平成 25 年度　卒業論文

マイクによる音声入力を利用した自動作曲アプリ

函館工業高等専門学校　情報工学科 5 年

東海林研究室　田村祐樹

目次

1章　序論

1.1　英文アブストラクト

Generally, it is recognized that music composition is difficult. Therefore, the purpose of this study is to simplify music composition by making an automatic music composition program running on Android terminal. At first, a user inputs a melody into the microphone of the Android terminal. Then, the automatic music composition program assigns chord progressions to the melody. Finally, the automatic music composition program makes the accompaniment based on chord progressions and makes music complete by mixing the melody and the accompaniment. The user can easily enjoy music composition only by inputting the melody. We performed further experiments of the program on some users. We also show the experimental results in the paper.

Key words:　automatic music composition, Android, chord progressions

1.2　研究背景

作曲に興味があるユーザーが PC 上で作曲を行おうとすると作曲アプリケーションを利用することになる。既に開発されている作曲アプリケーションには五線譜に音符を打ち込むもの、ピアノロールを用いたもの、音声入力を利用するものなどがある[1][2][3]。しかし、ユーザーが実際に作曲をしようと思うとコード進行、調、音楽用語などの作曲に関する知識が必要不可欠である。更に音声入力を作曲アプリケーションに使う場合にはマイクなどの機器が必要になる。これらの問題点により興味があったとしてもユーザーは作曲に躊躇してしまう。

そこで本研究では、このような問題点から作曲に躊躇してしまっている人々のために、作曲を簡単に楽しんでもらうアプリケーションを開発することにした。

2章　開発環境

開発環境は以下のとおりである。

使用 PC	Windows 7 Enterprise SP1 32bit version Intel Core 2 DUO E6550 2.33GHz RAM 2.0GB
使用ソフト	Visual C++ 2008 Express Edition AzPainter2 Domino
使用ライブラリ	MIDIData ライブラリ[4]
使用言語	C 言語

3章　自動作曲アプリケーションの開発

3.1　概要

初めに作成する曲のテンポを設定した後、声や楽器の音といった音声を PC のマイクから録音する。その音声を FFT により解析して曲のメロディを作成し、そのメロディに対してコード進行を生成する。その後、コード進行から伴奏パートが作成されて曲を完成する。完成した曲はフォーマット 1、分解能 120 の MIDI 形式データとして保存される。

3.2　メロディパートの作成

取り込んだ音声をサンプリングレート 22050Hz、量子化 Bit 数 16Bit、モノラルで Wav 形式データに保存する。そのデータに対して 2048 点 FFT を行う。FFT により解析された音声の音高とその長さの情報を MIDI 形式データに保存することでメロディパートを作成する。なおメロディの音階の範囲はオクターブ 1 のシからオクターブ 5 のシ(約 61Hz~987Hz)までである[5]。実際に五線譜に入力範囲を示したものが図 1 である。ここで、青い範囲は一般成人男性の音域(オクターブ 2 のラからオクターブ 4 のラ)、赤い範囲は一般成人女性の音域(オクターブ 3 のファからオクターブ 5 のレ)である。本アプリケーションの入力範囲は一般男性、一般女性の音域よりも広いため人間の声は問題なく認識される。

