**HW3**

Q36111281 張偉治

1. 程式碼說明

調高音高與調降音高的程式整體架構相同，差別只在於新舊訊號的頻率比率不同。在相同任務中有兩個版本，其一是採用short-time FFT將振幅與相位拆分，對相位計算後使用inverse short-time FFT與振幅重新合併，最後對計算後的訊號進行伸縮改變其頻率；其二是使用matlab內建的函式將訊號的時間拉伸，再對拉伸後的訊號進行伸縮改變其頻率。

1.1 pitch\_up

|  |
| --- |
| % 數值初始化  WindowLen = 256;  AnalysisLen = 85;  SynthesisLen = 90;  Fs = 16000;  Hopratio = SynthesisLen/AnalysisLen;  % 音檔讀取  reader = dsp.AudioFileReader('singing16k16bit-clean.wav', ...  'SamplesPerFrame',AnalysisLen, ...  'OutputDataType','double');  % FFT  win = sqrt(hanning(WindowLen,'periodic'));  stft = dsp.STFT(win, WindowLen - AnalysisLen, WindowLen);  istft = dsp.ISTFT(win, WindowLen - SynthesisLen );  logger = dsp.SignalSink;  % 初始化遍歷參數  unwrapdata = 2\*pi\*AnalysisLen\*(0:WindowLen-1)'/WindowLen;  yangle = zeros(WindowLen,1);  firsttime = true;  % 根據音窗遍歷  while ~isDone(reader)  y = reader();  % ST-FFT  yfft = stft(y);    % 分離出振幅與相位  ymag = abs(yfft);  yprevangle = yangle;  yangle = angle(yfft);  % 相位計算  yunwrap = (yangle - yprevangle) - unwrapdata;  yunwrap = yunwrap - round(yunwrap/(2\*pi))\*2\*pi;  yunwrap = (yunwrap + unwrapdata) \* Hopratio;  if firsttime  ysangle = yangle;  firsttime = false;  else  ysangle = ysangle + yunwrap;  end  % 將振幅與相位合成  ys = ymag .\* complex(cos(ysangle), sin(ysangle));  % IST-FFT  yistfft = istft(ys);  logger(yistfft) % Log signal  end  % 記憶體釋放  release(reader)  % 播放Time-Stretched訊號  loggedSpeech = logger.Buffer(200:end)';  player = audioDeviceWriter('SampleRate',Fs, ...  'SupportVariableSizeInput',true, ...  'BufferSize',512);  player(loggedSpeech.');    % 播放Pitch-Scaled訊號  Fs\_new = round(Fs\*(SynthesisLen/AnalysisLen));  player = audioDeviceWriter('SampleRate',Fs\_new, ...  'SupportVariableSizeInput',true, ...  'BufferSize',1024);  player(loggedSpeech.');  % 存檔  audiowrite('C:\AG\課程講義\digtal signal porcessing\HW3\singing16k16bit-clean\_1.wav',loggedSpeech.',Fs\_new); |

1.2 pitch\_up\_v2

|  |
| --- |
| % 數值初始化  WindowLen = 256;  AnalysisLen = 85;  SynthesisLen = 90;  Fs = 16000;  Hopratio = SynthesisLen/AnalysisLen;  play\_flag = 1;  % 音檔讀取  reader = dsp.AudioFileReader('singing16k16bit-clean.wav', ...  'SamplesPerFrame',AnalysisLen, ...  'OutputDataType','double');  % 時間調整  win = sqrt(hanning(WindowLen,'periodic'));  logger = dsp.SignalSink;  ats = audioTimeScaler(1/Hopratio,'Window',win,'OverlapLength',WindowLen-AnalysisLen);  % 根據音窗遍歷  while ~isDone(reader)    x = reader();  % Time-scale the signal  y = ats(x);  logger(y)  end  % 釋放記憶體  release(reader)  release(player)  % 播放Pitch-Scaled訊號  loggedSpeech = logger.Buffer(200:end)';  Fs\_new = round(Fs\*(SynthesisLen/AnalysisLen));  if play\_flag==1  player = audioDeviceWriter('SampleRate',Fs\_new, ...  'SupportVariableSizeInput',true, ...  'BufferSize',1024);  player(loggedSpeech.');  end  % 存檔  audiowrite('C:\AG\課程講義\digtal signal porcessing\HW3\singing16k16bit-clean\_2.wav',loggedSpeech.',Fs\_new); |

