中華大學資訊工程學系

100 學年度專題製作期末報告

音樂曲風辨識與應用

Identification and Application of Music Genre

專題組員: 陳淑華、陳璿、邵遵宇

指導老師: 周智勳老師

專題編號: PRJ2011-CSIE-10028

執行期間:100 年 3 月 至 101 年 6 月

目錄

一、摘要

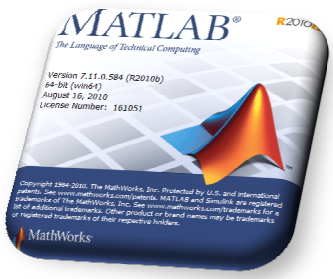
本專題使用 MATLAB 為開發平台，運用「梅爾倒頻譜係數」(Mel-frequency cepstral coefficients)及其改良的方法來算出用於語音辨識各曲風的特徵值，將音樂與各曲風的特徵值進行比對，用以辨識出該音樂之曲風。

二、簡介

由於現在的資訊科技發達，現在的人們聽音樂已經不再像以前購買黑膠唱片、

錄音帶或光碟片，取而代之的是在電腦上透過應用軟體與網際網路，就可以購買及下載到自己所喜愛的音樂，由於世界各地的音樂數量極為龐大，使用者在購買自己所喜愛的音樂後，如果想要從這麼龐大的資料中，找出和自己喜歡的音樂有著相似風格的音樂實在是不容易，而本專題正是為此而生，使用者可以從專題推薦的類似音樂找到其他可能會喜歡的音樂，造福了喜愛音樂的大眾。

三、專題進行方式



軟體
MathWorks MATLAB Software Version 7.11 (R2010b)

人員配置與職責	人員
專題分析、研究、規劃	
專案程式撰寫	陳淑華
曲風辨識率分析	陳　璿
成果報告&發表	邵遵宇

系統架構流程

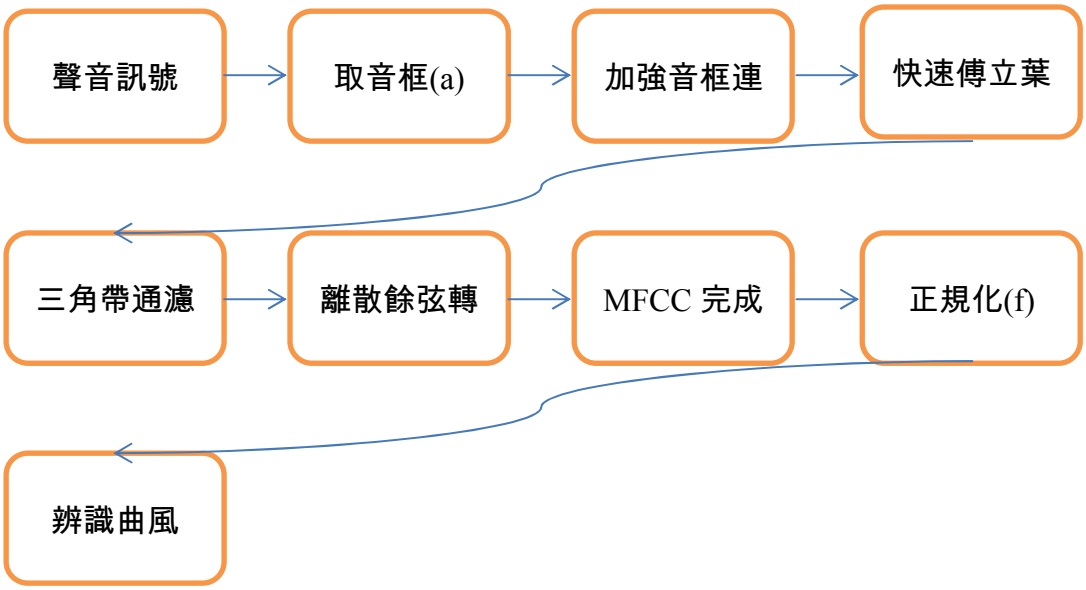


圖 3.1 系統架構流程圖

梅爾倒頻譜係數(Mel-frequency cepstral coefficients)

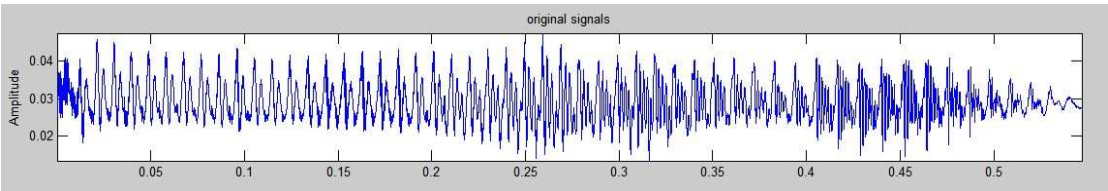


圖 3.2 原聲音檔

(a) 取音框（Frame blocking)

先將 N 個取樣點集合成一個觀測單位，稱為音框（Frame），通常 N 的

值是 256 或 512，本專題是採用 512 個取樣點。為了避免相鄰兩音框的變化過大，所以將兩相鄰音框之間重疊一段區域，此重疊區域包含了 M 個取樣點，通常 M 的值約是 N 的一半或 1/3，本專題是取 N 的一半值，也就是 256 個取樣點。通常語音辨識所用的音訊的取樣頻率為 8 KHz 或 16 KHz，以 8 KHz 來說，若音框長度為 512 個取樣點，則對應的時間長度是 512/8000*1000 = 64 ms。

