データベースシステム 119－9
（1999．7．21）

ハミングを用いた音楽検索システム

小杉尚子　西原祐一　紺谷精一　山室雅司　串間和彦
nao@isl.ntt.co.jp

NTT サイバースペース研究所
〒230-0847 横須賀市光の丘 1-1

本稿では、メロディを口ずさんで音楽を検索するためのシステムについて述べる。ユーザは、探したい曲のメロディの一部が分かっていれば、このシステムを用いてその曲を検索することができる。本システムでは、すべての曲を一定の拍数を含む音楽片に分割し、音楽片単位でマッチングを行うので、ユーザは曲のどの部分をハミングしても検索できる。各音楽片からは複数の特徴量を抽出する。これら特徴量をヒストグラムに集計し、それを多次元ベクトルとみなす。複数の特徴ベクトルに基づく類似検索は、柔軟で正確な検索を実現する。また、多次元インデクスは大規模データベースに対して、高速な検索を提供する。本稿では、上記システムにおける概要を述べるとともに、実際の被験者のハミングによる音楽検索実験の結果も報告する。

Music Retrieval System using a Hummed Tune as a Query

Naoko Kosugi, Yuichi Nishihara, Seiichi Kon'ya, Masashi Yamamuro, and Kazuhiko Kushima
NTT Cyberspace Labs.
1-1 Hikarinooka, Yokosuka-shi, Kanagawa 239-0847, Japan

This paper describes a music retrieval system which uses a hummed tune as a query to a database. This system enables a user to get the song's name even if he/she remembers only a part of the melody of the song.
All songs in the database are split into subdata. This system accepts a hummed tune of any part of the melody as a query, since matching is done using the subdata. Multiple features are extracted from each subdatum. The features are represented as high dimensional vectors based on histograms. Similarity retrieval using the indices of multiple feature vectors provides flexibility and accuracy as well as speed. This paper simply provides the techniques utilized for the system. Experimental results using hummed tunes of casual singers are also reported.

1　はじめに

マルチメディア・コンテンツは世界中で、急速な勢いで増え続けている。マルチメディア・コンテンツは、テキストや数値データの他に、画像データ、音楽データや動画データなど、さまざまな種類のデータを含む。このような、多様なメディアデータの急速な増加によって、高速な検索技術だけでなく、各メディアデータに合った検索方法の必要性が高まっている。

各メディアデータに合った検索方法の１つとして、**内容検索**が挙げられる。我々は内容検索が可能な高速画像検索システム *ExSight* を研究開発してきた [12, 13]。*ExSight* は全体画像による検索だけでなく、部分画像による検索にも対応している。例えばりんごの写真や絵を検索キーとして与えて、りんごの写っている画像を検索することができる。

また、**類似検索**もマルチメディア・データベースには欠かせない技術である。なぜなら、たとえばりんごの写っている画像を探す場合、検索キーとして与えたりんごの絵や写真とまったく同じりんごが映っている画像を探したいという要求はめったにない。ほとんどが、単にりんごが写っている画像が欲しいという要求であり、このような場合は「検索キーとして与えたりんごの画像に似ている画像を探す」といった、類似検索技術が必要である。我々は、多次元ベクトル間の距離計算に基づく、高速類似検索エンジンを研究開発してきた [6, 9]。

内容検索の新たなメディアへの対応として、我々は音楽データベースに関する研究開発を進めている。現在、検索キーとしてハミングや口ずさんだメロディなどを使って、曲名や歌手を検索することができる [5, 14, 2]。

この検索システムは、*ExSight* と同じ高速類似検索エンジンを使用している。

2 関連研究

ハミングによる音楽検索システムの研究は、日本ではかなり盛んでたくさんの研究成果が報告されている。それらは以下のような点で分類できる。

(1) ハミングする部分
- 曲の先頭やさびのみ [10]
- 自由 [1, 4, 11, 14]

(2) 検索に用いる情報
- 隣接音との相対音高差 [1, 4, 10, 11, 14]
- 初音との相対音高差 [14]
- 絶対音程 [4, 14]
- 相対音長差 [1, 10, 11]
- 隣接 3 音の音長・音高パターン [15]

