ボイスチェンジャーについて

2020 年度卒業レポート

木村友祐

2021 年 2 月 24 日

目 次

1 イントロ、ボイスチェンジャーの仕組みについて

私はぼんやりと卒業研究では音に関することをしたいなと思っていたのだが、そうすることになって先生からもらった本の中で一番興味のある分野がこのボイスチェンジャーについてだったので、これを取り扱うことにした。

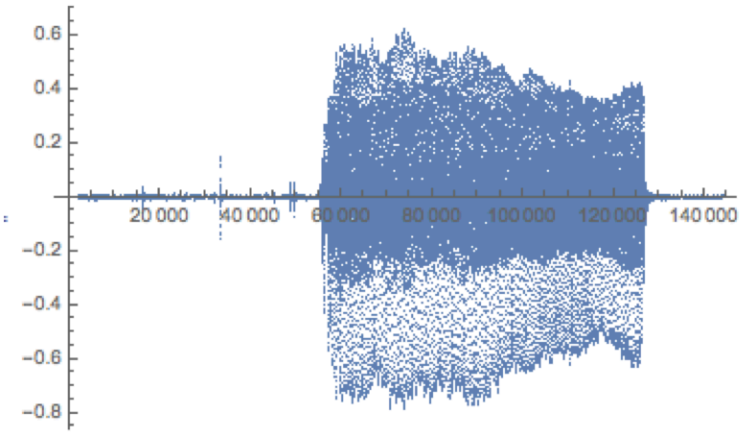
ここでボイスチェンジャーとは何か一度説明しておく。ボイスチェンジャーとはその字のごとく、声を変えられるシステムである。ニュースなどでインタビュー

する方がだれかわからないようにするために声を変える時などに使われる。
　そのボイスチェンジャーには「イコライザー」や「フォルマント変換」などいくつかやり方があるが、今回は「リング変調」という仕組みを使ったボイスチェンジャーのプログラムをしてみた。

2　Mathematicaにより周波数を調べる方法

　ボイスチェンジャーで実際の音の信号がどう変わるかを解析するに当たって準備が必要なので、それを見ていく。
　『信号処理とフーリエ変換』という授業内でも説明されている Mathematica で周波数を調べるという方法で今回音の解析を行っていった。(桂田 [2])
　以下では、録音したファイルを Mathematica を使って周波数を調べる手段を述べている。

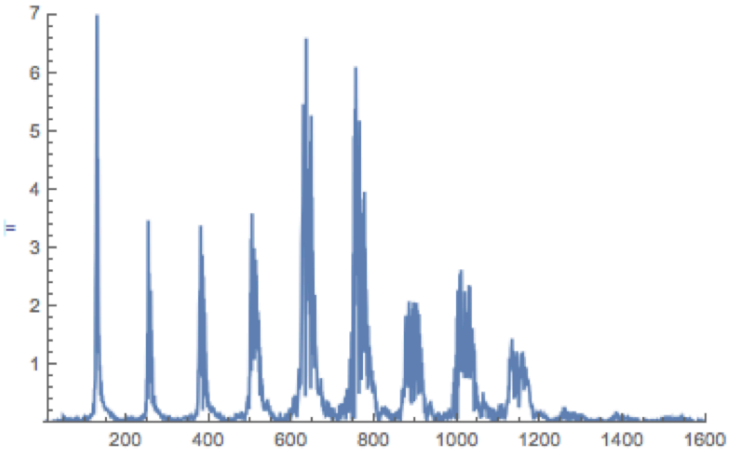
- ファイルを読み書きするには、作業フォルダを変更する必要がある。
  `SetDirectory[" /aoki/wavechange"]`
- “aaa.wav”のファイルを読み取る。
  (“aaa.wav”は、私があー、と普通に話す時の感じで発した音を録音したファイルである。)
  `(snd = Import["aaa.wav", "Sound"])`
- 読み込んだ信号のリストを得る。
  ```
  (tbl = snd[[1, 1, 1]];
  sr = snd[[1, 2]])
  ```
- sr から 3 秒分のデータを取り出してプロットしてみる。