1.3 pitch\_down

|  |
| --- |
| % 數值初始化  WindowLen = 256;  AnalysisLen = 95;  SynthesisLen = 90;  Fs = 16000;  Hopratio = SynthesisLen/AnalysisLen;  % 音檔讀取  reader = dsp.AudioFileReader('singing16k16bit-clean.wav', ...  'SamplesPerFrame',AnalysisLen, ...  'OutputDataType','double');  % FFT  win = sqrt(hanning(WindowLen,'periodic'));  stft = dsp.STFT(win, WindowLen - AnalysisLen, WindowLen);  istft = dsp.ISTFT(win, WindowLen - SynthesisLen );  logger = dsp.SignalSink;  % 初始化遍歷參數  unwrapdata = 2\*pi\*AnalysisLen\*(0:WindowLen-1)'/WindowLen;  yangle = zeros(WindowLen,1);  firsttime = true;  % 根據音窗遍歷  while ~isDone(reader)  y = reader();  % ST-FFT  yfft = stft(y);    % 分離出振幅與相位  ymag = abs(yfft);  yprevangle = yangle;  yangle = angle(yfft);  % 相位計算  yunwrap = (yangle - yprevangle) - unwrapdata;  yunwrap = yunwrap - round(yunwrap/(2\*pi))\*2\*pi;  yunwrap = (yunwrap + unwrapdata) \* Hopratio;  if firsttime  ysangle = yangle;  firsttime = false;  else  ysangle = ysangle + yunwrap;  end  % 將振幅與相位合成  ys = ymag .\* complex(cos(ysangle), sin(ysangle));  % IST-FFT  yistfft = istft(ys);  logger(yistfft) % Log signal  end  % 記憶體釋放  release(reader)  % 播放Time-Stretched訊號  loggedSpeech = logger.Buffer(200:end)';  player = audioDeviceWriter('SampleRate',Fs, ...  'SupportVariableSizeInput',true, ...  'BufferSize',512);  player(loggedSpeech.');    % 播放Pitch-Scaled訊號  Fs\_new = round(Fs\*(SynthesisLen/AnalysisLen));  player = audioDeviceWriter('SampleRate',Fs\_new, ...  'SupportVariableSizeInput',true, ...  'BufferSize',1024);  player(loggedSpeech.');  % 存檔  audiowrite('C:\AG\課程講義\digtal signal porcessing\HW3\singing16k16bit-clean\_3.wav',loggedSpeech.',Fs\_new); |

1.4 pitch\_down\_v2

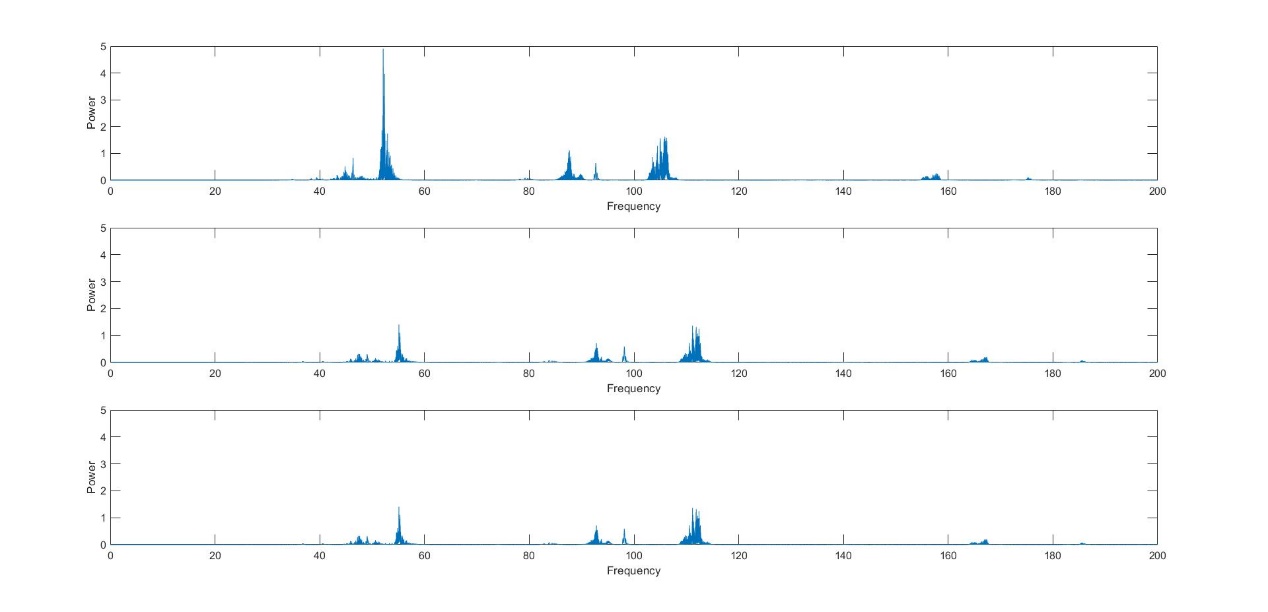
|  |
| --- |
| % 數值初始化  WindowLen = 256;  AnalysisLen = 95;  SynthesisLen = 90;  Fs = 16000;  Hopratio = SynthesisLen/AnalysisLen;  play\_flag = 1;  % 音檔讀取  reader = dsp.AudioFileReader('singing16k16bit-clean.wav', ...  'SamplesPerFrame',AnalysisLen, ...  'OutputDataType','double');  % 時間調整  win = sqrt(hanning(WindowLen,'periodic'));  logger = dsp.SignalSink;  ats = audioTimeScaler(1/Hopratio,'Window',win,'OverlapLength',WindowLen-AnalysisLen);  % 根據音窗遍歷  while ~isDone(reader)    x = reader();  % Time-scale the signal  y = ats(x);  logger(y)  end  % 釋放記憶體  release(reader)  release(player)  % 播放Pitch-Scaled訊號  loggedSpeech = logger.Buffer(200:end)';  Fs\_new = round(Fs\*(SynthesisLen/AnalysisLen));  if play\_flag==1  player = audioDeviceWriter('SampleRate',Fs\_new, ...  'SupportVariableSizeInput',true, ...  'BufferSize',1024);  player(loggedSpeech.');  end  % 存檔  audiowrite('C:\AG\課程講義\digtal signal porcessing\HW3\singing16k16bit-clean\_4.wav',loggedSpeech.',Fs\_new); |

