図 7** に楽曲におけるハミングされた旋律に対応する実際の楽曲部分のテンポ☆の分布を示す．図 6 からハミングのテンポは 120 に集中していることと，80〜120 のテンポで歌われているものが多いことが分かる．これは SoundCompass のメトロノームの初期値が 120 であることも影響していると思われるが，テンポ 120 は人にとってハミングしやすいテンポの 1 つである．また実際には 160 を超えるテンポを持つ曲もあることが

☆ 実際の楽曲では全曲の中でテンポは頻繁に変化する．最も変化の激しいもので，ボーカルのある部分だけで 1122 回もテンポが変化する曲もあった（テンポ情報のメタイベントをカウントすることで検出）．

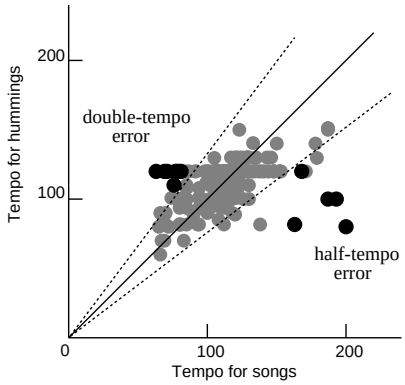

図 8 楽曲のハミングされた旋律に対応する実際の楽曲部分のテンポとハミングのテンポの関係
Fig. 8 Tempos of hummed parts of songs and hummings.

図 7 から分かるが，そのような速いテンポでハミングした人がいないことは図 6 から分かる．同じように，図 7 からテンポが 80 未満の曲は 10 曲以上あることが分かるが，そのような遅いテンポでハミングした人はそれよりもかなり少ないことが図 6 から分かる．

図 8 にハミングする際にユーザが選択したテンポと，ハミングされた旋律に対応する実際の楽曲部分のテンポの関係を示す．横軸はハミングされた旋律に対応する実際の楽曲部分のテンポを示し，縦軸はハミングのテンポを示している．中央の対角線（実線）を境界に，右側の黒の円は半テンポミス（half-tempo error）を起こしたケースを示している．左側の黒の円は倍テンポミス（double-tempo error）を起こしたケースを示している．グレーの円はテンポミスを起こしていないケースを示している．

黒の円が指すデータについてさらに詳しく調べた結果を**図 9** と**図 10** に示す．これらの図は，横軸はハミングされた旋律に対応する実際の楽曲部分のテンポを表している．縦軸は楽曲のテンポを表している．図中の縦棒は，楽曲の中でメロディが存在する部分の最小のテンポから最大のテンポ，つまりテンポの範囲を示している．棒の中の黒円は，その曲で最も長く使用されるテンポを示している．たとえば図 9 の最も左の縦棒は，ハミングされた旋律に対応する実際の楽曲部分のテンポは 63 で，その楽曲のメロディが存在する部分の最小のテンポは 43，最大のテンポは 63，最も長く使用されているテンポは 63 である．

図 9 は倍テンポミスに関するデータで，最小のテンポが 81 以下の楽曲にエラーが起きていることが分かる．一方，図 10 は半テンポミスに関するデータで，最大のテンポが 165 以上の楽曲にエラーが起きていることが分かる．また図 9 と図 10 から，多くの場合，

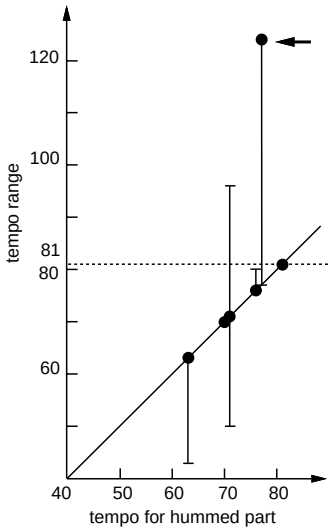

図 9 倍テンポミスを起こしたハミングの，ハミングされた旋律に対応する実際の楽曲部分のテンポと，その楽曲の最小/最大/最も長く使用されるテンポ
Fig. 9 Tempos of hummings that incur double-tempo error and the minimum, maximum and longest-used tempos of the hummed songs.

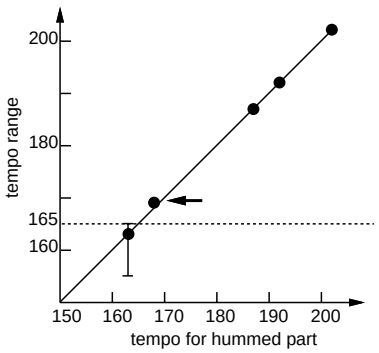

図 10 半テンポミスを起こしたハミングの，ハミングされた旋律に対応する実際の楽曲部分のテンポと，その楽曲の最小/最大/最も長く使用されるテンポ
Fig. 10 Tempos of hummings that incur half-tempo error and the minimum, maximum and longest-used tempos of the hummed songs.

