

【事前準備】

- ・ MATLAB(<http://www.mathworks.co.jp/>)のインストール
- ・ processing(<http://processing.org/>)のインストール
- ・ 使用するwavファイルはAudacity(<http://www.forest.impress.co.jp/library/software/audacity/>)、wavesurfer(http://sourceforge.jp/projects/sfnet_wavesurfer/releases/)等でモノラルに変換しておく

プログラムその1：ビート時刻を自動で検出し、クロマベクトルのコサイン類似度を求める

【必要なプログラム】

- ・ wavreads.m
- ・ crmean.m
- ・ beattime.m
- ・ crm_cos_slide.m
- ・ Music Audio Tempo Estimation and Beat Tracking(<http://labrosa.ee.columbia.edu/projects/beattrack/>からダウンロードし、すべて「MATLAB」フォルダに入れる)
- ・ demoChromatoolbox(<http://www.mpi-inf.mpg.de/resources/MIR/chromatoolbox/>からダウンロードし、すべて「MATLAB」フォルダに入れる)

【プログラム実行準備】

1. demoChromaToolbox.mをfcp1.mに別名保存する。
2. audio_to_pitch_via_FB.mの240～245行目を削除する。
3. pitch_to_chroma.mの163～177行目を削除する。
4. fcp1.mの1～4行目を削除し、代わりに下記を書く。

```
function [fcp_1, y1] = fcp1(filename)
[y1, fs] = wavread(filename);
```
5. fcp1.mの16行目のf_CPをfcp_1に書き換える。
6. fcp1.mの18行目以下をすべて削除する。

```

1 ① clear
2    close all
3
4    filename = 'sample.wav';
5    [f_audio,sideinfo] = wav_to_audio('', 'data_WAV/', filename);
6    shiftFB = estimateTuning(f_audio);
7
8    paramPitch.winLenSTMSP = 4410;
9    paramPitch.shiftFB = shiftFB;
10   paramPitch.visualize = 1;
11   [f_pitch,sideinfo] = ...
12       audio_to_pitch_via_FB(f_audio,paramPitch,sideinfo);
13
14   paramCP.applyLogCompr = 0;
15   paramCP.visualize = 1;
16   paramCP.inputFeatureRate = sideinfo.pitch.featureRate;
17   ③ [f_CP,sideinfo] = pitch_to_chroma(f_pitch,paramCP,sideinfo);
18
19   paramCLP.applyLogCompr = 1;
20   paramCLP.factorLogCompr = 100;
21   paramCLP.visualize = 1;
22   paramCLP.inputFeatureRate = sideinfo.pitch.featureRate;
23   [f_CLP,sideinfo] = pitch_to_chroma(f_pitch,paramCLP,sideinfo);
24
25   paramCENS.winLenSmooth = 21;
26   paramCENS.downsampSmooth = 5;
27   paramCENS.visualize = 1;
28   paramCENS.inputFeatureRate = sideinfo.pitch.featureRate;
29   [f_CENS,sideinfo] = pitch_to_CENS(f_pitch,paramCENS,sideinfo);
30

```

←削除



```

1 ② function [fcp_1, y1] = fcp1(filename)
2
3
4    [y1, fs] = wavread(filename);
5    [f_audio,sideinfo] = wav_to_audio('', 'data_WAV/', filename);
6    shiftFB = estimateTuning(f_audio);
7
8    paramPitch.winLenSTMSP = 4410;
9    paramPitch.shiftFB = shiftFB;
10   paramPitch.visualize = 1;
11   [f_pitch,sideinfo] = ...
12       audio_to_pitch_via_FB(f_audio,paramPitch,sideinfo);
13
14   paramCP.applyLogCompr = 0;
15   paramCP.visualize = 1;
16   paramCP.inputFeatureRate = sideinfo.pitch.featureRate;
17   ③ [fcp_1,sideinfo] = pitch_to_chroma(f_pitch,paramCP,sideinfo);
18
19   ④ paramCLP.applyLogCompr = 1;
20     paramCLP.factorLogCompr = 100;
21     paramCLP.visualize = 1;
22     paramCLP.inputFeatureRate = sideinfo.pitch.featureRate;
23     [f_CLP,sideinfo] = pitch_to_chroma(f_pitch,paramCLP,sideinfo);
24
25   paramCENS.winLenSmooth = 21;
26   paramCENS.downsampSmooth = 5;
27   paramCENS.visualize = 1;
28   paramCENS.inputFeatureRate = sideinfo.pitch.featureRate;
29   [f_CENS,sideinfo] = pitch_to_CENS(f_pitch,paramCENS,sideinfo);
30
31   paramCPP.coeftsToKeep = [55:190];

