Code :

Project3_cluster4.m (Main 主要執行) : (附註 : Matlab 需安裝 Audio System Toolbox)
(附註 : 會跑約半小時左右)

```
clear 'all';
close 'all';

[y1,fs1] = audioread('µù¥U»yªÌ01.wav'); %read auio
file
[coeffs1,delta1,deltaDelta1,loc1] = mfcc(y1,fs1);
%calculate MFCC1(coeffs1)

[y2,fs2] = audioread('µù¥U»yªÌ02.wav'); %read auio
file
[coeffs2,delta2,deltaDelta2,loc2] = mfcc(y2,fs2);
%calculate MFCC2(coeffs2)

[y3,fs3] = audioread('µù¥U»yªÌ03.wav'); %read auio
file
[coeffs3,delta3,deltaDelta3,loc3] = mfcc(y3,fs3);
%calculate MFCC3(coeffs3)

[y4,fs4] = audioread('µù¥U»yªÌ04.wav'); %read auio
file
[coeffs4,delta4,deltaDelta4,loc4] = mfcc(y4,fs4);
%calculate MFCC4(coeffs4)

[idx1, C1, sum1, D1] = kmeans(coeffs1,25) %VQ1 based
on k-means use 25 cluster
[idx2, C2, sum2, D2] = kmeans(coeffs2,25) %VQ1 based
on k-means use 25 cluster
[idx3, C3, sum3, D3] = kmeans(coeffs3,25) %VQ1 based
on k-means use 25 cluster
[idx4, C4, sum4, D4] = kmeans(coeffs4,25) %VQ1 based
on k-means use 25 cluster

[y,fs] = audioread('¹ï¸Ü¤º®e.wav'); %read auio file
```

```
[coeffs,delta,deltaDelta,loc] = mfcc(y,fs);
%calculate MFCC(coeffs)

matrix = zeros(length(coeffs),1) %the matrix save
result(speaker diarization)
matrix_p = zeros(4,length(coeffs)) %the matrix save
P(Fm|Sn)
prob1 = 0 %User1's prob
prob2 = 0 %User2's prob
prob3 = 0 %User3's prob
prob4 = 0 %User4's prob
TA = 0 %T * A
TB = 0 %T * B
TC = 0 %T * C
TD = 0 %T * D
TT = 0 %T.^2
AA = 0 %A.^2
BB = 0 %B.^2
CC = 0 %C.^2
DD = 0 %D.^2

for i = 1:length(coeffs)
  for p = 1 : 25
    prob1 = 0
    prob2 = 0
    prob3 = 0
    prob4 = 0
    TA = 0
    TB = 0
    TC = 0
    TD = 0
    TT = 0
    AA = 0
    BB = 0
    CC = 0
    DD = 0
    for q = 3:14
      TA = TA + coeffs(i,q) * C1(p,q)
      TB = TB + coeffs(i,q) * C2(p,q)
      TC = TC + coeffs(i,q) * C3(p,q)
      TD = TD + coeffs(i,q) * C4(p,q)
      TT = TT + coeffs(i,q).^2
      AA = AA + C1(p,q).^2
      BB = BB + C2(p,q).^2
      CC = CC + C3(p,q).^2
      DD = DD + C4(p,q).^2
    end

    prob1 = TA / (sqrt(TT) * sqrt(AA)) %Calulate
cosine similarity of Frame & User1
    prob2 = TB / (sqrt(TT) * sqrt(BB)) %Calulate
cosine similarity of Frame & User2
    prob3 = TC / (sqrt(TT) * sqrt(CC)) %Calulate
cosine similarity of Frame & User3
    prob4 = TD / (sqrt(TT) * sqrt(DD)) %Calulate
cosine similarity of Frame & User4

    if(prob1 > matrix_p(1,i)) %Find the largest
similarity
      matrix_p(1,i) = prob1
    end
    if(prob2 > matrix_p(2,i)) %Find the largest
similarity
      matrix_p(2,i) = prob2
    end
    if(prob3 > matrix_p(3,i)) %Find the largest
similarity
      matrix_p(3,i) = prob3
    end
    if(prob4 > matrix_p(4,i)) %Find the largest
similarity
      matrix_p(4,i) = prob4
    end
  end
end
```

```
for i = 3:length(matrix) - 2
  %Smooth the similarity
  prob1 = (matrix_p(1,i - 2) + matrix_p(1,i - 1) +
matrix_p(1,i) + matrix_p(1,i + 1) + matrix_p(1,i +
2)) / 5
  prob2 = (matrix_p(2,i - 2) + matrix_p(2,i - 1) +
matrix_p(2,i) + matrix_p(2,i + 1) + matrix_p(2,i +
2)) / 5
  prob3 = (matrix_p(3,i - 2) + matrix_p(3,i - 1) +
matrix_p(3,i) + matrix_p(3,i + 1) + matrix_p(3,i +
2)) / 5
  prob4 = (matrix_p(4,i - 2) + matrix_p(4,i - 1) +
matrix_p(4,i) + matrix_p(4,i + 1) + matrix_p(4,i +
2)) / 5

  %Update the similarity
  matrix_p(1,i - 2) = prob1
  matrix_p(1,i - 1) = prob1
  matrix_p(1,i) = prob1
  matrix_p(1,i + 1) = prob1
