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
2. **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

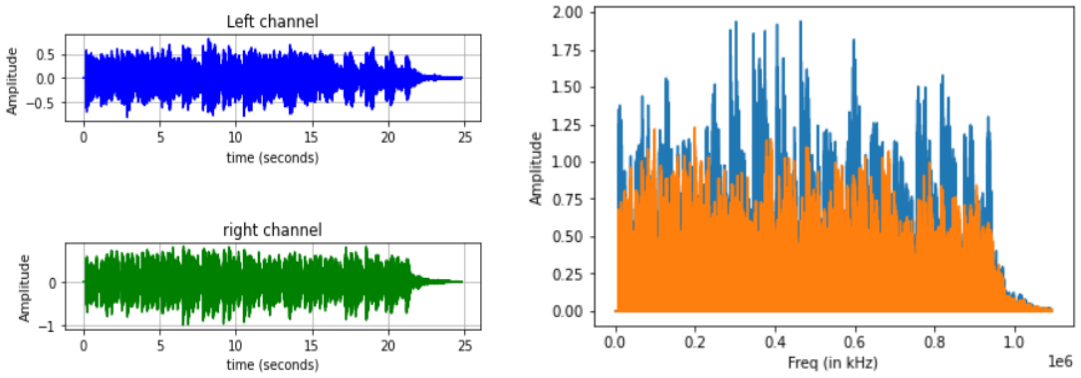
```
import os


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()
```



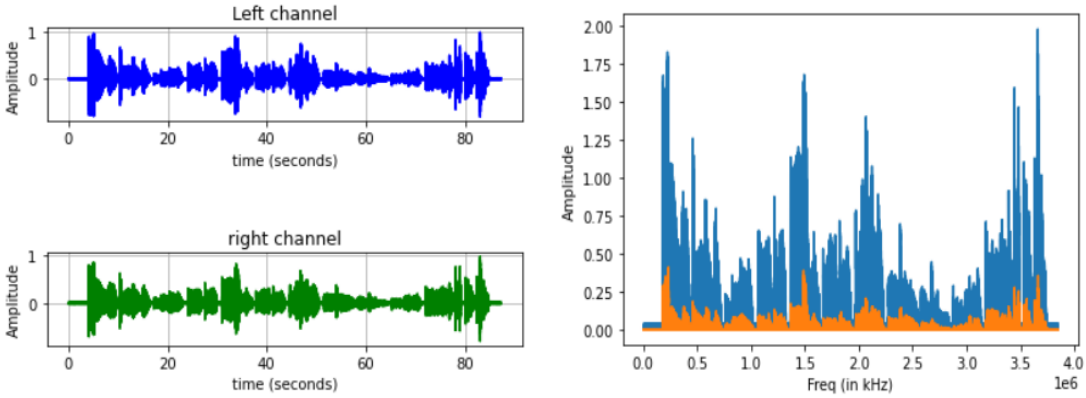
由音頻中可以清楚聽到鋼琴的高低音變換，可以清楚感受到雙聲道的存在;由兩張圖表可看出雙聲道的左右聲道的分別(左圖)，雖然大致上並無太大明顯的區別，但經過傅立葉轉換後的頻譜圖(右圖)即可清楚看出兩聲道的不同。

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()
```



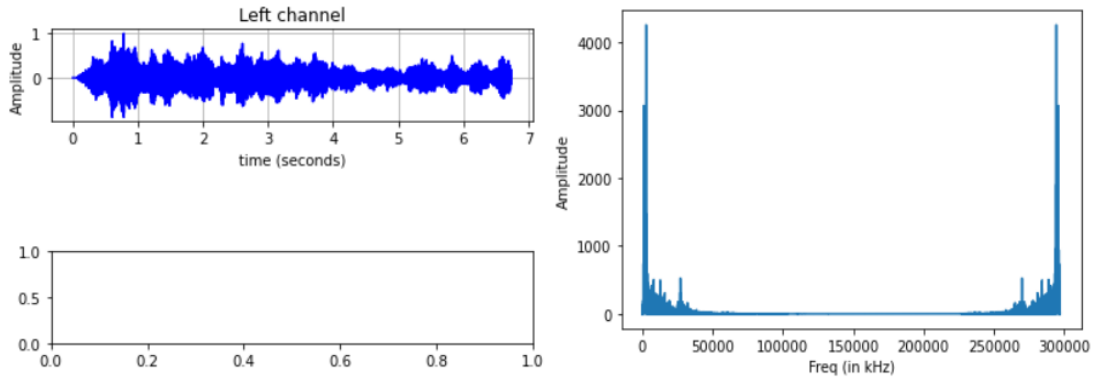
由音頻中雖然僅聽到薩可斯風的獨奏，並不像前面鋼琴音頻中清楚感受到雙聲道的存在，但由兩張圖表可看出雙聲道的左右聲道的分別(左圖)，雖然大致上並無太大明顯的區別，但經過傅立葉轉換後的頻譜圖(右圖)即可清楚看出兩聲道的不同，且相對於鋼琴的頻譜圖，兩聲道的差別極大。

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。