多媒體

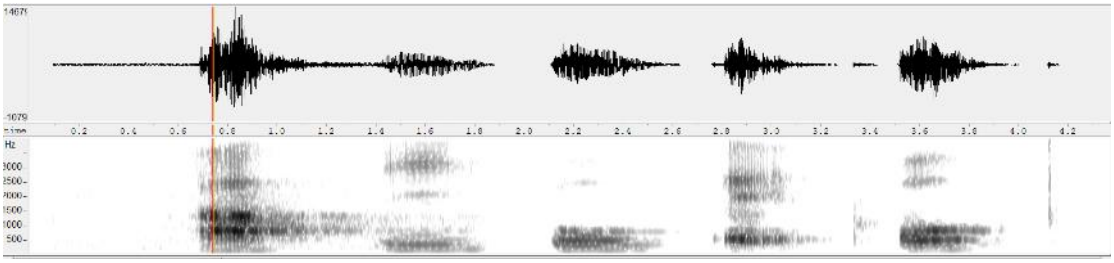
Mini project 2-1

姓名:張家瑋
學號:410886021

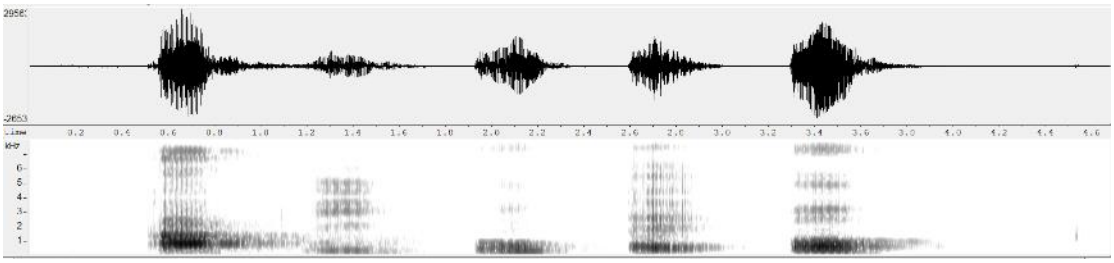
日期:2021/11/16

一、 Recording：

用採樣率 8000Hz 錄音成 vowel-8k.wav，spectrogram 圖示如下:



用採樣率 16000Hz 錄音成 vowel-16k.wav，spectrogram 圖示如下:



觀察可得兩者差異，16k 比 8k 來得更清晰，且 16k 的頻域範圍從[0,8000]，8k 的頻域範圍從[0,4000]。

二、 模擬程式：

首先，與第一次作業一樣在開頭宣告 wav 格式的標頭，如下圖:

```
#include <math.h>
#include <errno.h>
#include <time.h>

struct wavfile_header //設定wav檔格式
{
    char riff_tag[4];
    int riff_length;
    char wave_tag[4];
    char fmt_tag[4];
    int fmt_length;
    short audio_format;
    short num_channels;
    int sample_rate;
    int byte_rate;
    short block_align;
    short bits_per_sample;
    char data_tag[4];
    int data_length;
};
int i,n,k;
```

第二步要生成 cos 的 wav 檔，我利用函式 gen_cos()達成目的，如下:
首先宣告 wav 格式各參數如下圖:

```
void gen_cos(int fs, int f,char* filename){
    int T = 1;
    int SampleBit=16;
    int A=10000;

    struct wavfile_header header;
    strncpy(header.riff_tag, "RIFF", 4); //RIFF識別標誌
    strncpy(header.wave_tag, "WAVE", 4); //WAVE識別標誌
    strncpy(header.fmt_tag, "fmt ", 4);  //fmt識別標誌
    strncpy(header.data_tag, "data", 4); //資料標記符data

    header.riff_length = fs*(SampleBit/8)*T+36;//wav格式檔案長度
    header.fmt_length = 16;                    //過度位元組
    header.audio_format = 1;                   //格式類別
    header.num_channels = 1;                   //單聲道，設定1
    header.sample_rate = fs;                  //取樣率
    header.byte_rate = fs * (SampleBit / 8); //音頻傳送速率
    header.block_align = SampleBit / 8;        //一個樣本多聲道資料塊大小
    header.bits_per_sample = SampleBit;        //一樣本包含bit數
    header.data_length = fs*(SampleBit/8)*T;   //音頻長度
```

接下來生成 cosine 值，並存放入 file 檔案中，如下圖:

```
    FILE *file;
    file = fopen(filename,"wb+");           //以二進位制寫入的方式打開檔案
    fwrite(&header, sizeof(header),1, file);//將header寫入檔案

    short cosine;
    for( i=0 ; i < fs*T ; i++ ){
        cosine = floor(A*cos(2*M_PI*f*i/fs)+0.5);//生成cosine
        fwrite(&cosine,sizeof(short),1,file);
    }
    fclose(file);
};
```