(3) マッチング方法
- 文字列マッチング [1, 4, 10, 11]
- インデクスを用いたベクトル計算 [14, 15]

(4) データベースの規模 [1]
- 200 曲以下 [1, 11, 15]
- 200 ～ 1000 曲 [10]
- 1000 曲以上 [4, 14]

(5) データの種類
- 人の歌声 [11]
- MIDI 形式の音楽 [1, 4, 14]

(6) データとして持っている曲の部分
- 全曲 [4, 14]
- 曲の一部 (さびなど)[10, 15]

(7) マッチングに使うデータの形態
- 音程などを表す文字列 [1, 4, 10, 11]
- ビット列 [4]
- ベクトル [14, 15]

音楽は、音符の並びなどで表現され、個々の音符は長さと音程の 2 つの情報を持っている。したがって、例えば音程だけに注目して、個々の音符の音程情報をある種の文字列で表現することができる。この場合、文字列どうしを比較することによって、ある曲と別の曲がどのくらい似ているかを調べることができる。一方、たいていの人間は記憶を頼りに歌う場合、ぴったり楽譜通りに歌うことはできない。

上記に基づいて、人間が歌った曲と同じ曲を、その歌ったものをキーとして検索するシステムを構築する場合、音符の並びを文字列とみなすことで、エラーを許した文字列マッチング[2]を適用するのが従来の最も一般的なアプローチであった [1, 4, 10, 11]。その際、入力パターンに音符の**挿入**, **削除**, **置換**, **分割**, 結合といったエラーが存在することを仮定して、それらに柔軟に対応しなければならない [3]。

入力キーのエラーを考慮した文字列マッチング問題とは、ある文字列の中から、ある類似度の基準の下で、入力パターンに類似していると判断される部分をすべて見つけ出すことである。入力キーのエラーを考慮した文字列マッチングにおいて、もっとも良く知られた方法は、ダイナミック・プログラミング – 以下 DP マッチングと呼ぶ – である。もっとも良く知られた「類似度」は、編集距離[3]とよばれるもので、ある文字を別の文字に変換する場合のコストを指す。それぞれのコストは自由に決めることができるので、編集距離を使った類似度判定は非常に柔軟である。すなわち、ハミングを入力キーとする音楽検索のように、曖昧なキーを使い、かつ類似度の算出に自由度が必要な音楽検索システムには有効な手法である。しかし、DP マッチングでは、検索コストを音楽データ量に対して線形以下にすることはできないので、大規模なデータベースには不向きである。

一方、[3] の筆者は、入力キーのエラーを考慮した高速なテキスト検索方式の 1 つであるステート・マッ

[1] 実際にはデータベースの規模を登録されている曲数だけで単純に比較することはできない。検索の際は 1 曲あたりのデータ量 (音符数や長さなど) が大きな要素になるからである。

[2] approximate string matching

[3] edit distance

チング [8] を音楽検索に適用することを試みている。ステート・マッチングでは簡単なビット演算しか行われないので、大規模なデータベースでも高速に検索できる。しかし編集距離のように、各々のエラーに対してエラーのコストを自由に設定することができないので、適切な類似度を計算するのは難しい [3]。

3 音楽データベース

我々の音楽検索システムでは、音楽データベースにMIDI形式の曲を使用している。MIDIでは、PCMのように音の信号そのものではなく、音楽の演奏情報が記述されている。すなわち音符を「演奏時間」、「チャネル」、「音程」、「音量」の4つの数で表して曲を表現する。チャネルには、例えば1つの演奏パートが割り当てられており、複数の演奏パートをもつ曲では、複数のチャネルが使用されることもある。また多くのカラオケデータには、メロディ情報は1つのチャネルに入っている。このようにMIDIを使うと、ある特定のチャネルの音楽情報を簡単に抜き出すことができる。