程式碼 ：

```
waveFile='D:\H01.wav';
[y, fs]=wavread(waveFile);
y1=y(:,1);%取左聲道，右聲道(:,2);
n=length(y1);
t=(1:n)'/fs;
subplot(3,1,1);plot(t, y1,'-');
title('original signals');
xlabel(' '); ylabel('Amplitude');
axis  tight ;
```

(b) 加強音框連續性(Hamming window)

讓每個音框乘上漢明窗，目的是要加強音框左端和右端的連續性，這是因為在進行 FFT 時，都是假設一個音框內的訊號是代表一個週期性訊號，如果這個週期性不存在，FFT 會為了要符合左右端不連續的變化，而產生一些不存在原訊號的能量分佈，造成分析上的誤差。當然，如果我們在取音框時，能夠使音框中的訊號就已經包含基本週期的整數倍，這時候的音框左右端就會是連續的，那就可以不需要乘上漢明窗了。但是在實作上，由於基本週期的計算會需要額外的時間，而且也容易算錯，因此使用漢明窗來達到類似的效果。

將每一個分割出的音框分別乘上漢明窗，那麼乘上漢明窗後如下:

$$x[n] = s[n] \cdot w[n]$$

其中 x[n]為原始聲音序號，w[n]為窗函數(windows function)，本專題使用的窗函數為漢明窗，其公式如下:

$$w[n] = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{N-1}), & 0 \le n < N \\ 0 \quad , & \text{otherwise} \end{cases}$$

其中 N 代表著音框的大小，本專題用的值為 512。

程式碼 ：

```
% Plot of generalized Hamming windows
N=100;
n=(0:N-1)';
alpha=linspace(0,0.5,11)';
h=[];
for i=1:length(alpha),
        h = [h, (1-alpha(i))-alpha(i)*cos(2*pi*n/(N-1))];
end
plot(h);
title('Generalized Hamming Window: (1-\alpha)-\alpha*cos(2\pin/(N-1)), 0\leqn\leqN-1');

legendStr={};
for i=1:length(alpha),
        legendStr={legendStr{:}, ['\alpha=', num2str(alpha(i))]};
end
legend(legendStr);
```

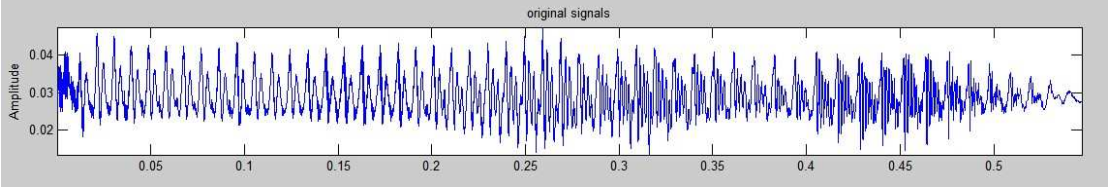


圖 3.3 為加強連續性

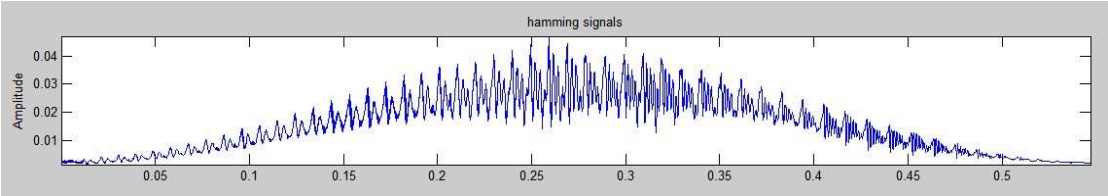


圖 3.4 加強連續性

(c) 快速傅立葉轉換（Fast Fourier Transform, or FFT）

由於訊號在時域（Time domain）上的變化通常很難看出訊號的特性，所以通常將它轉換成頻域（Frequency domain）上的能量分佈來觀察，不同的能量分佈，就能代表不同語音的特性。所以在加強音框連續性後，每個音框還必需再經過 FFT 以得到在頻譜上的能量分佈。

快速傅立葉轉換的公式如下:

