理工学部 数物・電子情報系学科 情報工学 EP
秋学期講義「画像・音声情報処理」

2018 年 12 月 18 日
担当：教授　長尾智晴
総合研究棟 S401 室（内線 4131）nagao@ynu.ac.jp

「画像・音声情報処理」中間レポート４（最終）について

今回が平成 30 年度の 4 回目，かつ最後の中間レポートです．

■ レポートの目的と内容

- 目的：　wave 形式の（モノラル）音声データを加工するプログラムを作ろう！

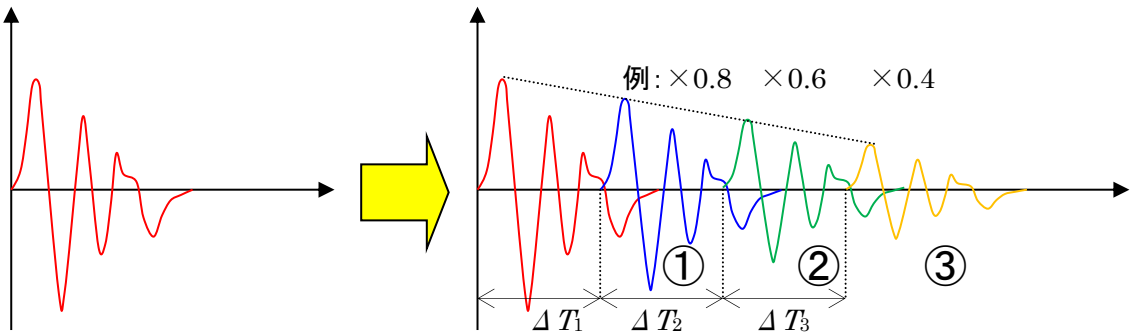
- 内容：

1. 原データについて：　各自で wave 形式の（モノラル）音声データを用意して下さい．フリー素材などを利用しても結構ですが，著作権などに注意し，本実験で用いるだけとし，外部には絶対に出さないで下さい．人の声の音声でも良いですし，短い音やメロディなど，人の声以外の音声データでも結構です．

2. 処理の内容について：　必須問題と応用問題があります．必須問題は必ずやって下さい．応用問題の作成・提出は任意としますが，応用問題を出した方が必須問題だけより若干高い評価にする予定です．

- 必須問題：　元の音声データ（org1.wav）にディレイをかけて，音が減衰しながら何度も繰り返されるような効果（“やまびこ”のような効果）を施した音声データ（out1.wav）を作るプログラム prog1.c を，先に紹介した wave ファイル入出力用ヘッダ wavelib.h を用いて作って下さい．ここでディレイとは，下図に示すように，元の音声データに対して振幅を減衰させ減衰音声信号を，時間をずらして加える処理を指すものとします（動画像処理のときにやった処理に似ています）．ただし，同図では最終的な音声信号（これらの音声信号の和）は示していません．図では元の音声信号に対して①～③の減衰音声信号を加えています，作成するプログラムでは以下をキーボードから指定できるようにして下さい．

    1. 加える減衰音声信号の回数（図では3回）．
    2. 減衰音声信号の間隔（図中の ΔT_i(i=1,2,…)）を等間隔にする場合はその間隔，次第に短くするならそのパラメータ（前回までの間隔に掛ける 0 以上 1 以下の実数など）．
    3. 図では振幅に対する減衰係数に等差減少関数を用いて一定値(=0.2)を減じていますが，等比減少関数（×0.8, ×(0.8^2), ×(0.8^3), …）も選べるようにして下さい．また，それらの際の公差および公比も指定できるようにして下さい．



元の音声信号　　ディレイ（エコー）をかけた音声の例

プログラムの作成手順：

1. 元の音声信号を読み込み，長さを5倍位に長くしてから，新しく増えた部分を無音（8bit なら 128，16bit なら 0）にする（このときファイル全体のサイズに関する RIFF サイズも更新することに注意）．
2. 元の音声信号に減衰率をかけた音声データを元の音声信号に加える．その際，上限と下限に注意すること（8bit なら 0～255，16bit なら−32768～32767．上限を超えたときは上限値，下限を超えたときは下限値にする）．同様の処理を指定回数だけ繰り返す．その際，音声信号の長さを超える部分にはデータを書かないこと．
3. 変換後の音声データをファイルに保存してからフリーソフトの SoundEngine で再生して効果を確認する．

➢ 応用問題： 元の音声データ（org2.wav）を入力すると，何らかの加工を施し，音声データ（out2.wav）として保存するプログラム prog2.c を wavelib.h を用いて作って下さい．ここでの加工方法・効果は任意としますが，できるだけ面白くてユニークなものを期待します．なお，周波数領域で変換する必要はなく，あくまでも空間領域，すなわち音声波形を直接加工するもので結構です．