- 音が鳴り始める 56000 番目あたりの音から 1 秒分取り出して、1600 個プロットした。

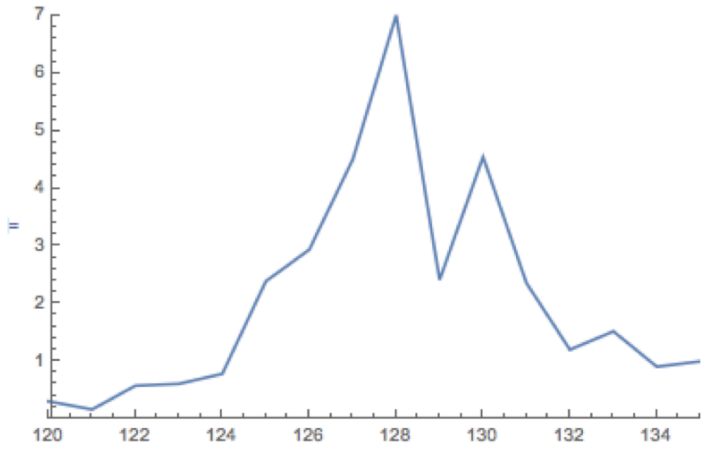
```
tb=Take[tbl,56000+1,56000+sr];
g=ListPlot[tb,Joined->True, PlotRange->{1,1600},-0.5,0.5]
```

- 音が鳴り出してから 1 秒間の信号を離散フーリエ変換して、1600 個の c を見る。
(離散フーリエ変換について:
x を周期 T の関数とする時、N $\in \mathbb{N}$ を固定して、
$\omega = \omega_N = exp\frac{2\pi i}{N}$
更に、j$\in\mathbb{Z}$ に対して
t_j=j$fracTN$, x_j=$x(t_j)$
とおく。　x_j は周期 N の数列である。フーリエ係数 c_n を台形公式で近似したものを離散フーリエ係数 C_n とすると、
$C_n = \frac{1}{T}\sum_{j=1}^{N-1} x(t_j)exp(\frac{-2\pi in}{T}t_j)\frac{T}{N}$
$= \frac{1}{N}\sum_{j=0}^{N-1} \omega^{(}-jn)x_j$
が成り立つ。C_n も n について周期 N の数列である。
写像 $x_j \to C_n$ を離散フーリエ変換という。



)(参考:松山 [3])

- ピークが 120-135 で出ているのでさらに範囲を絞る。



この時は、自分のあーと発した声のピークは約 127Hz であると言える。(Mathematica の Fourier 関数は最初のデータを 1 ではなく 0 から数えることから 1 ずれることを考慮している。)

3　リング変調

3.1　リング変調とは

リング変調とは、どんな仕組みなのか。

リング変調とは、音データにサイン波を掛け合わせて新しい信号を得る仕組みのことである。　例えば、音データが振幅 a_c、周波数 f_c のサイン波の場合、振幅 a_m、周波数 f_m のサイン波をかけ合わせると次のようになる。

$$s(n) = a_c \sin\left(\frac{2\pi f_c n}{f_s}\right) a_m \sin\left(\frac{2\pi f_m n}{f_s}\right) \quad (0 \leq n \leq N-1) \quad ...(1)$$

そして、この式は次のように展開することができる。

$$s(n) = \frac{a_c a_m}{2} \cos(\frac{2\pi(f_c + f_m)n}{f_s}) + \frac{a_c a_m}{2} \cos(\frac{2\pi(f_c - f_m)n}{f_s}) \quad (0 \leq n \leq N-1)$$

これはすなわち、周波数 f_c と周波数 f_m のサイン波から、周波数 $f_c + f_m$ と周波数 $|f_c - f_m|$ のコサイン波が生成されるということになります。
このように、音データにサイン波を掛け合わせることで周波数特性を変化させ、その結果として音色を変化させることがリング変調によるボイスチェンジャの原理になっています。

3.2　録音した音をリング変調してみる

このリング変調を使ったボイスチェンジのプログラムを実行して行くのだが、リアルタイム処理でのボイスチェンジの前にまずはあらかじめ録音しておいた音声ファイルをボイスチェンジして見て、その音の周波数特性などを見ていきたいと思います。

まず、録音をした wave ファイルをリング変調を用いてボイスチェンジするプログラムはこうなっている。(青木 [1] の第 1 章のプログラムを参考にして作った。)

また、青木 [1] で使われていた “`wave.h`” は Mac 環境では使えなかったので、修正した “`wave mk.h`” を使っている。この修正箇所については、付録として後に載せておく。

- ”wavechange.c” ソースコード

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wave_mk.h"

int main(void)
{
  MONO_PCM pcm0, pcm1;
  int n,fs;
  double am,fm;

  wave_read_16bit_mono(&pcm0, "aaa.wav"); /* 音データの入力 */

  pcm1.fs = pcm0.fs; /* 標本化周波数 */
  pcm1.bits = pcm0.bits; /* 量子化精度 */
  pcm1.length = pcm0.length; /* 音データの長さ */
  printf("length= %d\n",pcm1.length);
  pcm1.s = calloc(pcm1.length, sizeof(double)); /* 音データ */

  fs = pcm0.fs; /* 標本化周波数 */
  am = 1.0; /* 振幅 */
  fm = 300.0; /* 周波数 */


  for (n = 0; n < pcm1.length; n++)
  {
    pcm1.s[n] = am * sin(2.0 * M_PI * fm *  n / fs) *pcm0.s[n]; /* 音
データのコピー */
    printf("%f \n ",pcm1.s[n]);
  }

  wave_write_16bit_mono(&pcm1, "aaachange.wav"); /* 音データの出
力 */

  free(pcm0.s);
  free(pcm1.s);
```

```
    return 0;
}
```

このプログラム内の “aaa.wav”が、“aaachange.wav”では、テレビのニュースなどでプライバシーを保護する際にきくような音に変わっている。

このプログラムでは、(1) 式の a_m を 1、f_m を 300Hz に設定している。

このボイスチェンジの前と後でどのように音が変化したかを詳しく見るために、準備の段落で行なった手段に従い Mathematica を使って詳しく見て行くことにした。(参考:桂田 [2])

まず、”aaa.wav”での解析結果は、私があああと普通に発した音の周波数は約 127Hz になった。次に、”aaachange.wav”で同様の解析を行った時の波形については次のようになった。

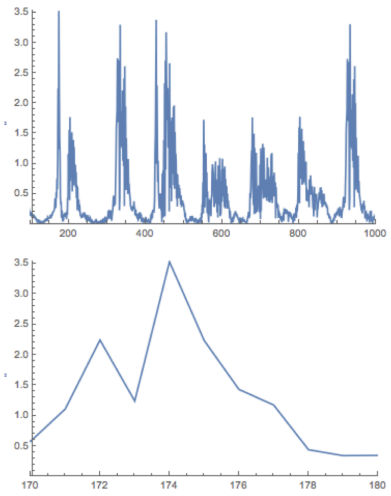


図 1: “aaachange.wav”

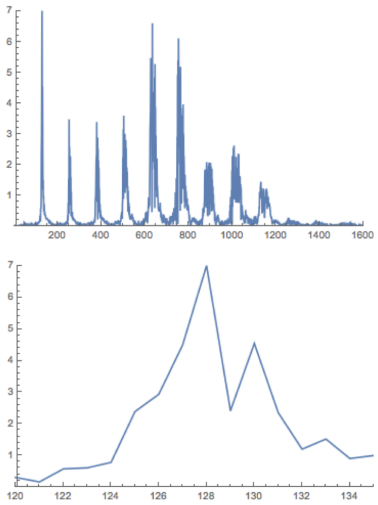


図 2: “aaa.wav”

解析結果としては、約 173Hz のところが一番高い波ではあるが、次に約 427Hz のところでも高い波が見られるというような結果になった。

これは、リング変調の仕組み通り、元の音の周波数 127Hz と、ボイスチェンジするためにかけあわせたサイン波の周波数 f_m300Hz を足したもの、つまり $f_c + f_m$ である 427Hz, そして $|f_c - f_m|$ である 173Hz という結果が出た。

次に、ギターの 5 弦の 2 フレットを押さえてドの音を出したものをボイスチェン

ジしたらどうなるかやって見た。
(“guitar5gen.wav”はギターの 5 弦の 2 フレットを押さえてドの音を出したものを録音した wave ファイル、“guitar5genchange.wav”はそれをリング変調でボイスチェンジした wave ファイル)

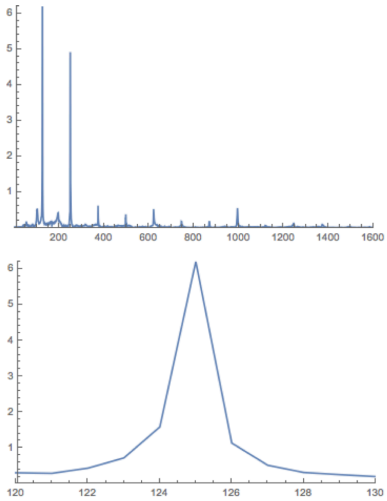


図 3: “guitar5gen.wav”

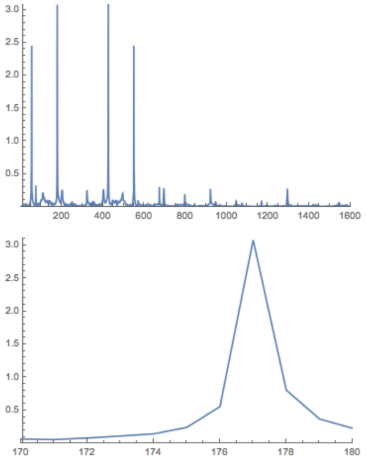


図 4: “guitar5genchange.wav”

この結果としては、“guitar5gen.wav”での音の周波数は約 124Hz になった。次に、“guitar5genchange.wav”での周波数は、約 176Hz と、約 424Hz のところで高い波が見られて、これもしっかりリング変調の仕組み通りサイン波の周波数同士を足したものと、引いたものの絶対値という結果が出た。また、この “guitar5gen.wav” と “guitar5genchange.wav” の結果の方が、とてもわかりやすく出ている。

これらの結果から、リング変調の仕組みがこのプログラムでは実行できていることと、混じり気の少ない音であればあるほどその結果はわかりやすいものになっていることがわかる。

3.3 ボイスチェンジされた音を元に戻す

リング変調でボイスチェンジされた音を元に戻すことはできるのかプログラムを使ってやってみた。そのプログラムは下記のものである。

- ソースコード

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wave_mk.h"

int main(void)
{
    MONO_PCM pcm0, pcm1;
    int n,fs;
    double am,fm;

    wave_read_16bit_mono(&pcm0, "aaachange.wav"); /* 音データの入
力 */

    pcm1.fs = pcm0.fs; /* 標本化周波数 */
    pcm1.bits = pcm0.bits; /* 量子化精度 */
    pcm1.length = pcm0.length; /* 音データの長さ */
    printf("length= %d\n",pcm1.length);
    pcm1.s = calloc(pcm1.length, sizeof(double)); /* 音データ */

    fs = pcm0.fs; /* 標本化周波数 */
    am = 1.0; /* 振幅 */
    fm = 300.0; /* 周波数 */



    for (n = 0; n < pcm1.length; n++)
    {
        pcm1.s[n] = pcm0.s[n]/(am * sin(2.0 * M_PI * fm *  n / fs)) ; /* 音
データのコピー */
        printf("%f \n ",pcm1.s[n]);
    }

    wave_write_16bit_mono(&pcm1, "motoaaa.wav"); /* 音データの出
力 */

    free(pcm0.s);
```

```
    free(pcm1.s);

    return 0;
}
```