$$X[k] = \sum_{n=0}^{N} e^{-j2\pi\frac{k}{N}n} \quad , \qquad 0 \le k < N$$

其中 X[k]是 X[n]的傅立葉轉換，k 代表第 k 的 Frequency Bin。

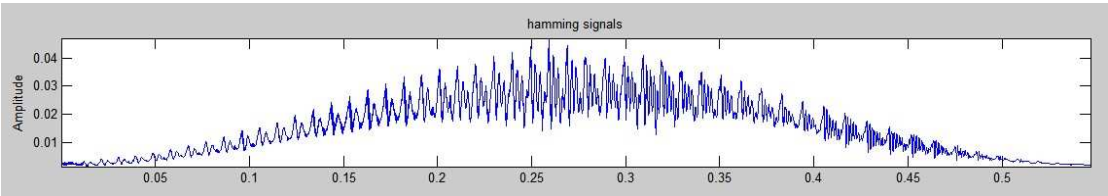


圖 3.5 傅立葉轉換前

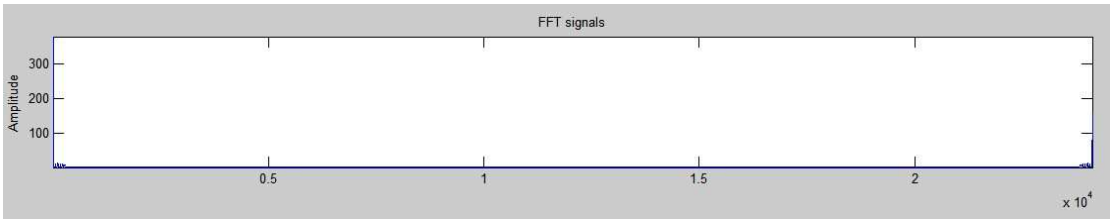


圖 3.6 傅立葉轉換後

(d) 三角帶通濾波器（Triangular Band-pass Filters）

使用三角帶通濾波器有兩個主要目的：

1)對頻譜進行平滑化，並消除諧波的作用，突顯原先語音的共振峰。因此一段語音的音調或音高，是不會呈現在 MFCC 參數內，換句話說，以 MFCC 為特徵的語音辨識系統，並不會受到輸入語音的音調不同而有所影響。

2)降低資料量。

人類的聽覺系統可接收的頻率範圍是 20 至 20000Hz 之間，但是人類的聽覺系統並非對每個頻率都有相同的敏感度。在低頻部份，人耳的感受比較敏銳，而在高頻部份，人耳對頻率變化量的感受就不是那麼明顯。在聲音辨識系統裡，最常使用的是一種較簡單且近似感知頻率刻度的轉換公式，稱之為梅爾頻率刻度(Mel-scal frequency)，其公式如下：