3. レポートの提出： レポートは電子メールで提出して下さい．ソースプログラム（wavelib.h は添付不要），元の音声データ，加工後の音声データを添付して下さい（ファイルを1つのフォルダに入れて zip などで圧縮して1つのファイルにして添付しても結構です．ファイルが大きくなり過ぎる場合は「宅ふぁいる便」などの商用サービスを利用しても構いません．） 電子メールについては次のようにして下さい．

宛先 to： nagao@ynu.ac.jp	← 長尾のメールアドレス
Cc： 自分のメールアドレス	← 確認のため入れて下さい
件名：report4（1644*** 横浜太郎）	← （）の中に自分の学籍番号・氏名を入れて下さい
内容：	
学籍番号： 1644***	← 自分の学籍番号
氏名： 横浜太郎	← 自分の氏名
レポートについて：	
1）ソースプログラムについて：	← 数行以内で簡単に説明して下さい．
2）実験結果について：	← 数行以内で簡単に説明して下さい．
3）感想・意見など：	← 自由記述．

レポート提出期限：2019年 2月12日（火） 深夜24：00（＝13日の00：00）

本レポートについて何か不明な点や相談したいことがあれば，お気軽に
- 電子メール：nagao@ynu.ac.jp
- 電話：045-339-4131（学内からの内線電話の場合は 4131）
- 居室：総合研究棟 4 階 S401 室（秘書は S402 室）
- オフィスアワー：毎週火曜日～金曜日の 12:00～13:00（会議などを除く）

までご相談下さい．

注意！：実験にネット上の音声データを使う場合は著作権・版権その他の権利を侵害することがないように注意して下さい．例えば，実験で作った音声を勝手に自分のブログなどに掲載する行為等も違法になります．あくまでも本レポートのためだけに使用し，外部には絶対に出さないようお願いします．

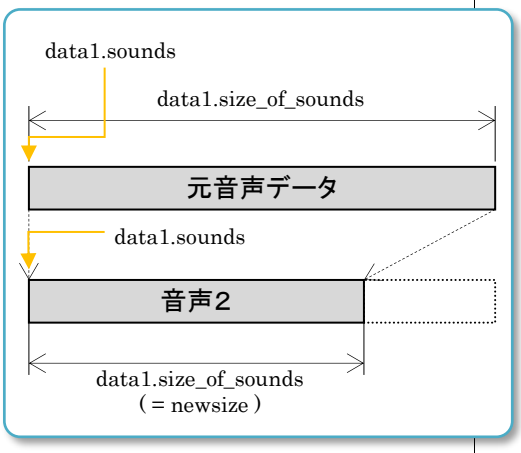
サンプルプログラムとその解説

元の音声データの再生速度を [0, 1] の範囲で短縮した音声を作るプログラムを次に示します.

```
/* このプログラムの名前：highspeed.c      */
/* 音声の再生速度を上げるプログラム */
#include<stdio.h>
#include"wavelib.h"

int main(void)
{
    /* RIFF チャンク用構造体変数 */
    struct RIFF RIFF1, RIFF2;
    /* fmt チャンク用構造体変数 */
    struct fmt   fmt1, fmt2;
    /* data チャンク用構造体変数 */
    struct data data1, data2;
    double alpha;   /* 短縮係数 */
    int newsize, i, n;   /* 作業用変数 */

    printf("音声の再生速度を上げる¥n");

    /* wave ファイルの読み込み */
    printf("元音声データを読み込む．¥n");
    load_wave_data( &RIFF1, &fmt1, &data1, "" );

    /* 音声２へのコピー */
    RIFF2 = RIFF1;        fmt2 = fmt1;        data2 = data1;

    /* 音声２のデータを詰める */
    do{
        printf("¥n 音声を時間軸方向に縮めるときの係数（ [0, 1] の実数）：");
        scanf("%lf",&alpha);
    }while( alpha <= 0.0 || alpha >= 1.0 );

    /* 音声２を作る */
    /* サイズの変更 */
    newsize = (int)( data2.size_of_sounds * alpha );
    /* RIFF チャンク全体の大きさも変わるので修正する */
    RIFF2.SIZE = RIFF2.SIZE - data2.size_of_sounds + newsize;
    data2.size_of_sounds = newsize;
    /* 元音声のデータを間引いてコピーする */
    for(i=0;i<data2.size_of_sounds;i++){
        /* 元の何番目のデータにするかを決める */
        n = (int)( data1.size_of_sounds * ( i / (double)newsize ) );
        /* データのコピー */
        *( data2.sounds + i ) = *( data1.sounds + n );
    }

    /* 音声２を wave ファイルとして保存 */
    printf("¥n 生成した音声を保存します．¥n");
    save_wave_data( &RIFF2, &fmt2, &data2, "" );

    return 0;
}
```