楽曲中で最も長く使用されるテンポでハミングされるが，そうではない場合もある（図 9 に 1 件，図 10 に 1 件．該当するケースを矢印で指示）ということも分かる．

4.1.2 部分ハミング片の選択エラー

ユーザがウィンド長より長く歌った場合，1 つのハミング・データから複数の部分ハミング片を作成できる．文献 15) では，このような場合は検索にはハミ

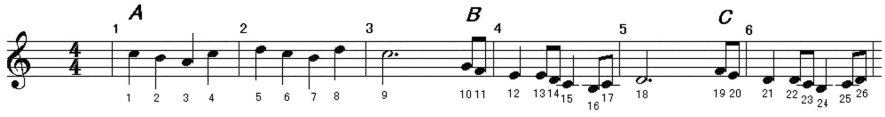

図 11 出だしの初拍の位置とサビの初拍の位置が一致しない曲の例
Fig. 11 A tune in which the start beat of the beginning part (A) is different from that of the bridge part (B).

表 2 良い検索結果を出した部分ハミング片の分布
Table 2 Distribution of hummed pieces which give good retrieval results.

生成された部分ハミング片数	ハミング・データ数（件）	歌い始め（件）	中央部分（件）	歌い終わり（件）	特定できない（件）
1	82	—	—	—	—
2	90	35	—	25	30
3	40	15	10	7	8
4	7	3	1	2	1
5	21	2	6	3	10
合計	240	55	17	37	49

ングの中央部分☆から切り出される部分ハミング片のみを使用していた．しかし分析の結果，必ずしも中央の部分ハミング片が有効であるとは限らないことが分かった．したがって，ハミング・データ全体を有効に利用する方式の検討が必要である．

ウィンド長（16 拍）より長いハミング・データは 158 件あった．被験者には 16 拍のハミング・データを収録した時点で十分な長さを得られたことを通知したが，それを承知で長く歌ったものである．なお，どんなに長く歌ってもウィンド長の 2 倍（=32 拍）ハミングされた時点で，録音は自動的に終了した．この 158 件について部分ハミング片の数と，その中でより良い検索結果をもたらした部分ハミング片について調べた結果を**表 2** に示す．「特定できない」というのは，どの部分ハミング片を用いても正しい検索結果を得られなかったものと，どの部分ハミング片を用いても正しい検索結果を得られたものの両方を含んでいる．

表 2 から部分ハミング片が 2 つの場合は，歌い始めが適していたのは 35 件，歌い終わりが適していたのは 25 件で，それぞれあまり差はない．また部分ハミング片が 3 つ以上生成されるハミングデータについては，歌い始めが最も適していたものは 17 件，中央部分が適していたものも 17 件，歌い終わりが適していたものが 12 件となり，やはりこれも部分ハミング片間にあまり差はない．このように，実際には検索キーとしての有効性はどの部分ハミング片もほぼ等しいということが分かる．

☆ 部分ハミング片が n 個の場合，$\lfloor n/2 \rfloor + 1$ 個目を使用．

4.1.3 特徴ベクトルの先頭の不一致

ハミングの先頭が，曲の出だしやフレーズの先頭であるとは限らない．また，各部分音楽片の先頭が，必ずしもフレーズなどの先頭と一致しているとも限らない．特に最近のポップスでは，曲の出だしの拍と，サビやフレーズの先頭の拍が合っていない曲が増えてきている（弱起）．つまり人の歌い出しと先頭が一致している部分音楽片が，必ずしもデータベースに存在しているというわけではないのである．

たとえば**図 11** のようなメロディを考えてみる．このメロディは，出だし（A）は 1 拍目から始まるが，サビは 3 小節目の 4 拍目（B）から始まる曲で，サビに関していえば弱起である．この曲をウィンド長 16 拍，スライド幅 4 拍で分割すると，すべての部分音楽片の先頭は各小節の先頭になる．よってユーザがサビの部分をサビの出だし（B）からハミングしても，データベース中には先頭が一致する部分音楽片は存在しない．このように，部分音楽片と部分ハミング片の先頭の間には，最大でスライド幅の半分（この場合は 2 拍）のずれが生じる可能性がある．

実際にこのようなハミングは 240 件中 48 件（全体の 20%）あり，そのうち 46 件が上位 100 位以内にも正解を検索することができなかった．これは約 5 回に 1 回のハミングは，これが原因で楽曲を検索できないということを示しており，検索精度の低下の最も深刻な原因になっている．メロディ・データを分割する際のスライド幅を小さくする（たとえばスライド幅 1 拍）という解決策も考えられるが，その場合は部分音楽片の数が増加し，データベースのサイズが大きくなってしまう．