```

←書き足す

←書き換える

←18行目以下
すべて削除

7. マッシュアップで使用する伴奏曲をwavreads.mの11行目に入力する。

```

1  function [fileName, wavData] = wavreads
2
3     files = dir('*.wav');
4
5     for n = 1 : length(files)
6         [path, name, ext] = fileparts(files(n).name);
7         fileName(n, :) = cellstr(name);
8         disp(name);
9     end
10
11     [fcp_1, y1] = fcp1('sample.wav'); ←伴奏曲に使用する
12     fs = 44100;                        wavファイル名
13     b1 = beat(y1, fs);
14     [c1, ch, c2] = crmean(b1, fcp_1, y1, fs);
15
16     for m = 1 : length(files)
17         [path, name, ext] = fileparts(files(m).name);
18         [y, fs, bits] = wavread(files(m).name);
19         [fcp_2, y2] = fcp1(files(m).name);
20         b2 = beat(y2, fs);
21
22         [d1, dh, d2] = crmean(b2, fcp_2, y2, fs);
23         [X, Y, Z] = crm_cos_slide(c1, d1);
24         wavData(m, 1) = X; %類似度
25         wavData(m, 2) = Z; %キーのスライド数
26         wavData(m, 3) = size(d1, 2); %ビート数
27     end

```

8. マッシュアップで使用するすべての曲を「MATLAB」と「data_WAV」フォルダ両方に入れる。

【プログラム実行方法】

`[fileName, wavData] = wavreads`

上記を実行すると、

- filenameにメロディで使われた曲名のリストを出力する。
- wavDataの1列目に伴奏曲とメロディ曲の1ビート毎のクロマベクトルにおけるコサイン類似度を出力する。
- wavDataの2列目にメロディ曲のキーの上げ数を出力する。
- wavDataの3列目にメロディ曲の拍数を出力する。