  matrix_p(1,i + 2) = prob1

  matrix_p(2,i - 2) = prob2
  matrix_p(2,i - 1) = prob2
  matrix_p(2,i) = prob2
  matrix_p(2,i + 1) = prob2
  matrix_p(2,i + 2) = prob2

  matrix_p(3,i - 2) = prob3
  matrix_p(3,i - 1) = prob3
  matrix_p(3,i) = prob3
  matrix_p(3,i + 1) = prob3
  matrix_p(3,i + 2) = prob3

  matrix_p(4,i - 2) = prob4
  matrix_p(4,i - 1) = prob4
  matrix_p(4,i) = prob4
  matrix_p(4,i + 1) = prob4
  matrix_p(4,i + 2) = prob4
end

for i = 1:length(matrix)
  %Find the biggest similarity of different user and
save result
  if (matrix_p(1,i) > matrix_p(2,i)) &&
(matrix_p(1,i) > matrix_p(3,i)) && (matrix_p(1,i) >
matrix_p(4,i))
    matrix(i) = 1
  elseif (matrix_p(2,i) > matrix_p(1,i)) &&
(matrix_p(2,i) > matrix_p(3,i)) && (matrix_p(2,i) >
matrix_p(4,i))
    matrix(i) = 2
  elseif (matrix_p(3,i) > matrix_p(1,i)) &&
(matrix_p(3,i) > matrix_p(2,i)) && (matrix_p(3,i) >
matrix_p(4,i))
    matrix(i) = 3
  elseif (matrix_p(4,i) > matrix_p(1,i)) &&
(matrix_p(4,i) > matrix_p(2,i)) && (matrix_p(4,i) >
matrix_p(3,i))
    matrix(i) = 4
  end
end

num1 = 0
num2 = 0
num3 = 0
num4 = 0
%Smooth the picture
for i = 1:length(matrix)
  %Calculate the count of different user
  if (matrix(i) == 1)
    num1 = num1 + 1
  elseif (matrix(i) == 2)
    num2 = num2 + 1
  elseif (matrix(i) == 3)
    num3 = num3 + 1
  elseif (matrix(i) == 4)
    num4 = num4 + 1
  end

  %Based on 100 frame
  if(mod(i,100) == 0)
    %The most count is the user
    if(num1 > num2) && (num1 > num3) && (num1 > num4)
      matrix(i - 99:i) = 1
    elseif(num2 > num1) && (num2 > num3) && (num2 >
num4)
      matrix(i - 99:i) = 2
    elseif(num3 > num1) && (num3 > num2) && (num3 >
num4)
      matrix(i - 99:i) = 3
    elseif(num4 > num1) && (num4 > num2) && (num4 >
num3)
      matrix(i - 99:i) = 4
    end
    num1 = 0
    num2 = 0
    num3 = 0
    num4 = 0
  end

  %if the end can't mod 100
  if(i == length(matrix))
    %The most count is the user
    if(num1 > num2) && (num1 > num3) && (num1 > num4)
      matrix(i - mod(length(matrix),100):i) = 1
    elseif(num2 > num1) && (num2 > num3) && (num2 >
num4)
      matrix(i - mod(length(matrix),100):i) = 2
    elseif(num3 > num1) && (num3 > num2) && (num3 >
num4)
      matrix(i - mod(length(matrix),100):i) = 3
    elseif(num4 > num1) && (num4 > num2) && (num4 >
num3)
```

```
      matrix(i - mod(length(matrix),100):i) = 4
    end
    num1 = 0
    num2 = 0
    num3 = 0
    num4 = 0
  end
end

% for compare the time and speaker separately, so we
can see the graph more clearly
figure(1),
for i = 1:length(matrix)
  if matrix(i) == 1
    plot(i,matrix(i),'b+');
    hold on;
  elseif matrix(i) == 2
    plot(i,matrix(i),'r+');
    hold on;
  elseif matrix(i) == 3
    plot(i,matrix(i),'g+');
    hold on;
  elseif matrix(i) == 4
    plot(i,matrix(i),'y+');
    hold on;
  end
end
text(1600,1.9,'b+ : user1');
text(1600,1.7,'r+ : user2');
text(1600,1.5,'g+ : user3');
text(1600,1.3,'y+ : user4');
hold off;

%the final result
figure(2),
for i = 1:length(matrix)
  if matrix(i) == 1
    plot(i,1,'b+');
    hold on;
  elseif matrix(i) == 2
    plot(i,1,'r+');
    hold on;
  elseif matrix(i) == 3
    plot(i,1,'g+');
    hold on;
  elseif matrix(i) == 4
    plot(i,1,'y+');
    hold on;
  end
end
text(1600,1.9,'b+ : user1');
text(1600,1.7,'r+ : user2');
text(1600,1.5,'g+ : user3');
text(1600,1.3,'y+ : user4');
hold off;
```