再來便可自行輸入參數來產生 wav 檔，如下:

```
int main(int argc, char *argv[]){
    gen_cos(16000,50,"cos_050Hz-16k.wav");
    gen_cos(16000,220,"cos_220Hz-16k.wav");

    gen_cos(8000,50,"cos_050Hz-8k.wav");
    gen_cos(8000,220,"cos_220Hz-8k.wav");
```

再來我用 gen_spectrogram()函式做 framming、window function、DFT 後生成 spectrogram:

```
void gen_spectrogram(int A_WT,double A_WS,double DFT_WS,double FrameInterval,int fs,char filename[],char txtname[],int T){
    int N,P,m,M;
    P= A_WS*fs;             //P為做window func的樣本數
    N = DFT_WS*fs;          //做DFT的樣本數
    M = FrameInterval*fs;   //framing間隔樣本數
    m = (1/FrameInterval)*T;//Frame總數量

    FILE *wav;
    wav = fopen(filename,"rb+");//以二進位制讀取的方式打開檔案
```

其中我用 A_WT 數值來判定 window function 是 Set1 或是 Set2，A_WT 為 0 代表 Set1，A_WT 為 1 代表 Set2，程式碼部分如下圖:

```
    float *w;               //初始化window function w
    w =  calloc(N,sizeof(double));//calloc將w初始值預設為零
    if(A_WT==0){        //rectangular
        for(n=0;n<P;n++){
            w[n]=1;
        }
    }
    else if(A_WT==1){   //hamming
        for(n=0;n<P;n++){
            w[n]=0.54-0.46*cos(2*M_PI*n/(P-1));
        }
    }
```

為要生成 spectrogram，我用 DFT()函式將每一個 frame 裡的點做 DFT，並計算複雜度，再將 DFT 完的結果輸出到 txt 檔裡，如下圖:

```
void DFT(float* X,int num,int* add,int* multi,char* filename,FILE* fp){
    float Real=0,Image=0;//將e^(jwt)拆成cos(wt)-i*sin(wt)
    float Xn;

        for(k=0;k<num;k++){
            Real=0;
            Image=0;
            for(n=0;n<num;n++){
                Real += cos((2*M_PI/num)*n*k)*X[n];//X[n]*e^(jwt) = X[n]*(cos(wt)-i*sin(wt))
                Image -= sin((2*M_PI/num)*n*k)*X[n];
                *add+=2;
                *multi+=2;
            }
            Xn =20*log10(sqrt(Real * Real + Image * Image));

            *multi+=1;
            fprintf( fp, "%f ", Xn);
        }
}
```

最後即可輸出加法數與乘法數，完成檔案讀寫，可依照輸入參數產生 12 組 txt 檔，達成作業要求，如下:

```
        DFT(X,N,&add,&multiple,filename,txt);
        fprintf( txt, "\n" );
    }

    printf("%s   1.addition: %d   2.multiplication: %d\n",txtname,add,multiple);

    free(value);
    free(w);
    free(X);

    fclose(txt);
}
```

```
gen_spectrogram(0, 0.005, 0.008, 0.005, 16000, "cos_050Hz-16k.wav", "cos_050Hz-16k.{Set1}.txt",1);
gen_spectrogram(0, 0.005, 0.008, 0.005, 16000, "cos_220Hz-16k.wav", "cos_220Hz-16k.{Set1}.txt",1);
gen_spectrogram(0, 0.005, 0.008, 0.005, 16000, "vowel-16k.wav", "vowel-16k.{Set1}.txt",5);
```

三、 複雜度：

$$X(n\Delta_t, m\Delta_f) = \sum_{p=n-Q}^{n+Q} w((n-p)\Delta_t)x(p\Delta_t)e^{-j2\pi pm\Delta_t\Delta_f}\Delta_t$$

