**CS3570 Introduction to Multimedia**

Homework #2 Report

103062234 張克齊

**Q1. (50%) Create your own FIR filters to filter audio signal**

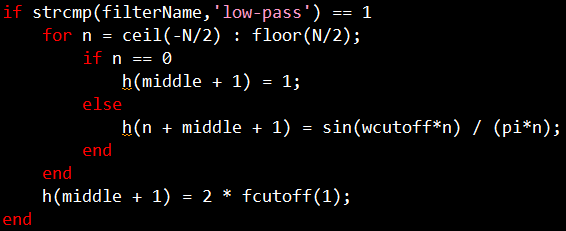
[實作方法]

**1. my\_filter.m**

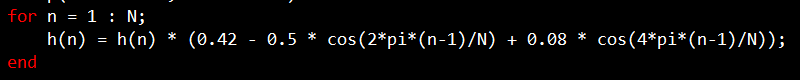
為主要實作的函式，傳入的變數中，功能為註解中所提到的，其中要注意的是，我傳入的fcutoff是利用vector的型式傳入，裡面有兩個值，在low-pass和high-pass中，只會用到第一項的值，傳入vector會像是 [500 0]；而當我們在實作band-pass和band-stop時，會需要下界和上界，因此需要兩個值，這時傳入的vector就會像是 [200 900]，而在呼叫時就能利用fcutoff(1)和fcutoff(2)代表第一和第二項。

依照講義slide #80中的步驟，我們首先要先對傳入的fcutoff做Normalization， ，並找出我們FIR size的中間點middle，而因為middle的值是奇數，這邊要再取floor()以不考慮remainder。

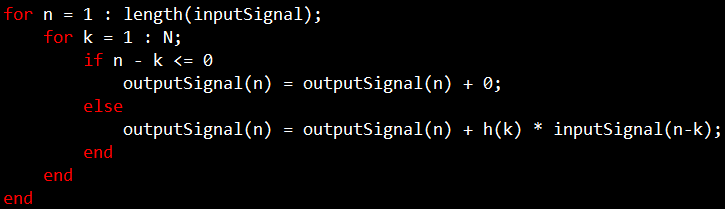
第二步是我們要依照講義slide #76公式做出我們可以使用的四種filter(low-pass, high-pass, bandpass, bandstop)，利用strcmp()的方式比較字串決定要用哪種filter，其中要小心matlab中matrix是從1開始所以要從0~N-1向右平移到1~N(index + 1)。下圖是low-pass的實作，其他三種改成對應公式的值即可。



第三步中，我們要決定我們的windowing function，在這邊用的是Blackman，其中要將slide #79公式中的'+'改成'-'，原因為ilms討論區說的範圍是從-N/2~N/2，這樣就能透過window得到可以處理input signal的filter。

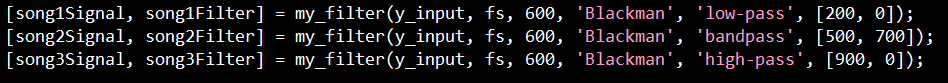


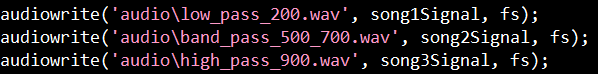
最後，利用slide #64中FIR公式對Input和filter做1D-convolution，其中要記得處理n-k<=0的情況(該項會為0)，就能得到最後filtered後的結果。



**2. HW2\_Q1.m**

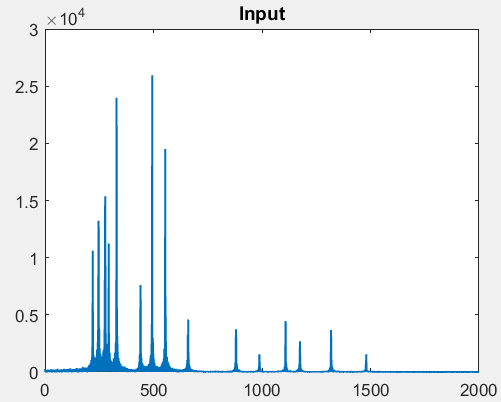
主程式中，主要是呼叫my\_filter函式得到結果後並畫圖，傳入的變數中，我N設成600，並分別利用low-pass, band-pass, high-pass分出三首歌，fcutoff要記得如同前面所述傳入vector，就能得到output的Signal和Filter；在畫圖方面，因為我們給的N為600，所以在畫time domain filter時我們可以把x軸限制在0~600間，方法為利用xlim([0 600])；後面畫frequency domain filer和output signal則可以仿效畫input的方法得到結果，最後再利用audiowrite()存下分離出來的三首歌。



