**訊號與系統期末報告** B0843042 黃子庭

**單聲道及雙聲道的比較**

1. **單聲道**

* 指只有一個聲音的通道
* 只能感受到聲音、音樂的前後位置及音色、音量的大小，不能感受到聲音從左到右等橫向的移動
* 單聲道伴音品質欠佳

1. **雙聲道**

* 有兩個聲音的通道
* 實現立體聲的原理
* 兩個聲道訊號在相位上有所差別

**程式實作**

**載入相關套件**

import os

if not os.path.exists('thinkdsp.py'):

    !wget <https://github.com/AllenDowney/ThinkDSP/raw/master/code/thinkdsp.py>

import numpy as np

import wave

from thinkdsp import read\_wave

import matplotlib.pyplot as plt

from scipy.io import wavfile

**對鋼琴音訊進行分析及取樣**

# 分析前先聽鋼琴音訊(以此音訊可清楚聽到存在不同聲道)

piano = read\_wave('piano.wav')

piano.normalize()

piano.make\_audio()

piano = wave.open(r"piano.wav", "rb")

params\_p = piano.getparams() # 讀取格式資訊

nchannels, sampwidth, framerate, nframes = params\_p[:4] # 聲道數、量化位數、取樣頻率、取樣點數

params\_p

str\_pdata = piano.readframes(nframes) # 讀取"取樣點數"個數據，返回字串格式

piano.close()

wave\_pdata = np.frombuffer(str\_pdata, dtype=np.short) # 將字串轉換為陣列，得到一維的short型別的陣列

wave\_pdata = wave\_pdata\*1.0/(max(abs(wave\_pdata))) # 賦值的歸一化(使得數據被限定在一定的範圍內，以消除其他樣本數據導致的不良影響)

wave\_pdata = np.reshape(wave\_pdata,[nframes,nchannels]) # 整合左聲道和右聲道的資料

time = np.arange(0, nframes) \* (1.0 / framerate) # 通過取樣點數和取樣頻率計算出每個取樣的時間

plt.figure()

# 左聲道波形

plt.subplot(3,1,1)

plt.plot(time, wave\_pdata[:,0], c="b")

plt.xlabel("time (seconds)")

plt.ylabel("Amplitude")

plt.title("Left channel")

plt.grid()

# 右聲道波形

plt.subplot(3,1,3)

plt.plot(time, wave\_pdata[:,1], c="g")

plt.xlabel("time (seconds)")

plt.ylabel("Amplitude")

plt.title("Left channel")

plt.title("right channel")

plt.grid()

plt.show() #由圖可表示存在兩聲道

sampling\_freq, audio\_p = wavfile.read(r"piano.wav")

audio\_p = audio\_p / np.max(audio\_p)   # 歸一化，標準化

fft\_signal = np.fft.fft(audio\_p) # 傅立葉變換

print(fft\_signal)

Freq = np.arange(0, len(fft\_signal)) # 建立時間軸

# 繪製訊號頻譜圖

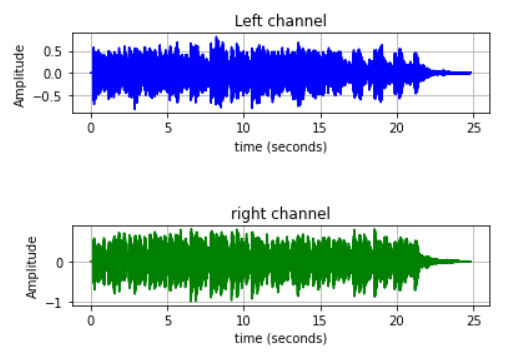
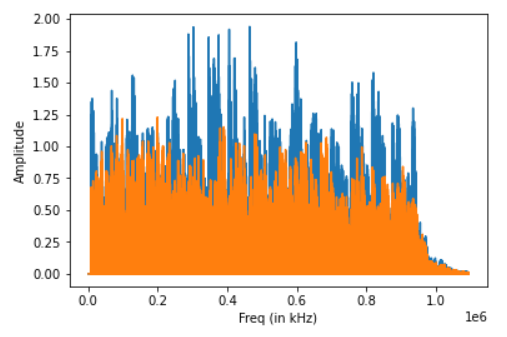
plt.figure()

plt.plot(Freq, abs(fft\_signal))

plt.xlabel('Freq (in kHz)')

plt.ylabel('Amplitude')

plt.show()