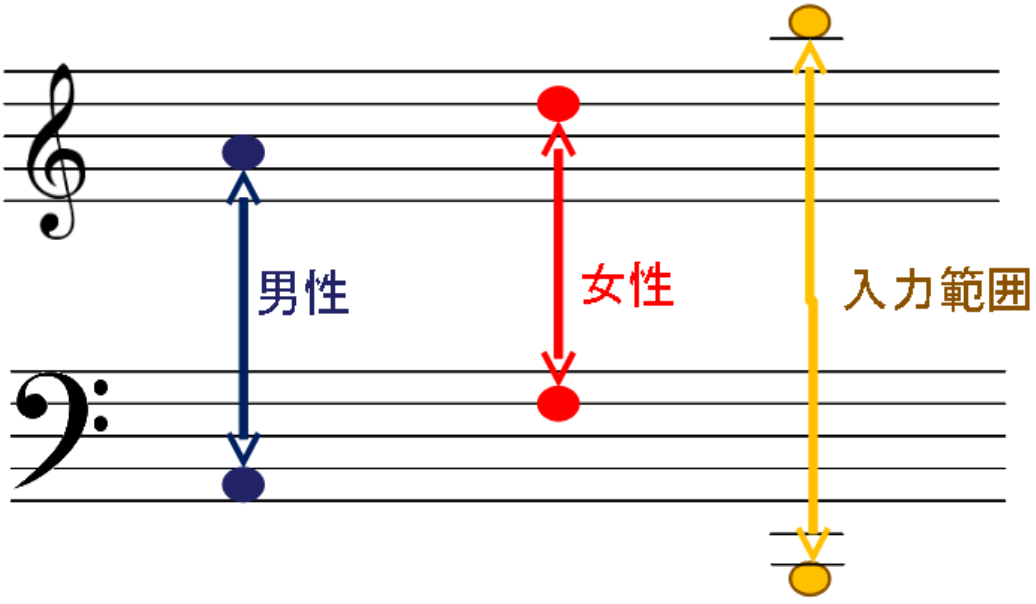

図 1　音階の入力範囲を五線譜で表した図

3.3　調の判定

3.2 で解析したメロディから調の判定を行う。メロディ中に含まれているド、ド#(レ♭)、レ、レ#(ミ♭)、ミ、ファ、ファ#(ソ♭)、ソ、ソ#(ラ♭)、ラ、ラ#(シ♭)、シの計 12 音の数を調べることによって判定する。

調に含まれている#(♭)の数はそれぞれ違うが、ある音に#(♭)が付く順番には法則が存在する。具体的には#の場合はファ→ド→ソ→レ→ラ→ミ→シの順に付く。♭の場合はこの逆でシ→ミ→ラ→レ→ソ→ド→ファの順である[6]。

そこで、この順番を利用して#(♭)の数を判定する。例えば、#の数を調べる場合は最初にファとファ#の数を調べる。ファ#の数がファの数よりも圧倒的に多い場合はそのメロディではファの音は全て半音上がっているものと判定する。次にドの音に関して同様に#が付くか調べていく。もしドに#が付かなければこのメロディは#を 1 つだけ含むト長調と判断する。もしドにシャープが付けば次はソの音を調べる。この様な操作を繰り返し行なって最終的な調を判断する。なお長調・短調の判断は主音によって判断を行っている。

3.4　コード生成

3.3 で決定した調から 7 つのコードを生成する。まず、その調でよく使われる音である「主音」から 7 つの音を並べて音階の作成をする。例えばト長調の主音は「ソ」であるのでソから 7 つの音を順番に並べた「ソラシドレミファ#」をいう音階を作成する。そして並べた音を根音とするコードを 7 つ作成する[7]。ト長調の場合に作成されるコードを図 2 に示す。コードは左から順にⅠ、Ⅱ、…Ⅶとギリシャ文字で表される。

