1ZG-8

歌声に合わせた楽曲自動再生システム　～ここカラ～

原　大輔　　太田　祐揮　　矢野　千晶　　濱川　礼
中京大学　情報理工学部　情報システム工学科

1. 概要

本論文では, 歌声を用いてユーザが探している楽曲を検索し, 歌声に合わせた位置から再生することを実現したシステム『ここカラ』について述べる. 手間取る楽曲の検索と再生に着目し, 楽曲を手軽に, ユーザの意図したフレーズから再生することを目的とする.

2. 背景

近年, ユーザの保有する楽曲数が増加している. [1]によれば, デジタルライブラリに保存している楽曲数が, 2009 年は, 1 人当たり平均 586 曲であったが, 2011 年では, 平均 1042 曲と増加している. このことは, メディアプレイヤ内での楽曲検索の重要性が増したと言える. 現在, web サービスとしての楽曲検索システムは, midomi[2]や SoundHound[3]などがある. これらは, 曲名を検索することができるが, [2]では, 視聴機能によりユーザが投稿した歌声が再生され, [3]では, 検索結果と同じタイトルの YouTube へのリンクが貼り付けられるなど, 原曲は再生されない. また, 既存研究[4][5]の多くは, MIDI ファイルを用いることで主旋律抽出を行い, DP マッチングによる楽曲検索を行なっている. しかし, 現在 MIDI ファイルは, 一般ユーザが保有していることは稀であり, ユーザの保有楽曲からの検索には現実的ではない. そこで我々は, ユーザの保有する楽曲から検索するシステム『ここカラ』を開発した.

3.『ここカラ』について

『ここカラ』は, 歌声を用いてユーザが探している楽曲を検索し, 歌声に合わせた位置から再生するシステムである. ユーザが保有している楽曲を検索対象としている. ユーザ保有の楽曲を活用することにより, 従来システムでは困難であった楽曲の再生を可能とした. また, この再生に関しても, 歌声と同じフレーズから再生を行うことを実現した. 実現に際して技術的問題については以下があげられる.

・MIDI ファイルを使用していない
・主旋律を忠実に再現することが不可能である
・主旋律以外の楽器音が楽曲中に混在している

これらの 3 つの問題を考慮した手法を次章で述べる.

Automatically playback music system to the singing voice ~KOKO*KARA*~

Daisuke Hara , Yuki Ota , Chiaki Yano and Rei Hamakawa

4. 提案手法

4. 1　概要

『ここカラ』は, 保有楽曲解析・楽曲検索という 2 つの処理を行っている. 保有楽曲解析処理[図 1]では, 保有楽曲を解析部で解析し, そのデータを保有楽曲データベースに格納する. 楽曲検索処理[図 2]では, ユーザの歌声を解析部で解析し, そのデータと保有楽曲データベースを使用し, マッチング部で楽曲検索を行う. 楽曲検索では, 蓄積したデータを使用し, 歌声と保有楽曲の類似度を総当たりで算出し, 検索結果として出力された楽曲を再生する. ただし, ユーザの歌声(データ)は, 保有楽曲と同じテンポ, キーでの歌唱を前提とする.

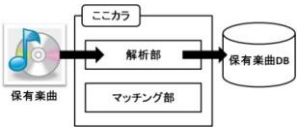

図 1：保有楽曲解析処理

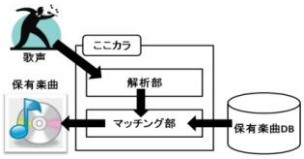

図 2：楽曲検索処理

4. 2　内部処理

『ここカラ』は, 大別して解析部とマッチング部から成る.

4. 2. 1　解析部

『ここカラ』は, MIDI ファイルを使用しないため, 保有楽曲の wav ファイルを解析する必要がある. 解析部では, 保有楽曲とユーザの歌声に対し, 周波数ごとの音の大きさ(以下, 特徴量)を抽出し, その結果を保有楽曲データベースに出力する処理を行う. なお, 主旋律は他の音より大きい傾向があると考え, 解析を行う.

4. 2. 1. 1　特徴量抽出

保有楽曲とユーザの歌声は wavext[6]を利用し wav フォーマットへ変換する. 変換済みファイルから 0.2 秒ずつ(以下, 1 フレーム)音データを読み込み, フーリエ変換を利用して特徴量抽出を行う. 特徴量を格納した配列(以下, 特徴量配列)の各要素には, $0Hz$から $4000Hz$までを $3.90625Hz$ごとに分割した特徴量が格納されている.

4. 2. 1. 2　マスク処理

特徴量配列について, 音の大きさが閾値よりも小さいデータを 0 として, その部分を無音とする.

