平成 24 年度　卒業論文

歌唱支援アプリケーションの開発

函館工業高等専門学校　情報工学科

須田　祥平、　東海林　智也

目次

1章　序論

1.1　英文アブストラクト

In this study, we develop an application for Android to correct a pitch of song which is sung at karaoke, and replay it. In most cases, we cannot avoid singing a song poorly. For some people, it might give them complex. On the other hand, over the past few years, a smartphone and tablet terminals working on Android OS and iOS have spread rapidly. Accordingly, a variety of applications have been developed. Therefore, we develop the application for the handy smartphone to support the tone-deaf and verify the effect.

1.2　研究背景

音楽業界では「VOCALOID」と呼ばれる技術を用いた楽曲が、アマチュア・プロを問わず制作、公開され爆発的な人気を得ている[1]。これに伴い、その曲を VOCALOID ではなく人間が歌いそれを公開する「歌い手」と呼ばれる人間が出現し始め、現在の動画投稿サイトで最も人気のあるジャンルのひとつとなっている。しかし、歌い手の投稿する動画の質が高くなるにつれて聞き手からの要求が引き上げられて『歌がうまくなければ投稿するな』といった論調も見られる[2]。

1.3　目的

本研究では、歌の音程修正を支援するアプリケーションを開発する。歌を歌う際に、音程を外すことは不可避である。人によってはそれがコンプレックスであったり、悩みの種になり得る。そこで、歌いたい歌の音程を自分が歌っている声で確認するアプリケーションを開発し本アプリケーションを使用することで、音痴の克服を支援し、歌い手がその歌を正しい音程で歌えるようにすることが本研究の目的である。

2章　開発環境

開発環境は以下の通りである。

CPU:Intel(R) Core(TM) i5-2300 CPU @ 2.80GHz
メモリ: 4103264 kB
OS：Ubuntu Linux 12.04
開発言語：C
コンパイラ：gcc

3章　前提知識

3.1　WAVEデータ

WAVE（RIFF Waveform Audio Format）は、MicrosoftとIBMにより開発された音声データ記述のためのフォーマットでRIFF（Resource Interchange File Format）の一種である[3][4]。通常、非圧縮の音声データを格納している。

WAVEファイルはヘッダと波形データから成っていて、ヘッダにはファイルサイズやサンプリングレートなどが記録されている。本アプリケーションで使用する情報はヘッダ情報「サンプリング周波数」と波形データの２つである。

3.2　MIDIデータ

MIDI（Musical Instrument Digital Interface)は、電子楽器の演奏データを機器間でデジタル転送するための世界共通規格である。実際の音ではなく音の高さ、大きさや種類などの演奏情報が登録されているため、実際の音声波形をサンプリングしたデータ形式（WAVEやMP3など）と比べてデータサイズが非常に小さいという特徴がある[5][6]。

MIDIデータのもっとも一般的な形式がSMF（Standard MIDI Format）である。本アプリケーションでもこのSMF形式のMIDIデータを採用している。SMFの構造は以下の通りとなっている。

・ヘッダ（データの大きさ、フォーマット、トラック数、時間単位情報が記録されている）
・トラックデータ１（Conductor Trackとも呼ばれる、フォーマット0ではこれ以降存在しない）
・トラックデータ２（演奏情報などが記録される）
・トラックデータ３（トラックデータ２と同様）
　…以降トラックの数だけ、トラックデータが続く

SMFのフォーマットは3つ（フォーマット0，1，2）存在するが、本アプリケーションでは主に複数のトラックから構成されているフォーマット１を使用している。

トラックデータは、以下のデータの組み合わせがトラック末尾まで続いている。

・「デルタタイム」＋「MIDIイベント」
・「デルタタイム」＋「SysExイベント」
・「デルタタイム」＋「メタイベント」

MIDIイベントとは、いわゆる演奏情報を示したものである。本アプリケーションではMIDIイベントの中から「ノートオン／オフ」を使用する。またメタイベントは、演奏情報に含まれないテンポや曲タイトルなどを納めるために用いるイベントである。本アプリケーションではメタイベントの中から「テンポ」を使用する。