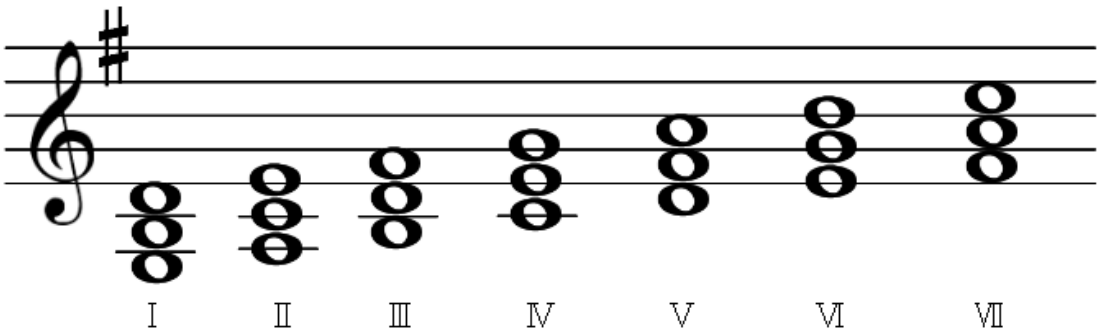

図 2　ト長調の場合に作成される 7 つのコード

3.5　コード進行

3.4 でコードを生成したら、メロディと照らし合わせながらコード進行を作成していく。ただしコード進行には規則性がある。その規則を図 3 に示す[8]。

Ⅰ→Ⅲ	Ⅲ→Ⅵ	Ⅴ→Ⅰ	Ⅵ→Ⅳ
Ⅰ→Ⅳ	Ⅳ→Ⅰ	Ⅴ→Ⅱ	Ⅶ→Ⅲ
Ⅰ→Ⅵ	Ⅳ→Ⅱ	Ⅴ→Ⅲ	Ⅶ→Ⅰ
Ⅱ→Ⅳ	Ⅳ→Ⅲ	Ⅴ→Ⅳ	
Ⅱ→Ⅶ	Ⅳ→Ⅴ	Ⅴ→Ⅵ	
Ⅲ→Ⅳ	Ⅳ→Ⅶ	Ⅵ→Ⅱ	

図 3　本アプリケーションで使用しているコード進行規則

図 3 はどのコードからどのコードへ進行できるかというものを表したものである。例えば左上の「Ⅰ→Ⅲ」という規則は「ⅠのコードからはⅢのコードへ進行可」と言った意味を表している。他に「Ⅰ→Ⅳ」「Ⅰ→Ⅵ」という規則もあるのでⅠのコードからはⅢ、Ⅳ、Ⅵのコードへ進行できるということである。また、「Ⅰ→Ⅰ」のように同じコードへ進行することもある[9]。

このコード進行規則とメロディを照らし合わせることによってコード進行を作成していく。ここでコードは 1 拍(4 分音符)の間隔で割り当てられる。まずメロディ中の音符を 1 拍ずつ見ていき､この 1 拍の間にどの音高の音符がいくつ含まれているのかを判定したあと、その音高を含むコードを候補として挙げていく。候補の中からコード規則に準じているものをその部分のコードとして割り当てる。もし当てはまるコードが複数ある場合はランダムでコードを割り当てる。これをメロディの終了まで繰り返していくことによりコード進行の作成を行う。

例として、図 4 で示すようなメロディが入力されたとする。点線部分はメロディを 1 拍ずつ区切ったということを表している。

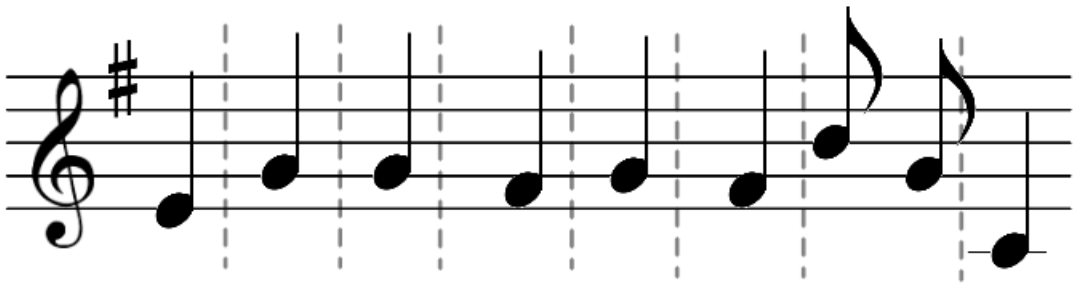

図 4　入力されたメロディの例

1 拍ずつ見ていくと最初は「ミ」の音符がひとつだけであるので「ミ」の音を含むコードを調べると「Ⅱ」「Ⅳ」「Ⅵ」のコードが該当する。候補が複数あるのでこの場合はランダ

ムに「Ⅵ」のコードを割り当てた。次の「ソ」の音符に対しても同様の操作を行うと「Ⅰ」「Ⅳ」「Ⅵ」のコードが候補として挙げられるが「Ⅵ→Ⅰ」という規則はないため「Ⅰ」のコードは省かれる。残った「Ⅳ」「Ⅵ」のコードからランダムで再び「Ⅵ」のコードを割り当てた。これを繰り返し行うことで図 5 のようなコード進行が作成された。

