**Code :**

**Main.m (主要執行function):**

% Clear and close all previous data

close all;

clear;

clc;

figure(2);

% Open audio file

filePath = "sample\_1.wav";

[y, Fs] = audioread(filePath); % Get audio sample data & sample rate

info = audioinfo(filePath); % Get audio info

% Set frame length & overlap length

frameLength = 20;

overlapLength = 10;

% Plot waveform of audio file

subplot(5, 1, 1);

% Plot the waveform according to the bitpersample, also denormalize it

[t, bitSample] = Waveform(y, Fs, info);

waveform = plot(t, bitSample);

title("Waveform");

xlabel("Time(s)");

ylabel("Audio Data(" + info.BitsPerSample + " bits)");

% Plot energy contour

subplot(5, 1, 2);

[t, energy] = Energy(bitSample, Fs, info, frameLength, overlapLength);

plot(t, energy);

title("Energy contour");

xlabel("Time(s)");

ylabel("Energy");

% Plot zero-crossing rate contour

subplot(5, 1, 3);

[t, zeroCrossingRate] = ZeroCrossingRate(bitSample, Fs, info, frameLength, overlapLength);

plot(t, zeroCrossingRate);

title("Zero-crossing rate contour");

xlabel("Time(s)");

ylabel("Rate");

% Plot pitch contour

subplot(5, 1, 4);

[t, pitch] = Pitch(y, Fs, info, frameLength, overlapLength);

plot(t, pitch);

title("Pitch contour");

xlabel("Time(s)");

ylabel("Hz");

% Plot end point detection

endplot = subplot(5, 1, 5);

[frontEnd, backEnd] = EndPointDetection(y, Fs, info, frameLength, overlapLength, energy, zeroCrossingRate);

hold on;

copyobj(waveform, endplot); % Copy wave from waveform

line([frontEnd, frontEnd], [2^(info.BitsPerSample-1), -2^(info.BitsPerSample-1)], 'Color',[1 0 0]);

line([backEnd, backEnd], [2^(info.BitsPerSample-1), -2^(info.BitsPerSample-1)], 'Color',[1 0 0]);

hold off;

title("End point detection");

xlabel("Time(s)");

ylabel("Audio Data(" + info.BitsPerSample + " bits)");

**Waveform.m (畫出waveform跟計算bits)**

function [t, bitSample] = Waveform(y, Fs, info)

t = 0:(1/Fs):(info.Duration-1/Fs); % Set time duration to audio file's length

% Denormalize the value of sample

bitSample = zeros(1, length(y));

switch info.BitsPerSample

case 8

bitSample = y.\*(2^7)+(2^7); % 0 <= y <= 255 (int8)

case 16

bitSample = y.\*(2^15); % -32768 <= y <= +32767 (int16)

case {24, 32}

bitSample = y.\*(2^31); % -2^31 <= y <= 2^31¡V1 (int32)

otherwise

bitSample = y;

end

end

**Energy.m (畫出Energy)**

function [t, energy] = Energy(y, Fs, info, frameLength, overlapLength)

frameSize = floor(frameLength\*Fs/1000); % Setting frame time to 20ms, so we'll have 882 sample in one frame

frameOverlap = floor(overlapLength\*Fs/1000); % Setting frame overlap to 10 ms, so two consecutive frames will have 15 ms overlap

% Calculate hamming window of frameSize

w = zeros(1, frameSize); % Store w(n)

for n = 0:1:frameSize-1

w(n+1) = 0.54-0.46\*cos(2\*pi\*n/(frameSize-1));

end

% Calculate the energy with frame and overlap

temp = 0;

energy = zeros(1, (floor(info.TotalSamples/frameOverlap)-floor(frameSize/frameOverlap)));

for i = 1:(floor((length(y))/frameOverlap)-floor(frameSize/frameOverlap))

for j = (((i-1)\*frameOverlap)+1):(((i-1)\*frameOverlap)+frameSize) % For loop sum energy with overlap

temp = temp + (y(j)\*w(j-(i-1)\*frameOverlap))^2; % (x(m)w(n-m))^2

end

energy(i) = temp;

temp = 0;

end

% Calculate the time duration for energy with frame and overlap

t = 0:((info.Duration)/length(energy)):(info.Duration-1/Fs);

end

**ZeroCrossingRate.m (畫出ZeroCrossingRate)**

function [t, zeroCrossingRate] = ZeroCrossingRate(y, Fs, info, frameLength, overlapLength)

frameSize = floor(frameLength\*Fs/1000); % Setting frame time to 20ms, so we'll have 882 sample in one frame

frameOverlap = floor(overlapLength\*Fs/1000); % Setting frame overlap to 10 ms, so two consecutive frames will have 15 ms overlap