4. 2. 1. 3　量子化処理

人間の歌声の音域を考慮して, マスク処理をした特徴量配列を C3($131Hz$)から A#7($3729Hz$)の 59 の音高へ分割し, まとめる. ある音高$X_n(n = 0,1,\cdots,58)$の周波数を$x_n(Hz)$としたとき, その前後の音高との間を境界として, $(x_{n-1} + x_n)/2$から$(x_n + x_{n+1})/2$の範囲にある特徴量配列の要素を音高X_nとしてまとめる. ここで, マスク処理を

した特徴量配列をR，Rにおける音高X_nの始点インデックスをS_n，終点インデックスをT_n，まとめた後に格納する配列をR'とする．配列Rの要素をT_nからS_nの範囲で平均を求め，$R'[n]$に代入する式を[図 3]に，例を[図 4]に示す．

$$R'[n] = \frac{1}{T_n - S_n + 1} \sum_{m=S_n}^{T_n} R[m]$$

図 3：量子化の計算式

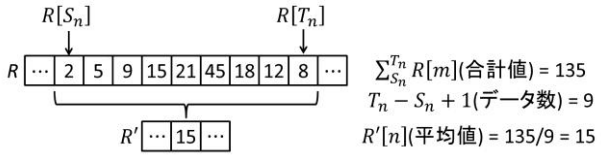

図 4：量子化処理の例

4．2．1．4　出力処理

最初に，1 フレーム毎に，R'の最大値を用いて，次式のように変換し，配列R''へ格納する．

$$R''[n] = R'[n] / \max_{0 \le x \le 58} R'[x] \quad (0 \le n \le 58)$$

これは，フレーム毎の全体の音の大きさの差をなくすための処理である．全フレームにわたり本処理を行い，その結果をフレームと音高の 2 次元配列で構成する．1 行を 1 フレーム，1 列を 1 音高として曲の長さだけ出力する．このとき，閾値より大きいデータは，主旋律を構成する候補として 1 を出力し，閾値以下のデータは 0 を出力する．閾値は，実験の結果最も精度の良かった 0.2 に指定した．

4．2．2　マッチング部

保有楽曲データに対して，歌声データをスライドしながらマッチングを行う．[図 5] にアルゴリズムを，[図 6] [図 7]に要素比較例を示す．歌声データの末尾が保有楽曲データの末尾を超えたら処理を終了し，次の楽曲の処理を行う．合計値が最も高いフレームを再生対象とする．

```
1:for(比較開始位置=0;比較開始位置<保有楽曲フレーム数;比較開始位置++){
2:      for(i=0;i<歌声フレーム数;i++){
3:              for(j=0;j<音高数;j++){
4:                      if(保有楽曲データ[i+比較開始位置][j]>0){
5:                              比較合計値+=(保有楽曲データ[i+比較開始位置][j])*歌声データ[i][j];
6:                              }
7:                      }
8:              }
9:              合計値配列[比較開始位置]=比較合計値;
10:             比較合計値=0;
11:     }
12:}
```

図 5：要素比較のアルゴリズム

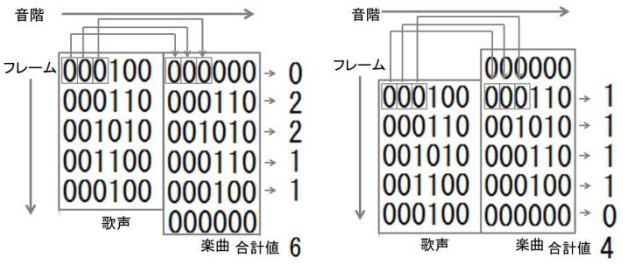

図 6（左）：歌声配列とオリジナル配列の要素比較例
図 7（右）：1 フレームスライド後の要素比較例

5．予備実験

我々が楽曲を準備し，100 曲，1000 曲(平均 4 分 30 秒/曲)を検索対象とした場合の検索時間と精度についてそれぞれ 50 回，85 回の楽曲検索を行った[*]．検索時間は，100 曲の場合平均 4.8 秒，1000 曲の場合平均 40.5 秒であった．また，1 位検索率は，100 曲の場合 62.0%，1000 曲の場合 42.4%であった．

6．評価・成果

大学生 15 名に対し，実際に各自が保有している楽曲 100 曲を使用し，『ここカラ』を用いて楽曲検索を行った．評価者は，検索したい楽曲 5 曲に対して，各曲異なる 2 か所，計 10 回検索を行った．その結果，精度に関しては，歌った楽曲の 1 位検索率 56.0%，5 位以内検索率 60.0%と，比較的，高評価を得ることができた．しかし，1 位検索率は 20.0～100%と個人差があった．また，「同じ楽曲が上位に検索される」という意見が多数あった．

7．考察

上記に示すように，ユーザの保有する楽曲の検索に有用なシステムを開発することができた．「同じ楽曲が上位に検索される」原因として，楽器音等の主旋律以外の音が大きい楽曲が含まれていることが考えられる．実際，予備実験の 100 曲で原因と考えられた 2 曲を人為的に除いて，再度実験を行ったところ，予備実験では 5 位以内に検索されなかった 19 か所中 15 か所が 5 位以内に検索され，精度が向上した．

8．展望

7 で述べたように，頻出する楽曲データに対する解析処理を考慮することで，精度が向上すると考える．現状では一意に指定している閾値を，各楽曲に適した値に変更する処理を検討中である．

9．参考文献・関連研究

[1]一般社団法人　日本レコード協会
http://www.riaj.or.jp/report/mediauser/index.html
[2]『midomi』　http://www.midomi.co.jp/
[3]『SoundHound』　http://www.soundhound.com/
[4]小杉尚子　他：“ハミングによる音楽検索システム”，情報処理学会論文誌，vol.45，No.1，2004
[5]市川拓人　他：“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”，情報処理学会研究報告，No.12，2008
[6]wavext　http://hp.vector.co.jp/authors/VA046927/wavext/

[*]実験環境
OS:Windows7, CPU:Intel(R) Core(TM) i7-3770 3.40GHz, メモリ:16.0GB