プログラムその2：ビート時刻を手動で求め、クロマベクトルのコサイン類似度を求める
※こちらはプログラムその1で正しくないビート時刻が出てしまったとき用です。

【必要なプログラム】

〈processing〉

- cmx
- tapping

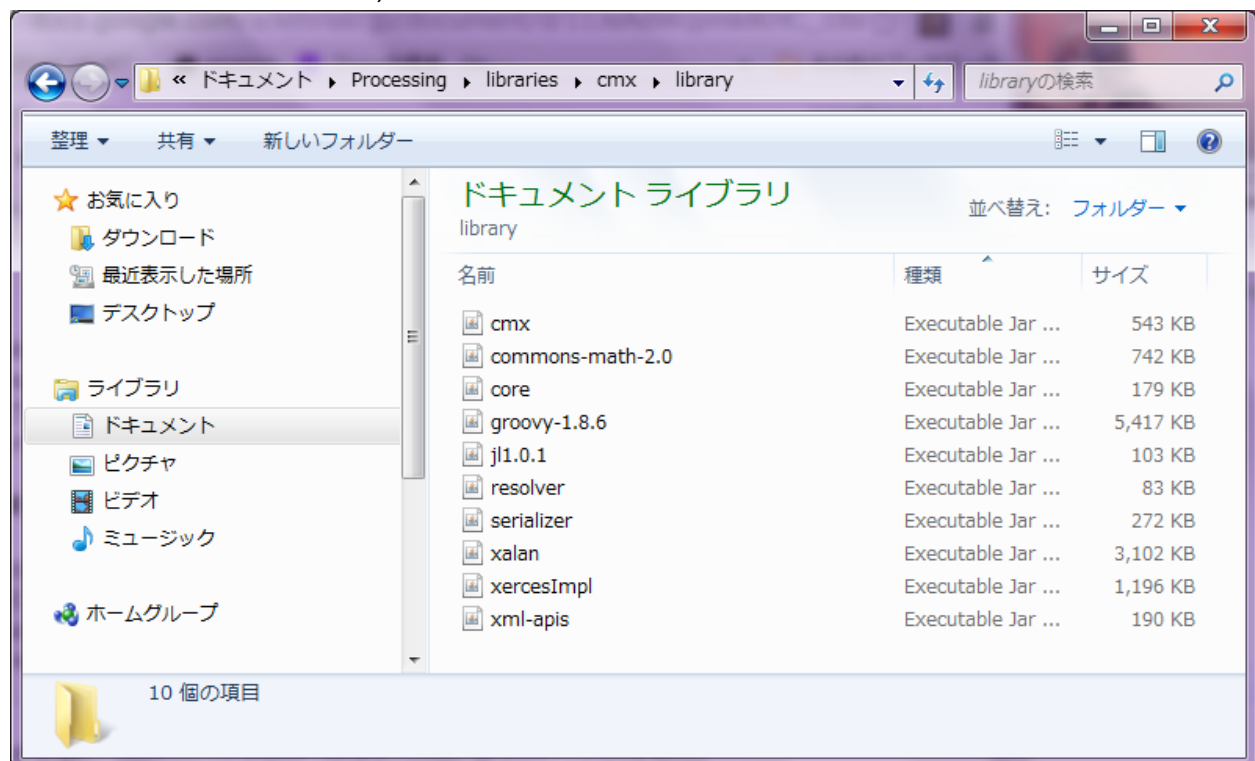
〈Matlab〉

- crmean.m
- beattime.m
- crm_cos_slide.m
- new_hikaku.m
- demoChromatoolbox(<http://www.mpi-inf.mpg.de/resources/MIR/chromatoolbox/>からダウンロードし、プログラムをすべて「MATLAB」フォルダに入れる)
- プログラムその1で作成したfcp1.m