% Calculate window of frameSize

w = 1/(2\*frameSize); % Store w(n), in our calculation will always be 1/2N

% Calculate the zerocrossing with frame and overlap

temp = 0;

zeroCrossingRate = zeros(1, (floor((length(y))/frameOverlap)-floor(frameSize/frameOverlap)));

for i = 1:(floor(info.TotalSamples/frameOverlap)-floor(frameSize/frameOverlap))

for j = (((i-1)\*frameOverlap)+2):(((i-1)\*frameOverlap)+frameSize) % For loop sum zerocrossing rate with overlap

temp = temp + abs(sign(y(j))-sign(y(j-1)))\*w;

end

zeroCrossingRate(i) = temp;

temp = 0;

end

% Calculate the time duration for zerocrossing with frame and overlap

t = 0:((info.Duration)/length(zeroCrossingRate)):(info.Duration-1/Fs);

end

**Pitch.m (畫出Pitch)**

function [t, pitch] = Pitch(y, Fs, info, frameLength, overlapLength)

frameSize = floor(frameLength\*Fs/1000); % Setting frame time to 20ms, so we'll have 882 sample in one frame

frameOverlap = floor(overlapLength\*Fs/1000); % Setting frame overlap to 10 ms, so two consecutive frames will have 15 ms overlap

Tau = frameSize-1; % Setting Tau to

% Calculate hamming window of frameSize

w = zeros(1, frameSize); % Store w(n)

for n = 0:1:frameSize-1

w(n+1) = 0.54-0.46\*cos(2\*pi\*n/(frameSize-1));

end

pitch = []; % Store pitch

frameAcr = []; % Store local frame autocorrelation

temp = 0;

for i = 1:(floor(info.TotalSamples/frameOverlap)-floor(frameSize/frameOverlap)) % loop through all the frame with overlap

for k = 0:Tau

for m = (((i-1)\*frameOverlap)+1):(((i-1)\*frameOverlap)+frameSize-k) % loop through elements in each frame

temp = temp + y(m)\*w(mod(m-1, frameSize)+1)\*y(m+k)\*w(mod(m+k-1, frameSize)+1); % sum R(n)

end

frameAcr(k+1) = temp; % store R(n)

temp = 0;

end

peaks = findpeaks(frameAcr); % get all the peak of acr wave in this frame

max2 = max(peaks(peaks<max(peaks))); % get second peak of acr wave in this frame

pitch(i) = find(frameAcr==max2); % get this peak's index, it's the pitch

frameAcr = [];

end

% Calculate the time duration for energy with frame and overlap

t = 0:((info.Duration)/length(pitch)):(info.Duration-1/Fs);

end

**EndPointDetection.m (畫出EndPointDetection)**

function [frontEnd, backEnd] = EndPointDetection(y, Fs, info, frameLength, overlapLength, energy, zeroCrossingRate)

% set silence interval to 100 ms

intervalSilence = 100;

frameNumber = length(energy);

soundStart = floor((intervalSilence-(frameLength-overlapLength))/overlapLength);

soundEnd = frameNumber - soundStart;

% Take second 100ms to set Tu, Tl, Tzc

Tu = max(energy(soundStart:soundStart\*2));

Tl = median(energy(soundStart:soundStart\*2));

Tzc = max(zeroCrossingRate(soundStart:soundStart\*2));

%% Calculate front end point

frontEnd = 0;

N = 0;

% Get initial N by Tu

for i = 1:frameNumber

if energy(i) > Tu

N = i;

break;

end

end

% Adjust N by Tl

for i = N:-1:1

if energy(i) < Tl

N = i;

break;

end

end

% Adjust N by Tzc

for i = N:-1:1

if zeroCrossingRate(i) < Tzc

N = i;

break;

end

end

frontEnd = N/frameNumber\*info.Duration;

%% Calculate back end point

backEnd = 0;