根據式子 ，假設在時域 t 軸有 T 個取樣點，在頻域 f 軸有 F 個取樣點，故總共要對 TF 個點做[2Q+1](即為本次實驗的 N)次運算，所以複雜度為 TF(N)，時間複雜度為 O(TFN)。下圖為各檔案計算出來的次數:

```
C:\Users\wesle\OneDrive\桌面\mini project 2>.\main
cos_050Hz-8k.{Set1}.txt   1.addition: 1638400   2.multiplication: 1664000
cos_220Hz-8k.{Set1}.txt   1.addition: 1638400   2.multiplication: 1664000
vowel-8k.{Set1}.txt   1.addition: 8192000   2.multiplication: 8320000
cos_050Hz-16k.{Set1}.txt   1.addition: 6553600   2.multiplication: 6604800
cos_220Hz-16k.{Set1}.txt   1.addition: 6553600   2.multiplication: 6604800
vowel-16k.{Set1}.txt   1.addition: 32768000   2.multiplication: 33024000
cos_050Hz-8k.{Set2}.txt   1.addition: 13107200   2.multiplication: 13158400
cos_220Hz-8k.{Set2}.txt   1.addition: 13107200   2.multiplication: 13158400
vowel-8k.{Set2}.txt   1.addition: 65536000   2.multiplication: 65792000
cos_050Hz-16k.{Set2}.txt   1.addition: 52428800   2.multiplication: 52531200
cos_220Hz-16k.{Set2}.txt   1.addition: 52428800   2.multiplication: 52531200
vowel-16k.{Set2}.txt   1.addition: 262144000   2.multiplication: 262656000
```

四、 Matlab：

先用 load()函式將各個 txt 檔匯入成矩陣:

```
A = load("cos_050Hz-8k.{Set1}.txt");
B = load("cos_050Hz-8k.{Set2}.txt");
C = load("cos_050Hz-16k.{Set1}.txt");
D = load("cos_050Hz-16k.{Set2}.txt");
E = load("cos_220Hz-8k.{Set1}.txt");
F = load("cos_220Hz-8k.{Set2}.txt");
G = load("cos_220Hz-16k.{Set1}.txt");
H = load("cos_220Hz-16k.{Set2}.txt");
```

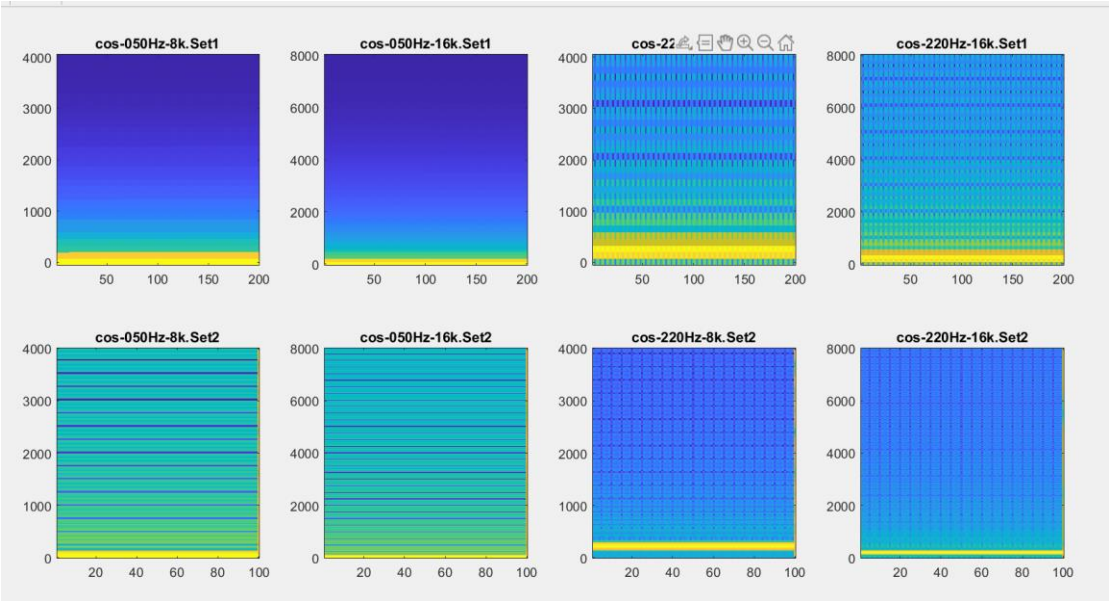
接下來便一一 subplot 各個 spectrogram 至圖上，下圖舉其中一組為例:

```
figure('Name','cos','Numbertitle','off');
subplot(2,4,1)
x=[1 200];
y=[0 4000];%設定頻率軸範圍
imagesc(x,y,A(:,1:ceil(end/2))');%因頻域分布由[-pi,pi]，magnitude大小左右對稱，故只需繪出txt檔前一半的值即可
axis xy;            %將y軸順序顛倒，改為往上為正
title('cos-050Hz-8k.{Set1}');

subplot(2,4,5)
```

五、 輸出結果:

由觀察下圖結果可發現，Set2 比 Set1 輸出的結果更加清晰且精準，用 rectangular function 當作 window type 的時頻譜模糊、不易分析:



觀察下圖可發現 Set2 大幅清晰於 Set1 的圖，且仔細觀察 8k.Set2 和 16k.Set2 的差別，取樣率為 16k 的圖更為精確、不模糊: