4C-5

# 自己相関係数を用いたカバー曲検索への試み

津田 啓太郎　小早川 倫広　星 守　大森 匡
電気通信大学大学院情報システム学研究科

## 1　はじめに

本研究の目的は、ある楽曲の一部を問い合わせとして入力し、その曲のオリジナル曲やカバー曲を検索することの出来る、楽曲内容に基づいた検索システムの構築にある。カバー曲とは、過去に演奏し発表された楽曲（原曲）を、本人もしくは他のアーティストにより再び演奏、発表された曲のことをいう。本研究におけるカバー曲は、あまり大きなアレンジがされていない楽曲とする。このような場合でも、ボーカル（アーティスト）が変わることにより楽曲の音域が変化していたり、演奏速度の変化が生じている。カバー曲検索を実現しようとした場合、楽曲の音域（ピッチ）、演奏速度（テンポ）の変化に対しての頑強な検索は最低限必須となる。

カバー曲検索の実現にあたり、本研究では、ビットレートに依存しない検索手法 [1][2][3] や、構造抽出 [4] でも用いられている自己相関係数列に着目する。本稿では、TwinVQ 圧縮過程で算出される自己相関係数列がピッチ、テンポの変化に対してどの程度頑強な検索性能を有しているかを検証し、カバー曲検索へ適用可能かどうかを調べる。

## 2　自己相関係数列による検索の手順

図 1 に沿って検索手順を説明する。

**Step1** TwinVQ エンコーダを適用し、第 $dim$ 次の自己相関係数列 $\boldsymbol{r}^q_{dim,n}= r^q_{dim,1}, r^q_{dim,2}, \ldots, r^q_{dim,w}$ $(dim = 1, \ldots, 20)$ を得る（$w$ はフレーム数）。

**Step2** 問い合わせ部分とデータベースの楽曲とをフルスキャンしながらユークリッド距離を計算し、その距離が最も小さくなるときの距離を、その曲と問い合わせ曲との距離とする。

**Step3** Step2 で求めた距離により順位をつけ、類似した部分を持つ楽曲を順に出力する。

## 3　データ集合

本節では実験に用いているデータ集合について説明する。ピアノ伴奏曲 557 曲、カバー曲を含む一般のジャズ、ロック、ポップの楽曲 4,024 曲の計 4,581 曲を基本楽曲集合 $\mathbb{D}$ と呼ぶ。集合 $\mathbb{D}$ からランダムに 60 曲 $A_i$ $(i = 1, \ldots, 60)$ を選抜し、楽曲 $A_i$ に対してピッチ、テンポの変更を施す。

ピッチ変更とは、楽曲 $A_i$ に対し半音（半階）を 1 単位としてピッチを変更することで、半音ピッチを上げることを 1 度上げるという。$A_i$ に対してピッチを $j$ 度上げた楽曲を $A_i^{P+j}$、$j$ 度下げた楽曲を $A_i^{P-j}$ と表す。本稿では、ピッチ $j$ を $-3, -2, -1, +1, +2, +3$ のように変更した。計 $360 (= 60 \times 6)$ 曲をピッチ変更データ集合 $\mathbb{P} = \bigcup_{i,j} A_i^{P+j}$ と呼ぶ。

テンポ変更とは、楽曲 $A_i$ に対し楽曲演奏時間を伸縮することである。$A_i$ に対してテンポ $T$ を $t\%$ 縮めた楽曲を $A_i^{T+t}$、$T$ を $t\%$ 伸ばした楽曲を $A_i^{T-t}$ と表す。本稿では、テンポ $t$ は $-14, -10, -6, -4, -2, +2, +4, +6, +10, +14$ のように変更した。計 $600 (= 60 \times 10)$ 曲をテンポ変更データ集合 $\mathbb{T} = \bigcup_{i,j} A_i^{T+t}$ と呼ぶ。

**Procedure** retreival($\boldsymbol{D}$, $\boldsymbol{q}$, $k$)

**input**　$\boldsymbol{D}$: $M$ 曲からなる楽曲データベースの自己相関係数列 $\boldsymbol{r}^m_{dim}$ $(m = 1, \ldots, M\ ,\ dim = 1, 2, \ldots, 20)$ の集合
$\boldsymbol{q}$: 利用者が入力する楽曲データ
$k$: 出力する楽曲数

**output**　$L$: 楽曲リスト

// Step 1 TwinVQ エンコード //
・$\boldsymbol{q}$ に対して，TwinVQ エンコーダを適用.
・自己相関係数列 $\boldsymbol{r}^q_{dim} = r^q_{dim,1}, \ldots, r^q_{dim,w}$ を抽出
($w$ は自己相関係数列の長さ).

// Step 2 ユークリッド距離算出//
**for** $m \leftarrow 1$ **to** $M$ **do**
　**for**　$n \leftarrow 1$ **to** $N^m - w + 1$ **do**
　・$\boldsymbol{r}^q_{dim}$ と $\hat{\boldsymbol{r}}^m_{dim} = r^m_{dim,n}, \ldots, r^m_{dim,n+w-1}$ 間のユークリッド距離 $dist(\boldsymbol{r}^q_{dim}, \hat{\boldsymbol{r}}^m_{dim})$ を算出.
　・$Dist(\boldsymbol{r}^q_{dim}, \boldsymbol{r}^m_{dim})$
　$\leftarrow min_{n=1}^{n=N^m-w+1}$ ( $dist(\boldsymbol{r}^q_{dim}, \hat{\boldsymbol{r}}^m_{dim})$).

// Step 3 類似曲リストの出力//
・距離 $Dist(\boldsymbol{r}^q_{dim}, \boldsymbol{r}^m_{dim})(m = 1, \ldots, M)$ を昇順ソートし，上位 $k$ 曲の類似曲リスト $L$ を出力.

**end** procedure

図 1: 検索手順

## 4　ピッチ、テンポに対する頑強性

本節では、TwinVQ 圧縮過程で算出される自己相関係数列が、ピッチ、テンポの変化に対してどの程度頑強な検索性能を有しているかを検証する。その際、基本楽曲集合 $\mathbb{D}$ にピッチ変更データ集合 $\mathbb{P}$ とテンポ変更データ集合 $\mathbb{T}$ を加えた集合を、音楽データベースとして用いた。今回の検索実験では 1 次と 20 次の自己相関係数列と、それらを組み合わせたものを用いて実験を行った。しかし、本稿では 20 次の自己相関係数列を用いた実験結果について述べる。

### 4.1　ピッチの頑強性に対する実験

ピッチ変更データ集合 $\mathbb{P}$ 内の曲 $A_i^{P+j}$ と原曲のサビ部分を問い合わせとして与え、2 節の手順により検索を行い、20 位までの検索結果のリスト $L_i^{P+j}$ を得た。$L_i^{P+j}$ 中に $A_i^{P+d}$ $(d = -3, -2, -1, \pm 0, +1, +2, +3)$（ただし $A_i^{P\pm 0}$ は原曲を表す）が出力されていれば、表の $j$ 行の $d$ 列のセルに 1 を加える、という作業を $i = 1, \ldots, 60$（曲）に対して行い、60 曲に対する平均を算出する（表 1）。表 1 の最右列は各行における行平均を、最も下の行は各

表 1: ピッチ変化における 20 位までの検索結果（単位：%）

| | $d$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $j$ | +3 | +2 | +1 | ±0 | −1 | −2 | −3 | 平均 |
| +3 | 100 | 100 | 100 | 100 | 95.0 | 86.7 | 76.7 | 94.0 |
| +2 | 100 | 100 | 100 | 100 | 100 | 91.7 | 88.3 | 97.1 |
| +1 | 100 | 100 | 100 | 100 | 100 | 98.3 | 93.3 | 98.8 |
| ±0 | 98.3 | 100 | 100 | 100 | 100 | 100 | 98.3 | 99.5 |
| −1 | 83.3 | 95.0 | 100 | 100 | 100 | 100 | 98.3 | 96.7 |
| −2 | 75.0 | 86.7 | 96.7 | 100 | 100 | 100 | 100 | 94.0 |
| −3 | 65.0 | 80.0 | 90.0 | 96.7 | 100 | 100 | 100 | 90.2 |
| 平均 | 88.8 | 94.5 | 98.1 | 99.5 | 99.3 | 96.7 | 93.6 | 95.8 |

表 2: テンポ変化における 20 位までの検索結果（単位：%）

| | $d$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $t$ | +6 | +4 | +2 | ±0 | −2 | −4 | −6 | 平均 |
| +6 | 100 | 98.3 | 55.0 | 21.7 | 13.3 | 6.7 | 5.0 | 42.9 |
| +4 | 93.3 | 100 | 91.7 | 53.3 | 23.3 | 8.3 | 8.3 | 54.0 |
| +2 | 36.7 | 93.3 | 100 | 96.7 | 51.7 | 11.7 | 10.0 | 57.1 |
| ±0 | 8.3 | 45.0 | 93.3 | 100 | 93.3 | 50.0 | 15.0 | 57.9 |
| −2 | 6.7 | 10.0 | 40.0 | 93.3 | 100 | 95.0 | 40.0 | 55.0 |
| −4 | 3.3 | 6.7 | 13.3 | 45.0 | 86.7 | 100 | 91.7 | 49.5 |
| −6 | 3.3 | 3.3 | 6.7 | 10.0 | 41.7 | 90.0 | 100 | 36.4 |
| 平均 | 36.0 | 51.0 | 57.1 | 60.0 | 58.6 | 51.7 | 38.6 | 50.4 |

列における列平均を、右下端のセルは全平均を表す。全平均は全体の性能を示している。

例えば、$j = +3$ の行、$d = -1$ の列のセルの値は 95.0% であった。これは、ピッチを +3 度変更した楽曲データを問い合わせに入力し、検索を実行した場合、ピッチを −1 度変更した楽曲データが 20 位以内に 95.0% 出現していることを意味する。

### 4.2 テンポの頑強性に対する実験

テンポ変更データ集合 $\mathbb{T}$ 内の曲 $A_i^{T+t}$ と原曲のサビ部分を問い合わせとして与え、2 節の手順により検索を行い、20 位までの検索結果のリスト $L_i^{T+t}$ を得た。4.1 節と同様な手順で表 2 を得る。ただし、表 2 では $t = -6, \ldots, +6$ に関して示す。

例えば、$t = +6$ の行、$d = +4$ の列のセルの値は 98.6% である。これは、テンポを +6% 変更した楽曲データを問い合わせに入力し、検索を実行した場合、テンポを +4% 変更した楽曲データが 20 位以内に 98.6% 出現していることを意味する。

## 5 考察

本節では、ピッチ、テンポの変化への頑強性について表 1、表 2 を用いて考察を行う。

**ピッチに関して** まず、表 1 の最も右の列に着目する。行平均の値はすべて 90.2% 以上であり、このことは、$j$ 度ピッチ変更された楽曲を問い合わせにして検索を行うとき、平均的に 90% 以上の割合で原曲とピッチ変更楽曲が上位 20 位以内に出現することを示している。ここで、行平均の列の最大値 99.5% をとった ±0 行（原曲が問い合わせ）に着目する。$d = -2, -1, \pm 0, +1, +2$ 列の値は、すべて 100% である。このことは、20 位以内に必ず $A_i^{P+d}$ が出現することを意味する。また、±3 列で最小をとるが、その値は 98.3% と高い。このことは、20 次の自己相関係数列を問い合わせに用いた場合、データベースにピッチが異なる楽曲データが存在していても、20 位以内に検索可能であることを示している。

次に最の下の行に着目する。列平均の値はすべて 88.8% 以上である。列平均の行の最大値 99.5% をとった ±0 列に着目すると、±0 列 $j = -2, -1, \pm 0, +1, +2, +3$ 行の値は、すべて 100% である。このことは、ピッチが異なる任意の問い合わせ $A_i^{P+j}$ に対して、20 位以内に 99.5% の割合で原曲 $A_i^{P\pm 0}$ が出力されるという意味である。

全平均は 95.8% である。このことは、ピッチが異なる楽曲データが存在しているデータベースに対して、$j$ 度ピッチ変更された楽曲を問い合わせにして検索を行うとき、平均で 95.8% の割合で 20 位以内に出現することを示している。

**テンポに関して** まず、表 2 の対角に着目する。対角は問い合わせ自身が 20 位以内に出現している割合を示しているが、当然、全ての問い合わせ $A_i^{T+t}$ $(t = -6, \ldots, +6)$ において 1 位に出現した。次に各 $t$ 行において、$(t \pm 2)$ 列のセルに着目する。これらのセルは、問い合わせ $A_i^{T+t}$ に対する楽曲 $A_i^{T+t\pm 2}$ の出現率であり、どの行においても 90% 前後である。このことは、テンポ $t$ の問い合わせ楽曲に対して、テンポ $t \pm 2$ の楽曲は、高い割合で 20 位以内に検索可能であることを示している。さらに各 $t$ 行において、$(t \pm 4)$ 列のセルに着目する。これらのセルは、問い合わせ $A_i^{T+t}$ に対する楽曲 $A_i^{T+t\pm 4}$ の出現率であり、どの行においても 50% 前後の検索結果であった。さらに、問い合わせ $A_i^{T+t}$ に対する楽曲 $A_i^{T+t\pm 6}$ の出現率は 20% 前後であった。このことから、自己相関係数列を用いた検索は、テンポ ±2 以内の変更の楽曲に対して頑強であるといえる。しかし、テンポ ±6 以上の変更の楽曲に対しては、工夫が必要である。

## 6 まとめ

本稿では、カバー曲検索を実現するためには、楽曲のピッチとテンポの変化に対して頑強であることが必要であるとの考えから、自己相関係数列のピッチとテンポの変化に対する頑強性について検討した。ピッチ変更に対しては頑強であり、この手法の有効性を確認することができたが、テンポ変更に対しては改善の必要があることがわかった。今後はテンポ変更に対しより頑強な手法を開発し、実際にカバー曲に対しての検索実験を行う。なお、楽曲内容に基づいた検索については、[5][6] を参照されたい。

## 参考文献


[1] K. Onishi, M. Kobayakawa, M. Hoshi and T. Ohmori, " A Feature Independent of Bit Rate for TwinVQ Audio Retrieval ", Proc. ICME2001, 2001, pp. 293-296.

[2] M. Kobayakawa, M. Hoshi and K. Onishi, " A Method for retrieving music data with different bit rates using MPEG-4 TwinVQ Audio Compression ", Proc. ACM MM, 2005.

[3] M. Kobayakawa and M. Hoshi, " A Partial Retrieval of Music Data with Different Bit Rate using MPEG-4 TwinVQ Audio Compression ", Proc. MDDM2007, 2007 (to appear).

[4] 中西 基浩, 小早川 倫広, 星 守, 大森 匡, " 楽曲圧縮過程において算出される自己相関係数列を用いた楽曲の節長抽出と構造分析 ", 情報処理学会 研究報告 2005-MUS-59-4, Vol. 2005, No. 14, pp. 19-26, (2005) .

[5] Chih-Chin Liu and Po-Hun Tsai," Content-Based Retrieval of MP3 Music Objects ", Proc. the eleventh International Conference on Information and Knowledge Management(CIKM2002), pp. 506-511, 2002.

[6] Daniel P. W. Ellis, " Extracting Information From Music Audio ", Communications of the ACM, August 2006, Vol. 49, No. 8, pp. 32-37.