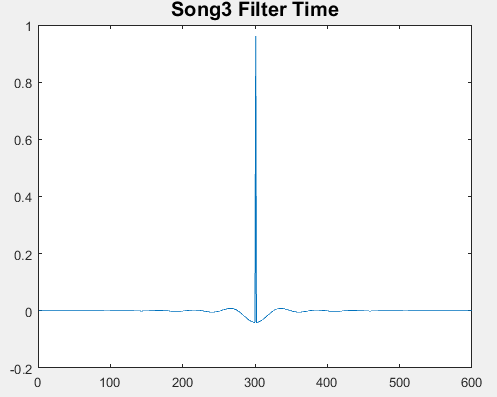
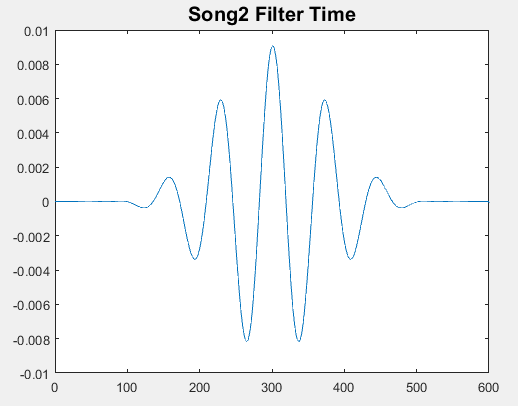
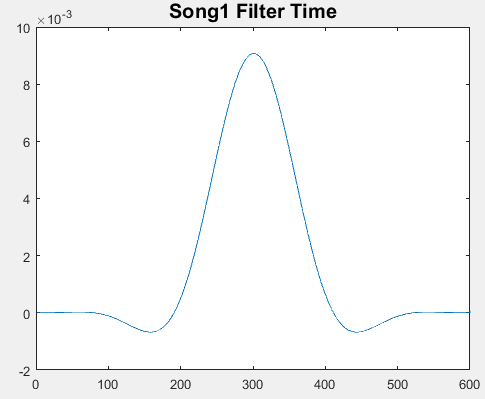


[結果圖]

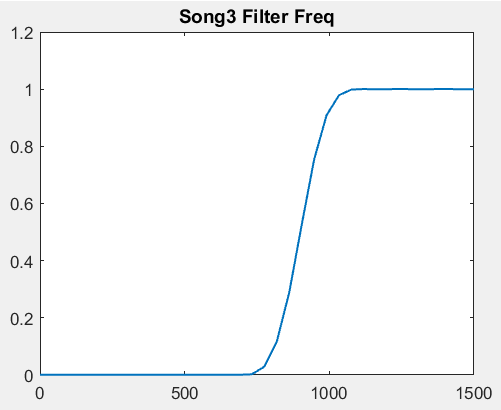
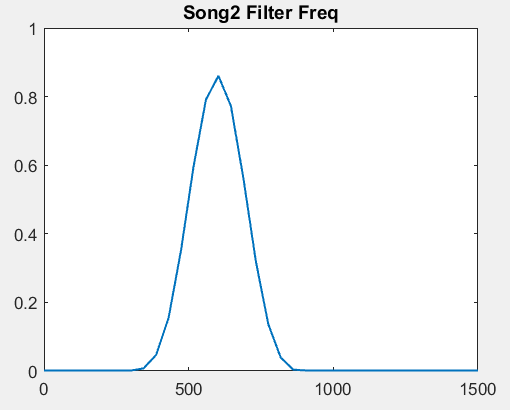
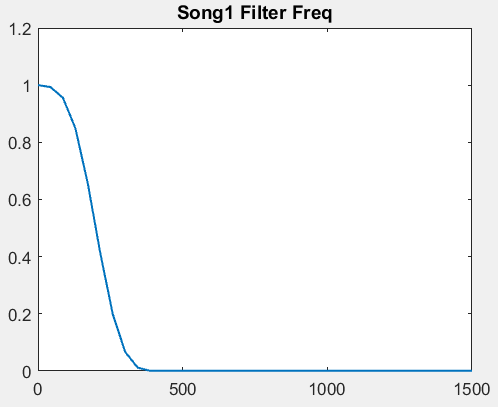
The spectrum of the input signal