4.2 楽曲データの調査

本節では曲内での特徴ベクトルの冗長性に関する調査結果について報告する．調査の対象とした楽曲は，21,804 曲の MIDI データである．

多くの曲において，似たようなあるいはまったく同じ部分が曲の中で繰り返し出現することはよく知られている[16]．また多くの曲が 2 番，3 番を持つ繰返し構成になっていることもよく知られている．**図 12** は全

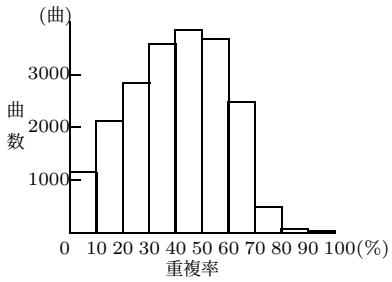

図 12 1つの曲内に同じ特徴ベクトルを生成する部分音楽片が存在する割合の度数分布

Fig. 12 Histogram of the percentage of songs that have song-pieces which generate the same feature vectors for all features of a song.

21,804 曲について，スライド幅 4 拍，ウィンド長 16 拍で楽曲を部分音楽片に分割したときに，1 曲の中にまったく同じ特徴ベクトルを生成する部分音楽片が含まれている割合（重複率）を示している．横軸は重複率を表しており，縦軸は曲数を表している．

図 12 から 30%から 60%の割合で同一曲内に同一の特徴ベクトルを生成する部分音楽片が存在する曲が多いことが分かる．特にダンス・ソングなどのリズム重視の曲や，サッカーチームの応援歌などのように，だれもが覚えやすく試合中に選手に声援を送るフレーズが繰り返し用いられる曲は重複率が高く，80%を超えていた．重複率が高いことは，蓄積コストのみならず，検索速度の性能劣化の要因となる．

4.3 分析/調査結果のまとめ

4.1 節と 4.2 節の分析/調査結果から，以下のような問題が明らかになった．

- 半テンポミス/倍テンポミス
  楽曲中の四分音符を八分音符や二分音符と間違えてハミングした場合は，正しい検索結果を得られない．倍テンポミスは，最低テンポが 81 以下の楽曲に関して生じた．半テンポミスは，最大テンポが 165 以上の楽曲に関して生じた．
- 部分ハミング片の選択エラー
  ハミングから複数の部分ハミング片がとれる場合，検索キーとしての有効性はどの部分ハミング片もほぼ等しいことが分かった．
- 特徴ベクトルの先頭の不一致
  部分ハミング片の先頭と部分音楽片の先頭が一致しない場合は，正しい検索結果を得られない．ハミングデータの約 20%について，このエラーが生じていることが分かった．
- 特徴ベクトルの冗長性
  楽曲データには 30%から 60%の割合で，同一曲内に同一の特徴ベクトルを生成する部分音楽片が存在する曲が多いことが分かった．

5. SoundCompass の改良

本章では 4 章で明らかにした問題点に対する解決策を提案する．またこれらの解決策を導入して改良した，新しい SoundCompass について述べる（5.5 節）．

5.1 半テンポ/倍テンポ曲の重複登録

4.1.1 項より，最小のテンポが 81 以下の曲については倍テンポミスが発生し，最大のテンポが 165 を超える曲については半テンポミスが発生することが分かった．そこで実用化にあたって最小のテンポが 85 以下である曲については，その 2 倍のテンポの曲を，また最大のテンポが 160 以上である曲については，その半分のテンポの曲を自動生成し，データベースに重複登録することで解決を試みる．この方式の有効性は 6.1.1 項で定量的に評価する．

5.2 部分ハミング片間 OR 検索

4.1.2 項より，検索キーとしての有効性はどの部分ハミング片もほぼ等しいことが分かった．したがって，すべての部分ハミング片から得られる検索結果を用いて最終結果をまとめる方法を検討する．

今までの実験から，複数の部分ハミング片を使って検索した場合，ある 1 つの部分ハミング片が正解の曲の部分音楽片と正しくマッチすると，その距離はきわめて小さくなるが，必ずしもその前後の部分ハミング片と正解曲の他の部分音楽片の距離も小さくなるとは限らない，ということが分かっている．このことをふまえて，複数の部分ハミング片を検索に使用しても，それぞれの検索結果を独立に評価して最終的な検索結果をまとめる方式を提案する．この方法を部分ハミング片間 OR 検索と呼ぶことにする．

ハミング h から作成した部分ハミング片数を m，曲 a の部分音楽片数を n とする．曲 a の i 番目の部分音楽片と j 番目の部分ハミング片との距離を $d(a_i, h_j)$ としたとき，曲 a とハミング h との距離 $D(a, h)$ を，

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

とする．

部分ハミング片間 OR 検索方式を具体例を使って説明する．ハミングから 3 つの部分ハミング片 h_1，h_2，h_3 を切り出したとする．データベースには，A〜D の 4 曲が登録されているとする．これら 3 つの検索キー（部分ハミング片 h_1，h_2，h_3）を使って，**表 3** のような検索結果を得たとする．() 内は距離を示す．この中で最小の距離は 0.3 で 2 番目の検索キーが曲 A に対し

表 3 各部分ハミング片による検索結果の例
Table 3 Example of retrieval results of each hummed piece.

