平成 27 年度 卒業研究

自動採譜プログラムの開発

函館工業高等専門学校 情報工学科 5 年

3 番 池田由希

指導教員 東海林智也

目次

1 章 序論

1.1 英文アブストラクト

Automatic transcription is a theme that a lot of researchers have been studied so far. The purpose of this study is to develop a program that transcribes from the music signal automatically. First, the system obtains Fourier coefficients F[k] from the music signal f(i). Second, the system evaluates the basic frequency from F[k]. Third, the system obtains the correlation between F[k] and the instrument data that is has been prepared in advance. Finally, the system determines the instrument from the correlation, and outputs the information such as the instrument type, pitch and time as a file.

Keyword: digital music, automatic transcription, multi-pitch analysis

1.2 研究目的

コンピュータを用いて音楽信号から楽譜を自動生成するという自動採譜のアルゴリズムの研究を行う。更に、音声ファイル（WAVE 形式）を入力すると、その波形を解析し結果を別ファイル（MIDI 形式）として出力するプログラムの開発を行う。

1.3 研究背景

今日ではデジタル音楽についての技術は大きく進歩し、アナログで演奏された音をデジタルのデータで取り込むことや、そのデータからアナログの音に限りなく近い音を出力することが可能となっている。更には全てデジタル上で音楽を作成することも簡単になった。

しかしながら、その音楽を構成する単音に分解する採譜技術は未だ確立された方法はなく、研究の域を出ないままでいる[1]。例えばデジタル音楽を分析するために、音高解析、楽器判定、曲中のテンポ（速さ）の解析など、様々な研究が進められている。本研究では「多重音高解析」、「楽器判定」についての処理を行うプログラムを製作した。

2 章 プログラムの仕様

2.1 概要

本研究で作成するプログラムは、WAVE 形式ファイルの名前と曲中のテンポを入力すると自動採譜を行った結果を MIDI 形式ファイルで出力するものである。

WAVE（RIFF Waveform Audio Format）という形式は、音声データ記述のために開発されたフォーマットであり、RIFF（Resource Interchange File Format）の一種である[2][3]。WAVE 形式ファイルは 44 バイトのヘッダと波形データから成っており、ヘッダにはそのファイルの情報が記述されている。本研究では、ヘッダ中の「サンプリングレート」、「チャンネル数」、「データサイズ」、そして波形データの情報を使用する。

また MIDI（Musical Instrument Digital interface）という形式は、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に策定された規格である。

本プログラムは WAVE 形式ファイルに記述された数値データをフーリエ変換し、その結果から周波数解析を行って音高解析や楽器判定を行う。大まかな処理の流れを以下に示す。

① パラメータ取得

WAVE 形式ファイルから各種パラメータを取得する。

② 高速フーリエ変換（以下 FFT）

N 個（$N = 2^X$）の数値データから周波数スペクトルを求める。

③ 基音検出

周波数スペクトルからピークを検出し、その中から中心周波数を求める。

④ 音高解析

中心周波数はどのオクターブで、12 音技法で表すとどの音なのかを求める[4]。

⑤ 楽器判定

用意された楽器データと比較し、何の楽器が音を発しているのかを求める。なお、楽器数、登場楽器は予め判明しているものとした。

※⑤までを行った後もまだ未判定の音が残っていると判定された場合、③ ～ ⑤を繰り返し行う。

⑥ ファイル出力

解析結果を MIDI 形式で出力する。

各項目の詳細については 3 章で解説する。

2.2 開発環境

開発環境は以下の通りである。

OS:	Windows8
使用ソフト:	Microsoft Visual Studio 2013
	SoundEngine Free
	Domino(音源は Microsoft GS Wavetable Synth)
使用ライブラリ:	MIDIData ライブラリ[5]
使用言語:	C 言語

3 章 自動採譜プログラムの作成

3.1 パラメータ取得

最初に WAVE 形式ファイルを読み込み、ヘッダの情報全てとデータを取得する。なおテンポは予め判明しているものとしユーザが事前に指定する。テンポは 1 分間中の 4 分音符の数を示すもので BPM（Beats Per Minute）とも呼ばれる。例えばテンポ 60 なら 1 分で 60 個、つまり 1 秒おきに 4 分音符が登場する。