$$mel = 2595\log_{10}\left(\frac{f}{700}+1\right)$$

其中f為實際頻率，mel 為梅爾頻率。

梅爾頻率代表一般人耳對於頻率的感受度，由此也可以看出人耳對於頻率 f 的感受是呈對數變化的：

- 在低頻部分，人耳感受是比較敏銳
- 在高頻部分，人耳的感受就會越來越粗糙

如圖3.7所示，實際頻率與梅爾頻率在頻率小於1kHz時接近線性關係；當頻率大於1kHz時，兩者則呈現對數關係。

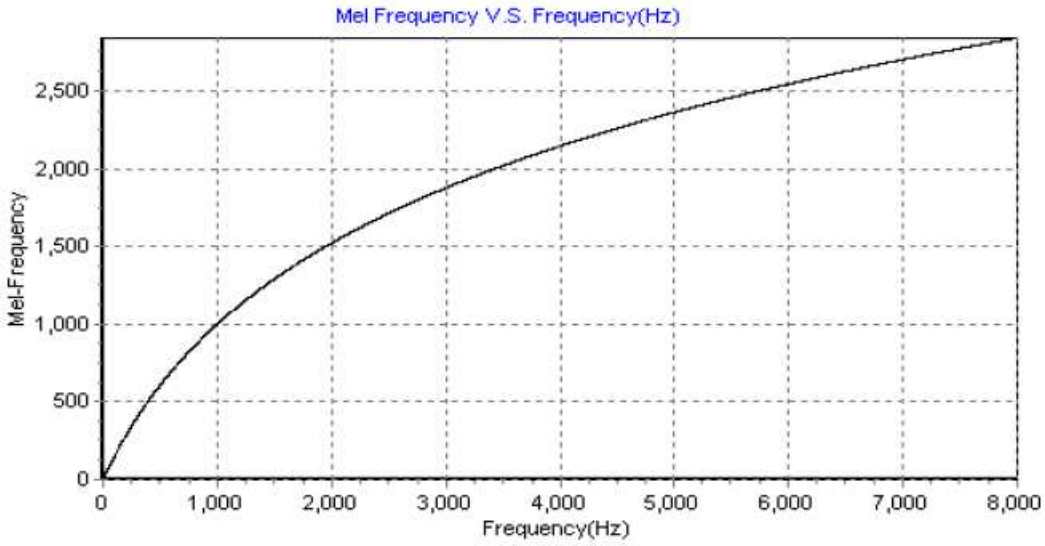


圖 3.7 實際頻率和梅爾頻率之間的關係圖

由圖 3.8 可看出三角帶通濾波器組是由多個三角帶通濾波器所組成，其設計上參照了人耳的特性，在低頻時三角濾波器組的間隔較密集，頻寬也較窄，而隨

著頻率增加，每個三角濾波器的間距與頻寬也隨之增加，模擬出人耳在低頻比高頻有更佳敏銳度的現象。利用濾波器來處理每個框內之訊號，即可獲得該框內訊號的頻譜能量值參數。

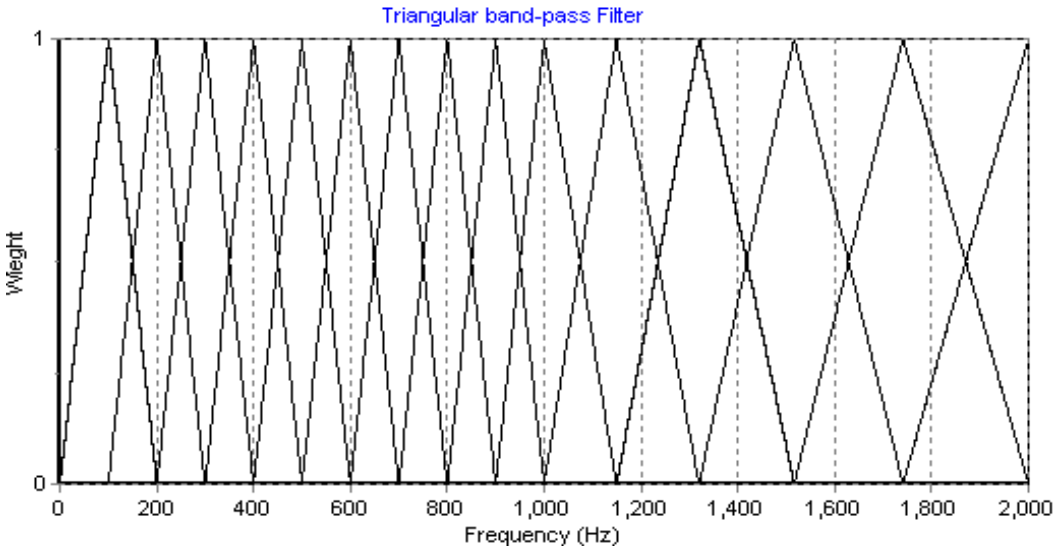


圖 3.8 三角帶通濾波器組模型

將快速傅立葉轉換後產生之頻譜圖經過三角帶通濾波器將訊號分割成數個頻帶併計算每個頻帶的能量值其公式如下 ：

$$A_k = \left|X[k]\right|^2, \quad 0 \le k < \frac{N}{2},$$

$$E_j = \sum_{k=0}^{\frac{N}{2}-1} \emptyset_j \ (k) A_k, \quad 0 \le j < J$$

E_j為頻帶的能量值，J 為三角帶通濾波器個數，本論文取 15 個，$\emptyset_j(k)$為第 j 個三角帶通濾波器振幅大小，A_k為頻譜能量值，由於訊號經過傅立葉轉換後成左右對稱，故計算至$\frac{N}{2}$範圍即可。

程式碼 ：

```
ak=sum;
xlabel('Time '); ylabel('Amplitude');
axis([0,512,0,15]);
ffn=44100/512;
 %----------------E1------------------
i=0; phi=0;
for i=1:255,
    k=i*ffn ;
        if 0 < k && k <= 100,
            phi=(1/100)*k*ak(i);
        elseif k > 100 && k < 200,
            phi=(-1/100)*(k-200)*ak(i);
        else k = 0 || k >= 200;
            phi=0; end;%phi0
            E1 = E1 + phi;
 end;
%----------------E2------------------
 i=0;phi=0;
 for i=1:255,
```

...

```
            phi=(1/206)*(k-1518)*ak(i);
        elseif k > 1724 && k < 1949,
            phi=(-1/225)*(k-1949)*ak(i);
        else k <= 1518 || k >= 1949;
            phi=0; end;
            E14 = E14 + phi;
 end;
    %----------------E15------------------
  i=0;phi=0;
  for i=1:255,
    k=i*ffn ;
        if 1724 < k && k <= 1949,
            phi=(1/225)*(k-1724)*ak(i);
        elseif k > 1949 && k < 2195,
            phi=(-1/246)*(k-2195)*ak(i);
        else k <= 1724 || k >= 2195;
            phi=0; end;
            E15 = E15 + phi;
 end;
```

(e)離散餘弦轉換(Discrete Cosine Transform,DCT)

將上述的 15 個對數能量 E_j帶入離散餘弦轉換，求出 L 階的 Mel- scale Cepstrum 參數，這裡 L 取 15。離散餘弦轉換公式如下：

$$c_i(m) = \sum_{j=0}^{J-1} \cos\left(m\frac{\pi}{J}(j+0{,}5)\log_{10}(E_j)\right)^{-}, \quad 0 \le m < L$$

由於之前作了 FFT，所以採用 DCT 轉換是期望能轉回類似 Time Domain 的情況來看，又稱 Quefrency Domain，其實也就是 Cepstrum。又因為之前採用 Mel- Frequency 來轉換至梅爾頻率，所以才稱之 Mel-scale Cepstrum。

離散餘旋轉換程式碼 ：

```
%========= C(m=0)
    Csum0 = 0;
    for j=1:15,
        C0=cos((0)*(pi/J)*((j-1)+0.5))*log10(Ej(j));
        %C0
        Csum0 = Csum0 + C0 ;
        %Csum
    end;
%========= C(m=1)
    Csum1 = 0;
    for j=1:15,
        C0=cos((1)*(pi/J)*((j-1)+0.5))*log10(Ej(j));
        %C0
        Csum1 = Csum1 + C0 ;
        %Csum
    end;
```

```
    end;
%========= C(m=13)
    Csum13 = 0;
    for j=1:15,
        C0=cos((13)*(pi/J)*((j-1)+0.5))*log10(Ej(j));
        %C0
        Csum13 = Csum13 + C0 ;
        %Csum
    end;
%========= C(m=14)
    Csum14 = 0;
    for j=1:15,
        C0=cos((14)*(pi/J)*((j-1)+0.5))*log10(Ej(j));
        %C0
        Csum14 = Csum14 + C0 ;
        %Csum
    end;
```

(f)正規化(Normalization)

由於算出來的各特徵值範圍可能會因為差距太大而受較大的值影響辨識率,因此使用正規化讓所有的值都介於 0~1 之間，其公式如下:

程式碼 ：

```
A=[0.011862682434 0.000086723549 -0.000145695949 -0.000114724184

%B=[10 11 12 13 5 6 7 8 9 14 15 1 2 3 4];

for i=1,

    Min=A(i,1);
    Max=A(i,1);
    for j=1:15
        if A(i,j) < Min
        Min = A(i,j);
        end;
        if A(i,j) > Max
        Max = A(i,j);
        end;
    end;
    for j=1:15
        A(i,j)=(A(i,j)-Min)/(Max-Min);
        A
    end;
        fid=fopen('D:\1minu.txt','a+');
        fprintf(fid,'%10.12f %10.12f %10.12f %10.12f %10.12f %10.
        fprintf(fid,';');
        fclose(fid);
end;
```

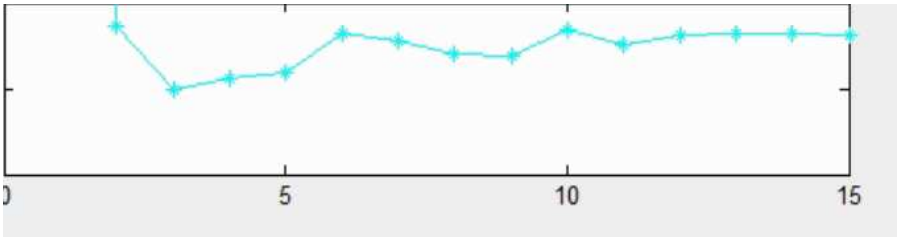


圖 3.9 使用正規化前的值

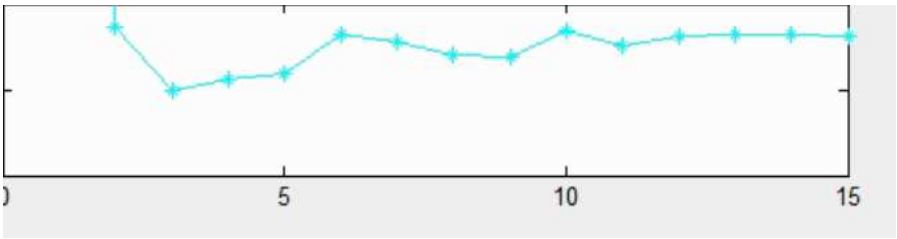


圖 3.10 使用正規化後的值

(g)曲風辨識：

(1) 將兩個音樂資料庫 Train_729 首及 Test_694 首(來源為網路上的 open source，每個音樂資料庫裡有六種曲風(資料庫已分類好音樂資料庫中每首歌的曲風)，分別為 Classical、Electronic、Jazz&Blue、Metal&Punk、Rock&Pop、World) 中的每首歌取其特徵係數及正規畫後，進而辨識分析歌曲曲風。

(2) 本專題辨識分析曲風方法為，將 Train 資料庫設定為樣本資料庫，Test 資料庫設定為測試資料庫，把 Test 測試資料庫中的 1 首歌其正規畫後之特徵係數與 Train 樣本資料庫中的 729 首歌做距離分析對比，分析 Test 1 首的距離與 Train 729 首中哪首距離最接近(即為距離為最小值)，以判斷 Test 為何種類型的曲風。例如：Test_17(第 17 首)與 Train_264(第 264 首)的距離最接近，Train_264 為 Classical 曲風，所以 Test_17 分析為 Classical 曲風。

程式碼 ：

```
for rp = 1 : 25 ,

    A=[1.000000000000000000 0.020599404524196563 0.000000000000000000 0.012645264766647809
        ];
CS=[Rock_Pop( rp,1),Rock_Pop( rp,2),Rock_Pop( rp,3),Rock_Pop( rp,4),Rock_Pop( rp,5),Rock_Pop( rp,
    ];
SUA_4=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

Total = 100;
for i=1:813
    SUA = [A(i,1)-CS(1,1),A(i,2)-CS(1,2),A(i,3)-CS(1,3),A(i,4)-CS(1,4),A(i,5)-CS(1,5),A(i,
           ;];
    SUA_2=SUA.^2 ;
    SUA_3 = sqrt(SUA_2);
    SUM = SUA_3(1,1)+SUA_3(1,2)+SUA_3(1,3)+SUA_3(1,4)+SUA_3(1,5)+SUA_3(1,6)+SUA_3(1,7)+SUA_

    if SUM < Total
        Total = SUM;
        Type = i;
     end;

     if 1 <= Type && Type <= 449,
         TYPE = 'Classical' ;
     elseif 450 <= Type && Type <= 556,
         TYPE = 'Electronic' ;
     elseif 557 <= Type && Type <= 582,
         TYPE = 'Jazz_blue' ;
     elseif 583 <= Type && Type <= 626,
         TYPE = 'Metal_punk' ;
     elseif 627 <= Type && Type <= 722,
         TYPE = 'Rock_pop' ;
     else 723 <= Type && Type <= 813,
         TYPE = 'World' ;
     end;

end;    fid=fopen('D:\Compare_Nor_Test_JazzBlue_25.txt','a+');
        fprintf(fid,'%d %d,%s ' , rp , Type ,TYPE );
        fprintf(fid,';');
        fclose(fid);

end;
```