データベースを作るために、まず音楽MIDIファイルからメロディのデータを抜き出す。ほとんどの人がメロディで曲を認識・記憶しているので、現段階ではメロディ・データのみに対してマッチングを行っている。次に、メロディ・データを、重なりをもたせた小さな音楽片に切り出す。以下「**スライディング・ウィンドウ方式**による分割」と呼ぶ。マッチングはこれらの音楽片を使って行うので、ユーザは曲を検索する際に、曲のどの部分をハミングしても構わない。1つの音楽片からは、6章で定義する複数の特徴量を抽出し、特徴ベクトルとする。特徴ベクトルは特徴量ごとに個別のインデクスを作成する。

4 検索システム

ユーザにはマイクを通してハミングしてもらい、それをMIDI形式に変換する。ハミングは、市販の作曲ソフトウェア [7] を用いてMIDIファイルに変換し保存する。次に、特徴ベクトルを生成するところまで3章で述べたデータベースの作成と同じようにハミングデータを処理する。すなわち、まずハミングからデータベース中のメロディ・データと同じ大きさと同じずらし幅でハミング片を切り出す。次に各ハミング片から6章で定義する特徴量を抽出し多次元特徴ベクトルに変換する。

検索は、まずハミング片から作られた各特徴ベクトルに対して、各特徴量ごとに類似度を計算する。次に、各特徴量における類似度を重み付線形和によって統合し、最終的な類似度とする。この音楽片ごとの統合類似度に基づいて、上位の音楽片を含む曲を検索結果とする。したがって、ユーザはハミングしたメロディを含む曲を探し出すことができる。この検索においては、多次元特徴ベクトルのための高速類似検索エンジン [9] が使われている。

図1は、3章と4章で述べた一連の処理を簡単に図で示している。右側がデータベースの作成、左側上がハミングの処理、左側下がそれらを用いた検索の様子を表している。

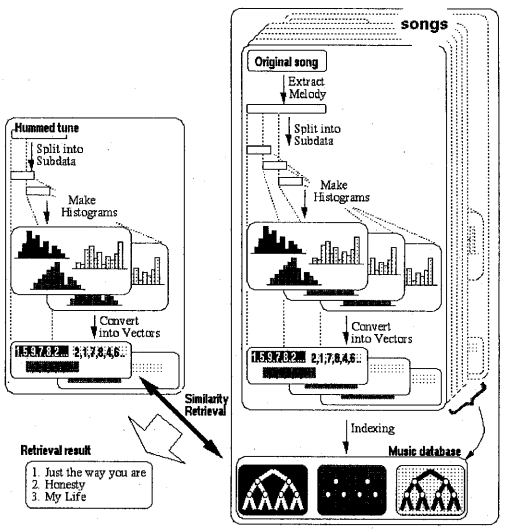

図 1: データベースの作成、ハミング・データの処理、曲の類似検索のイメージ図

5 時間正規化

本研究では、音楽片とハミング片を比較して類似検索を行っているので、音楽片とハミング片が何らかの単位で同じ量の情報を持ち合わせていることが前提となる。単位としては、たとえば音符数や実時間などが考えられる。しかし、2章でも述べたように、ハミングがもつあいまい性の問題で音符数を単位とするのは適切ではない。また、ハミングの速さは人によって千差万別なので、実時間も単位とするのは不適切である。

そこで、我々は「拍数」を単位とした。なぜなら、同じ曲をゆっくり演奏しても速く演奏しても、拍子は普遍的な単位だからである。そこで、同じ拍数を含むように、音楽片やハミング片を切り出す。我々はこれを**時間正規化**と呼ぶ。

音楽データの時間正規化の処理は容易である。なぜなら音楽データはMIDIを使用しているため、音符の長さは拍子に対する相対的な長さで記述されており、あらかじめ時間正規化されていると考えて良い。一方、本システムでは、ユーザにメトロノームに合わせてハミングしてもらう。ハミング時にセットしたメトロノームのテンポに基づいて、拍子を単位に処理すれば時間正規化されたことと等価である。

6 特徴ベクトル

我々は、特徴ベクトルをヒストグラムを基にして作ることを提案する。ほとんどの人にとって、たとえメトロノームを使ったとしても一定のテンポや音程で歌い続けることは難しいので、生成する特徴ベクトルにはハミングの不正確さに対処できることが要求される。

本節では、本システムで用いた、上記の要求を満足するようなヒストグラムの作り方と、それらヒストグラムを特徴ベクトルに変換する方法について述べる。これらの特徴ベクトルのもつ利点については7章で述べる。