1.5 signal\_analysis

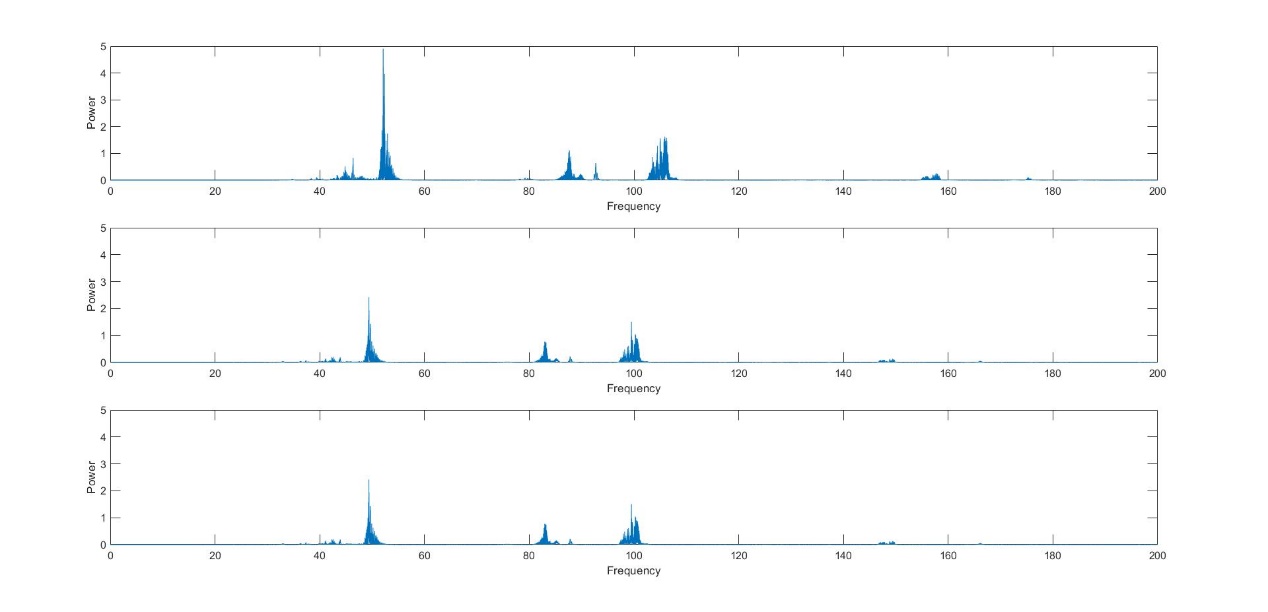
|  |
| --- |
| [y0,fs0] = audioread('singing16k16bit-clean.wav');  [y1,fs1] = audioread('singing16k16bit-clean\_3.wav');  [y2,fs2] = audioread('singing16k16bit-clean\_4.wav');  m0 = length(y0);  n0 = pow2(nextpow2(m0));  y0\_f = fft(y0,n0);  f0 = (0:n0-1)\*(fs0/n0)/10;  power0 = abs(y0\_f).^2/n0;  m1 = length(y1);  n1 = pow2(nextpow2(m1));  y1\_f = fft(y1,n1);  f1 = (0:n1-1)\*(fs1/n1)/10;  power1 = abs(y1\_f).^2/n1;  m2 = length(y2);  n2 = pow2(nextpow2(m2));  y2\_f = fft(y2,n2);  f2 = (0:n2-1)\*(fs2/n2)/10;  power2 = abs(y2\_f).^2/n2;  tiledlayout(3,1)  ax1 = nexttile;  plot(f0(1:floor(n0/2)),power0(1:floor(n0/2)))  xlabel('Frequency')  ylabel('Power')  ax2 = nexttile;  plot(f1(1:floor(n1/2)),power1(1:floor(n1/2)))  xlabel('Frequency')  ylabel('Power')  ax3 = nexttile;  plot(f2(1:floor(n2/2)),power2(1:floor(n2/2)))  xlabel('Frequency')  ylabel('Power')  linkaxes([ax1 ax2 ax3],'xy')  ax1.XLim = [0 200]; |

