左右位置と打楽器音分離を用いた音楽データのパート分離

2013SE196 杉浦 洋平　2013SE250 山本 佳広
指導教員 後藤 邦夫

1　はじめに

私達は楽器の演奏が趣味で原曲は見つけることができても，様々な楽器音が含まれているため，練習する楽器音だけを聞き取ることが難しく効率が低下していた．

本研究では，2014 年度の先行研究 [3] を参考としたフーリエ変換やメディアンフィルタ，ステレオフィールド内で panning により音を分離 [1] し，周波数を分析し，多楽器で演奏された楽曲から Percussive/Harmonic 音に分離 [2] する．そして抽出した音源から左右位置を判定する．また，マルチプラットフォームにするために linux を使い，Java でプログラム作成にあたる．Windows, Mac OS X で動作することを目標とする．GUI で操作し，再生しながらリアルタイムでフィルタを変更，ファイルを保存，スペクトログラムを目で見ながらフィルタ設定する機能やフィルタが自動で作れることを目標とする．実験準備として音の特性を解析するためにドラムセットの単音を同じ録音環境下で録音し，単音一つ一つの周波数範囲，左右位置を Audacity を使い解析する．

実験では，プリセットフィルタ，マニュアルフィルタ，自動フィルタを使い，取り出したい楽器の単音をどれだけ正確に取り出すことができ，取り出した楽器の単音それぞれを再合成し，元の音楽にどれだけ近づいているかで評価する．予想される成果として各楽器の単音の分離，自動で指定した楽器音を分離できることが挙げられる．音声分離についてのプログラムの作成は杉浦，GUI 作成については山本が担当する．

2　システムの概要

本研究では，Pure Java でプログラム作成した．プログラムを実行すると GUI で 2 つのウィンドウが表示される．

Filtered Player
: 音声の出力先，音源の選択，再生速度の設定，音量の設定，再生，一時停止，音声のファイル保存の機能を実現する．

Advanced Filter Setting
: 縦軸に周波数 (0Hz から 28160Hz)，横軸に定位 (LR) を取る GUI を表示し，フィルタ設定を変更する，また設定したフィルタの保存/ロードする機能を実現する．ボタンの機能説明 (THROUGH:全ての音を流す，MUTE:音を消す，PERCUSSIVE:打楽器音を分離，HARMONIC:弦楽器音を分離)

3　STFT 係数の加工

STFT(short-time Fourier transform) は短時間フーリエ変換のことである．

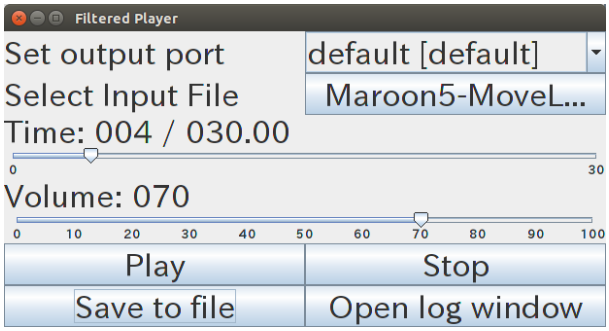

図 1　Filtered Player

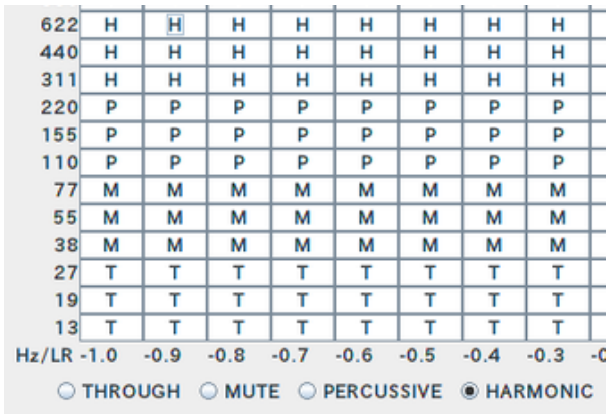

図 2　Advanced Filter Setting

本研究では，音声など時間変化する信号の周波数と位相を解析するために STFT を用いて，入力のサンプリング周波数が 44.1kHz で十分な周波数分解能力を確保するために window サイズを 4096 に設定する．窓関数には一般的な Hann window を用いる．