3.3　エンディアン

エンディアンは、2バイト以上のデータを記録や転送する際に行う順番のことである[7]。バイトオーダとも言う。バイトオーダの違うコンピュータ同士などでデータをやり取りする場合、そのバイトオーダを考慮しないと値が狂ってしまうため、変換や統一を行う必要がある。MIDIデータは、最上位のバイトから順番に記録されるビッグエンディアン形式で格納されているため、最下位のバイトから順番に記録や転送を行うリトルエンディアンの環境を用いている場合はエンディアン変換を行う必要がある。変換方法は以下の通りである[8]。

①データを読み込み、データサイズ分の1次元配列を用意する。

②データの最下位と最上位を入れ替える。さらに最下位の次のデータと最上位の次のデータを入れ替えて①で用意した1次元配列に代入する。この操作をデータ全体に対して行う。

3.4　平均律

平均律とは、1オクターブなどの音程を均等な周波数比で分割した音律のことで、均一に12等分した音律のことを特に十二平均律という[9]。ある基準音の1オクターブ上の音は基準音の2倍の周波数を有していることから、隣り合う音、すなわち半音の周波数比は（2の12乗根）：1となる。

3.5　STFT（Short-Time Fourier Transform：短時間フーリエ変換）

STFTは、関数に窓関数をずらしながらかけて行うフーリエ変換のことである[10][11]。波形を一定の大きさのフレームごとに分割して各フレームに窓関数をかけ、各フレーム毎にFFTを実行する。データ数が増えるにつれて莫大な計算量が必要になるDFTや高速だがフーリエ変換を行うデータ数に制限があるFFTの欠点を補った手法である。

3.6　音量正規化

音量（または音声）正規化は、音声データ中の最大音量を特定し、一定の音量レベルに収まるように全体の音量を調整する処理のことであり、音割れしないように全体の音量を上げる際に用いる[12][13]。信号のピークを分析して調整する手法とRMS（Root Mean Square:平均自乗偏差）を分析して調整する手法の２つが存在するが、本アプリケーションでは信号のピークを分析する手法を用いた。

4章　音程修正アプリケーションの開発

4.1　仕様

はじめに端末に向かって歌ってそれを録音し、WAVE フォーマット形式でデータを作成する。以後これを「歌データ」と呼ぶ。次に、あらかじめ用意してある MIDI データを用いて教師データを作成する。作成した教師データを基に歌データのピッチシフトを行って、新たに WAVE フォーマット形式のデータを作成する。

4.2　教師データの作成

教師データの作成は、以下の手順で行われる。

①教師データを保存する 1 次元配列を作成する。

②MIDI データからヘッダとトラックデータを読み込む。実行環境がリトルエンディアン形式を採用している場合は、データを読み込んだ後にビッグエンディアン形式に変換する。

③トラックデータを解析する。各イベントを読み込んだ際の操作を下に示す。

(1)「テンポ設定」からテンポを取り出し、1tick あたりの実時間を計算する。
(2)「ノートオフ」からデルタタイムを取得する。
(3)「ノートオン」からノートナンバーから周波数を計算する。さらにその時までのデルタタイムの総計を用いてノートがオンになった実時間を取得する。こうして求めた周波数と実時間を教師データとして登録する。また、ノート数をカウントしておく。

デルタタイムは次に続くイベントまでの時間を表す時間情報のことである。単位は tick であり、ヘッダ情報の Division（四分音符の分解能）とメタイベントの「テンポ」を用いて実時間への変換を行う。「テンポ」は、1 分間における四分音符の数を表している音楽におけるテンポとは異なり、MIDI データでは四分音符の長さが実時間で表されている[6]。単位はマイクロ秒である。テンポを Division で割って 1tick の実時間を求め、その時点でのデルタタイムの総計と求めた 1tick の実時間の長さをかけて実時間に変換する[14]。

ノートオン／オフは、指示されたノートナンバーの音を発音／消音するイベントである。オンの場合は音の大きさも指定する。ゼロを指定した場合はノートオフと同義になる。ノートナンバーにはその番号に対応した音の高さが割り当てられている。このノートナンバーと平均律を用いて周波数を算出する。MIDI ではノートナンバー 69 がオクターブ 4 のラ、すなわち国際標準で認められている 440Hz となっている。このノートナンバーと 440 という数値を参考に、十二平均律から周波数を計算する式を求めると式(1)になる[15]。

$$440\times 2^{\frac{NoteNumber-69}{12}}=frequency\cdots(1)$$

これ以外にもさまざまな MIDI イベントやメタイベントが存在しているが、それらが教師データにあたる MIDI データ内に存在する場合は全て読み飛ばす。

4.3　ピッチシフト処理

4.3.1　重ね合わせと窓関数

重ね合わせは、フレーム同士を一定の大きさだけ重ね合わせて処理を行うことである。プログラム上では、フレームをさらにステップに分割することで、重ね合わせの割合を実現する。本アプリケーションではステップの大きさをFFTのフレームサイズの32分の1で設定している。