由音頻中可以清楚聽到鋼琴的高低音變換，可以清楚感受到雙聲道的存在;由兩張圖表可看出雙聲道的左右聲道的分別(左圖)，雖然大致上並無太大明顯的區別，但經過傅立葉轉換後的頻譜圖(右圖)即可清楚看出兩聲道的不同。

**對薩克斯風音訊進行分析及取樣**

# 分析前先聽薩克斯風音訊(以此音訊無法清楚聽到存在不同聲道)

saxophone = read\_wave('saxophone.wav')

saxophone.normalize()

saxophone.make\_audio()

saxophone = wave.open(r"saxophone.wav", "rb")

params\_s = saxophone.getparams() # 讀取格式資訊

nchannels, sampwidth, framerate, nframes = params\_s[:4] #聲道數、量化位數、取樣頻率、取樣點數

params\_s

str\_sdata = saxophone.readframes(nframes) # 讀取"取樣點數"個數據，返回字串格式

saxophone.close()

wave\_sdata = np.frombuffer(str\_sdata, dtype=np.short) #將字串轉換為陣列，得到一維的short型別的陣列

wave\_sdata = wave\_sdata\*1.0/(max(abs(wave\_sdata))) # 賦值的歸一化(使得數據被限定在一定的範圍內，以消除其他樣本數據導致的不良影響)

wave\_sdata = np.reshape(wave\_sdata,[nframes,nchannels]) # 整合左聲道和右聲道的資料

time = np.arange(0, nframes) \* (1.0 / framerate) # 通過取樣點數和取樣頻率計算出每個取樣的時間

plt.figure()

# 左聲道波形

plt.subplot(3,1,1)

plt.plot(time, wave\_sdata[:,0], c="b")

plt.xlabel("time (seconds)")

plt.ylabel("Amplitude")

plt.title("Left channel")

plt.grid()

# 右聲道波形

plt.subplot(3,1,3)

plt.plot(time, wave\_sdata[:,1], c="g")

plt.xlabel("time (seconds)")

plt.ylabel("Amplitude")

plt.title("Left channel")

plt.title("right channel")

plt.grid()

plt.show() #由圖可表示雖然聽音訊時無法僅為薩克斯風獨奏,但仍存在兩聲道

sampling\_freq, audio\_s = wavfile.read(r"saxophone.wav")

audio\_s = audio\_s / np.max(audio\_s)   # 歸一化，標準化

fft\_signal = np.fft.fft(audio\_s) # 傅立葉變換

print(fft\_signal)

Freq = np.arange(0, len(fft\_signal)) # 建立時間軸

# 繪製訊號頻譜圖

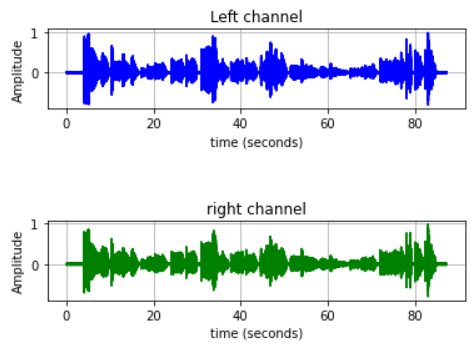
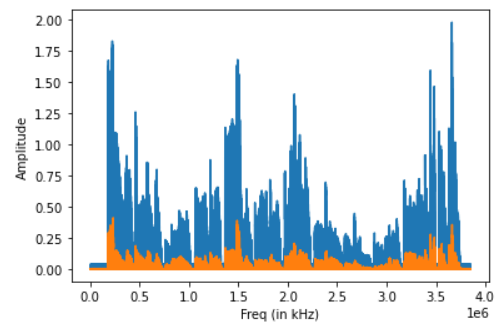
plt.figure()

plt.plot(Freq, abs(fft\_signal))

plt.xlabel('Freq (in kHz)')

plt.ylabel('Amplitude')

plt.show()

