# DSP Final Project - Paper 實作

b05902066 資工三 蔡翔陞

## 參考paper

### An Industrial-Strength Audio Search Algorithm

Avery Li-Chun Wang
avery@shazamteam.com
Shazam Entertainment, Ltd.

USA:
2925 Ross Road
Palo Alto, CA 94303

United Kingdom:
375 Kensington High Street
4th Floor Block F
London W14 8Q

Shazam 透過高精確度的音訊比對演算法，幫助使用者在短時間內利用環境內的片段旋律找出原曲，被譽為找歌神器，本次final project將實作Shazam公司發表的找歌演算法，並設法提升精準度。

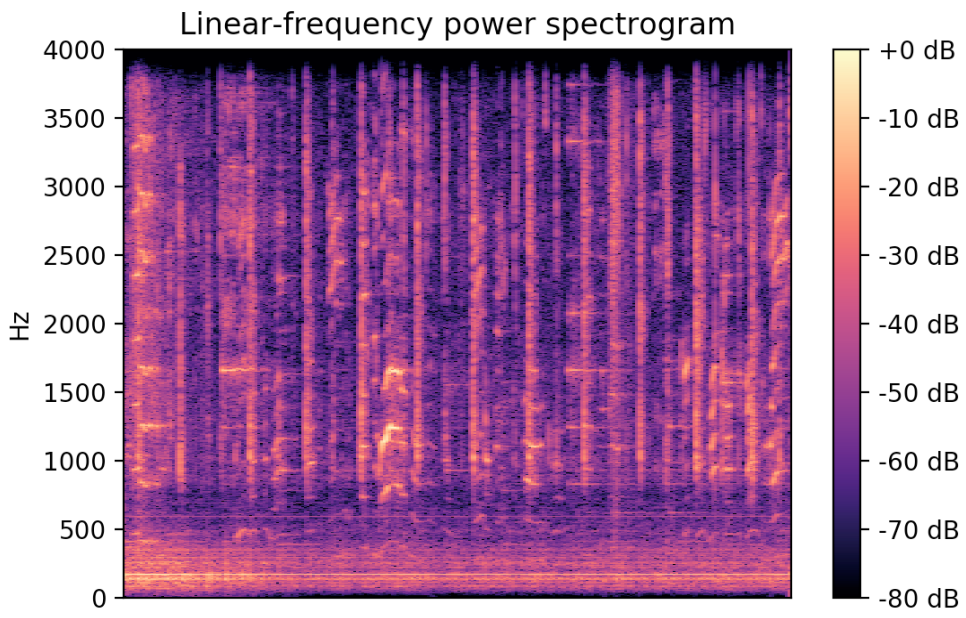
## 作法

1. 找出音訊特徵點(Star)
2. 產生hash entries
3. 將錄音與資料庫比對

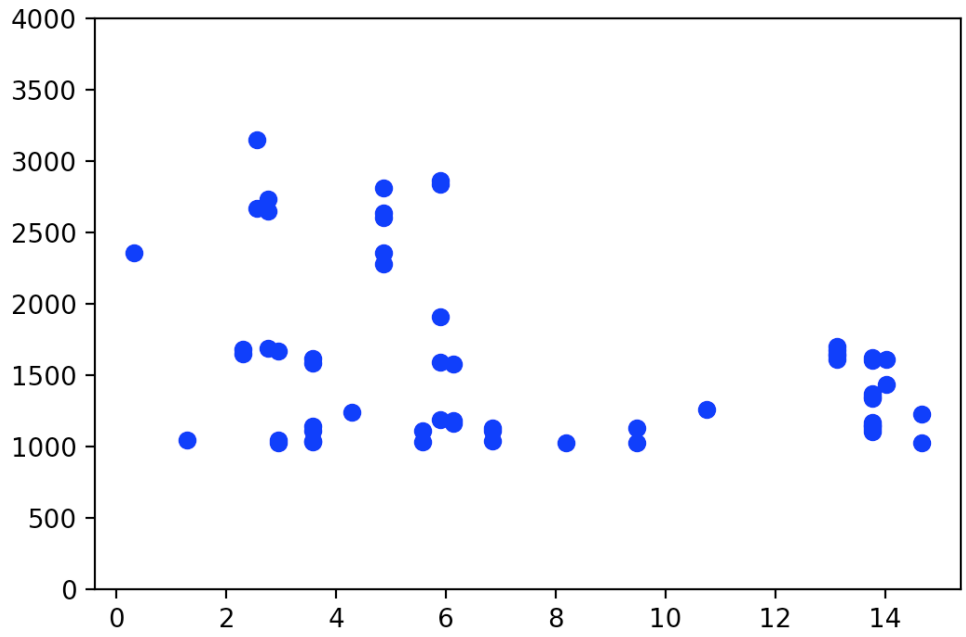
### 找出音訊特徵點(Star)

