

NTU

# 生醫工程實驗期末專題報告

手語辨識系統實作

第一組

B02901145 黃以瑄 B02901109 陳意雯 B02901057 王子毅



105-1

# 目錄

● 動機 .....	1
● 實驗設計 .....	1
一、 製作 database .....	1
二、 即時單手手語判斷 .....	5
三、 即時雙手手語判斷 .....	7
● 結果分析與未來展望 .....	9
● 心得 .....	10
參考資料 .....	11
附錄 .....	12

## ● 動機

手語是一種不使用語音，而用手勢、肢體動作等來表達意思的特殊語言，在聾人的社群中，手語是相當常見的溝通方式，但大部分的聽人並沒有學習手語，這也就造成了這兩個社群間的隔閡。我們這組的組員中有兩個人學過手語，也有認識一些聾人，就我們所知台灣並沒有任何可以翻譯手語的工具，頂多只有可以查詢的手語字典。因此希望可以做出一個藉由影像辨識來翻譯手語的系統，讓不會手語的人也可以更沒有障礙的跟聾朋友溝通。

## ● 實驗設計

(所有程式皆放在 **GitHub** 上，[https://github.com/b02901145/bio\\_final](https://github.com/b02901145/bio_final)，並附於報告後的附錄中)

分成製作 database、即時單手手語判斷、即時雙手手語判斷三個部分

### 一、製作 database

儘管世界各國的手語有許多相似的地方，手語仍然跟一般的語言一樣，在不同地區會有不同的用法，所以我們在網路上可以找到的資料大多是國外的，與台灣的手語不同，所以我們打算自己製作一個手語的 database。

因為不同人的手部形狀差異不大，所以我們假設“個體間的差異並不會影響手語的判斷”，而先以同一個人的照片來製作 database，之後經過測試發現個體差異果然不會影響判斷的結果。下面我們將製作 database 分為三個部分。

#### 1. 選定要判斷的手型

手型就像英文字母一樣，是構成不同手語的基礎，我們從台灣的手語中，選了 **20 個常見的手型**，如下圖 1-1 所示。分別是一、二、三、四、五、六、七、八、九、手、方、童、錢、龍、WC、守、百、四十、零、借，共二十個。在之後判斷的過程中，我們將這些手型分別稱為 1-one, 2-two, 3-three, 4-four, 5-five, 6-six, 7-seven, 8-eight, 9-nine, 10-hand, 11-square, 12-child, 13-money, 14-dragon, 15-wc, 16-wait, 17-hundred, 18-fourty, 19-zero, 20-borrow。



圖 1-1: 我們選擇的手型

## 2. 拍攝照片

我們為每一個手型各拍攝 50 張手的角度略有不同的照片，總共一千張，分別存在名為 1, 2, 3, 4, ..., 20 等 20 個資料夾中。如圖 1-2 所示。