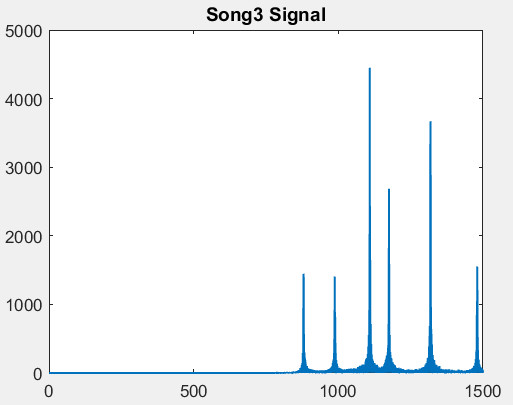
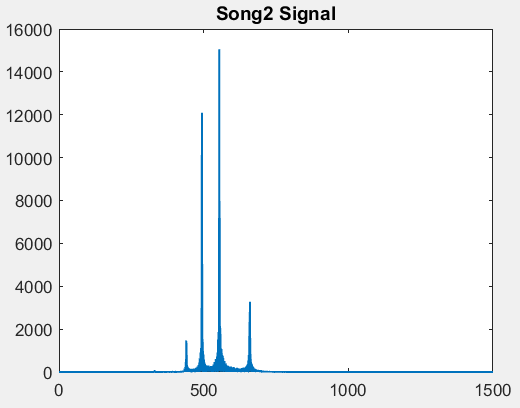
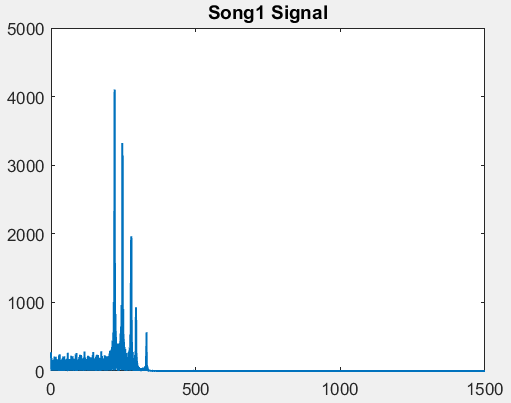
The shapes of the filters (time domain)



The spectrums of the filters



The spectrums of the output signals



[結論]

1. 決定filter的方法: 因為要分出三首歌，我自然就會想到可能分別在低頻、中頻、高頻的地方，所以我分別使用low-pass, band-pass, high-pass三種filter來分出三首歌，而其中的cutoff frequency是經過慢慢調整到沒有其他歌的介入為止。

2. filter與convolution的使用: 實作方法中有提到步驟，首先算出我們要用來convolution的filter，並給他我們要的window function(這邊是用Blackman)，因為這是一個有限的音頻，所以我們可以利用FIR對input和filter做convolution，這樣就能濾掉input的signal，達到分離歌曲的結果。

3. 圖形比較:

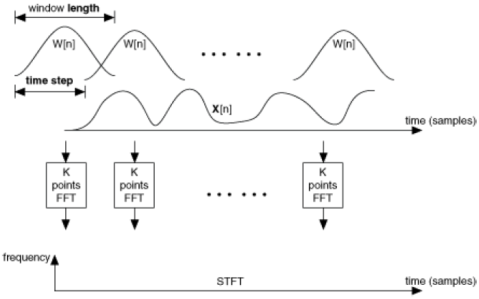
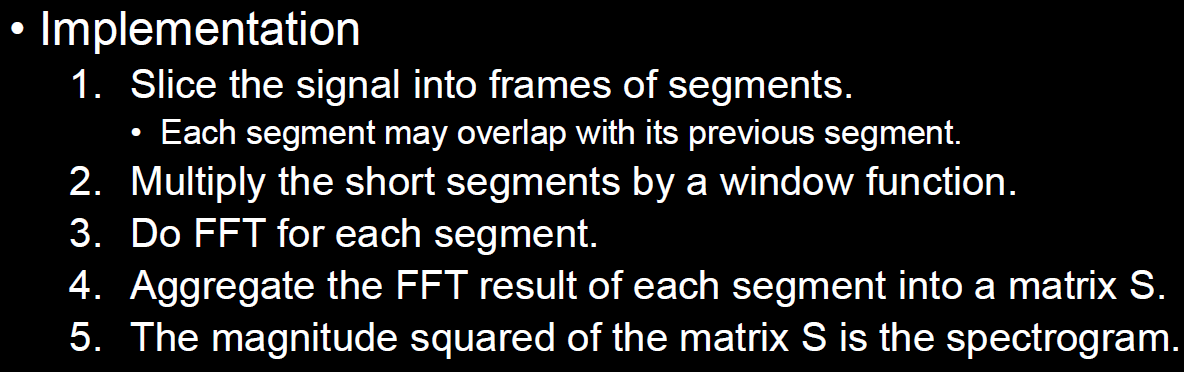
(1) Filter: 在Time Domain中，會得到這樣的圖是因為頻率域轉時域是利用sinc函數轉換，公式是。當我們取的頻率愈高時，會使中間的sinc週期增加，可以注意的是因為除上了pi\*x，所以當頻率愈高時我們可以發現y軸的值變小。在Frequency Domain的圖中，可以得到類似slide #68的理想圖形，但因為在計算時我們是利用sin函數，所以並不會很明顯的垂直向下(向上)，而是會有一些坡度的向下(向上)。