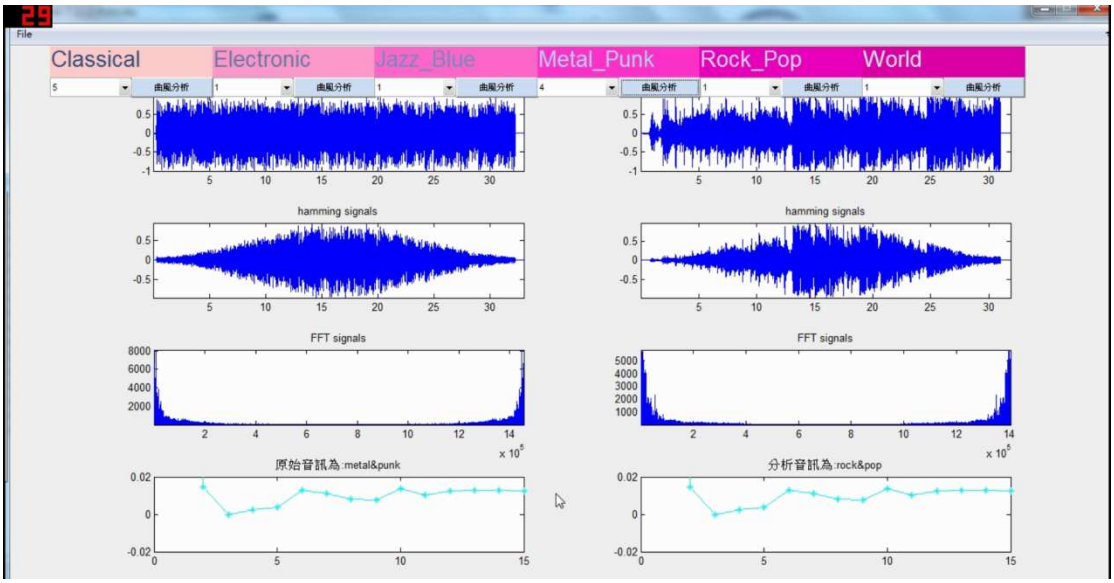


圖 3.11 曲風辨識結果圖

四、主要成果

音樂資料庫:

	Train	Test
Classical	320	310
Electronic	115	114
Jazz/Blue	26	26
Metal/Punk	45	45
Rock/Pop	101	98
World	122	101

表 4.1 訓練與測試音樂資料庫各曲風音樂數量

分析結果：

	Classical	Electronic	Jazz_Blue	Metal_Punk	Rock_Pop	World
Classical	284	6	1	2	4	13
Electronic	19	58	3	3	17	14
Jazz_Blue	0	6	8	7	5	0
Metal_Punk	0	13	0	20	11	1
Rock_Pop	8	27	1	13	45	4
World	34	15	2	6	11	33