在將音訊經過Fourier transform並產生Spectrogram後，我們可以發現重要的資訊（說話、聲音特色）會保留在能量較大的特定時間與特定頻率上，因此在這個演算法中，將這些在Spectrogram上能量突出的點視為特徵點(Star)，因此可將原本的Spectrogram過濾成為Constellation Map。

**Spectrogram(x軸:時間, y軸:頻率)**



**Constellation Map(x軸:時間, y軸:頻率)**

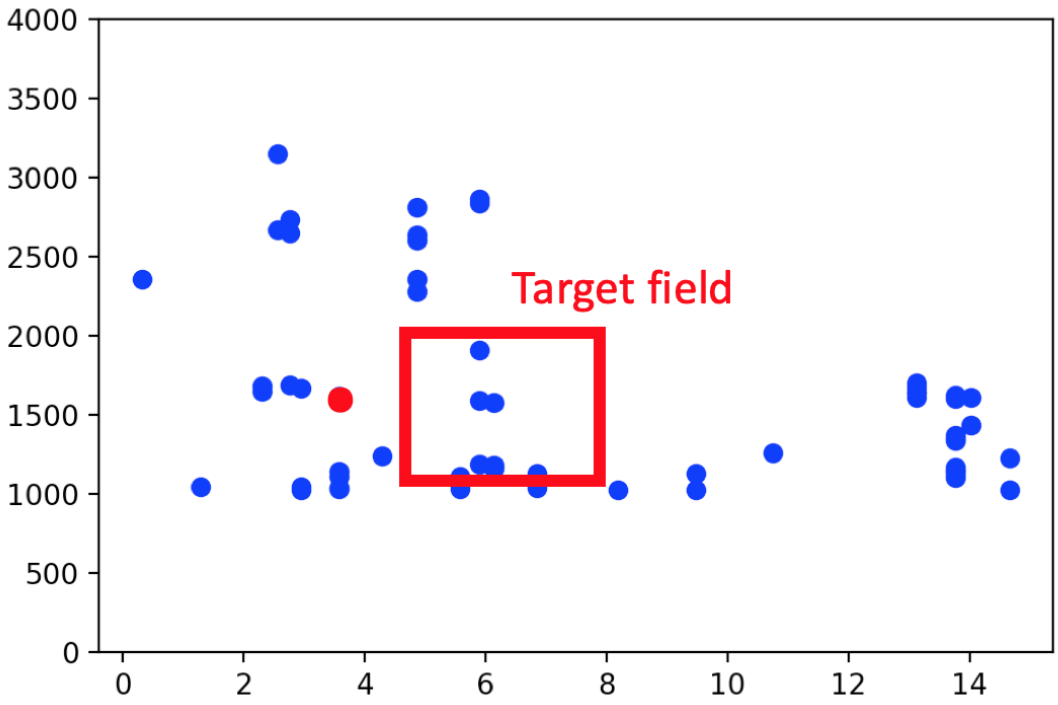



## 產生hash entries

由於每個音訊皆會產生獨一無二的constellation map，因此只要將data set與所得音訊的constellation map互相比對即可計算相似度，但由於：

1. 若要以圖找圖的方式比對，將會耗費大量時間
2. constellation map的star十分分散，map的空間遠大於star的個數
3. 音訊的特徵在時間與頻率相近的部分較具相關性

因此這個演算法進一步將constaellation map轉為hash list，將相鄰stars的資訊轉為hash entry儲存，減少儲存量與比對時的計算量。Paper中採用的方式為將每個star(座標：`(t1, f1)`)作為基準點，並與特定相對範圍中的star(座標：`(t2, t2)`)產生hash entry `[f1:f2:t2-t1]:t1` 。