窓関数は、ある有限区間外の値を0とする関数である[16][17]。これをFFTを行う前の各フレームにかけることでスペクトル分解能が向上する。さまざまな種類が存在し、その関数により性能も異なってくる。本アプリケーションでは一般的な窓関数であるHanning窓を採用している。

4.3.2　部分周波数

FFTを行った場合、周波数は（サンプリング周波数）÷（データ数）の間隔でしか解析できず、この間に相当する周波数の結果は最も近い周波数帯に移動する必要がある[18]。より正確な周波数を求めるために、FFTの結果から得られる位相を記録して用いる。

4.3.3　ピッチシフト倍率

教師データの周波数と部分周波数を用いてピッチシフト倍率を計算する。

①振幅スペクトルを求める。

②直流成分を除いた振幅スペクトルが最大になるデータ番号を求める。

③②で求めたデータ番号の部分周波数を取り出す。

④教師データの周波数と③で取り出した部分周波数の割合がピッチシフト倍率となる。

4.3.4　ピッチシフト操作

ピッチシフトは振幅スペクトルと周波数データのシフトによって実現される。また、周波数データにピッチシフト倍率をかけて周波数を変化させる必要がある。手順は以下の通りである[19]。

①データ番号にピッチシフト倍率をかけて振幅スペクトルと周波数データのシフト位置を算出する。

②振幅スペクトルと周波数データをシフト位置を移動する。

③周波数データにピッチシフト倍率をかける。

以上の操作を音声波形終端まで繰り返し行った後、IFFTと重ね合わせ・窓関数処理を行って波形を生成し、最後に正規化処理を行なってピッチシフト処理は終了である。新たに生成された音声波形をWAVEフォーマット形式でデータを作成する。

５章　結果と考察

5.1　入力音声と出力音声の比較

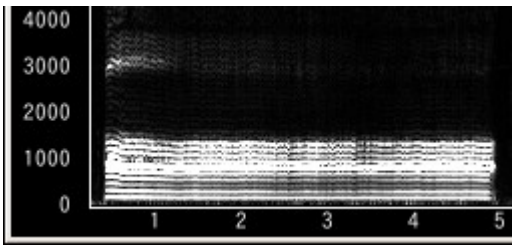

図１　入力音声のパワースペクトル

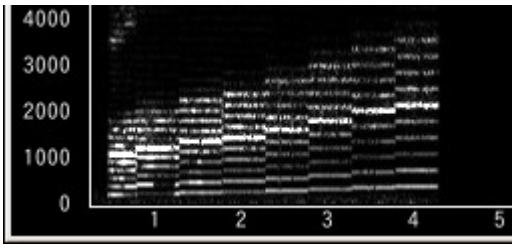

図２　出力音声のパワースペクトル

```
time : 0.00000[s]       freq : 261.62557[Hz]
time : 0.01874[s]       freq : 261.62557[Hz]
time : 0.51842[s]       freq : 293.66477[Hz]
time : 1.01810[s]       freq : 329.62756[Hz]
time : 1.51778[s]       freq : 349.22823[Hz]
time : 2.01746[s]       freq : 391.99544[Hz]
time : 2.51714[s]       freq : 440.00000[Hz]
time : 3.01682[s]       freq : 493.88330[Hz]
time : 3.51650[s]       freq : 523.25113[Hz]
time : 4.01618[s]       freq : 0.00000[Hz]
```

図３　教師データ

図１は、入力時の歌データのパワースペクトル、図２はピッチシフト処理後の歌データのパワースペクトルである。横軸は時間（秒）、縦軸は周波数（Hz）また、図３は教師データである。パワースペクトルを比較すると、処理前と処理後でパワースペクトルが変化していることがわかる。さらに、入力時とピッチシフト処理後の歌データをそれぞれ聞いて比較すると、パワースペクトルの表示からわかるように、音程が一本調子だった歌データが教師データに合わせてピッチシフト処理を行ったことで音程がついているかのように聞こえた。しかし、音声がとても機械的で人間の発声と呼ぶには程遠いという問題があった。