このプログラムは、3.2 で使われた”aaachange.wav”にリング変調の際に掛け合わせた振幅 a_m, 周波数 f_m を割ることで、元の音に近いファイル”motoaaa.wav”を生成したものになっています。

この “motoaaa.wav”が元の音に近いものになっているかを、スペクトルを見て確認すると次のようにかなり近いものに戻っていることがわかります。

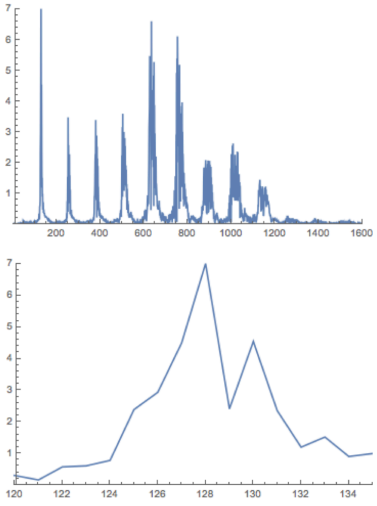


図 5: “aaa.wav”

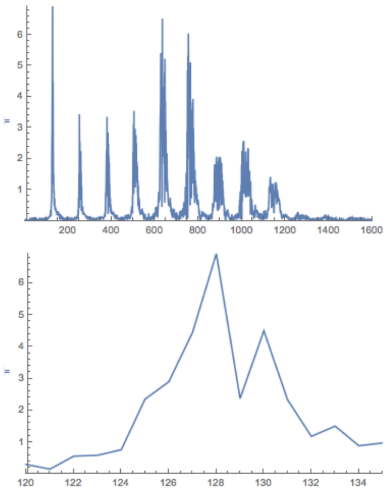


図 6: “motoaaa.wav”