ヒストグラムの横軸は音程又は音程差とした。音程の場合は0～127の128の整数の区分であり、音程差の場合は-127～127の255の整数の区分である。MIDI形式では音を128の整数で表現するからである。次に、縦軸は拍数又は回数とした。たとえば、1つの音符の出現に対して、拍数の場合、2拍の音は2と記録されるが、回数の場合は1と記録される。

抽出する特徴量は以下の3種である。

音程 各音符の音程そのもの
前音との差分 連続する2音間の音程差
初音との差分 音楽片の最初の音と各音との音程差

上記の特徴量について、ハミングの音程の揺らぎを許容するためにファジー化したヒストグラムを作る。ファジー化するとは、本当の音程(true tone)に対して、その音の両側n半音の範囲に、重み(w)つきの余分な累積(extra accumulation)を施すことである(図 2参照)。具体的には、本当の音程ド(C4)が**2**拍記録されるとき、nが**1**でwが**0.5**の場合、ドの両側**1**半音のド$^{\#}$(C#4)とシ(B3)の区分にも、1 (=2×0.5)が加算される。

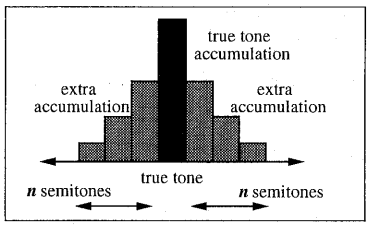

図 2: ファジー・ヒストグラム概念図

ヒストグラムをベクトルに変換するにあたって、マッチングを高速化するために、ヒストグラムの区分数を減らす。これは特徴ベクトルの次元の縮小に相当する。図3と図4は、ヒストグラムを(*left*+1+*right*)区分に減らす様子を示している。音程特徴量ヒストグラムでは、ハミングとデータベース中の曲の調の違いを吸収するために、平均を中心とした正規化を行う。次に、ヒストグラムの区分数を減らすために、平均音$\bar{N}$を中心に、一定の値 *left, right* をそれぞれ左側、右側の区分の範囲とする。*left, right* よりも外の値は、無視するかマージするかのどちらかで処理する。結果的には(*left*+1+*right*)次元の特徴ベクトルができあがる。

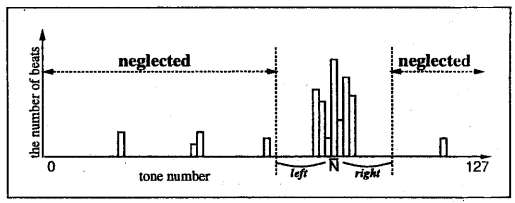

図 3: ヒストグラムの区分数を減らして、ベクトルの次元を縮小することを表すイメージ図。ベクトルの次元は、平均の音$\bar{N}$を中心に"*left*+1+*right*"になる。*left*、*right*区分に入らなかった値は無視する。

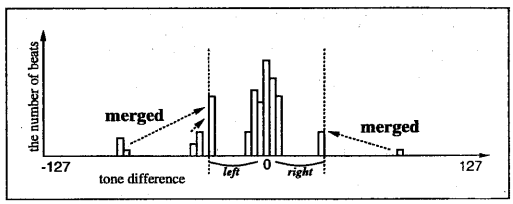

図 4: ヒストグラムの区分数を減らして、ベクトルの次元を縮小することを表すイメージ図。ベクトルの次元は"*left*+1+*right*"になる。*left*、*right*に入らなかった値は、範囲内の一番外側の区分にマージする。

特徴ベクトルは、各特徴量ごとにインデキシングされる。類似度は各特徴量インデクスからの検索結果の重み付き線形和で表される。この重みは自由に調節することができるので、自由度の高い類似検索を実現することができる。