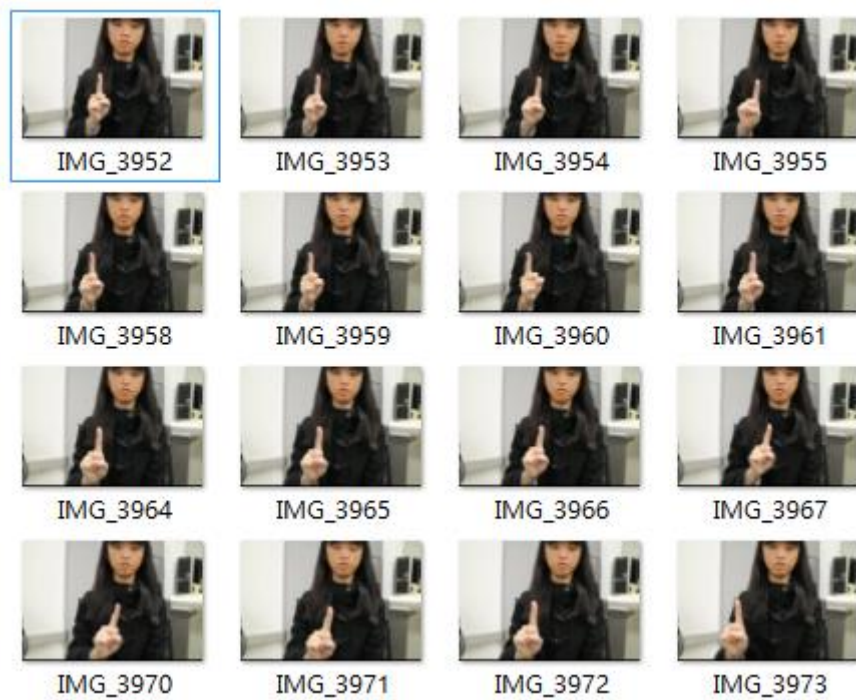


圖 1-2: '1-one' 的部分 database，可以看出圖中手的角度略有不同。

### 3. 處理照片 (svm.m)

為了加快判斷手語的速度還有去除人臉、背景、衣服、膚色等對判斷的影響，我們將圖片中手的部分擷取下來，並將三維的 RGB 圖片轉成 YCbCr 的圖片，只保留亮度 Y，最後將處理好的結果存成一維的照片。

#### (1) 找出膚色:

我們用 SVM(support vector machine)找出照片中屬於膚色的部分，如圖 1-3 所示。



圖 1-3: 將圖片中膚色的部分標示為 1，非膚色部分標示為 0 的結果

#### (2) 找出手:

從圖 1-3 可以看出找到的膚色區塊有手、臉、跟一些零星的點，我們在拍攝的過程中，保持手的位置比臉低，因此我們可以計算膚色區塊的大小，保留較大的兩塊中，位置比較低的那塊，結果如圖 1-4 所示。

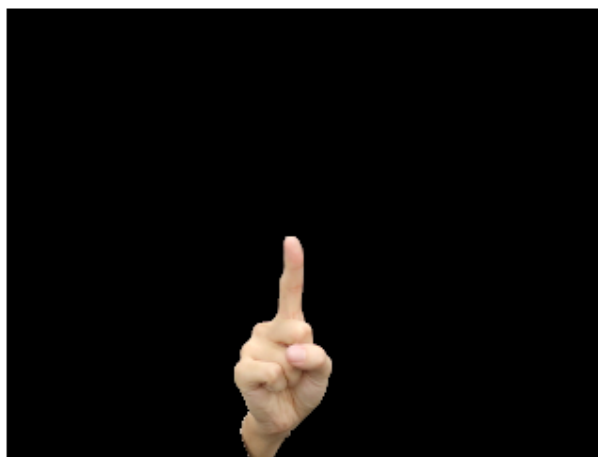


圖 1-4: 僅保留照片中手的部分

#### (3) 裁切圖片:

為了方便之後的處理，我們找出手的區塊的座標邊界，並將圖片裁

切成與長邊相同大小的正方形，且只保留亮度的資訊，得到如圖 1-5 的圖片後，將所有處理好的圖片存進資料夾 1, 2, 3, ..., 20 中。



圖 1-5: 處理好的圖片

## 二、即時單手手語判斷

這部分的程式是用 python 寫的，主要分成 **training** 和 **testing** 兩個部份。

### 1. Training

#### (1) 調整圖片大小 ([resize.py](#))

Dataset 中的圖片雖然都是正方形的，但大小並不固定，所以我們先將所有圖片大小調整為固定的 80\*80 並儲存。

#### (2) 分割 training data 和 testing data ([train.py](#))

Preprocess 後的 data 為 256 階 gray scale，大小 80\*80 的手部圖片，共 20 種手型，每種 50 張。其中 4/5 當作 training data，1/5 當作 validation data。

#### (3) CNN(Convolutional neural network) model ([train.py](#))

在 train.py 的程式中，我們利用 Python 的 neural network library “Keras” 建構一個 CNN(convolutional neural network)的 model。如圖 2-1 所示。