一般に音楽演奏には，20 m sec 以下の遅延が望ましいので，楽音の分析間隔も 20 m sec 以下が望ましい．Hann window を用いる場合，通常，window サイズの $\frac{1}{2}$ ずつシフトして変換し，逆変換後に和を取ると，元の信号に戻る．

音楽では，短時間の変化を捉える必要があるので，Window サイズの $\frac{1}{4}$(1024 サンプル、23 m sec) ずつシフトする．$\frac{1}{4}$ シフトの場合は，逆変換後に和を取るとサンプル値が 2 倍になるので，サンプル値に $\frac{1}{2}$ をかけて，出力する．

3.1　panning による音の分離

文献 [1] より panpot を用いて，左右の音声出力バランスの差に基づいて分離する．panpot の領域で取り出したい音の周波数が存在する左右位置を明らかにするため，ゲインの大きさを調整し位相を消す．その後，周波数の大きさを推定し，再合成をする．

• 分離方法

右の音を R(t)，左の音を L(t) とし，ある時刻における Window 内の N サンプルのフーリエ係数を $FR(k)$, $FL(k)$ とする．k の範囲は 1≦k≦N とする．ω_n^{kn} は $\omega_n^{kn} = e^{-j2\,\pi/n}$ とする．a，b は R(t) と L(t) の比率を考える．a，b の範囲は，0≦a≦1，0≦b≦1 とする．

このとき，右の音が大きい場合を $F\_R(k) = |FL(k) - aFR(k)|$ とし，左の音が大きい場合を $F\_L(k) = |FR(k) - bFL(k)|$ とする．

指定した位置に近い値を取り出すので，それぞれ式 (1)，(2) が最小になるときの a，b を求める．文献 [2] の離散の方法から改良された文献 [3] は，計算量か少なく，簡潔でわかりやすい二次関数の最大，最小問題に持ち込むことができる．$F\_R(k)$，$F\_L(k)$ を 2 乗した式の二次関数が下に凸な二次関数になり，最大，最小になるときの a，b を求めた [3]．

この二次関数の軸を d とおく．$FL(k) = x_L + iy_LR$，$FR(k) = x_R + iy_R$ とおく．また $absL = \sqrt{x_L^2 + y_L^2}$，$absR = \sqrt{x_R^2 + y_R^2}$，$absLR = \sqrt{(x_L - y_R)^2 + (y_L - y_R)^2}$ となる．ここで，FL(k)，FR(k) の実部 (内積) を dp，虚部を (外積) を cp とおくと，$dp = x_Lx_R + y_Ly_R$，$cp = x_Ry_L - x_Ly_R$ となる，$d = \frac{dp}{(absR)^2}$ と表すことができる．

右の音が大きい場合の最大値：max，最小値：min を定義し [3]，最大値から最小値を引いて，その時間での周波数の大きさを推定し，再び再合成する．

$$max - min = absLR - absL(d < 0) \quad (1)$$

$$max - min = absLR - \frac{abscp}{absR} \quad (0 \le d < \frac{1}{2}) \quad (2)$$

$$max - min = absL - \frac{abscp}{absR} \quad (\frac{1}{2} \le d < 1) \quad (3)$$

$$max - min = absL - absLR \quad (1 \le d) \quad (4)$$

3.2 Percussive/Harmonic な音の分離

メディアンフィルタを用いてモノラル音声信号の harmonic な音と percussive な音を分離する．ここで言う，harmonic な音 (弦楽器) とは周波数範囲が広く長時間であることを指し，また percussive な音 (打楽器) とは周波数範囲が狭く短時間であるもののことを指す．