2.輸出結果

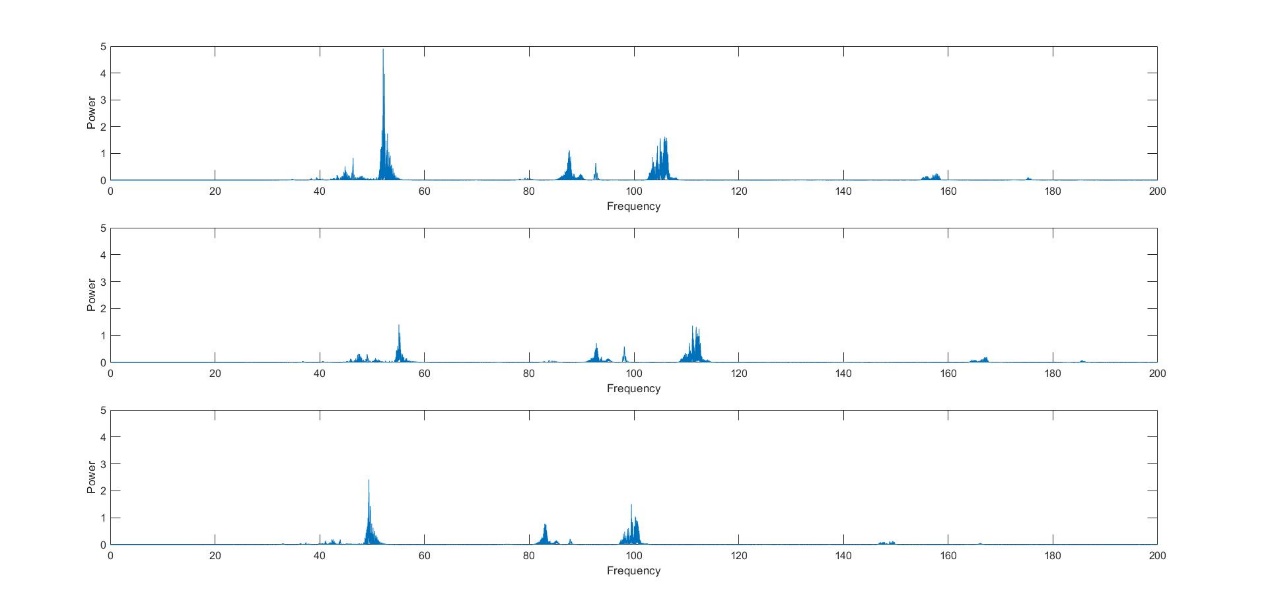
2.1 不同方法升高六個半音頻譜圖比較



2.2 不同方法降低六個半音頻譜圖比較



2.3 相同方法頻譜圖比較(system method)



2.4 相同方法頻譜圖比較(matlab函式)