3.4 2回ボイスチェンジすると？

では、ボイスチェンジしたものもう一度ボイスチェンジする、すなわち 2 回ボイスチェンジしたものはどういうスペクトルを描くのか見て見たいと思います。

スペクトルの前に、簡単にどのようにして 2 回ボイスチェンジしたかを説明しておくと、先述の”wavechange.c”に 1 度ボイスチェンジされた”guitar5genchange.wav”

を入力して、2 回ボイスチェンジされた wave ファイル”guitar5gennikai.wav”を生成するという流れである。

それでは、スペクトルを見比べてみよう。

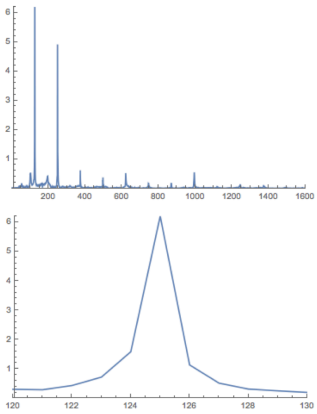


図 7: “guitar5gen.wav”

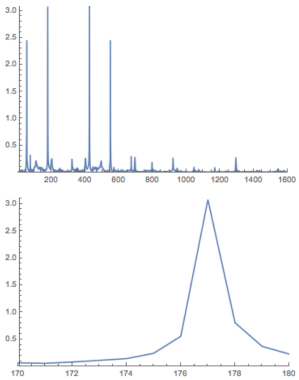


図 8: “guitar5genchange.wav”

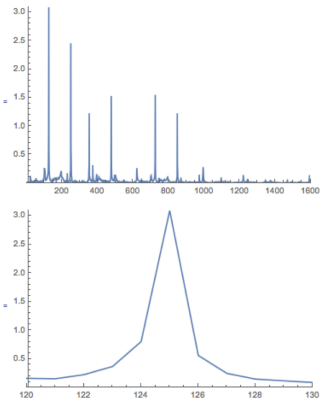


図 9: “guitar5gennikai.wav”

これを見ると、“guitar5gen.wav”と “guitar5gennikai.wav”はピークが約 124Hz でほとんど同じところになっている。これも、リング変調の式から考えると足し引きした周波数のコサイン波が出てくることから考えると妥当な結果になっているのではないかと思われる。

しかし、“guitar5gen.wav”と “guitar5gennikai.wav”の 2 つは同じような高さに聞こえるものの、元に戻そうとするプログラムで生成されたものほど同じ音には聞こえない。これはスペクトルから読み取れる限り、ピーク以外の部分がリング変調される際にいくつもできていることからそうなっているのではないかと読み取れる。

4 終わりに

様々な実験をする際に、ギターの 1 音をボイスチェンジしてみた時より人の声をボイスチェンジしてみた時の方がかなり音のスペクトルが複雑になっていたことから、ボイスチェンジを使えばプライバシー保護はかなりできているんだな、と感じた。　また、リング変調についての色々な実験を行いリング変調の仕組みはサイン波の周波数の足し引きでできている簡単な仕組みであることはわかったが、

よりこれについて実験をするのであれば位相が違うものを元に戻す場合のプログラムなどは組むことができなかったのでそういう実験や、リアルタイム処理のものについてももっと詳しい実験をしてみても面白そうだなと感じた。

5　付録

“wave mk.h”修正箇所
①文字コードを UTF に変換
② long riff chunk size; を long riff chunk size = 0; に,long data chunk size; を long data chunk size = 0; に変換
③ WAVE HEADER という型を追加
int check wave(WAVE HEADER *pcm, char *file name) という関数を追加
void wave read 16bit mono() を修正

6　参考文献

参考文献

[1] 青木直史, サウンドプログラミング入門-音響合成の基本と C 言語による実装-, 技術評論社 (2013).

[2] 桂田祐史, 信号処理とフーリエ変換　講義ノート,http://nalab.mind.meiji.ac.jp/~mk/fourier/fourier-lecture-notes.pdf

[3] 松山周五郎, 音の Fourier 解析 (明治大学理工学部数学科　卒業研究レポート) (2013).