部分ハミング片	検索結果（距離）			
h_1	D(0.9),	B(1.5),	C(1.8),	A(5.8)
h_2	A(0.3),	B(1.2),	C(2.0),	D(5.9)
h_3	B(1.0),	C(1.2),	D(1.5),	A(6.0)

表 4 表 3 の結果例から得られる最終的な検索結果
Table 4 Final retrieval result based on table 3 results.

順位	曲名	距離
1	A	0.3
2	D	0.9
3	B	1.0
4	C	1.2

て出力している．この状態から，各検索キーによる検索結果を独立に評価して最終結果を作成する．まず各検索結果を楽曲単位にまとめ，式 (1) より，ハミングと楽曲との距離を算出する．次に楽曲との距離でソートして，得られた曲名リストを検索結果とする．すなわち表 3 の結果からは，最終的にはハミングされたのは曲 A の可能性が最も高いとする曲名リスト（**表 4**）を得る．この方式の有効性は 6.1.2 項で定量的に評価する．

5.3 音高推移部分特徴ベクトル

4.1.3 項で述べた問題を解決するために，音高推移特徴ベクトル[15]を改良した，「音高推移部分特徴ベクトル」を考案する．この新しい特徴ベクトルは，ベクトルの先頭を部分音楽片の中で，あるルールに従って自由に決められるという特徴を持ち，音高推移特徴ベクトル[15]の一部から構成される．拍粒度が r，ウィンド長が w 拍のとき，先頭から適切な s 拍区間内の，あるルールに従って選択した次元（たとえば最も高い音が最初に出現する次元）を特徴ベクトルの先頭（起点）とし，全長 τ 次元の特徴ベクトルとなる（$\tau = (w - s) \times r + 1$）．なお，区別のために，以下では「音高推移特徴ベクトル[15]」を「音高推移全体特徴ベクトル」と表記する．

たとえば**図 13** の旋律を考える．$r = 2$ である．$w = 8$，$s = 4$ とし，基準音を 0（C-1）とした場合，このメロディから TT1 $= (60, 60, \ldots, 60, 60)$，TT2 $= (64, 64, \ldots, 59, 59)$ という 2 つの 16 次元の音高推移全体特徴ベクトルを作成できる．一方，たとえば各部分音楽片の最初の 4 拍の中で，最も高い音が最初に出現する次元をベクトルの先頭とする（他のルールについては 6.1.3 項を参照）音高推移部分特徴ベクトルを作成するならば，最初の部分音楽片では E4 の出る次元

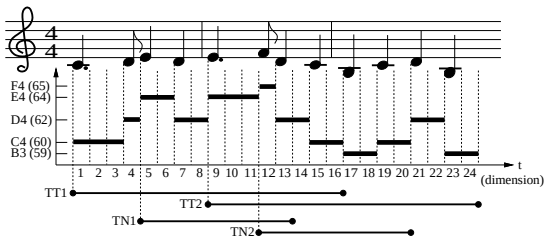

図 13 音高推移全体特徴ベクトル（TT）と音高推移部分特徴ベクトル（TN）
Fig. 13 Entire tone transition feature vector (TT) and partial tone transition feature vector (TN).

がベクトルの先頭になり，TN1 $= (64, 64, \ldots, 65, 62)$ という 9（$= (8 - 4) \times 2 + 1$）次元の音高推移部分特徴ベクトルを作成できる．また 2 番目の部分音楽片に関しては F4 の出る次元がベクトルの先頭となり，TN2 $= (65, 62, \ldots, 60, 60)$ という 9 次元のベクトルを作成できる．

同様に図 11 のメロディからこの方式で特徴ベクトルを作成すると，4 番目の部分音楽片までは，ベクトルの先頭が部分音楽片の先頭に一致した特徴ベクトルが作成されるが，5 番目にはパート C（$= 4$ 拍目）を先頭とする，部分音楽片の先頭とベクトルの先頭が一致しない特徴ベクトルを作成できる．したがってユーザがこの 5 小節目を含んでサビの部分をハミングした場合，部分音楽片から作られた特徴ベクトルの中に，部分ハミング片から作られた特徴ベクトルの 1 つと先頭が一致するものがあることになり，適切なマッチングを行うことができる．この方式の有効性は 6.1.3 項で定量的に評価する．