(2) Output signal: 根據不同的filter，可以發現對應不同的output結果，例如如果是low-pass，那就會發現input中只有低頻被保留，其他頻率的signal都會被濾掉。從上面input跟output signal的圖相互對照，效果是很好的。

**Q2. (50%) Music classification through spectrograms and human eyes:**

[實作方法]

**1. my\_stft.m**



在實作STFT時，主要是參照reference中所提示的步驟進行，再分別得到最後的S、T、F。我的作法是將切下的每一段做完處理並儲存後再處理下一段，也就是利用一個for迴圈來完成每一段的處理，但如上右圖所示知道要處理overlap，會變成，這樣就可以利用分割出每一段segment。

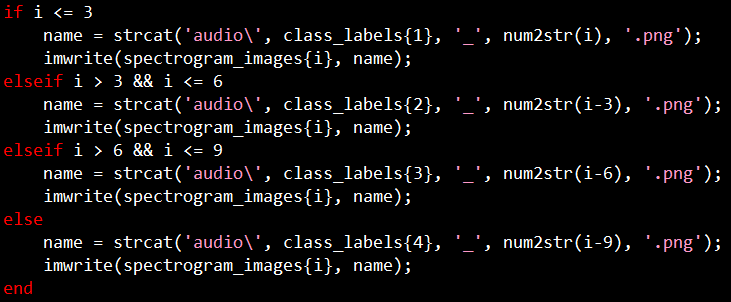
第二步中，跟第一題一樣要做window function的規劃，而因為這題沒有限制函式，所以我是直接使用hann()來做Hann window，並利用矩陣點乘的方法省略for迴圈的計算，會變成。接著第三步中，很簡單的可以利用fft()得到該slice轉換到頻域的結果；第四步中，就可以儲存得到的結果S和T，S是把經fft轉換的每一段的結果(變數FFT)存成n(slice數量) \* segment\_duration的matrix，T則是每一段slice的中點對應到原本的index，因此是個1 \* n(slice數量)的matrix，所以我設了一個變數now來決定每段slice的儲存位置。

算完存完每一段slice後，要將T中所有的數除上sample rate才是我們要的；最後，要決定我們的F，F代表它們的週期頻率，算法是，(now-1)會是slice的數量，這樣就能得到我們要的S、T、F了。

**STF公式參考網站**https://www.mathworks.com/matlabcentral/fileexchange/45197-short-time-fourier-transformation--stft--with-matlab-implementation

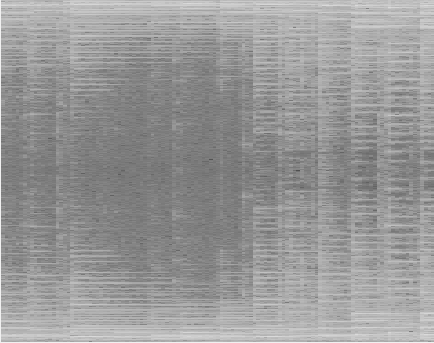
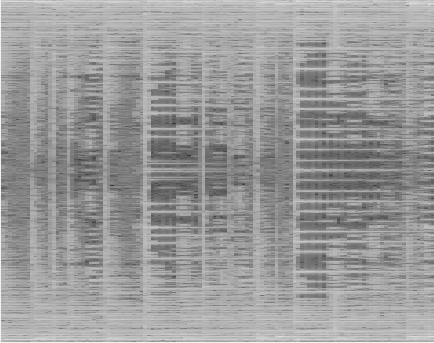
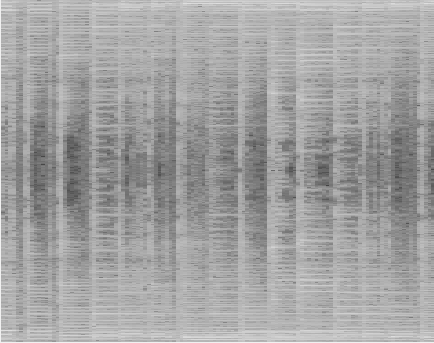
**2. HW2\_Q2.m**