表 4.2 曲風分析結果

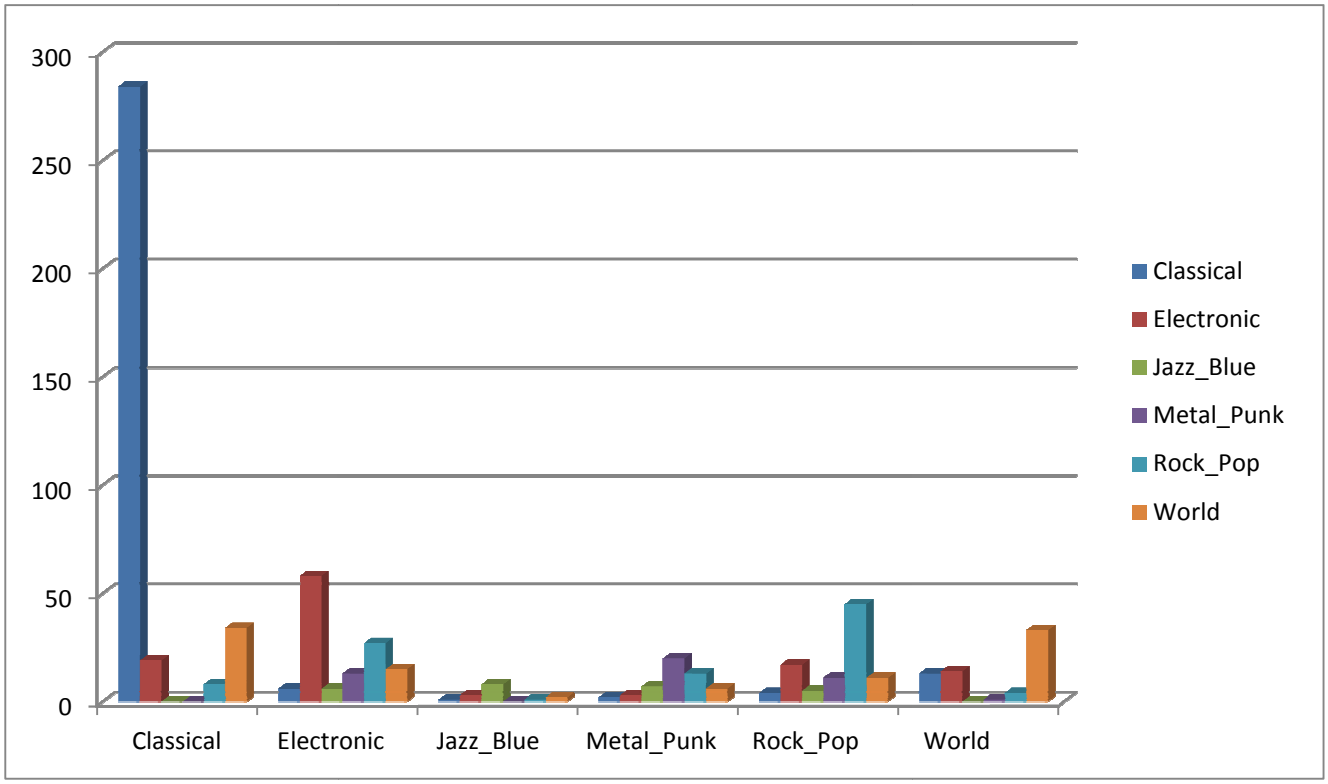


圖 4.1 使用 MFCC 正規化後統計長條圖

	辨　識　成　功	總　共　首　數
Classical	284	310
Electronic	58	114
Jazz/Blue	8	26
Metal/Punk	20	45
Rock/Pop	45	98
World	33	101
總共結果	448	694

表 4.3 使用 MFCC 正規化後統計表

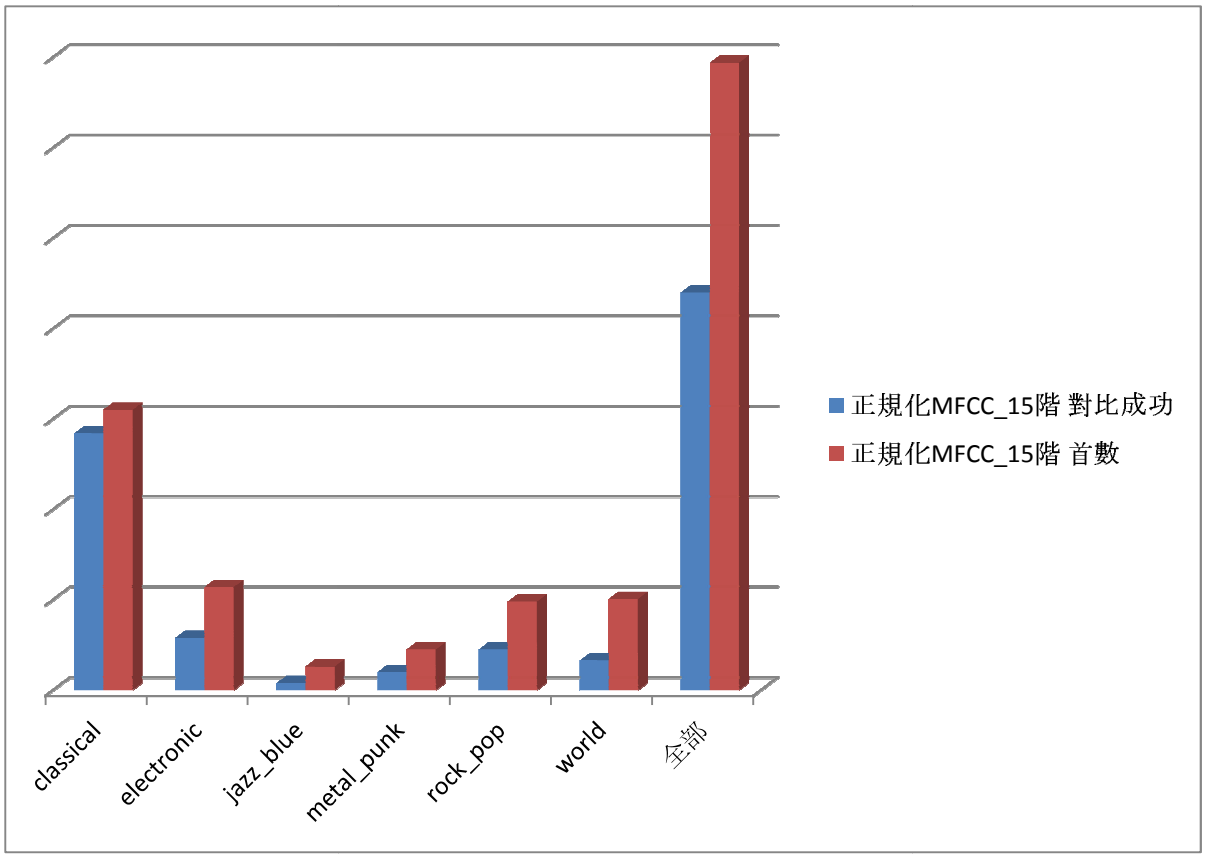


圖 4.2 使用 MFCC 正規化後統計長條圖

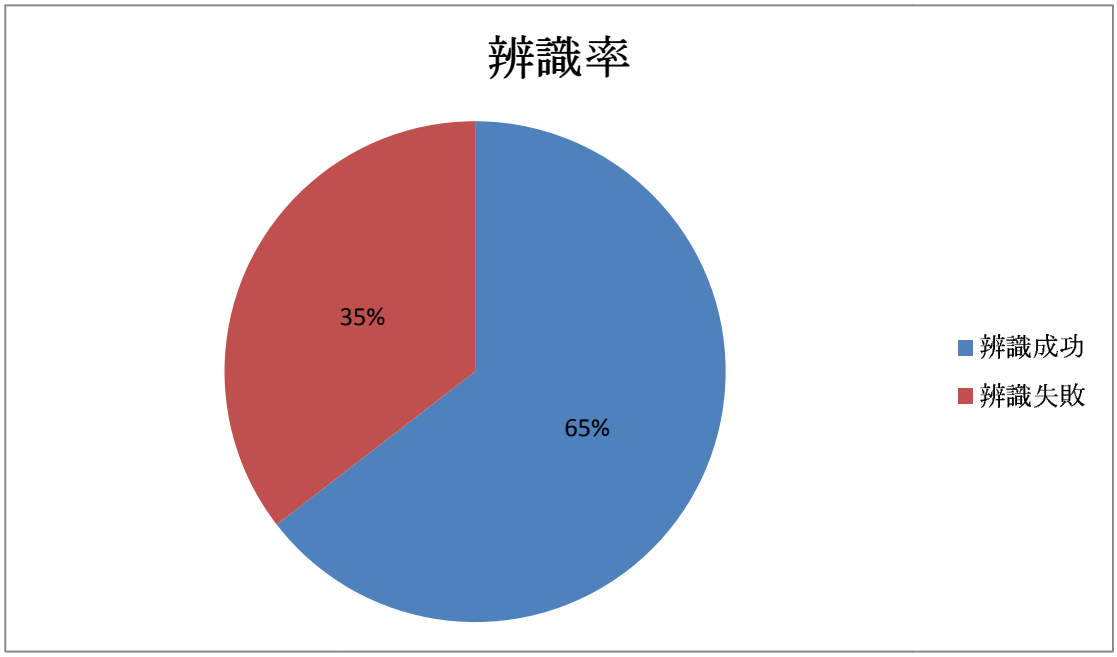
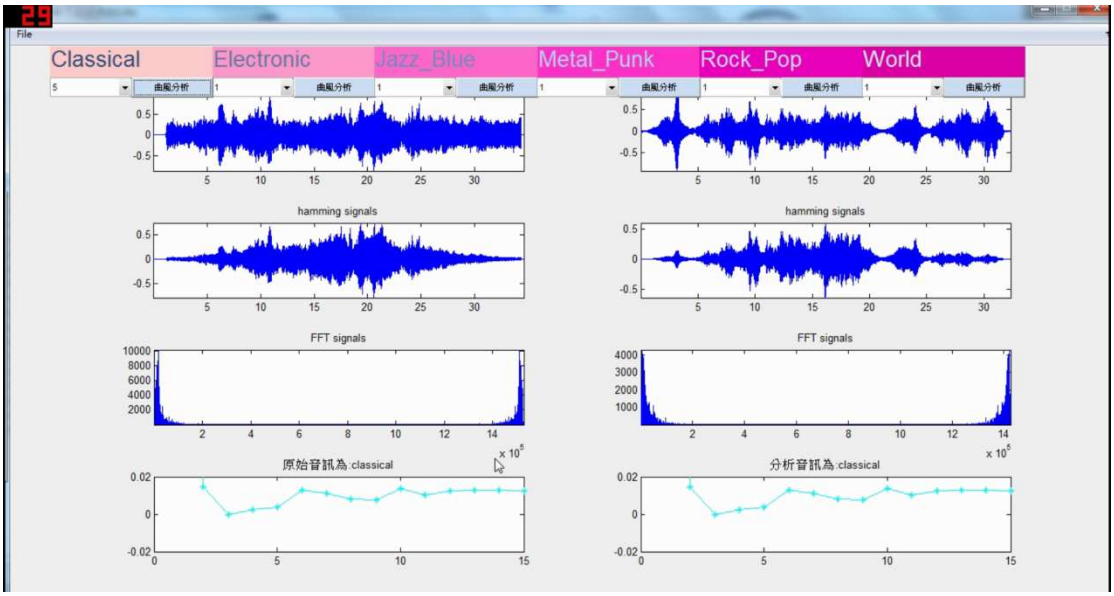


圖 4.3 使用 MFCC 正規化後辨識率圓餅圖

介面設計



五、評估與展望

在我們所製作的專題中，我們最終選擇使用”正規化 MFCC”方法辨識曲風，實驗中，使用梅爾倒頻譜係數的平均值與標準差對 694 首的測試音樂進行辨識，辨識率達到 65%，目前因使用的方法辨識率並不高，未來可以尋找更多的方法實驗，進而提升辨識率。

六、結語

專題過程中遇到了許多的難題，從開始的讀取音訊檔，還有MFCC 中的加強音框連續性、對音訊檔快速傅立葉轉換，一個困難的數學式完成後，接著另一個複雜的三角帶通濾波器，及離散餘弦轉換的數學式，但是老師非常有耐心的一一指導公式，細心的解釋多遍給我們聽，最終得以完成本專題。

七、銘謝

首先非常感謝指導老師周智勳老師提供許多的資源與協助，假日更是犧牲自己的時間與我們一同奮戰，也感謝周老師實驗室的學長姊們的協助，感謝他們願意花時間指點我們的疑惑，也感謝他們提供的音樂資料庫，最後感謝一路走來的同組夥伴們。

八、參考文獻

[1] 張智星 “MATLAB 程式設計入門篇” 碁峰,2011

[2] 洪維恩“MatLab 7 程式設計” 旗標,2005

[3] 李顯宏 “Matlab 介面開發與編譯技巧 第二版” 文魁,2008

[4] Roger Jang (張智星)

http://mirlab.org/jang/books/audioSignalProcessing/