program code :

from scipy.io import wavfile #提供wav檔的處理

import matplotlib.pyplot as plt #提供類似matlab的繪圖API

import numpy as np #提供數學計算函式庫

rate,data = wavfile.read("./sample\_speech\_signal.wav") #讀取.wav檔。返回一個tuple，分別為採樣率和數據的numpy數組

print("rate : " + str(rate) + " Hz")

print(str(len(data))+" datas")

#(1) Waveform

plt.figure(figsize = [25,10])

plt.plot(data)

plt.title('Waveform')

plt.xlabel('time (1/44100 s)')

plt.ylabel('Amplitude')

plt.show()

#(2) Energy contour

#let frame size = 20ms, and 1 frame has 20ms\*44100Hz = 882 datas, so there are (140288//882) = 159 frames.

#刪掉多餘的datas, (140288-140288%882) = 140238, 140288%882 =50.

data3 = data[:-50]

print("length of data3 :",len(data3))

Energy = []

for i in range(159):

begin = i\*882

end = (i+1)\*882

w = np.zeros(data3.shape)

w[begin:end] = 1 # Rectangular window # w[begin] = (1,1) ... w[end] = (1,1)

E = sum((data3\*w)\*\*2)

Energy.append(E)

plt.figure(figsize=[25,10])

plt.plot(Energy)

plt.title('Short-Time Energy')

plt.xlabel('frame')

plt.ylabel('Energy')

plt.show()

#(3) Zero-crossing rate contour

data5 = np.zeros(data3.shape)

data6 = np.zeros(data3.shape)

for i in range(len(data3)): #sgn[x(m)]

if data3[i][0] >= 0:

data5[i] = 1

else:

data5[i] = -1

data6[0] = 1

for i in range(len(data3)-1): #sgn[x(m-1)]

if data3[i][0] >= 0:

data6[i+1] = 1

else:

data6[i+1] = -1

Zero\_crossing = []

for i in range(159):

begin = i\*882

end = (i+1)\*882

w = np.zeros(data3.shape)

w[begin:end] = 1/2

ZCR = sum(abs(data5-data6)\*w)

Zero\_crossing.append(ZCR)

plt.figure(figsize=[25,10])

plt.plot(Zero\_crossing)

plt.title('Zero-crossing rate')

plt.xlabel('frame')

plt.ylabel('rate')

plt.show()

#Zero-crossing rate of unvoiced sounds and environmental noise are usually larger than voiced sounds.

#(4) End point detection

sound\_volume = []

Avg = []

Std = []

for i in range(159):

begin = i\*882

end = (i+1)\*882

result = data3[begin:end] - np.median(data3[begin:end]) #apply median subtraction for zero justification

sound\_volume.append(sum(abs(result))[0]) #method of "abs sum"

for i in range(0,159,20):

if i >= 140:

Avg.append(sum(sound\_volume[i:])/len(sound\_volume[i:]))

Std.append(np.std(sound\_volume[i:]))

else:

Avg.append(sum(sound\_volume[i:i+20])/len(sound\_volume[i:i+20]))

Std.append(np.std(sound\_volume[i:i+20]))

start\_point = []

end\_point = []

Start = False

for i in range(len(sound\_volume)):

if sound\_volume[i] > Avg[i//20] + 0.05\*Std[i//20] and Start == False and sound\_volume[i] > 300000:

Start = True

start\_point.append(i)

if sound\_volume[i] < Avg[i//20] - 0.05\*Std[i//20] and Start == True and sound\_volume[i] > 300000:

Start = False

end\_point.append(i)

plt.figure(figsize=[25,10])

plt.plot(sound\_volume)

plt.title('End point detection')

plt.xlabel('frame')

plt.ylabel('volume')

for i in range(len(start\_point)):

plt.axvline(start\_point[i],color='#2ca02c') #green

for i in range(len(end\_point)):

plt.axvline(end\_point[i],color='#d62728') #red

plt.show()