程式碼解說：

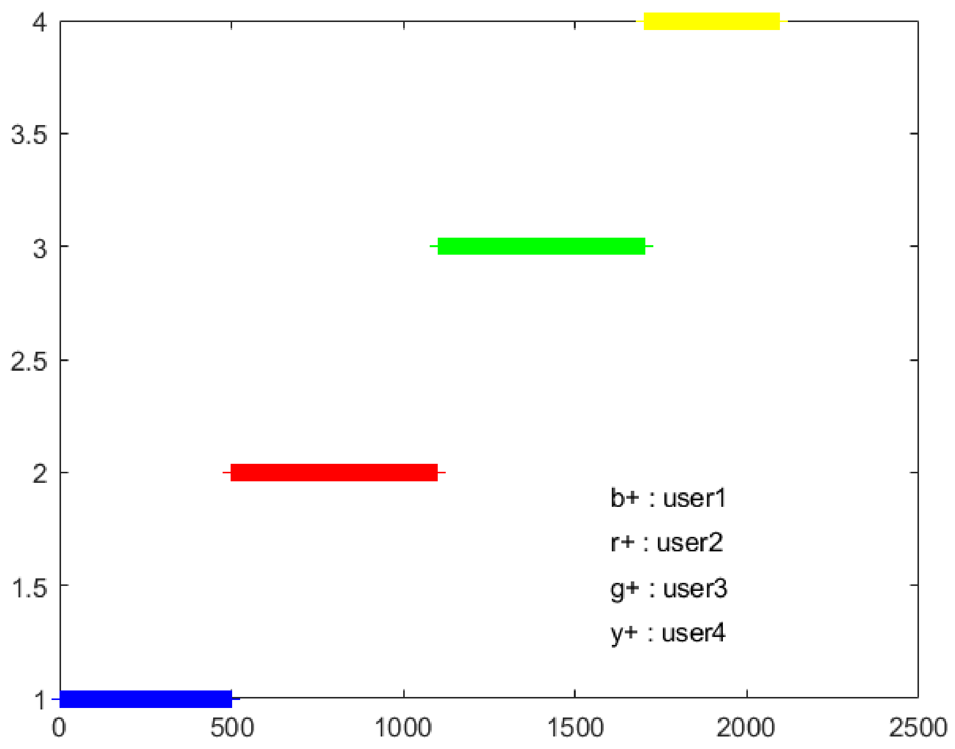
首先，先分別將四位註冊者的資料進行讀取，然後再利用 mfcc()計算 MFCC，再利用 kmeans 取出 centroid，並另用這些 centroid 直接作為 VQ。