また、何分音符ごとに採譜するかを決める採譜間隔を事前に設定しておく。この採譜間隔とテンポ、サンプリング周波数、そしてチャンネル数から 1 つの採譜間隔分音符のデータ数を求め、それを解析幅とする。次に行う FFT はこの解析幅おきに実行する。以下に解析幅を求める式を示す。

解析幅 = サンプリングレート × チャンネル数 × 4 ÷ 採譜間隔 × 60 ÷ テンポ

チャンネル数が 2 であった場合、データは L、R、L、R……(L : 左チャンネルのデータ、R : 右チャンネルのデータ)と記述されるため解析幅が 2 倍になる。WAVE ファイルが 1 秒の間に持つデータ数はサンプリングレート×チャンネル数分であるため、テンポ 60、4 分音符で採譜する場合に解析幅がサンプリングレート×チャンネル数になるよう解析幅を設定した。

またこの解析幅を用いて、ファイルのデータサイズ(byte)からデータが採譜間隔で設定した音符何個分の長さなのかを計算し、FFT を行う総回数を求めておく。なお、FFT を行う際に音の立ち上がりを考慮してデータ取得位置をある程度ずらしノイズカットを行うため、この分をデータサイズから引く必要がある。以下に式を示す。

FFT 総回数 =(データサイズ ÷ 2 − ノイズカットデータ数) ÷ 解析幅

一般的な WAVE 形式ファイルのデータは short 型(2byte)で構成されているのでデータサイズを 2 で割ることでデータ総数を求めている。また、解析幅が FFT に用いるデータ数 N を下回るとき、最後の FFT を行う際に要素が足りなくなる可能性があるため、FFT の総回数を 1 減らす。

3.2　FFT（高速フーリエ変換）

3.1 で求めた解析幅おきに FFT 総回数分 FFT を行う。データの開始位置の求め方を以下に示す。

データ開始位置 = 解析幅 × 現在の FFT 回数 + ノイズカットデータ数

FFT で数値データをフーリエ係数へと変換すると、使用したデータ数 N と同じ個数だけフーリエ係数が求まる。しかしフーリエ係数は中央の値を挟んで点対称的に値が並んでいるので本プログラムでは前半の N÷2 個のフーリエ係数のみ使用する。

このフーリエ係数は 0 [Hz] ～サンプリングレート÷2 [Hz] までの周波数の分布を N÷2 個の要素で表している。このため、1 つのフーリエ係数が表す周波数の幅は、

周波数幅 =（ サンプリングレート ÷ 2）÷（N ÷ 2）[Hz]

となる。音のピッチを判定するためにはサンプリングレート÷2 と N÷2 の値がなるべく近いほうが良いが、N を増やすと使用データ数が増え次の音符のデータを含んで精度が落ちることが予想されるので、サンプリングレートに対して N を適切な値に定めることが必要である。

なお次の基音検出でピークの検出を行うため、FFT を終えるごとにフーリエ変換の結果を配列へ保存することを必要回数分だけ繰り返す。

3.3 基音検出

基音とは、様々な周波数のサイン波から構成される音の中で基準になる大元のサイン波のことをいう[6]。この基音は、音を構成するサイン波の中で一番大きい成分であるというわけではない。音には基音を底とした整数倍音や非整数倍音が含まれており、その倍音成分のほうが基音より大きいこともある。このことを図 1（ピアノの周波数構成例）、図 2（トランペットの周波数構成例）を用いて説明する。

これらのグラフは FFT によって得られたフーリエ係数（振幅成分）をグラフ化したものである。ピアノは周波数が最も低いピークが一番大きいが、トランペットはそうではない。しかし、双方の音は同じオクターブ 4 番目のド（C）の音である。以上を考慮し、本プログラムの基音検出ではしきい値を設定してある程度の大きさを持つ最も低いピーク周波数を基音とした。

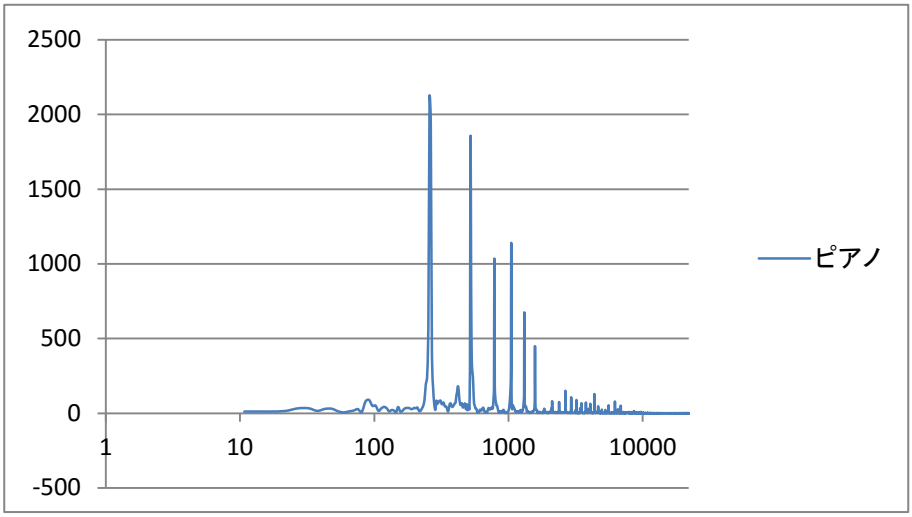

図 1. ピアノのド（C）の音の周波数構成例

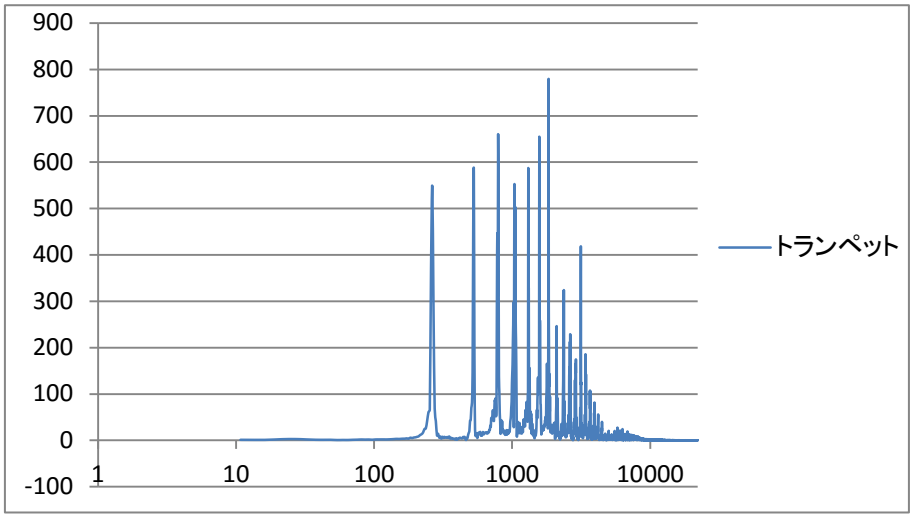

図 2. トランペットのド（C）の周波数構成

さらに基音検出を行った後、鳴っている音の状態を以下の 3 通りに分類し、次の音高解析に進むかどうかの判定を行う。

状態 1. 音が発せられた状態（発音）
状態 2. 音が前の時間から継続して鳴っている状態（継続音）
状態 3. 音がない状態（消滅）

状態 1 のときは音高解析へ進み、状態 2 または状態 3 の場合は音高解析以降の処理は行わずに次の時刻の FFT 処理へ移る。これらの状態を判別するために、条件式を 2 つ用いて処理を行った。図 3 はそのフローチャートである。

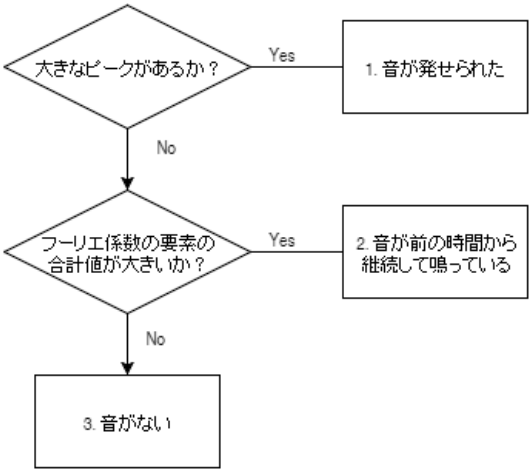

図 3. 音の状態を分類するフローチャート

3.4 音高解析

音高解析では、基音の周波数がどのオクターブで、十二音技法の何の音なのかを計算で求める。具体的には、基音の周波数が基準ピッチの何倍であるかを求めることで実装する。ここで基準ピッチとは、オクターブ 4 番目のラ（A）の周波数のことを指すが、曲によって基準となる周波数が異なることがあるため注意が必要である[7]。今回は国際標準ピッチである A = 440 [Hz]を使用した。

十二音技法では、一つのオクターブは 12 個の音に分けられ、各音の周波数は基準ピッチの周波数値から表 1 のように求めることができる。

基音の周波数値を乗算もしくは除算し 440 することで近づけていき、ある程度まで近づいた時点で計算を終了し、次の楽器判定に移る。なお今回はラの音の上下 1 音との中間点をしきい値とした。

表 1. 音の周波数の計算方法（等分平均律音階[8]）

オクターブ	音名	周波数 [Hz]
2	ラ	$440 \div 2^{24/12} = 110$
		……
3	ラ	$440 \div 2^{12/12} = 220$
		……
4	ソ	$440 \div 2^{2/12} = 392$
4	ソ♯	$440 \div 2^{1/12} = 415.3$
4	ラ	440
4	ラ♯	$440 \times 2^{1/12} = 466.16$
4	シ	$440 \times 2^{2/12} = 493.88$
		……
5	ラ	$440 \times 2^{12/12} = 880$
		……
6	ラ	$440 \times 2^{24/12} = 1760$

3.5 楽器判定

楽器判定はテンプレートマッチングを用いて行う。判定を行いたい楽器の音データを予め用意しておき、そのデータから数点選んで相関 R を求め、R の値が 1 に最も近い楽器を解析結果として採用する。今回はピアノとトランペットの二つの楽器にのみ対応しているため、違いがわかりやすい基音、第 2 倍音、第 3 倍音の 3 点から相関 R を求めた。

結果は各楽器の出力用配列へ格納する。なお、それぞれの楽器の同時発音可能数を考慮して、それぞれの時間ごとにピアノは 10 音まで、トランペットは 1 音までを出力用配列へ保存できるようにした。

出力用配列へ保存する値は以下の式で求める。

$$\text{出力値} = (\text{オクターブ} + 1) \times 12 + \text{ピッチクラス}$$

MIDI 形式はオクターブが-1 番目のド（C）から 9 番目のソ（G）まで扱うことができ、それぞれに 0～127 までの値が割り当てられている[10]。そのため出力値の計算ではオクターブの出力値を MIDI 形式に合わせるため 1 を足している。またピッチクラスは、ド（C）を 0 とした 0～11 までの値である[9]。

楽器判定を終えた後は FFT 結果の配列から該当する楽器の単音データを減算し、その後もまだ音があると判定されれば基音検出から楽器判定までを繰り返し行う。

3.6 ファイル出力

作成した出力用配列から MIDI 形式ファイルを作成する。今回はオープンソースである MIDIData ライブラリを用いる[5]。

楽器が複数ある場合はそれぞれにトラックを割り当てるようにした。図 4 はトラック分けされた MIDI 形式データを MIDI 作成・編集用フリーソフトウェア Domino で表示したものである。青がトラック 1、赤がトラック 2 に記述されたデータである。長方形のバーをノートイベントと呼ぶ。

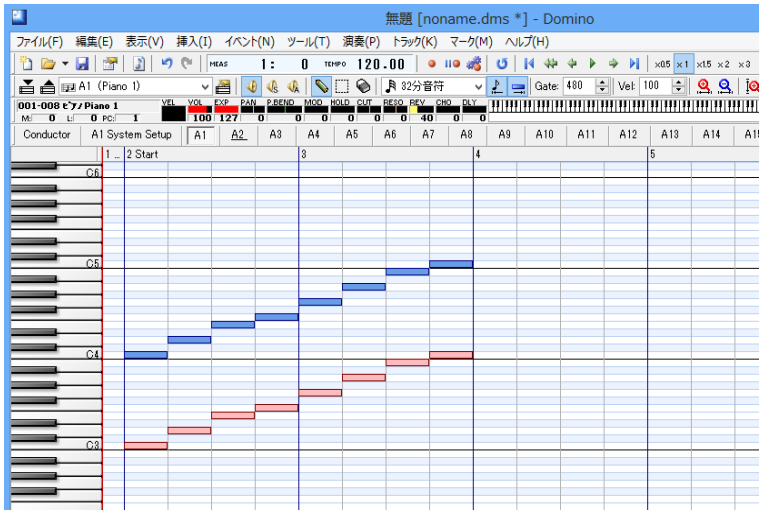

図 4. トラック分けされた MIDI 形式データを表示したピアノロール例

4 章 実験

実験は、Windows の初期内臓音源である Microsoft GS Wavetable Synth を使って MIDI 形式ファイルからオーディオファイル（WAVE 形式ファイル）を作成し、それをプログラムで解析するという手法で行った。また、楽器判定に用いる楽器の音データとしてオクターブ 4 の 12 音のフーリエ係数（振幅成分）を用意した。

MIDI 形式ファイルの作成、結果の確認には Domino を使用した。Domino の画面は図 4 に示された通りで、縦が音の高さ、横が時間を表している。ノートイベントが上にあるほど音程が高く、下にあるほど音程が低い。

4.1 楽器が一つの場合

4.1.1. サンプルデータの準備

楽器が一つの場合のテストに用いるサンプルデータには、ピアノ 3 種類、トランペット 1 種類を用意した。

① ピアノ オクターブ 4 の「ドレミファミレド」
② トランペット オクターブ 4 の「ドレミファミレド」
③ ピアノ オクターブの「ドレミファミレド」
④ ピアノ オクターブの「ドレミファミレド」

なお、採譜間隔は 16 分音符としたが、それぞれの音はテンポ 60 の 4 分音符の長さで作られている。例として図 5 に①のピアノロールを示す。

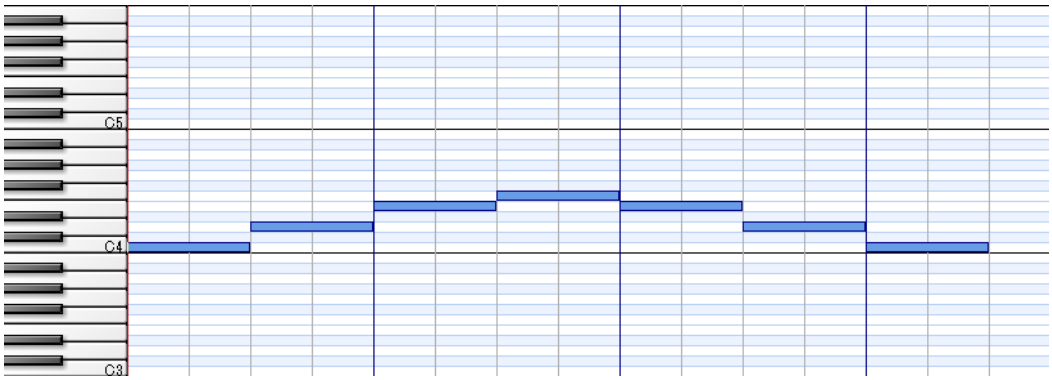

図 5. サンプルデータのピアノロール表示

4.1.2. 実験結果

実験結果を以下の表 2、図 6、7 に示す。

表 2. 楽器が一つの場合の実験結果

番号	1 音目	2 音目	3 音目	4 音目	5 音目	6 音目	7 音目
①	完全一致						
②	完全一致						
③	一致	一致	長さの不一致	検出なし	長さの不一致	検出なし	一致
④	一致	一致	一致	楽器誤判定	一致	一致	一致