#------------------------------------------------------

plt.figure(figsize=[25,10])

t = np.linspace(0,159,140238)

plt.plot(t, data3)

plt.title('End point detection')

plt.xlabel('frame')

plt.ylabel('Amplitude')

for i in range(len(start\_point)):

plt.axvline(start\_point[i],color='#2ca02c') #green

for i in range(len(end\_point)):

plt.axvline(end\_point[i],color='#d62728') #red

plt.show()

#(5) Pitch contour

# The frame size for pitch should cover at least two fundamental periods

# fundamental period = 44100 Hz / 100 Hz = 441 datas, frame size for pitch = 882 datas

# auto-correlation function (ACF)

voice\_pitch = []

k = 20 # k is the pitch period and varies from 20 to 400

for i in range(159-k): # 159

begin = i\*882

end = (i+1)\*882

data9\_1 = data3[begin:end]

data9\_2 = data3[begin+k:end+k]

x1 = np.zeros(data3.shape)

x1[begin:end] = data9\_1

x2 = np.zeros(data3.shape)

x2[begin+k:end+k] = data9\_2

w1 = np.zeros(data3.shape)

w1[begin:end] = 1

w2 = np.zeros(data3.shape)

w2[begin+k:end+k] = 1

autocorrelation = sum(x1\*w1\*x2\*w2) # sum[x(m)\*w1(n-m)\*x(m+k)\*w2(n-m-k)]

#數學上，音高和基本頻率的關係：p = 69 + 12\*log2(f/440)

if autocorrelation[0] == 0:

p = 69

else:

p = 69 + 12\*np.log2(autocorrelation[0]//440)

voice\_pitch.append(p)

plt.figure(figsize=[25,10])

plt.plot(voice\_pitch)

plt.title('Pitch')

plt.xlabel('frame')

plt.ylabel('Hz')

plt.show()

plt.figure(figsize=[25,10])

xx = np.arange(0,len(voice\_pitch))

plt.scatter(xx,voice\_pitch)

plt.title('Pitch')

plt.xlabel('frame')

plt.ylabel('Hz')

plt.show()

#男生的音高範圍約在頻率 62 ~ 523 Hz。

import librosa

wave,fs = librosa.load('sample\_speech\_signal.wav')

#print(fs) # 22050

#print(len(wave)) # 70144

pitches,magnitudes = librosa.core.piptrack(wave,fs,hop\_length=int(10 \* fs /1000),threshold=0.75)

#print(len(pitches)) # 1025

max\_pitch = []

for i in range(pitches.shape[1]):

max\_ = np.max(pitches[:,i])

if max\_>1000 :

max\_ = 0 # 如果超過合理的音高範圍，就認定沒有音高存在

max\_pitch.append(max\_)

#print(len(max\_pitch)) #319

plt.figure(figsize=[25,10])

plt.plot(max\_pitch)

plt.title('Pitch')

plt.ylabel('Hz')

plt.show()

plt.figure(figsize=[25,10])

xx = np.arange(0,len(max\_pitch))

plt.scatter(xx,max\_pitch)

plt.title('Pitch')

plt.ylabel('Hz')

plt.show()

#男生的音高範圍約在頻率 62 ~ 523 Hz。

#(5) Pitch contour

# The frame size for pitch should cover at least two fundamental periods

# fundamental period = 44100 Hz / 100 Hz = 441 datas, frame size for pitch = 882 datas

# auto-correlation function (ACF)

# Center Clipping

data7 = []