音声サンプルを STFT して，時間軸から周波数軸に変換する．次に変換された周波数の共役複素数をスペクトログラムに表し，harmonic 要素の集まり ($h_{k,\,i}$ と percussive 要素の集まり ($p_{k,\,i}$) にメディアンフィルタを掛ける．図 3 はメディアンフィルタを掛けた部分を図で表した．

メディアンフィルタの連続するフレームにおいて harmonic な音を高めるために percussive な音を抑制し，また反対に percussive な音を高めるために harmonic な音を抑制する．これは，元のスペクトログラムに適用するマスクを生成するために用いる．

本研究では，文献 [3] より中央値を 15 から 30 の値で変化させたが劇的な変化が見られなかったため，中央値を 17 個に設定している．

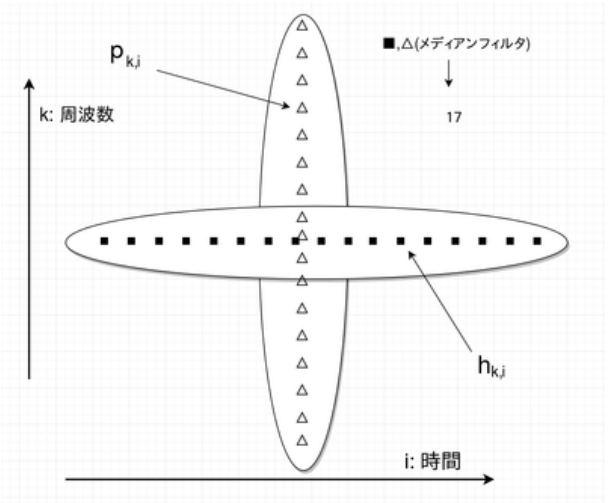

図 3 スペクトログラム

4 実現

表 1 使用したクラスの一覧

クラス一覧	主要機能 (コードの行数)
FilteredPlayerApp	メイン GUI (344)
FilteredPlayer	音声処理，再生，ファイル書き出し (250)
FilterGUI	ボタンで Pan，Freq 範囲を選択，フィルタの保存，読み出し (347)
GetPortsInfo	サウンド入力，出力ポートのリストを読む (69)
Utils	byte，float 変換，mp3 から wav ファイルへ変換 (79)
FFTChunk	データ定義，外部で作成した配列の受け取り (89)
FFTforward	順方向 FFT の計算 (89)
IFFToverlapAdd	FFT 逆変換とシフトしたものを再合成 (80)
BufferedEffector	イフェクタのバッファ処理のための抽象クラス (41)
PanCalculator	Pan 計算 (FFTChunk に代入) (100)
PercCalculator	左右の音の percussive/harmonic 値を独立に計算 (159)

ここからは，この Java プログラムの概要や処理，工夫した要点を説明する．

4.1 クラスの概要

プログラムを実行し，GUI について説明する．

panel

panel. setLayout(new GridLayout(2,2)); で (縦，横)=(2,2) のパネルを作成．JLabel で Set output port を追加．JComboBox で Set output Ports の項目名を表示．JLabel で select Input File を追加．JButton で File を追加．

panel2

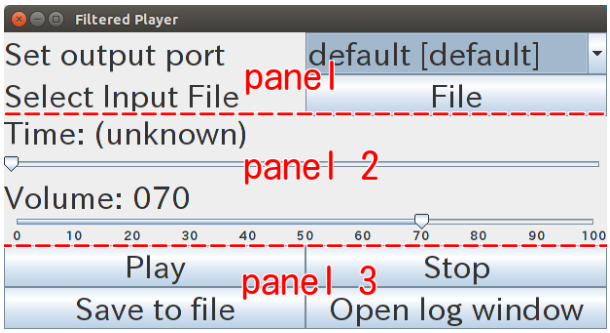

図 4 Filtered Player

panel2. setLayout(new GridLayout(6,1)); で (縦, 横)=(6,1) のパネルを作成. JLabel で Time, Volume を設定し, 同様に JLabel, JSlider を設定.