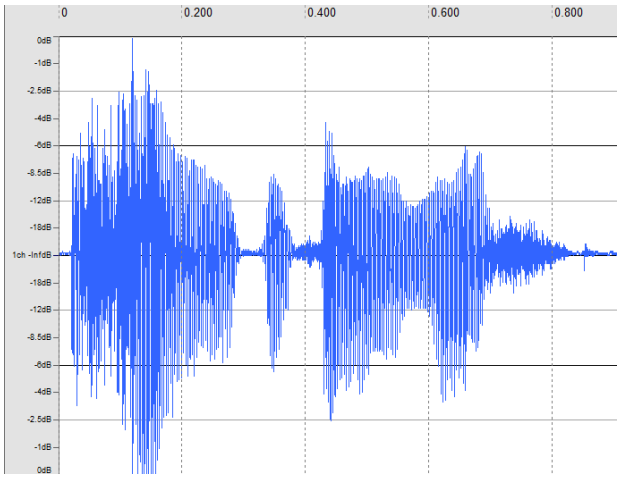
表1 WAVE ファイルのブロックによるデータ構造

長さ [byte]		内 容	意 味	チャンク
4		文字列“RIFF”	チャンク ID	RIFF チャンク
4	8	これ以降 RIFF チャンクサイズ (ファイルサイズ－8)	RIFF チャンクサイズ	
4		文字列“WAVE”	RIFF の種類	
4		文字列“fmt”＋英数空白	チャンク ID (フォーマット定義)	fmt チャンク
4		バイト数	fmt チャンクサイズ　↓16 進表記 リニア PCM なら 16 (10 00 00 00)	
2	16	フォーマットタイプ (詳細省略)	リニア PCM なら 1 (01, 00)	
2		チャンネル数	モノラル 1 (01 00), ステレオ 2 (02 00)	
4		サンプリングレート[Hz]	44.1kHz なら 44100 (44 AC 00 00)	
4		データ速度[byte/sec]	44.1kHz 16bit ステレオなら 44100×2×2=176400 (10 B1 02 00)	
2		ブロックサイズ [byte/sample×チャンネル数]	16 bit ステレオなら 2×2=4 (04 00)	
2		サンプルあたりのビット数 [bit/sample]	WAVE フォーマットでは 8 or 16 bit 16bit なら 16 (10 00)	
2		拡張部分のサイズ	リニア PCM の場合は存在しない	
n		拡張部分	リニア PCM の場合は存在しない	
4		文字列“data”	データチャンク (波形データ) ID	data チャンク
4		波形データのサイズ[byte]	データサイズ (=size)	
size		波形データ	8bit or 16bit PCM データが時間順に記録されている (ステレオの場合は LRLR….). 8bit: unsigned (0～255, 無音は 128) 16bit: signed (-32768～+32767, 無音は 0)	

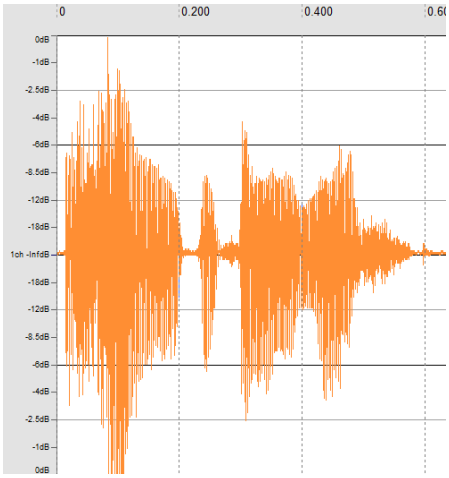


一見，難しそうですが，要領をつかめばそんなに難しくありません

● 元の音声データ波形（「おはようございます」）



● 加工後の音声データ波形



波形の概要は変わらず，再生時間が短くなっていることがわかる．

その他のプログラミング上のヒントについて

1．実数値をキーボードで読み込むには次のようにします．テストプログラムを次に示します．

```
/* double.c */
#include<stdio.h>

int main(void)
{
    double param;   /* 実数値用変数 */

    printf("input parameter(0-1):");      /* 入力を促す表示 */
    scanf("%lf",&param);                  /* パーセント・エル・エフの書式 */
                                          /* & を忘れないように */
    printf("parameter = %f",param);       /* 正しく入力されたか確認 */

    return 0;
}
```

実行例（キーボードから 0.5 を入力した場合）

```
input parameter(0-1):0.5
parameter = 0.500000
```

2．音割れを防ぐには？

音声波形を重ねて足した場合，結果的に大きな音になって「音割れ」が生じることがありますので，動画像で多数のフレームを重ねたときと同様に，音圧の上限値を指定して，それより大きくならないようにする必要があります．