N = 0;

% Get initial N by Tu

for i = soundEnd:-1:1

if energy(i) > Tu

N = i;

break;

end

end

% Adjust N by Tl

for i = N:soundEnd

if energy(i) > Tl

N = i;

break;

end

end

% Adjust N by Tzc

for i = N:soundEnd

if zeroCrossingRate(i) < Tzc

N = i;

break;

end

end

backEnd = N/frameNumber\*info.Duration;

end

**程式碼解說 : (詳細都有註解，以下簡單解說)**

**Main :**

讓使用者call來使用，讀進wav檔以及呼叫其他function來計算

**Waveform.m :**

先利用info來抓取此wav檔案位元率，進而調整audioread得到的y，讓畫出來的圖形可以呈現實際儲存的位元數字，而非range(-1, 1)

**Energy.m :**

設定frame length和 overlap各為20ms&10ms，而頻率為44100Hz，所以在一個frame裡面會有882個sample point，而overlap為441個sample point，然後藉由以上的frame size跟overlap來逐一利用short-time energy以及w(n) = hamming window畫出energy曲線

**ZeroCrossingRate.m :**

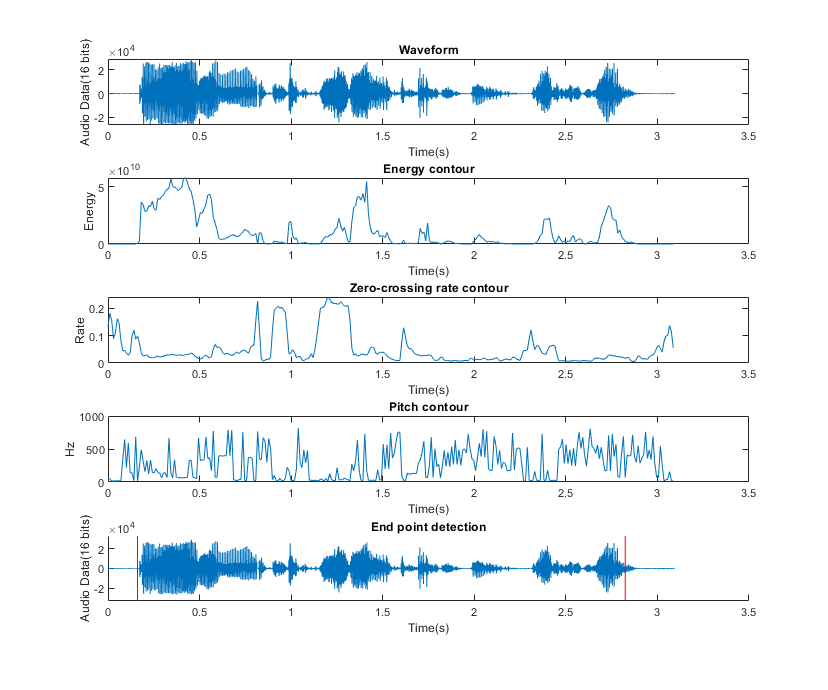
設定frame length和 overlap各為20ms&10ms，而頻率為44100Hz，所以在一個frame裡面會有882個sample point，而overlap為441個sample point，然後藉由以上的frame size跟overlap來逐一利用zero crossing rate fuction以及w(n) = 1/2N 畫出zeroc crossing rate曲線

**Pitch.m :**

設定frame length和 overlap各為20ms&10ms，而頻率為44100Hz，所以在一個frame裡面會有882個sample point，而overlap為441個sample point，然後藉由以上的frame size跟overlap來逐一計算出每個frame的auto correlation後，曲第一高wave peak跟第二高wave peak的週期差值(一定會小於882因為一個frame最多882個點)，將這些差值畫出就是pitch曲線，缺點就是太高頻率的畫不出來(但剛好也可以把背景高頻雜音濾掉)

**EndPointDetection.m :**

利用energy前面100ms ~ 200ms(0~100ms我視為無聲音)，將此處的max設為Tu，此處的median設為Tl，同樣區間下的zerocrossingrate的max為Tzc，然後從energy頭找起，找到第一個大於Tu的值，之後回頭找第一個小於Tl的值，然後有此index後再繼續往前找第一個小於Tzc的值，此值就是開頭的end point，找尾巴的亦然。

**Result :**