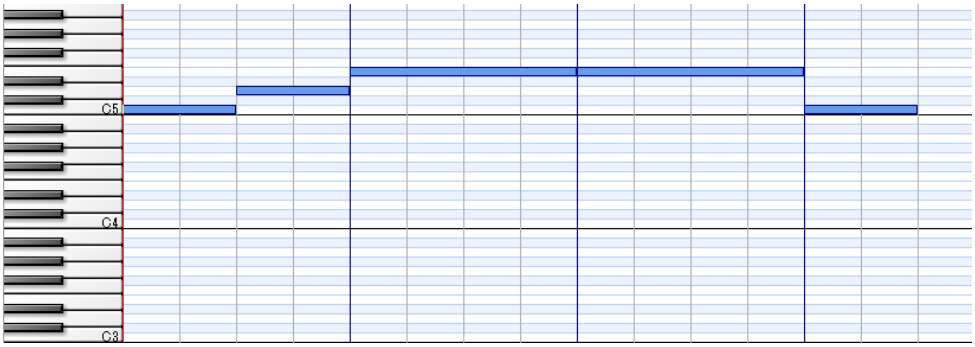

図 6. ③のテスト結果のピアノロール表示

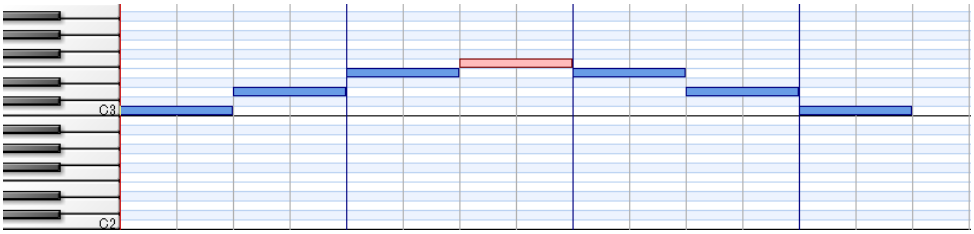

図 7. ④のテスト結果のピアノロール表示

①、②のテストではサンプルデータに完全一致した結果が得られた。しかし③では 2 箇所の音が検出されず、前の音が続いていると解析された（図 6）。さらに④では 4 箇所目の音がピアノではなくトランペットだと判定された（図 7）。

4.2 楽器が二つの場合

4.2.1 サンプルデータの準備

楽器が 2 つの場合の実験に用いるサンプルデータには、2 つの楽器が違う音を発している場合、同じ音を発している場合の 2 種類を用意した。

① ピアノ オクターブ 4 の「ドレミファミレド」
　トランペット オクターブ 4 の「ミファソラソファミ」
② ピアノ・トランペット オクターブ 4 の「ドレミファミレド」

4.2.2 実験結果

実験結果を以下の表 3、図 8、9、10 に示す。

表 3. 楽器が 2 つの場合の実験結果

番号	1 音目	2 音目	3 音目	4 音目
①ピアノ	一致	長さの不一致	一致	一致
①トランペット	一致	長さの不一致 発音検出位置の不一致	一致	一致
②ピアノ	一致	一致	検出なし	一致
②トランペット	検出なし	一致	一致	検出なし

番号	5 音目	6 音目	7 音目
①ピアノ	一致	長さの不一致	一致
①トランペット	一致	長さの不一致 発音検出位置の不一致	一致
②ピアノ	一致	検出なし	一致
②トランペット	検出なし	長さの不一致 発音検出位置の不一致	検出なし

図 8. ①の実験結果のピアノロール表示

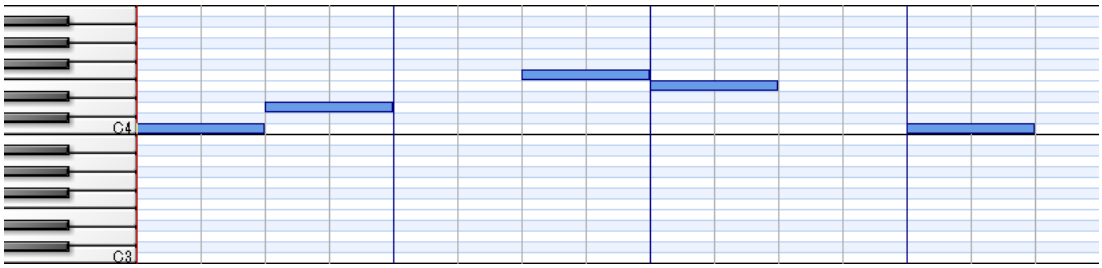

図 9. ②の実験結果(ピアノ)のピアノロール表示

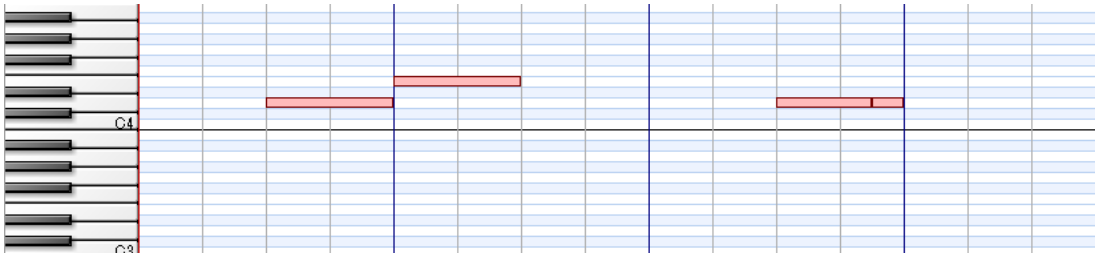

図 10. ②の実験結果(トランペット)のピアノロール表示

①の実験では、一致する箇所も多かったが、ピアノ・トランペット共に 2 音目、6 音目で発音の誤検出があった（図 8)。しかし音程の誤検出は無く、概ね正しい検出ができたと言える。