然後讀取對話內容，一樣先計算出 MFCC，然後將每個 frame 與四位使用者的 25 個 VQ 做 cosine similarity，並於各 25 個中取最大值當作當前的機率，這部分為”**speaker detect**”

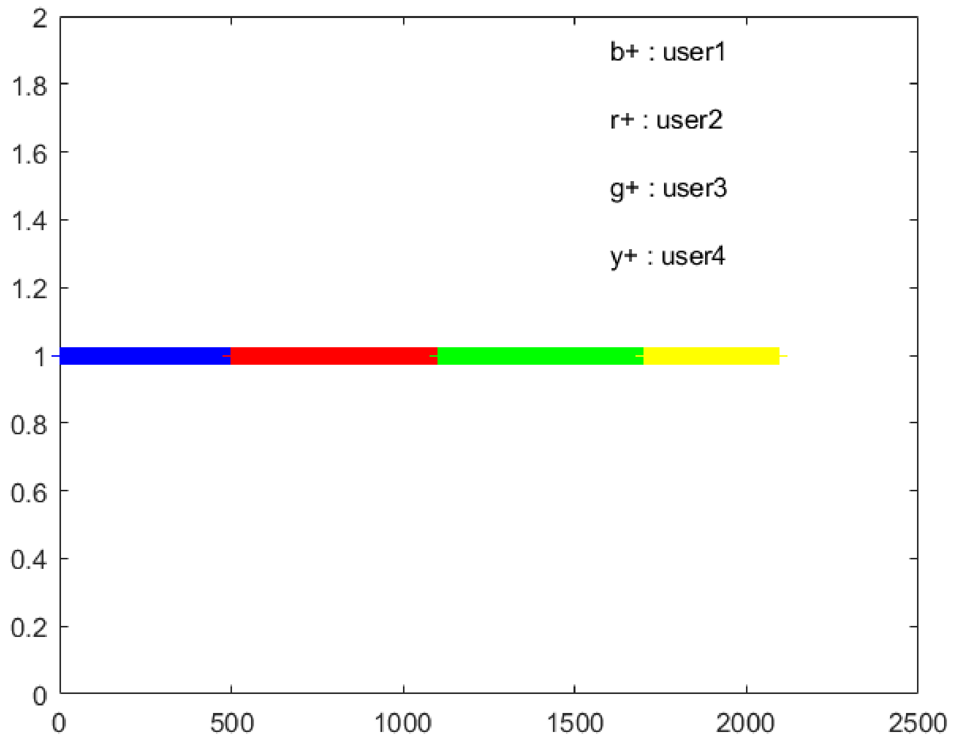
等全部都取好之後，再將機率取前後各兩個加自身做 smooth，這部分為”**smooth**”

接著再將根據得到的最大機率來判斷與哪位使用者相似度最高，並將其存起來；然後在每 100 個 frame 裡面去比較，看哪一個 user 次數最多，就將那 100 個 frame 都換成那個 user，最後再將圖印出來，共有兩張，第一張是將四位使用者分開來，比較明顯看出誰在哪一段(result1.png)，第二章則是將其合起來(result2.png)

Result :



4
3.5
3
2.5
2
1.5
1
0
500
1000
1500
2000
2500
b+ : user1
r+ : user2
g+ : user3
y+ : user4



2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
b+ : user1
r+ : user2
g+ : user3
y+ : user4