5.4 重複した部分音楽片の排除

図 12 から楽曲内には約 30%から 60%の重複した部分音楽片が存在することが分かったので，同じ特徴ベクトルを生成する部分音楽片はデータベースに登録しないことで，データベースサイズを縮小する．具体的には，すべての特徴量に対して等しい特徴ベクトルを生成する部分音楽片どうしは，楽曲中で最初に出現する部分音楽片のみをデータベースに登録するという方式をとる．この方式の有効性は 6.2 節と 6.3 節で定量的に評価する．

5.5 システム構成とデータベースの構築/検索処理

本節では，**図 14** を用いて新しい SoundCompass の処理の流れについて説明する．文献 15) のシステム（図 1 参照）に加えた新しい処理を太字で示す．図 14 の左側はデータベースの構築処理を示し，図 14 の右側は検索処理を示している．なお，システム構成は SoundCompass[15]（図 3）と同じである．

図 14 新しい SoundCompass の処理の流れ
Fig. 14 Processes for new SoundCompass.

5.5.1 データベースの構築処理

最初に，抜き出したメロディ・データに対してテンポの解析を行い，最大と最小のテンポを検出する．指定した閾値を超える/下回るテンポを持つ楽曲については，元の楽曲の倍/半分のテンポを持つ楽曲を複製する（5.1 節）．次にその複製楽曲も含めてすべての楽曲から部分音楽片を作成する．各部分音楽片からは，音高推移全体特徴ベクトル，音高差分布特徴ベクトル，音高推移部分特徴ベクトル（5.3 節）を作成する．重複した部分音楽片を排除（5.4 節）してから，各特徴ベクトルは特徴量ごとに多次元空間インデクスに格納し，データベースが完成する．

5.5.2 検索処理

マイクから入力されたハミングは，MIDI に変換した後，部分ハミング片を作成し，データベースと同種の特徴量の特徴ベクトルを作成する（5.3 節）．検索では，ハミングから作られた特徴ベクトルと距離の近いベクトルを，データベース内の各多次元空間インデクスから探し出す．本論文では,「音高推移全体特徴ベクトルによる距離と音高差分布特徴ベクトルの距離による合算値」と,「音高推移部分特徴ベクトルによる距離と音高差分布特徴ベクトルによる距離の合算値」のより小さい方を，ハミングとその楽曲の最終的な距離とする．ウィンド長より長いハミングを収録した場合は，複数の部分ハミング片を作成し，部分ハミング片間 OR 方式で検索結果を統合する（5.2 節）．統合した検索結果は，ハミングしたメロディに似ている部分を持つ曲名の，距離順のランキングリストとして出力する．

6. 実験結果と考察

本章では 5 章で提案した各手法の有効性を定量的に評価する．本評価実験に用いるハミングは，4.1 節の分析で使用した 240 件のハミング・データで，楽曲データベースに登録されているのは 4.2 節で調査した MIDI 形式の 21,804 曲である．

6.1 各新技術の効果と検索精度の改善

6.1.1 半テンポ/倍テンポ曲の重複登録の効果

4.1.1 項で述べたように半テンポミス/倍テンポミスは 240 件中 11 件あった．それらに対し 5.1 節のような解決策を施すことにより，11 件のうち 4 件を 5 位以内に検索することができるようになった．さらに 5.2 節と 5.3 節の提案手法も合わせて使用すると，11 件中 7 件を 5 位以内に検索することができるようになった．

残りの 4 件については，1 件は 6 位に検索された．2 件はベクトルの先頭の不一致も併発しており，音高推移部分特徴ベクトル方式を導入してもベクトルの先頭は一致せず，検索できなかった．残りの 1 件は，ハミングの後半の音程が不安定で検索できなかった．

6.1.2 部分ハミング片間 OR 検索の効果

検索に使用する特徴量にも依存するが，文献 15) で検索精度を評価するために使用した特徴量☆を用いた場合は，5 位以内の検索精度は 7.9%向上した．

改良後の SoundCompass においては，5 位以内の検索精度は 84.2%となり，部分ハミング片間 OR 検索方式を用いない場合に比べて 11.7%向上した．

6.1.3 音高推移部分特徴ベクトルの効果

音高推移部分特徴ベクトルは，起点の選択に関して複数のルールが考えられる．本論文では，以下の 3 種のルールによる音高推移部分特徴ベクトルについて検討し評価する．

ルール A 最も高い音の音符が最初に出現する次元を起点とする．

ルール B 最も低い音の音符が最初に出現する次元を起点とする．

ルール C 最も長く継続する音符が最初に出現する次元を起点とする．

表 5 に 240 件のハミングデータについて，音高推移部分特徴ベクトルの導入によって，部分音楽片から作られる特徴ベクトルと部分ハミング片から作られる特徴ベクトルの先頭が一致したものと，一致しなかったもの（不一致）の件数を，それぞれのルールに関して調べた結果を示す．表 5 から先頭が一致する件数は

☆ 音高推移全体特徴ベクトルと音高差分布特徴ベクトルの組合せ．

表 5 各起点の，先頭が一致した特徴ベクトルの数と一致しなかった特徴ベクトルの数
Table 5 Number of vectors whose beginnings are mutually correspondent to those made from humming according to each start point.