panel3

panel3. setLayout(new GridLayout(2,2)); で (縦, 横)=(2,2) のパネルを作成. panel3 に JButton で Play, Stop, Save to file, Opne log window を追加. これらの panel, panel2, panel3 を NORTH, CENTER, SOUTH に追加した. コンストラクタで処理を行い, Button の処理は, ActionEvent で行い, Slider の処理は, ChangeEvent で行った. TimeSlider, VolumeSlider は, if 文で場合分けし処理した.

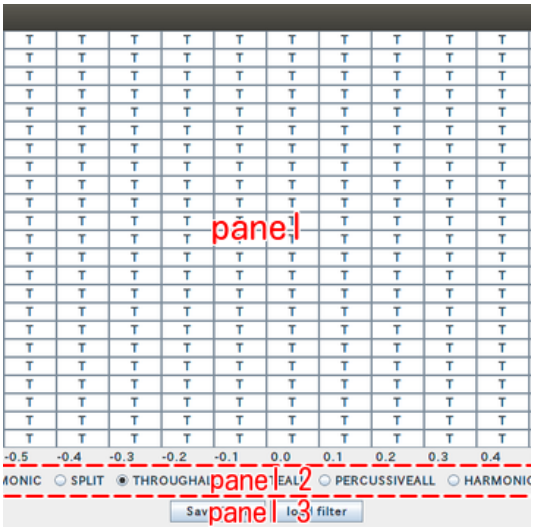

図 5 Filtered Player

panel

panel. setLayout(new GridLayout(y+1,x+1,0,0)); でパネルを作成. for 文の 2 重ループ内で処理し, button[p][f]. setText("T"); で左上から順に追加. JLabel で Hz/LR を追加.

panel2

JPanel で THROUGH, MUTE, PERCUSSIVE, HARMONIC, SPLIT, THROUGHALL, MUTEALL, HARMONICALL, PERCUSSIVEALL, SPLITALL, karaoke-Male, drumCover を追加.

panel3

panel3 に JButton で Save filter, load filter を追加.

また, 私達が実験を行う上でのデータ処理の流れを示す.

データ処理の流れ

1. ループはじめ
2. 少しずつサンプル読み込み, float に変換 (全部読んだら終了)
3. Window サイズ (4096) 単位で FFT (Hann window, 4096/4 単位でシフト)
4. FFT 係数から Pan 計算
5. FFT 係数から percussive/pan 計算
6. フィルタをかける
7. Window サイズ単位で IFFT, 1024 シフトなのでオーバーラップ加算 (float で出力)
8. byte に変換して, 出力装置に書く
9. ループ終わり

4.2 Java サウンド機能の利用

GetPortsInfo.java

Java VM で認識されるオーディオ入出力ポート情報を取得. public Mixer. Info[] get(Input/output)Ports().

Util. java

16bit LE と float の音声サンプル変換、逆変換. public static float[] LE16ToFloat. public static byte[] FloatToLE16. float の状態でゲイン調整. public static void adjustFloatGain. mp3 ファイルを wav ファイルに変換して出力. public static void mp3ToWaveFile.

4.3 STFT ライブラリを用いた変換と逆変換

FFTforward. java

順方向 FFT を行い, window/4 ずつシフト, Han window かけて FFT.

入力 putSamples(float[]).

出力 public FFTChunk getFFTChunk() (出力なければ return null).

BufferedEffector. java

イフェクタのバッファ処理を行う.

(input/output)Buffer = new ArrayList<Float>().

public int putSamples(float[] data, int len) – バッファに格納.

abstract protected void process(); – putSamples で実行.

abstract public float[] getSamples(); – ここで取り出される.

abstract public void flush(); – 最後にたまったものを出すように指示.