## 將錄音與資料庫比對

將錄音運用相同的方式產生hash list，比對 `[f1:f2:t2-t1]` 是否相同，若相同則為hit，並同時將兩者的t1相減產生$dt$並記錄。

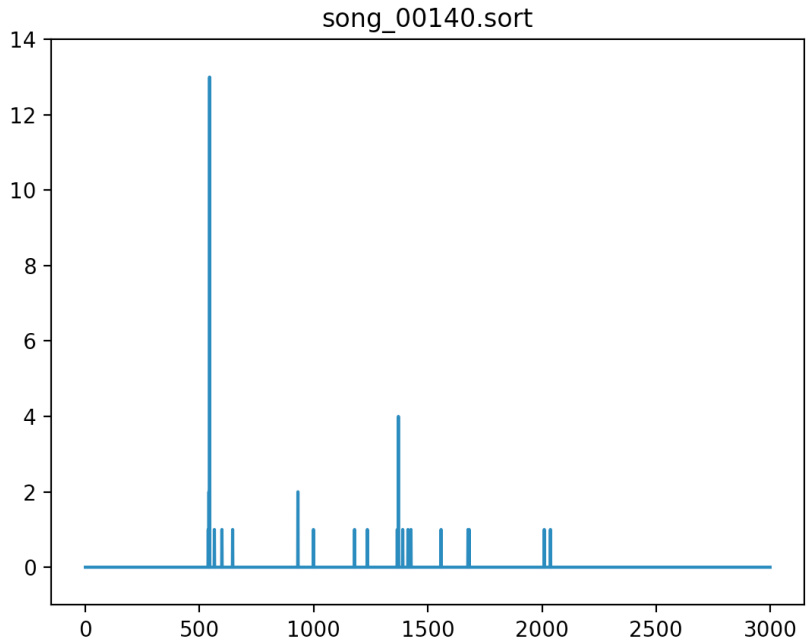
當某一錄音檔的片段出自$data_k$時，將預期產生一連串的hit，且這些hit的$dt$會近乎相同，因此若繪製成histogram將會在特定$dt$產生極大值。

因此我們可以利用此種性質，計算比對後的：

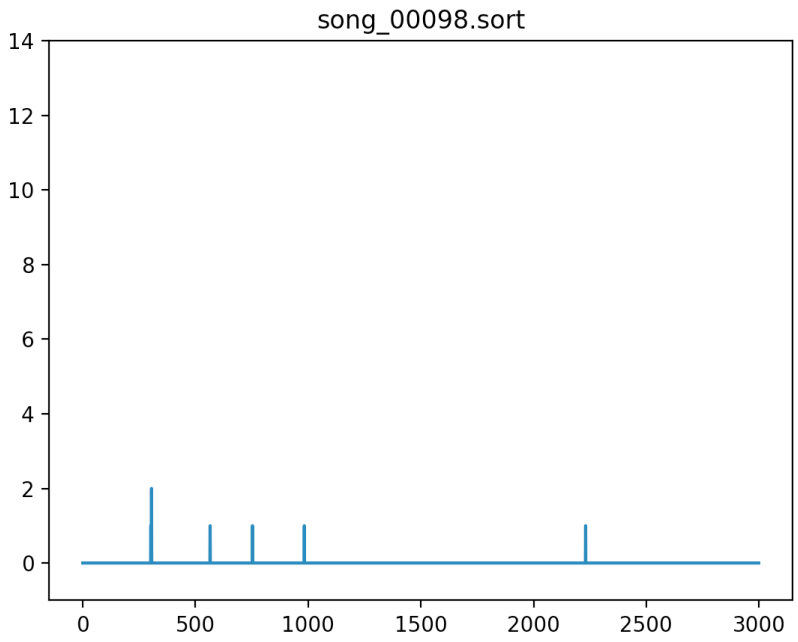
1. hit 數量
2. hit histogram pattern

即可有效判斷錄音與資料庫的相似程度，並進行評分，最後將data set依照分數進行排序，輸出結果。

**分數較高的分佈(x軸:dt, y軸:次數)**




**分數較低的分佈(x軸:dt, y軸:次數)**




## 實作上的trick

1. 將音訊down sample
   目前的錄音幾乎皆是以sample rate = 44100Hz，目的在於捕捉人耳所能接收的頻率(20000~20Hz)，然而一般的音樂並不會使用到如此高頻的頻率段(太過刺耳)，因此我在這次實作中將sample rate調整為8000Hz，能夠接收0~4000Hz的頻率，有效減少計算量並提高準確率。

2. 忽略1000Hz以下的頻率
   經過測試後發現收音後能量都集中在1000Hz以上的頻率段，猜測原因為音響對於低頻的播放效果有限，一般麥克風也對低頻的接收不佳，忽略後能有效降低計算量與增加準確率。

3. 適當容許頻率與時間差
   音響在播放音樂與麥克風在接收都時有可能會造成聲音變質(頻率改變)，因此在判斷hit時，可以適當容忍誤差，如此也能增加準確率。

## Future Work

1. 將錄音進行減噪
2. 使用平行化技術加快比對