ルール	一致	不一致
ルール A	219	21
ルール B	207	33
ルール C	206	34

表 6 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，先頭の一致/不一致の関係
Table 6 Correspondence/in-correspondence of the beginnings of feature vectors for entire/partial tone transition feature vector.

		全体特徴ベクトル	
		一致	不一致
部分特徴ベクトル	一致	160	59
	不一致	14	7

ルール A が最も多いことが分かる．これはおそらく，高い音は低い音や長い音よりも意識的に発声する人が多いので，人の歌唱から特徴を抽出する際に，基準や目印とするのに優れているためであると考えられる．

次に，音高推移部分特徴ベクトルと音高推移全体特徴ベクトルの関係を調べた結果を**表 6** に示す．音高推移部分特徴ベクトルには，ルール A によるものを使用した．表 6 から，音高推移全体特徴ベクトルでは部分音楽片と部分ハミング片の先頭が一致していたのに，音高推移部分特徴ベクトルでは不一致になったものが 14 件あることが分かる．したがって，検索に使用する場合はどちらかの特徴ベクトルだけを使用するのではなく，ベクトルの先頭が一致する方の特徴ベクトルを検索に使用する方が良いといえる．しかし，ベクトルの先頭が一致する特徴ベクトルは事前には分からないので，本論文では両方の特徴ベクトルを独立に使って検索し，より短い距離を算出した方の特徴ベクトルによる検索結果を採用することとした（5.5.2 項参照）．

4.1.3 項の分析で明らかになった，部分音楽片の特徴ベクトルの先頭と部分ハミング片の特徴ベクトルの先頭が一致しなかった 48 件について調べてみると，44 件がルール A による音高推移部分特徴ベクトルによりベクトルの先頭が一致していた．表 6 の中で，元々先頭が一致していなかったのに先頭が一致したものは 59 件となっており 44 件より多い．これは分析の過程で，主に採譜ミスやテンポずれなどが原因で先頭が一致しなかったものが 16 件あることが分かり，そのうちの 15 件は本ベクトルの導入で先頭が一致したためである．この場合のテンポずれとは，ある音符を長め/短めに歌った場合を指す．

表 7 表 1 の 84 件のうち，5 位以内に検索されるようになった 57 件の内訳
Table 7 Reasons behind 57 cases in Table 1 retrieved in the fifth rank.

要因	件数	要因	件数
A	4/4	A&B	1/2
B	13/19	A&C	1/3
C	14/21	B&C	20/25
D	3/8	A&B&C	1/2

A：半テンポミス/倍テンポミス（4.1.1 項）
B：部分ハミング片の選択エラー（4.1.2 項）
C：特徴ベクトルの先頭の不一致（4.1.3 項）
D：その他

このように，部分音楽片とハミングの先頭がずれる場合だけでなく，

- 採譜ミスで，ずれが生じた場合，
- 歌い間違いで，ずれが生じた場合，
- 途中でテンポをはずして，ずれが生じた場合，

でも，音高推移部分特徴ベクトルを導入することで検索が可能になることが分かった．

6.1.4 検索精度

文献 15) で提案した技術のみを用いた場合は，本データベースに対して 5 位以内に正解を検索できなかったのは 240 件中 84 件である．これに対して，半テンポ/倍テンポ曲の重複登録，部分ハミング片間 OR 検索，音高推移部分特徴ベクトルを導入することで，上記の 84 件中 57 件を 5 位以内に検索することができるようになった．その内訳を**表 7** に示す．表内の分数の分子に相当する数字が，検索されるようになったハミングの数である．

新たに 6 位以下にランクダウンしたのは 11 件で，最終的に 5 位以内に検索されなかったのは 84 件中の 27 件と合わせて 38 件となった．このうち 100 位以内にも検索されないのは 19 件で，主な原因は，特徴ベクトルの先頭の不一致や，特徴ベクトルの基準音の不一致，またはそれらの組合せである．

最終的には，5 位以内に正解が検索される件数が 156 件から 202 件に増加し，新 SoundCompass は 84.2%の検索精度を達成し，従来の我々のシステムと比較して，19.2%の向上が可能となった．