(flush() 後に getSamples()) – クラスで実装.

IFFToverlapAdd．java

FFT 逆変換とシフトしたものの再合成を行う.

入力 putFFTChunk(FFTChunk) (putSamples() は使わない)

出力 float[] getSamples()

処理内容 protected void process()

4.4 panning 分離

PanCalculator．java – Pan を計算して FFTChunk に代入

FFTChunk 単位処理でバッファリング不要

入力 putFFTChunk(FFTChunk chunk)

出力 FFTChunk getFFTChunk()

処理内容 protected void calcPan(FFTChunk)

4.5 Percussive/Harmonic な音の分離

PercCalculator．java

入力 putFFTChunk(FFTChunk)

出力 FFTChunk getFFTChunk(), flush()

処理 calcPMedian() – 周波数方向に指定 (17 が適当) した範囲の median

calcHMedian() – 時間方向に同上、さらに percussive の度合い (0. . 1) を FFTChunk に代入

5 実験

曲のジャンルに合わせてプリセットフィルタ，マニュアルフィルタを使い評価を行う.

5.1 サンプル音源の選択

表 2 サンプル音源一覧

ジャンル	アーティスト名，曲名
J-POP	Mr. Children，Fanfare
	宇多田ヒカル，First Love
Instrumental	Miles Davis，So what
	John Coltrane，Blue train
Male-vocal	Bruno Mars，Just the way you are
	Justin Bieber，WhatDoYouMean?
Female-vocal	Katy Perry，Firework
	Beyonce，halo

5.2 プリセットフィルタ結果の評価

既存のフィルタを選択するだけという簡単な操作で音の分離を行うことができる．しかし，左右位置の範囲指定が大まかなので他の楽器の音が入る，音が小さくなるなどの結果であった.

5.3 マニュアルフィルタ結果の評価

１つ１つのフィルタを押し分離を行っていくので操作に手間がかかるので，範囲指定しフィルタを変更できるようにするなど改善する．percussve 音のスネアを分離したい時にボーカルの音が入ってしまう．ボーカルの音を消そうとするとスネアの音が小さくなってしまうという結果になった．また，バスドラムの音も左右位置が近いことから被ってしまうので音は鮮明ではないが，なるべく指定する範囲を狭くする必要がある．しかしボーカル音の分離では，左右位置は真ん中で他の楽器よりも音が綺麗に取り出すことができた．実験結果を以下の URL に掲載する．URL:http://goto-lab.sc.nanzan-u.ac.jp/Thesis/index.html

5.4 自動フィルタ

実現するにあたって, 左右位置, 周波数ごとに一つの曲中のピークを記録し, pan freq Hz total percussive harmonic から特徴をつかみグループに分けられるかどうか確かめる．また目視により，左右位置，周波数範囲がまとまりそうか確認が必要なので gnuplot でグラフィック表示しグループを計算で算出する．本研究で実現することができなかった.

6 おわりに

本研究では,

- GUI で操作
- マルチプラットフォーム (Windows，Mac OS X) での動作
- 再生中にリアルタイムでフィルタを変更/保存機能
- 分離した wav ファイルを保存
- スペクトログラムを表示し目視でフィルタ設定機能

を実現した．自動フィルタは実現することができなかったため今後も取り組んでいきたい.

参考文献

[1] Barry, D., Coyle, E. and Lawlor, B.: Real-Time Sound Source Separation: Azimuth Discrimination and Resynthesis, *Audio Engineering Society Convention, San Francisco, 2004*, Vol. 117, pp. 1–7 (2004).

[2] FitzGerald, D.: Harmonic/Percussive Separation using Median Filtering, *International Conference on Digital Audio Effects*, Vol. 13, pp. 1–4 (2010).

[3] 倉内　慎，増田大輝：ステレオ多楽器音のドラムパートの抽出，南山大学情報理工学部 システム創成工学科 2014 年度 卒業論文 (2015).