7 ヒストグラムを使うことの利点

ヒストグラムを基に作られたベクトルによるマッチングは、2章でのべたような、音楽検索における問題に柔軟に対応することができる。

ヒストグラムは、どんな音が何拍出ているのかというような、全体の傾向をつかむことを可能にする。また、音符の分割・統合・挿入・削除はヒストグラムには部分的にしか影響しない。さらに拍数ベースのヒストグラムなら、全体の拍数の中でつじつまがあっていれば良いので、音符の分割・挿入の影響は無視できる。音符の置換は、ヒストグラムのファジー化で対処できる。どの程度の置換まで許容するかは、ヒストグラムをファジー化する際の範囲と重みで自由に調節できる。

我々の検索システムはテンポの違いや、テンポの狂いにも対応できる。従来の文字列マッチングによる検索では、文字列といった場合、音程を文字に置き換えたものが多く、テンポ情報またはリズム情報はあまり使用されなかったが、本方式では拍数を基準にハミング片の切出しやずらし幅を決めているので、テンポ情報はヒストグラムを作る過程で大きく影響を与えている。実際には、時間正規化によってハミングのテンポは、音楽データベース内の曲の基本テンポに補正される。メトロノームに合わせてハミングしてもらうので、補正処理は簡単である。したがって、根本的なテンポの差は問題にならない。テンポの揺らぎは、回数ベースのヒストグラムによって吸収される。個々の音符の長さに不確定さがあっても、音符の出現回数がほぼ正しければ、ヒストグラム上の影響は少ないからである。

8 実験結果

本章では実際のハミングを用いた音楽検索実験の結果について述べる。

まず実際のハミングを用いた音楽検索実験について述べる。音楽データベースには、現在1200曲を登録してある。ほとんどは日本の曲で、ヒットポップスを中心に演歌やアニメソングなどが含まれている。

この実験では、音楽片、ハミング片を作るためのウィンドウサイズとずらし幅は、それぞれ20拍と4拍とした。1曲あたりの平均音楽片数は91となった。ヒストグラムは、1半音の幅と0.5の重みでファジー化した。被験者にはテンポを一定に保ってハミングしてもらうために、メトロノームに合わせて歌ってもらった。ただし、メトロノームの速さは適宜調節した。

実験では男性14人、女性4人の合わせて18人にハミングしてもらい、77のハミング・データを得た。1曲あたりのハミングの平均の長さは28.5拍で、ハミング片は平均で3.12個できた。本実験ではウィンドウサイズを20拍としているので、20拍以上ハミングすると、複数のハミング片が生成される。表1は実際のハミングを使っての音楽検索実験において、以下のcase1) からcase3) のそれぞれの場合で、正解曲が1位でマッチしたハミング数と3位以内にマッチしたハミング数を示している。

case1) 検索キーとして、中央のハミング片のみを使う。各特徴量における類似度を統合する際の線形和の重みは同一とする。

case2) ハミング片の選択が検索精度に与える影響を調べるために、各ハミングのそれぞれのハミング片を1つずつ検索キーにしてマッチングを行い、最高位で正解にマッチしたハミング片を、そのハミングの検索キーとする。各特徴量における類似度を統合する際の線形和の重みは同一とする。

case3) case2) と同様、検索キーとして、中央のハミング片だけではなく、他のハミング片も使用する。さらに各特徴量における類似度を重み付で線形和する際の各重みはマッチングの結果が良くなるようにその都度調整する。

表1: ハミングを使っての音楽検索実験結果

	ハミングの数	1位でマッチしたハミング数	3位以内でマッチしたハミング数
case 1	77	32 (42%)	38 (49%)
case 2	77	47 (61%)	53 (69%)
case 3	77	64 (83%)	66 (86%)

最初の2つ (case1 と case2) の実験結果からほとんどの人が歌いはじめと歌い終りよりは、歌っている最中の方が上手にハミングできるということが分かる

(32/47)。しかし一方で、歌いはじめの方が上手に歌える人や、歌い続ければ歌い続けるほど調子が良くなる人がいるのも事実(15/47)なので、ハミングのどの部分を検索キーとして切り出すかは今後の検討課題である。また、case3)の実験結果は、各特徴量はそれぞれ特性を持っているので、それぞれの特徴量における類似度を線形和するにあたって、それぞれの特性を考慮した重みを付加したほうが有効であるということを示唆している。各特徴量をどのような重みで評価するかは今後の検討課題である。