6.2 蓄積コストの低減化

ウィンド長 16 拍，スライド幅 4 拍のスライディング・ウィンド方式で楽曲を分割すると，21,804 曲から生成される部分音楽片数は 3,241,038 となる．これは

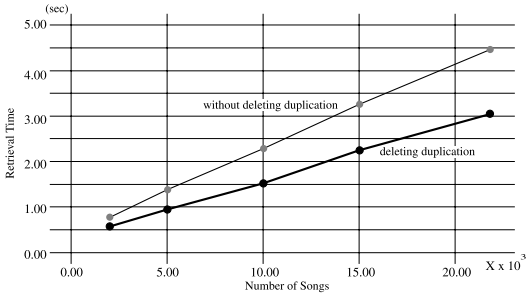

図 15 データベースを圧縮した場合の検索時間と，圧縮しなかった場合の検索時間
Fig. 15 Retrieval time evaluation where the database is/is not compacted.

5.1 節に基づいて重複登録した，半テンポ曲と倍テンポ曲の部分音楽片も含んでいる．このときの特徴ベクトルの容量は約 1.01 GB である．5.4 節で提案した方式を適用することにより，部分音楽片数は 1,858,115 になり，データベースのサイズは 57.3%に削減することができた．また本方式で圧縮したデータベースを用いても，検索精度には影響はなかった．

6.3 検索速度の改善

図 15 は，21,804 曲から上位 25 曲を検索した場合の検索時間を示している．実験には，Sun Ultra80（UltraSPARC-II 450 MHz×2，主記憶 4 GB）1 台を使用した．太い実線は 5.4 節で提案した方式でデータベースを圧縮した場合の検索時間を，細い実線は圧縮しない場合の検索時間を示している．横軸はデータベースに格納された楽曲の数，縦軸は検索時間を表している．

重複した特徴ベクトルを排除することによって，検索速度についても改善することができた．これは以下の 2 つの理由による．第 1 に，データベースサイズが削減されたことにより，データのアクセス数と計算時間が短縮された点である．そして第 2 に，同じ曲数を検索する場合でも本方式では重複した特徴量ベクトルがデータベースから排除されたために，検索しなければならない部分音楽片の数が少なくなった点である．

7. カラオケ選曲システムへの実用化のために

SoundCompass は，実際にカラオケ選曲システムの一部として，東京都内のカラオケボックスに導入されている[17),18)]．本章ではそれについて紹介する．

我々は，カラオケボックスでの選曲システム用に **図 16**，**図 17**，**図 18** のようなスクリーンイメージを持つグラフィカル・ユーザ・インタフェースを用意した．図 16 は最初の画面である．右はじの人形は 8 分音符の形の頭を持った SoundCompass のキャラクタ「音符ちゃん」で，ユーザに機器の操作をアドバイスしてハミング検索を誘導する．この画面では，左右に体を動かして，ユーザに “タ” を用いてテンポに合わせて歌うようにアドバイスしている．

図 16 最初の画面
Fig. 16 Opening screen.

図 17 テンポの調節用画面
Fig. 17 Tempo-adjusting screen.

図 18 検索結果の曲名リスト
Fig. 18 Retrieval result.

図 17 はハミング収録のためのテンポ調節の画面で，音符ちゃんが体を左右に動かしてメトロノームの役をしている．音符ちゃんの動きの速さは音符ちゃんの下の方向バーで自由に調節することができる．音符ちゃんは，ここではメトロノームの速さを自分の歌いやす

図 19 受話器付きタッチパネル
Fig. 19 Touch-panel with a receiver.

い速さに調節してから録音ボタンを押してハミングするようにユーザにアドバイスしている．

図 18 は検索結果表示画面を示している．ハミング収録後，検索実行のボタンを押すと数秒でこの画面を出力する．この画面では曲のタイトルのほかに，その曲がヒットした順位とその点数（類似度を 0〜100 点の点数に変換したもの）も表示される．たとえば 2 行目の「Love Train」はユーザのハミングに 2 番目に近い部分を持つ曲として表示されており，ユーザのハミングが 100 点満点表示で 80 点程度の点数で近かったということを表している．

図 19 はユーザインタフェース機器で，受話器付きのタッチパネルである．ハミング収録時のテンポの速さは，図 17 の音符ちゃんの左右の動きの速さでも分かるが，この受話器からメトロノームの音を聞いて確認することもできる．画面は検索結果を表示しており，一番下の右端のボタンを押して検索結果の曲をカラオケシステムに予約することができる．

本システムの使用感について複数回行ったアンケートでは，全般的にゲーム感覚で利用するユーザが多く，楽しんで使ってくれたようだ．ユーザの大多数がハミング検索を初めて使ったようであるが，一度使い方を覚えてしまえば，このシステムは直感的で受け入れやすいものだったようである．ほとんどが「また使いたい」,「検索が早くて驚いた」など好意的な意見だった．