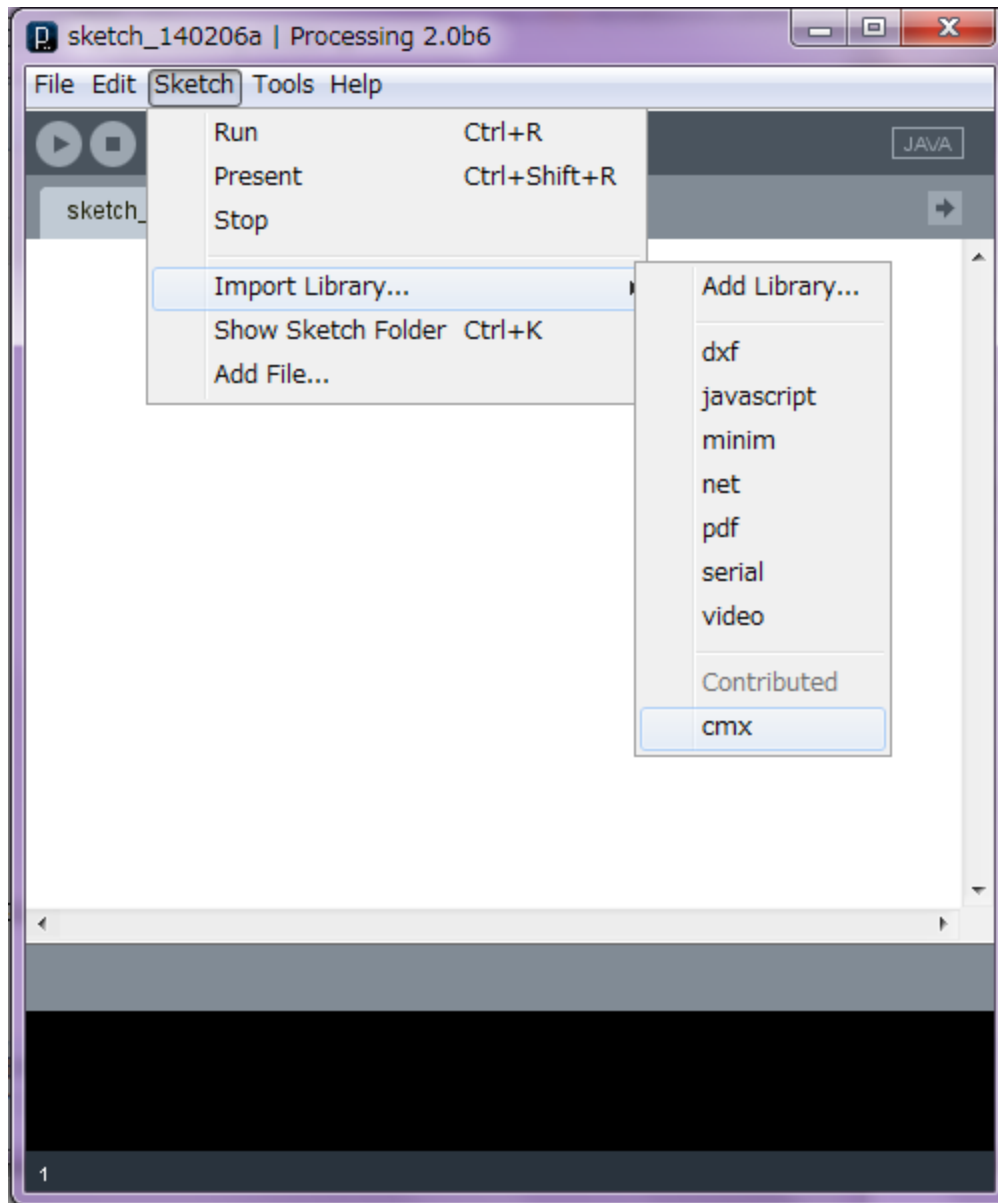
【プログラム実行準備】

〈processing〉

1. cmx-0.62.zipを解凍する。
2. cmx.jarとlibフォルダ内にあるjarファイル9個を、ドキュメント
>processing>libraries>cmx>libraryフォルダの中に入れる。(cmxフォルダとlibraryフォルダは
確かないはずなので、作る)



3. processingを起動し、SketchタブのImport Libraryにcmxが表示されていれば成功。



4. processingでtappingプログラムを開き、6行目の"sample.wav"の部分で、ビートを手動で検出したいファイル名にする。



〈Matlab〉

new_hikaku.mの4, 5行目にコサイン類似度で比較したい曲名を入力する。

※4行目に伴奏曲、5行目にヴォーカル曲を入力

```
1  function X = new_hikaku(b1, b2)
2
3  -      fs = 44100;
4  -      [fcp_1, y1] = fcp1('sample2.wav'); %伴奏
5  -      [fcp_2, y2] = fcp1('sample3.wav'); %ヴォーカル
6
7  -      [c1, ch, c2] = crmean(b1, fcp_1, y1, fs);
8  -      [d1, dh, d2] = crmean(b2, fcp_2, y2, fs);
9
10 -      [x, y, z] = crm_cos_slide(c1, d1);
11 -      X = [x, z];
```

↓ ここ

【プログラム実行方法】

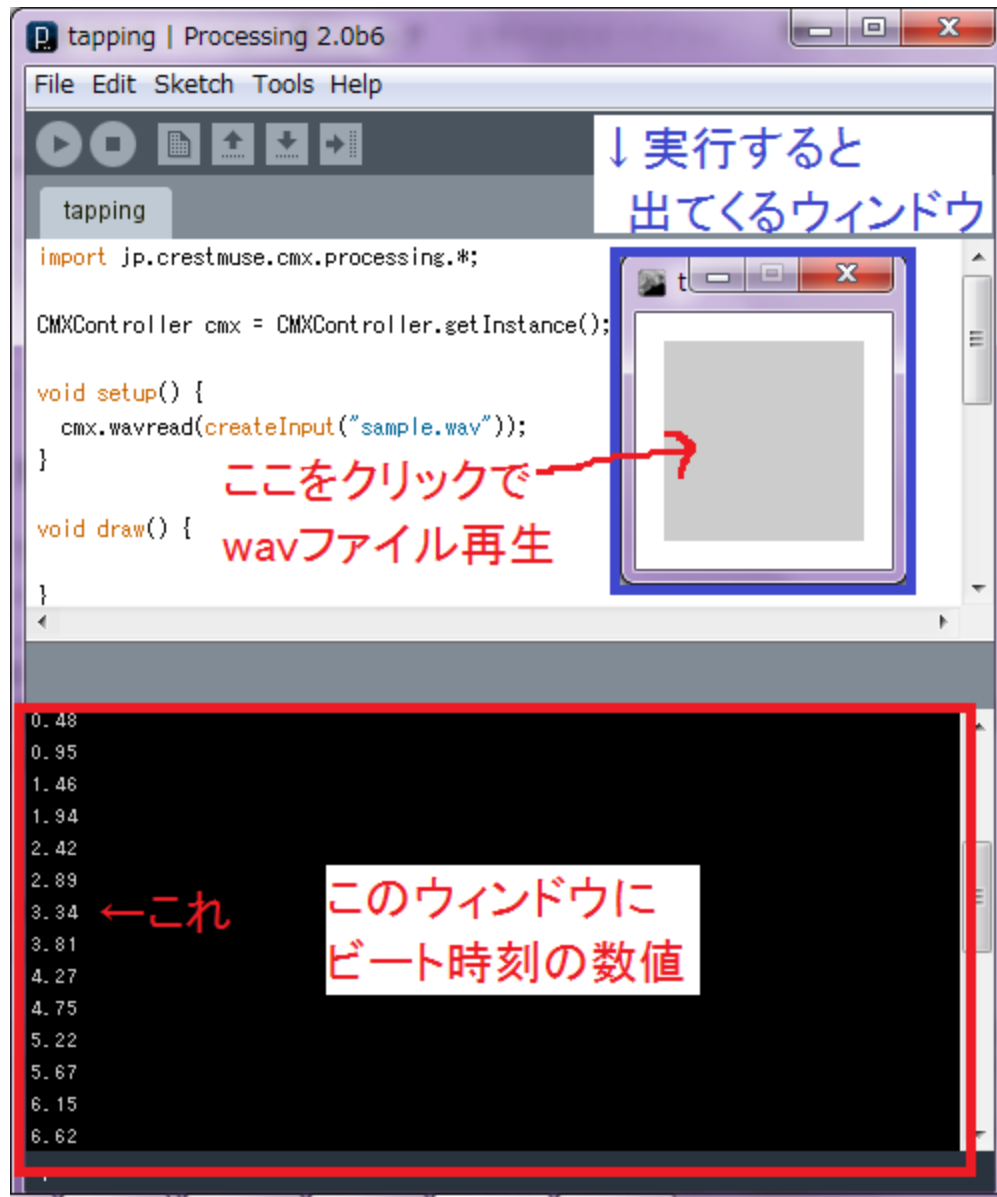
〈processing〉

まずtappingプログラムを実行する。

出てきた小さいウィンドウ内をクリックすると、wavファイルが再生されるので、ビートにあわせてスペースキーを押す。これがビート時刻となる。

ビート時刻の数値は、プログラム下のウィンドウ内に表示されるので、それをコピーして使用する。

注意：どうしても出だしはスペースキーを押すタイミングが遅れてしまうとおもうので、数値や数値の列数がビートと合うようにうまく調整してください。



〈Matlab〉

b1, b2にはそれぞれの曲のビート時刻を代入し、

`X = new_hikaku(b1, b2);`

上記を実行すると、

- Xの1列目に伴奏曲とメロディ曲の1ビート毎のクロマベクトルにおけるコサイン類似度を出力する。
- Xの2列目にメロディ曲のキーの上げ数を出力する。