また②の実験では、ピアノとトランペットが同時に検出されているのは 2 音目のみであった（図 9、10)。検出がされていない箇所も多々見られるが、片方に検出されていない音はもう片方の楽器の音として判定されて検出されている。

5 章 考察

実験の結果をもとに、3 つの観点から考察を行う。

5.1 発音・消滅時刻について

実験の結果から、音の発音・消滅時刻については誤検出が多かったと言える。7 音×4 つの実験×2 項目の計 56 項のテスト項目について、15 項目がサンプルデータと一致しなかった。

楽器が 1 つの場合のテストでは、オクターブが高い場合にだけ誤検出があった。原因としては音が移り変わる際に十分なピークが検出されなかったことが挙げられる。これはピークかそうでないかを決定するしきい値が高めだったこと、ノイズカットの要素数が多く、音の立ち上がり部分の多くがカットされてしまったことが要因として考えられる。

楽器が 2 つの場合のテストでは、①・②の両方でトランペットの継続音を発音と検出した結果が 3 箇所あった。トランペットは人が息を吹き込んで鳴らす楽器であり、音量の増減が激しく、誤検出がされやすいと考えられる。このためトランペットの発音・消滅処理にはただピーク検出をするのではなく、検出された音が前の時刻と同じものであれば音量差を比較し新規発音なのか継続音なのかを判定する処理を行うことが改善案として挙げられる。

また②では、2 つの楽器が鳴っているのにも関わらず、2 音目以外は片方の音しか検出されなかった。これは最初に解析された音の成分を減算する過程で本来の成分よりも大きな値を引いていることが予測される。適切な大きさを決定するにはピアノ・トランペット双方の周波数特性をより理解し、互いの成分が混ざり合う倍音成分以外の要素からそれぞれの大きさを決定することで実現できる可能性がある。

5.2 音高解析結果について

発音が検出された音の中でサンプルデータと異なる音高が解析されたものは無かった。よって、音高解析は成功していると言える。

5.3 楽器判定について

サンプルデータと異なる楽器が検出された実験は楽器が 1 つの場合の④である。同じ楽器内でもオクターブや音が違えば周波数構成も異なってくることがあるため、これに対応できるようオクターブの異なる音データを用意することで改善できる可能性がある。

6 章 まとめ

ピアノ・トランペットの 2 つの楽器のみの対応となったが、WAVE 形式ファイルの入力から MIDI 形式ファイルの出力までを自動で行う採譜プログラムを開発することができた。

今後は対応楽器を増やすこと、プログラムを GUI 化すること、任意に決定した変数を自由に変更可能としカスタマイズの幅を広げることを行う。

参考文献

[1] 自動採譜のための、多重音高解析,
http://www.isc.meiji.ac.jp/~iclab/research/musictrans/musictrans.htm/

[2] 槻ノ木流の「BB っと WORLD」
http://bb.watch.impress.co.jp/cda/bbword/16386.html

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

[4] シェーンベルクの 12 音技法についての概要 1
http://www11.plala.or.jp/komposition/musik01.html

[5] フリーでオープンソースの MIDI データ作成・編集用ライブラリ『MIDIData ライブラリ』
http://openmidiproject.osdn.jp/MIDIDataLibrary.html

[6] シンセサイザー研究室 - 極めろ！シンセ音色研究所,
https://synth-voice.sakura.ne.jp/synth-voice/html5/column06.html

[7] チューニングピッチの変遷 – ムジカアンティカ湘南
http://www.coastaltrading.biz/img/%E3%83%81%E3%83%A5%E3%83%BC%E3%83%8B%E3%83%B3%E3%82%B0%E3%83%94%E3%83%83%E3%83%81%E3%81%AE%E5%A4%89%E9%81%B7.pdf

[8] H・F・オルソン 著　工博 平岡正徳 訳, 音楽工学, p.41-p.53

[9] ジャズメンのためのピッチクラス/ピッチクラスセット/フォートクラス早見表 | Music Theory Workshop Japan
http://neralt.com/pitch-class-set-forte-class/

[10] MIDI の学習 2-1
http://www1.plala.or.jp/yuuto/midi/p0201.html