在這邊我們要做的只有把做完STFT後的結果的spectrogram存成png檔，用的是imwrite()的方法，在命名我利用比較彈性的方法，用strcat()組成我想輸出檔名的字串，這樣就能利用原本給的class\_label來命名，如下圖。

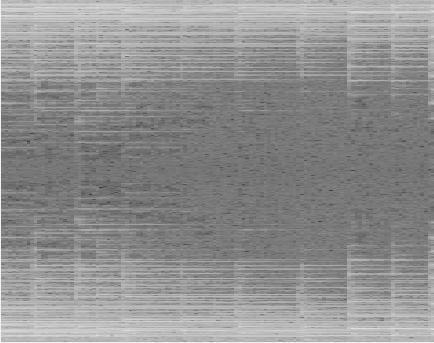
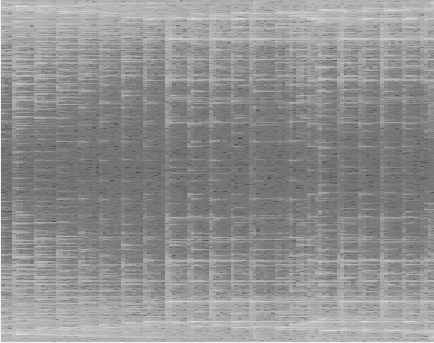
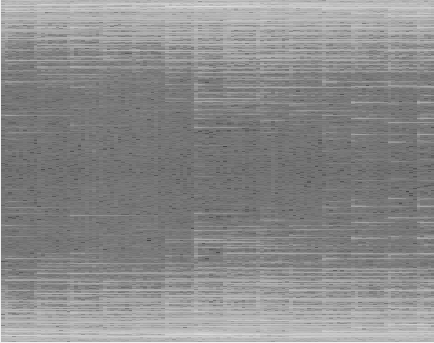


[結果圖]

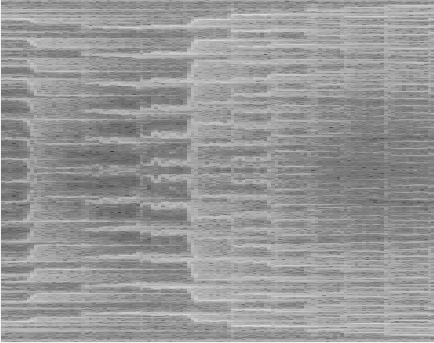
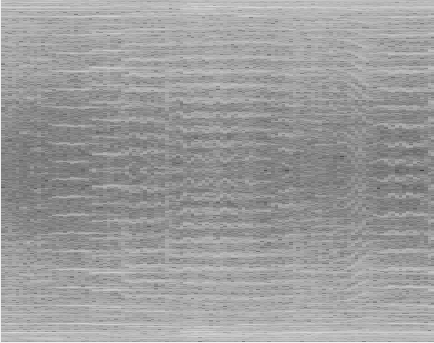
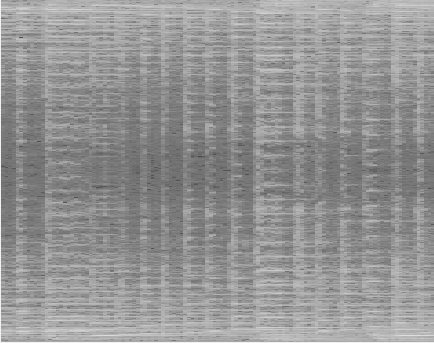
**Guitar**