5.2　考察

上述したピッチシフト処理後の音声が機械的な発声になる問題は、アルゴリズムの再検討をすることで解決できる。機械的な発声になる原因を調査したところ、ピッチシフト後の振幅スペクトルに対する位相スペクトルの形状の一致しないことが原因と判明した[20]。よって、ピッチシフト処理後に振幅スペクトルと位相スペクトルを一致させるアルゴリズムを追加することでこの問題を解決できると考えられる。

６章　総括

本研究では、音痴を克服したり歌の正しい音程を知って正しい音程で歌えるようになることを支援するアプリケーションの開発を行った。

今後はAndroidアプリケーションへの移植を行うことを予定している。ただし、今回は動作確認を全てPC上で行ったためAndroid端末のスペックを考慮した開発を行っていない。そのため実際にAndroid端末上で動作させた場合、計算速度が非常に遅くなることが考えられる。そこで、FFTやピッチシフト処理の計算をクラウド上のサーバに任せることも考える。

参考文献

[1]初音ミク “キャラ萌え”だけじゃない人気の本質とは ：日本経済新聞
http://www.nikkei.com/article/DGXNASFK1002B_Q2A410C1000000/

[2]「昔のニコ動は…」Twitterで話題のニコニコ動画への不満ツイートベスト25 - NAVER まとめ
http://matome.naver.jp/odai/2134440748265351001

[3]WAV - Wikipedia
http://ja.wikipedia.org/wiki/WAV

[4]wav ファイルフォーマット
http://www.kk.iij4u.or.jp/~kondo/wave/

[5]MIDI - Wikipedia
http://ja.wikipedia.org/wiki/MIDI

[6]about Standard MIDI format
http://www2s.biglobe.ne.jp/~yyagi/material/smfspec.html

[7]エンディアンとは【endian】（バイトオーダー） - 意味/解説/説明/定義 ： IT用語辞典
http://e-words.jp/w/E382A8E383B3E38387E382A3E382A2E383B3.html

[8]初心者のプログラミング体験記 C言語でMIDI(SMF)データを読んでみる！
http://torasukenote.blog120.fc2.com/blog-entry-104.html

[9]平均律について
http://stby.jp/heikinritu.html

[10]短時間フーリエ変換 - Wikipedia
http://ja.wikipedia.org/wiki/%E7%9F%AD%E6%99%82%E9%96%93%E3%83%95%E3%83%BC%E3%83%AA%E3%82%A8%E5%A4%89%E6%8F%9B

[11]ピッチシフトをしてみよう | ボロリン
http://blog.jyoken.net/?eid=793438

[12]株式会社ペガシス：サービス＆サポート
http://tmpgenc.pegasys-inc.com/ja/support/labo/sound.html

[13]音量正規化 - Wikipedia
http://ja.wikipedia.org/wiki/%E9%9F%B3%E9%87%8F%E6%AD%A3%E8%A6%8F%E5%8C%96

[14]General MIDI Lite - AMEI <一般社団法人 音楽電子事業協会>
http://amei.or.jp/specifications/General_MIDI_Lite_v1.0_japanese.pdf

[15]音楽研究所 研究テーマ->DTM->資料
http://www.asahi-net.or.jp/~hb9t-ktd/music/Japan/Research/DTM/freq_map.html

[16]窓関数 - Wikipedia
http://ja.wikipedia.org/wiki/%E7%AA%93%E9%96%A2%E6%95%B0

[17]窓関数：窓関数をかけてFFTの結果を最適化する - Developer Zone - National Instruments
http://www.ni.com/white-paper/4844/ja

[18]ＦＦＴによる周波数解析
http://mniwa.web.fc2.com/store/fft.html

[19]Pitch Shifting Using The Fourier Transform : The DSP Dimension
http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/

[20]MATLAB Note/音声の加工 - Miyazawa's Pukiwiki 公開版
http://shower.human.waseda.ac.jp/~m-kouki/pukiwiki_public/75.html#b7c530cc