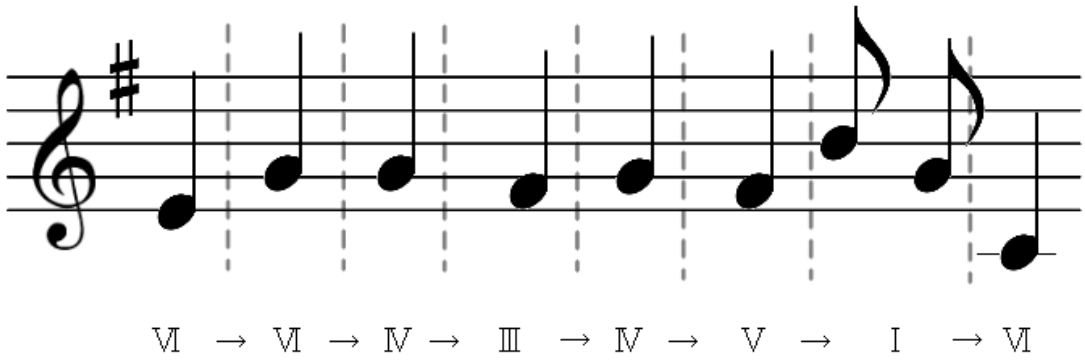

図 5　完成したコード進行

このコード進行は一例であり、違うコードがランダムに選択されるため実行するたびに全く異なるコード進行が作成される。

3.6　伴奏パートの作成

伴奏パートを作成する前に使用する楽器・音量・パン(音が左右のどのくらいの距離から聞こえてくるか)・ビブラートの有無をランダムに決定する。これらの設定は 0～128 の間の数字で決定される。楽器は 0～128 のそれぞれに数字に対応したものが割り当てられる。音量は 128 に近づくほど大きくなる。パンは 0 に近づくほど左に音が寄り、128 に近づくほど右に寄る。中央値である 64 は音が中央から聞こえてくる。ビブラードは数字が大きいほど音の揺れが激しくなる。ここで楽器は ProgramChange イベント、パン・ビブラートの有無に関しては ControlChange イベント、音量に関しては NoteOn イベントで設定を行っている。

以上の設定を終えたら伴奏を生成していく。例えば図 6 のようにト長調のⅠのコードがありその部分に伴奏を生成するものとする。まずⅠのコードの構成音である「ソ・シ・レ」

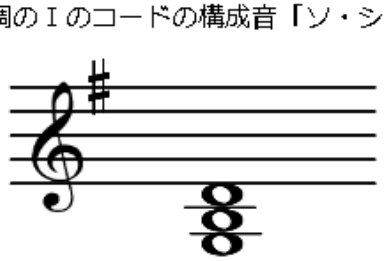

図 6　ト長調のⅠのコードの構成音

のうち一つ出力する音を決定する。なおこの時、ランダムで音を出力する代わりに休符となることもある。音を決定したあと、その音の音符長(何分音符か)をランダムに決定する。この動作を繰り返すことにより伴奏パートを作成していく。実際にト長調のⅠのコードから作成された例を図 7 に示す。図 7 ではソの 8 分音符、16 分休符、シの 16 分音符といった伴奏が生成された。

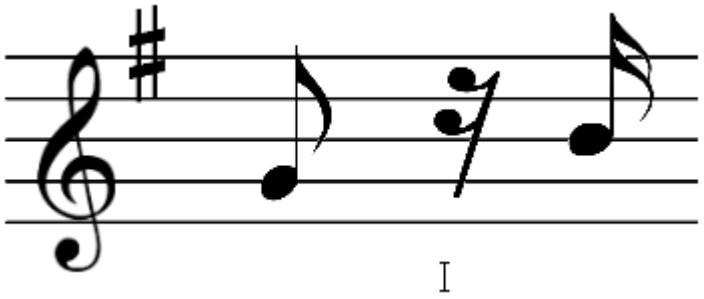

図 7　ト長調のⅠのコードに対して生成された伴奏

3.7　ベースパートの作成

ベースパートに関しても 3.6 の伴奏パートと同じように楽器をランダムに決定する。その後、以下で説明する 3 パターンの生成法によってベースパートが作成される。1 つ目のパターンは伴奏パートと同じようにしてメロディを生成するパターン(以下パターン A)である。2 つ目はコードの根音をそのまま利用するパターン(以下パターン B)である。3 つ目はその根音を 1 オクターブずつ上下させていくパターン(以下パターン C)である。実際に「Ⅰ→Ⅳ」の進行で作成されたベースパートを次ページの図 8 に示す。