****

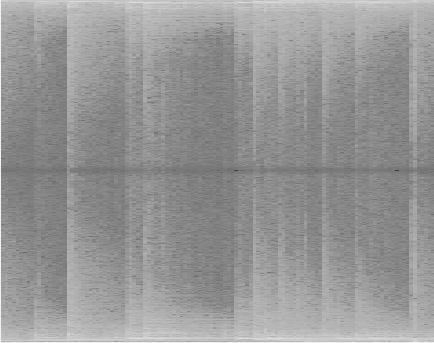
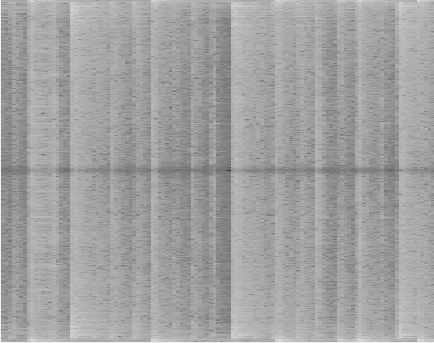
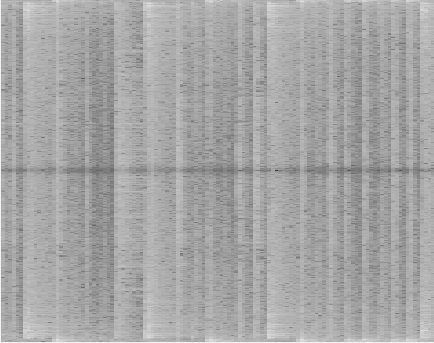
**Piano**

****

**Violin**

****

**Drum**

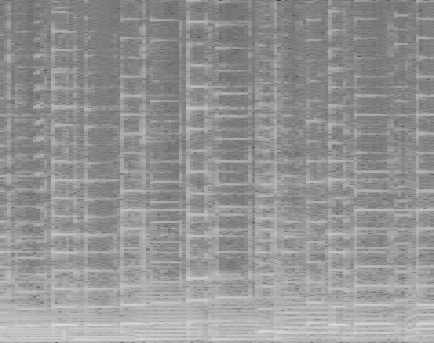
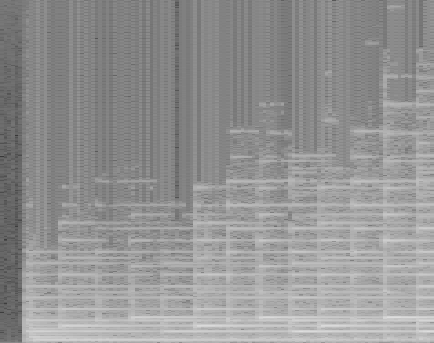


[比較]

1. 分辨不同樂器的spectrogram:

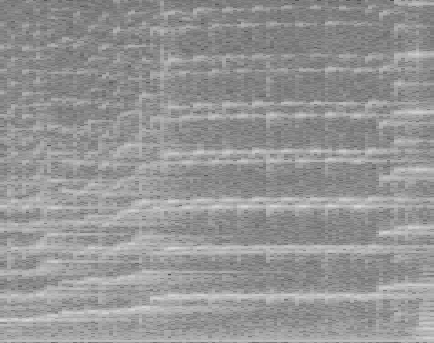
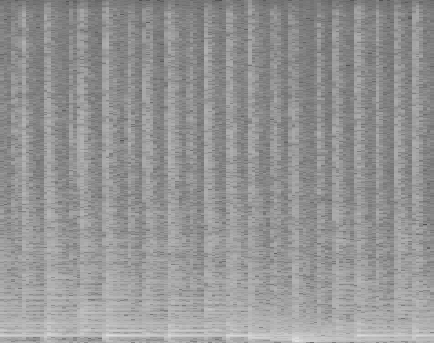
**test\_spectrogram\_1-> Piano** **test\_spectrogram\_2 -> Guitar**

排列整齊的格子狀 也是格子狀，但感覺比較不整齊

**test\_spectrogram\_3 -> Violin** **test\_spectrogram\_4 -> Drum**

右邊跟左邊的條紋規則不同但相像 很明顯的一條條直紋

2. 對STF transform的實作與心得: 在做STFT時，主要是依照reference中的步驟來完成，目的是將一個完整的音檔切成一段段的時間segment經轉換到頻域後並組合在一起，得到STF中的S，並利用T和F中的資訊畫出spectrogram，詳細實作方法在上面已說明。STFT跟DFT比較來說，STFT比DFT多了一個window function，可以分析出隨著時間變化的頻率，隨著window function大小的不同會有不同的頻率和時間解析度，再轉成spectrogram也能有更好的結果來比較，因此STFT被常常利用在聲音的比對。