9 まとめ

本稿ではハミングを検索キーとして使用する、音楽検索システムについて述べた。我々は、

- 音楽データをスライディング・ウィンドウ方式で、重なりをもたせた音楽片に分割した。これはユーザが好きな場所をハミングして検索することを可能にした。
- 音程や音長などの音楽情報をヒストグラム化し、それを特徴ベクトルとした類似検索を行うことで、ハミングのテンポのゆらぎや音符の挿入・削除・分割・統合に対処した。
- ファジーヒストグラムを作ることで、音符の置換の原因となるハミングの音程のゆらぎに対処した。
- ヒストグラムを多次元特徴ベクトルに変換することで、データのインデキシングを可能にした。結果として、大規模データベースに対して効率の良い検索を実現した。
- 各特徴量における類似度を重み付き線形和で統合する際の、各重みを調整することで、検索精度を向上させることができた。

今後の課題としては、まず8章でも言及した通り、ハミングから最適な検索キーを選択する方法や、特徴量インデクスごとの検索結果をより柔軟に統合するための方法などを検討する必要がある。さらに、

- データベースの規模の拡大
- 検索精度をより向上させるための、新しい特徴量の追加
- 音楽片やハミング片を作る際のウィンドウやずらし幅の調整

などにも取り組んでいく必要がある。

参考文献

[1] Asif Ghias, Jonathan Logan, and David Chamberlin. Query By Humming. In *Proc. ACM Multimedia 95*, pp. 231–236, November 1995.

[2] Naoko Kosugi, Yuichi Nishihara, Seiichi Kon'ya, and Masashi Yamamuro. Music Retrieval by Humming. In *Inproceedings PACRIM'99*, p. to be appeared. IEEE, August 1999.

[3] Rodger McNab. INTERACTIVE APPLICATIONS OF MUSIC TRANSCRIPTION. Master's thesis, Computer Science at the University of Waikato, 1996.

[4] Roger J. McNab, Lloyd A. Smith, David Bainbridge, and Ian H. Witten. The New Zealand Digital Library MELody inDEX. http://www.dlib.org/dlib/may97/meldex/05written.html, May 1997.

[5] Yuichi Nishihara, Naoko Kosugi, Seiichi Kon'ya, and Masashi Yamamuro. Humming Query System Using Normalized Time Scale. In *Proceedings of CODAS'99*, March 1999.

[6] Makoto Onizuka and Satoshi Okada. Query Model for Structured Objects. In Tharam S. Dillon, Robert Meersman, and Zahir Tari, editors, *Short Paper Proceedings*, pp. 32–45. International Conference on Database Semantics, January 1999.

[7] WILDCAT CANYON SOFTWARE. AUTOSCORE.

[8] Sun Wu and Udi Manber. Fast Text Searching Allowing Errors. *Communications of the ACM*, Vol. 35, No. 10, pp. 83–91, October 1996.

[9] キャサリン・カーティス, 谷口展郎, 中川純一, 山室雅司. 多次元特徴ベクトルを用いた画像類似検索エンジン. データベースシステム 113-28, pp. 167–172. 情報処理学会, July 1997.

[10] 薮山哲也, 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, Aug. 1994.

[11] 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, pp. 25–32. 電子情報通信学会, Feb. 1998.

[12] 串間和彦, 赤間浩樹, 紺谷精一, 山室雅司. 色や形状等の表層的特徴量にもとづく画像内容検索技術. 情報処理学会論文誌, pp. 171–184, 1999.

[13] 串間和彦, 赤間浩樹, 紺谷精一, 木本晴夫, 山室雅司. オブジェクトに基づく高速画像検索システム:ExSight. 情報処理学会論文誌, Vol. 40, No. 2, pp. 732–741, 1999.

[14] 西原祐一, 小杉尚子, 紺谷精一, 山室雅司. 時間正規化を用いたハミング検索システム. 音楽情報科学研究会予稿集 30-6, pp. 27–32. 情報処理学会, May 1999.

[15] 柳瀬隆史, 高須淳宏, 安達淳. メロディからの特徴抽出による曲検索システム. 情報処理学会研究報告, Vol. 99, No. 39, pp. 1–6, May 1999.