**** ****

由音頻中雖然僅聽到薩可斯風的獨奏，並不像前面鋼琴音頻中清楚感受到雙聲道的存在，但由兩張圖表可看出雙聲道的左右聲道的分別(左圖)，雖然大致上並無太大明顯的區別，但經過傅立葉轉換後的頻譜圖(右圖)即可清楚看出兩聲道的不同，且相對於鋼琴的頻譜圖，兩聲道的差別極大。

**對交響音訊進行分析及取樣**

# 分析前先聽交響音訊(以此音訊可聽到嘈雜聲)

mono = read\_wave('mono.wav')

mono.normalize()

mono.make\_audio()

mono = wave.open(r"mono.wav", "rb")

params\_m = mono.getparams() # 讀取格式資訊

nchannels, sampwidth, framerate, nframes = params\_m[:4] #聲道數、量化位數、取樣頻率、取樣點數

params\_m

str\_mdata = mono.readframes(nframes) # 讀取"取樣點數"個數據，返回字串格式

mono.close()

wave\_mdata = np.frombuffer(str\_mdata, dtype=np.short) #將字串轉換為陣列，得到一維的short型別的陣列

wave\_mdata = wave\_mdata\*1.0/(max(abs(wave\_mdata))) # 賦值的歸一化(使得數據被限定在一定的範圍內，以消除其他樣本數據導致的不良影響)

wave\_mdata = np.reshape(wave\_mdata,[nframes,nchannels]) # 整合聲道的資料

time = np.arange(0, nframes) \* (1.0 / framerate) # 通過取樣點數和取樣頻率計算出每個取樣的時間

plt.figure()

# 左聲道波形

plt.subplot(3,1,1)

plt.plot(time, wave\_mdata[:,0], c="b")

plt.xlabel("time (seconds)")

plt.ylabel("Amplitude")

plt.title("Left channel")

plt.grid()

# 右聲道波形

plt.subplot(3,1,3)

plt.plot(time, wave\_mdata[:,1], c="g")

plt.xlabel("time (seconds)")

plt.ylabel("Amplitude")

plt.title("Left channel")

plt.title("right channel")

plt.grid()

plt.show() #由圖可表示雖然聽音訊時非常嘈雜,但仍為單聲道

sampling\_freq, audio\_m = wavfile.read(r"mono.wav")

audio\_m = audio\_m / np.max(audio\_m)   # 歸一化，標準化

fft\_signal = np.fft.fft(audio\_m) # 傅立葉變換

print(fft\_signal)

Freq = np.arange(0, len(fft\_signal)) # 建立時間軸

# 繪製訊號頻譜圖

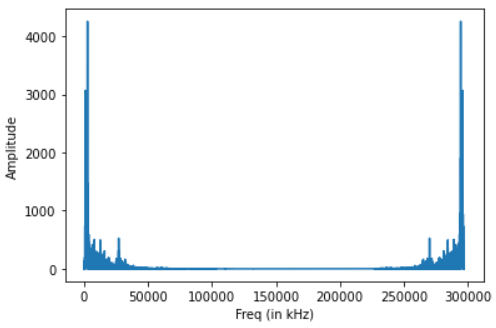
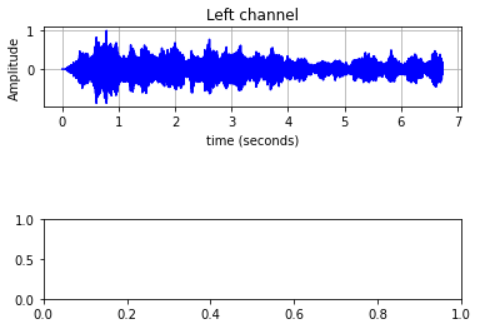
plt.figure()

plt.plot(Freq, abs(fft\_signal))

plt.xlabel('Freq (in kHz)')

plt.ylabel('Amplitude')

plt.show()

****

由音頻中雖然聽到相對於前面兩個音頻較為吵雜的音頻，但由兩張圖表可看出此音頻僅為單聲道，因為左圖僅有一圖形呈現，並不存在左右聲道，經過傅立葉轉換後的頻譜圖(右圖)亦可看出相對於前面兩個音頻，此音頻的頻譜圖較單一且平緩。