8. ま と め

本論文では我々が研究開発中のハミング検索システム（SoundCompass）を実用化するために考案した技術について述べ，それらの有効性を定量的に評価した．現在，データベースは 21,804 曲を保持しており，データベース・サーバはその中から約 3.1 秒で検索結果を出力する．検索精度は，人が聞いて曲の一部であると判断できるハミングの約 84%について，正しい曲名を 5 位以内に出力することができる．もちろん正しい曲名だけではなく，ハミングした部分を含んだ曲を一部に含むメドレーも検出することができる．

我々は特に，何度歌い直しても決して検索されないケースを明らかにし，それらを解決することを最重要課題として，音高推移部分特徴ベクトル，半テンポ/倍テンポのメロディ・データの複製とデータベースへの重複登録を提案しその有効性を示した．また，長いハミングについて，得られたハミング・データを有効に検索に活用するために，部分ハミング片間 OR 検索を提案した．さらにデータベースのサイズを縮小する方法も提案し，その効果について述べた．最後に実用化の一例として，カラオケボックスの選曲システムへの適用について紹介した．

参 考 文 献

1) Ghias, A., Logan, J. and Chamberlin, D.: Query By Humming, *Proc. ACM Multimedia 95*, pp.231–236 (1995).
2) 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).
3) Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D. and Yanker, P.: Query by Image and Video Content: The QBIC System, *IEEE Computer*, Vol.28, No.9, pp.23–32 (1995).
4) Kushima, K., Satoh, M., Akama, H. and Yamamuro, M.: Integrating Hierarchical Classification and Content-based Image Retrieval — ImageCompass, *Proc. Conference on Intelligent Information Processing*, pp.179–187 (2000).
5) Curtis, K., Taniguchi, N., Nakagawa, J. and Yamamuro, M.: A comprehensive image similarity retrieval system that utilizes multiple feature vectors in high dimensional space, *Proc. International Conference on Information, Communication and Signal Processing*, pp.180–184 (1997).
6) 歌ってみたかったあの歌タッタタッタタタ — 一発選曲カラオケ店登場，朝日新聞 2000.12.7 夕刊.
7) 蔭山哲也，高島洋典：ハミング歌唱を手掛かりとするメロディ検索，電子情報通信学会論文誌，Vol.J77-D-II, No.8, pp.1543–1551 (1994).
8) 園田智也，後藤真孝，村岡洋一：WWW 上での歌声による曲検索システム，信学技報，p.25–32，電子情報通信学会 (1998).
9) Kosugi, N., Nishihara, Y., Kon'ya, S., Yamamuro, M. and Kushima, K.: Let's Search for

Songs by Humming!, *Proc. ACM Multimedia 99 (Part 2)*, p.194 (1999).
10) Lemström, K. and Perttu, S.: SEMEX — An Efficient Music Retrieval Prototype, *International Symposium on Music Information Retrieval (MUSIC IR 2000)* (2000).
11) 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, *3rd International Conference on Music Information Retrieval* (2001).
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, *3rd International Conference on Music Information Retrieval* (2001).
13) McNab, R.: Interactive Applications of Music Transcription, Master's thesis, Computer Science at the University of Waikato (1996).
14) 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).
15) 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).
16) Foote, J.: Visualizing Music and Audio using Self-Similarity, *Proc. ACM Multimedia 99*, pp.77–80 (1999).
17) メロディを口ずさんで選曲，日本経済新聞 2001.1.19.
18) 音声認識で一発選曲 — 進化するカラオケ，朝日新聞 2001.11.26 科学面「技あり」. http://www.asahi.com/science/waza/011126.html.

(平成 14 年 9 月 9 日受付)
(平成 15 年 10 月 16 日採録)

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．

山室　雅司

1985 年早稲田大学理工学部数学科卒業．1987 年同大学院数学専攻修士課程修了．1990 年コロンビア大学大学院電気工学専攻修士課程修了．1999 年博士（工学，早稲田大学）．1987 年日本電信電話株式会社入社．以来，ネットワークオペレーション情報モデル化・ビジュアル化，データベース設計法，マルチメディア情報検索の研究に従事．現在，デジタル情報流通の研究に従事．1994 年電子情報通信学会学術奨励賞．電子情報通信学会，日本ソフトウェア学会，IEEE-CS 各会員．

串間　和彦（正会員）

1980 年京都大学工学部電子工学科卒業．2001 年博士（情報学，京都大学）．1980 年日本電信電話公社（現 NTT）入社．以来，知識ベースシステムの研究，大規模クライアントサーバシステムの実用化，マルチメディアデータベースの研究等を経て，現在はモバイルマルチメディアの研究開発に従事．