パターン A では先ほどの伴奏パートの作成で説明したようにそれぞれのコードの構成音からランダムに選ばれた音を出力することによってベースパートを生成する。今回の例ではⅠのコードの部分からはソとシの 8 分音符が、Ⅳのコードの部分からはソとドの 8 分音符が出力された。

次にパターン B、パターン C を説明する。今回の例で使用しているⅠのコードの根音は「ソ」、Ⅳのコードの根音は「ド」である。

パターン B の場合では根音をそのまま使うためⅠのコードの部分ではソの音のみが、Ⅳのコードの部分ではドの音のみが使われている。残りの小節の場合も同様に根音のみを使う。

一方、パターン C も同じように根音のみを使用しているがⅠのコードの部分ではオクターブ 2 のソとオクターブ 3 のソを交互に使用している。同じようにⅣのコードの部分もオクターブ 2 のドとオクターブ 3 のドを交互に使用している。残りの小節に対しても同じことを行っている。

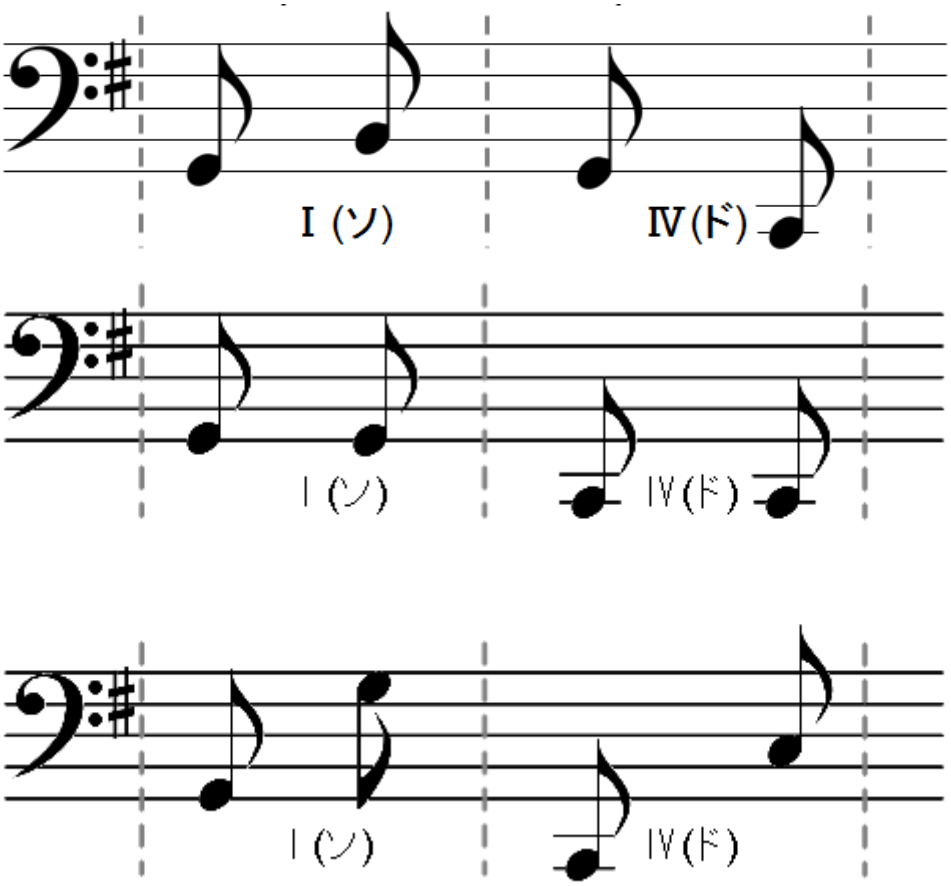

図 8　Ⅰ→Ⅳの進行で作成されたベースパート
(上がパターン A、中央がパターン B、下がパターン C)

3.8　ドラムパートの作成

ドラムパートはひとつのドラムセットのみを使うので他パートのようにランダムに楽器を決定することはない。ドラムパートは 4 ビートか 8 ビートかをランダムで決定したあと複数用意してあるリズムパターンの中からランダムに決定したものを繰り返し使用するものとなっている。なお、リズムパターンはアコースティックスネア、バスドラム、ハイハットクローズ、ハイハットオープン、クラッシュシンバルの 5 つから構成している。