CL = 2000

for i in range(len(data3)):

if data3[i][0] > CL:

data7.append(data3[i][0] - CL)

elif data3[i][0] < (-CL):

data7.append(data3[i][0] + CL)

else:

data7.append(0)

voice\_pitch = []

k = 20 # k is the pitch period and varies from 20 to 400

for i in range(159-k): # 159

begin = i\*882

end = (i+1)\*882

data9\_1 = data7[begin:end]

data9\_2 = data7[begin+k:end+k]

x1 = np.zeros(len(data7))

x1[begin:end] = data9\_1

x2 = np.zeros(len(data7))

x2[begin+k:end+k] = data9\_2

w1 = np.zeros(len(data7))

w1[begin:end] = 1

w2 = np.zeros(len(data7))

w2[begin+k:end+k] = 1

autocorrelation = sum(x1\*w1\*x2\*w2) # sum[x(m)\*w1(n-m)\*x(m+k)\*w2(n-m-k)]

#數學上，音高和基本頻率的關係：p = 69 + 12\*log2(f/440)

if autocorrelation == 0:

p = 69

else:

p = 69 + 12\*np.log2(autocorrelation//440)

voice\_pitch.append(p)

# Median Smoothing

for i in range(0,len(voice\_pitch)-len(voice\_pitch)%3,3):

median\_pitch = np.median(voice\_pitch[i:i+3])

for j in range(3):

voice\_pitch[i+j] = median\_pitch

plt.figure(figsize=[25,10])

plt.plot(voice\_pitch)

plt.title('Pitch')

plt.xlabel('frame')

plt.ylabel('Hz')

plt.show()

plt.figure(figsize=[25,10])

xx = np.arange(0,len(voice\_pitch)) # 159-k

plt.scatter(xx,voice\_pitch)

plt.title('Pitch')

plt.xlabel('frame')

plt.ylabel('Hz')

plt.show()

#男生的音高範圍約在頻率 62 ~ 523 Hz。

methods :

匯入所需的library

讀入音檔，得到採樣率和資料

(1) Waveform

依資料畫出波形

(2) Energy contour

設frame size為20ms，則一個frame有882個資料，共159個frame。刪掉多餘的50個資料。依簡報上Short-Time Energy的公式，並採用Rectangular window，算出每個frame的Short-Time Energy，依此畫圖。

(3) Zero-crossing rate contour

依簡報上Zero-Crossing Rate的公式，先取出sgn[x(m)]和sgn[x(m-1)]的資料，算出每個frame的Zero-Crossing Rate，依此畫圖。

(4) End point detection

透過資料算出各個frame的音量，算出每20個frame的平均值與標準差，再由音量的平均值與0.05個標準差找出起點與終點，將起點標為綠線，將終點標為紅線，分別畫出對應於音量和波形的圖，發現皆大致吻合，能大致看出「多媒體系統與應用」八字的起點與終點。

(5) Pitch contour

方法一（自相關函數法）：依簡報上Short-Time Autocorrelation的公式，算出每個frame的autocorrelation，再用頻率轉音高的公式，依此畫圖。基於簡報上的圖，分別畫出折線圖與散點圖。

方法二：匯入librosa，讀入音檔，得到音高，取每個區段中的最大值，依此畫圖。

方法三：將方法一加上前處理Center Clipping與後處理Median Smoothing，然後畫圖。

results :

一張含有 文字, 地圖, 桌, 橙色 的圖片

自動產生的描述

一張含有 船, 團體, 大, 差異 的圖片

自動產生的描述

一張含有 文字, 地圖 的圖片

自動產生的描述

一張含有 文字, 桌 的圖片

自動產生的描述

一張含有 地圖, 桌, 船, 男人 的圖片

自動產生的描述

一張含有 文字, 地圖, 桌, 男人 的圖片

自動產生的描述

一張含有 地圖, 大, 差異, 團體 的圖片

自動產生的描述

一張含有 文字, 桌, 小, 坐 的圖片

自動產生的描述

一張含有 文字, 膝上型電腦, 大, 桌 的圖片

自動產生的描述

一張含有 文字, 桌, 男人, 膝上型電腦 的圖片

自動產生的描述

一張含有 室內, 膝上型電腦, 電腦, 監視器 的圖片

自動產生的描述