3.9　出力・保存

以上の作業を終えて完成した曲を MIDI ファイルとして出力して保存する。図 9 に完成した MIDI ファイルを MIDI 編集ソフトで表示したものを示す。

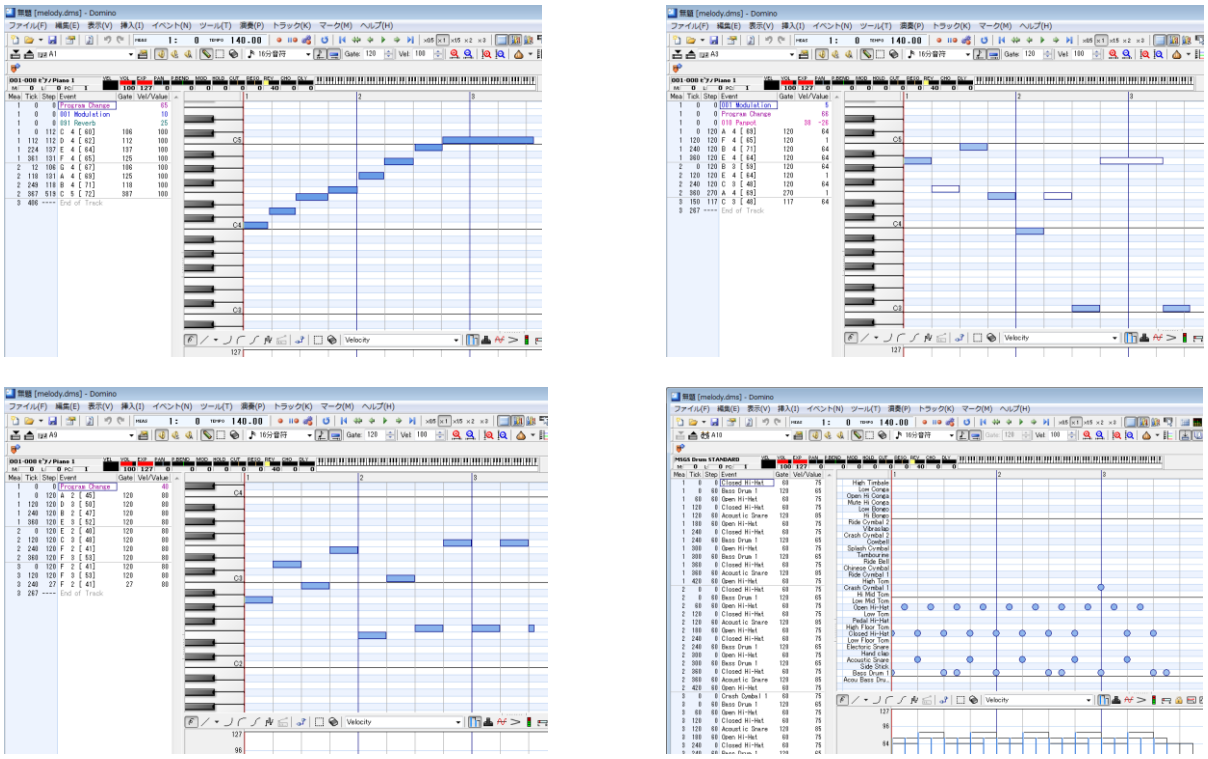

図 9　実際に生成された MIDI ファイルを表示したもの
(左上がメロディ、右上が複数ある伴奏パートのひとつ、左下がベースパート、右下がドラムパート)

4章　テスト

実際に本アプリケーションを使って作曲を行うテストを行った。対象者は 18~20 歳の男性 2 名と女性 2 名である。実施日は 2014 年 2 月 24 日(月)である。

はじめに PC に接続したマイクから音声を録音して、C 言語で作成した自動作曲のプログラムを実行して曲を作成し、最後にその曲を聞いてもらった後にアンケートを取った。アンケートの内容と結果は図 10 で示す通りである。

アンケート項目	男性		女性	
	1 人目	2 人目	1 人目	2 人目
簡単に作曲できたか	5	5	5	5
完成した曲のクオリティはどうか	5	3	5	5
実際に使ってみたいか	5	3	5	5

図 10　アンケート項目の一覧と回答　(5 段階で高いほど良い)

また、感想として以下のコメントを得た。

・作曲の時間が非常に短かった。

・一瞬で作曲できてた。すごい。

・入力した声の音程がずれずに楽器の音になっていてすごい。面白そうなので使ってみたい

・いろんな楽器の音が聞けて面白い。

5章　考察

5.1　アンケート結果からの考察

アンケートの結果から「簡単に作曲できたか」という項目が全て 5 であったことから本アプリケーションで作曲を簡単に行うことができるということがいえる。しかし、曲のクオリティに関しては評価が良くなかったので今後作成される曲のクオリティの向上を目指していきたい。

5.2　その他の問題点

その他の問題点のひとつとしてマイク入力から録音した音声を解析する関数の精度が良くないという点が挙げられる。録音された音声に雑音が入っていると雑音までメロディとして判定されてしまったり、音が小さいとしっかり認識されなかったりといった問題があった。これらの問題点を解決するために、音声ファイルの音量の正規化をした後でパワースペクトルから音の強さを判断するという処理を試す予定である。

もうひとつの問題点は、作成される曲のジャンルとドラムのリズムパターンが少ない点である。この問題点は作成される曲のバリエーションに関係する。特にドラムのリズムパターンは同じものが何度も使われてしまうため、曲のジャンルとドラムのリズムパターンを増やすことにより作成される曲のバリエーションを増やそうと考えている。

5.3　Android アプリケーションへの移植

ユーザーが更に手軽に作曲を楽しめるようにするため、本アプリケーションを現在普及している Android 端末向けのアプリケーションに移植している。現在は Android NDK[10]を用いて自分が作成した C 言語の関数を Android アプリケーション向けに移植している。また MIDI 操作に使用しているライブラリも Android 向けではなかったため、Android アプリケーション向けにソースを移植している。Android アプリケーションを起動した際に表示される画面と作曲ボタンを押した際に表示されるダイアログを図 11 に示す。また、使用ソフト、ライブラリ、スマートフォン、言語を以下に示す。

使用ソフト　　ADT Bundle For windows 32bit
Eclipse 3.8
ADT 21.1.0
Android NDK Revision 9c
AzPainter2
Domino

使用ライブラリ	MIDIData ライブラリ[4]
使用スマートフォン	KDDI HTC j butterfly HTL21 Android 4.1.1 RAM 2GB
使用言語	Java, C 言語

図 11　起動画面と作曲ボタンを押した際に表示されるダイアログ

6章　総括

本研究では PC に接続されたマイクへの音声入力を利用することによって誰でも簡単に作曲を行うことができるアプリケーションの開発を行った。更に実際に録音した音声ファイルから曲を作成して聞いてもらうというテストを行った。そのアンケート結果から、簡単に作曲ができるアプリケーションを開発できたことが分かった。

今後の課題としては、C 言語で作成した関数を Android NDK を利用して Android 向けアプリケーションに移植すること、音声認識を改良すること、曲のバリエーションとクオリティを向上させることが挙げられる。

参考文献

[1] 自分の声を自動的に合唱にしてくれるアプリ「合唱ひとり」
http://nlab.itmedia.co.jp/nl/articles/1307/19/news113.html

[2] 手軽で簡単！ 身近な音で曲作りできるアプリ「iMaschine」が面白い
http://rocketnews24.com/2011/10/28/146363/

[3] Pocketband - Music Studio
http://pocketband.net:8080/home

[4]おーぷん MIDI ぷろじぇくと
http://openmidiproject.sourceforge.jp/

[5]MIDI ノート番号と音名、周波数の対応表
http:/ /www.asahi-net.or.jp/~HB9T-KTD/music/Japan/Research/DTM/freq_map.html

[6] 水谷友香・キッシー岸田 共著,CD で覚えるやさしい楽譜の読み方 ,p.50-p.59(成美堂出版,2005)

[7] 水谷友香・キッシー岸田 共著,CD で覚えるやさしい楽譜の読み方 ,p.86-p.89(成美堂出版,2005)

[8] 成瀬正樹 著,コード進行スタイルブック,p.38(Rittor Music,2001)

[9] 若松正司 著,若松正司の音楽セミナー コードの使い方①,p.76(音楽之友社,1989)

[10] Android NDK の標準ライブラリに libunzip を追加して使用する
http://www.usefullcode.net/2011/01/android_ndk_libunzip.html