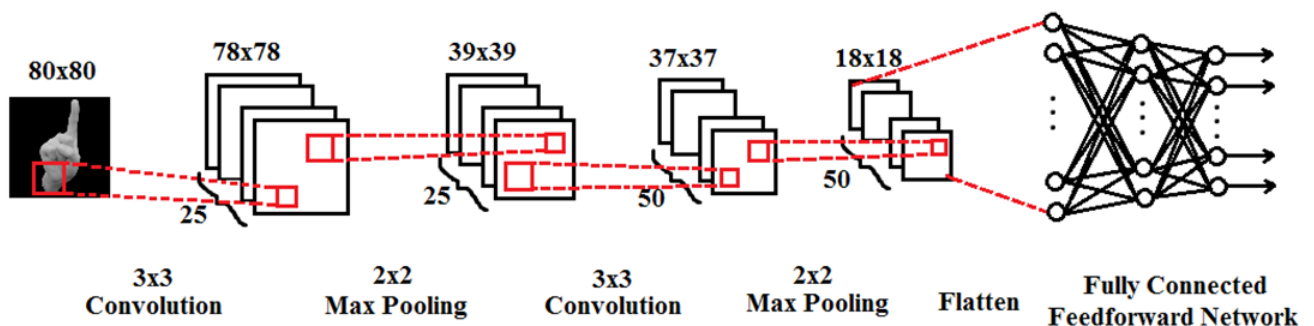


圖 2-1: CNN model 結構圖

圖 2-1 為 CNN model 的架構，以一張 80\*80 的圖片為 input，用 25

個大小為 3\*3 的 filter 和圖片做 convolution，每個 filter 可偵測不同的特徵，得到 25 個大小為 78\*78 的圖片，再將圖片分成 2\*2 的小區域，每個區域中只保留最大值，以降低圖片的大小。

Convolution 和 max pooling 的步驟重複數次後，可取得圖片中較小範圍的特徵，將這些特徵展開成一維向量，輸入到一個 neural network 中得到 20 個 output，每個 output 分別是屬於 20 種手型的機率。

完成後將 model 儲存起來，我們 training 後得到的 model 在 training data 和 validation data 的 accuracy 都約為 0.99，可以得知我們的 model 有相當高的可信度。

## 2. Testing (camera\_right.py)

在此程式中，我們使用和 preprocess 一樣的方法來擷取手部畫面，將 MATLAB 程式改成在 python 中實作，以達到 real time 的效果，並用前面得到的 model 來判斷手型種類，最後將結果即時的顯示在畫面上。

### (1) Train SVM model

與第一部份製作 database 的方法相同，我們將膚色 training data 的 RGB 值轉為 YCbCr 作為 feature，減去 mean 再除以 standard deviation，找出一個用於判斷是否為膚色的 SVM model。

### (2) Capture frame, resize

在這次的實驗中，我們用的是筆記型電腦內建的鏡頭。先從鏡頭取得大小為 640\*480 的照片，並將照片壓縮為 180\*135 以減少運算量和運算時間，經過我們測試，這樣的壓縮並不會影響判斷的正確率。

### (3) Color space transform, feature scaling, testing

將照片每個 pixel 的 RGB 轉為 YCbCr，減去 mean 再除以 standard deviation，以 SVM model 判斷每個 pixel 是否屬於膚色，屬於膚色的標記為 1，不是膚色的標記為 0。

### (4) Group the regions

將判斷為膚色的部分分成相連的數個區域，每個區域以不同的 label 標記，方便之後計算每個膚色區塊的大小還有位置。

### (5) Find the largest two regions, choose the lower one

我們預設畫面中會出現一張臉和一隻手，整張圖中膚色區域面積最大的兩塊會是臉和手，因此找出面積最大的兩塊。考量手跟鏡頭的距離

會影響手和臉的相對大小，所以我們是由高低來判斷兩個區塊中哪一個是手。由於手的位置較低，因此選擇較低的區塊。

#### (6) Resize, classify

將判斷成手的區塊的Y值(亮度)放入一個與手部長度或寬度切齊的黑色正方形中，並把圖片大小 resize 為 80\*80，輸入先前得到的 CNN model，判斷圖片中應該是哪一個手型。

#### (7) Show the result

在顯示畫面中，把判斷為不是膚色的部分亮度降低，並將判斷成手的區塊用紅色的框框標示出來，在框框左上角上方顯示手語的判斷結果(例: 1-one)，圖 2-2 為實際的顯示結果，**正確率約有九成**。

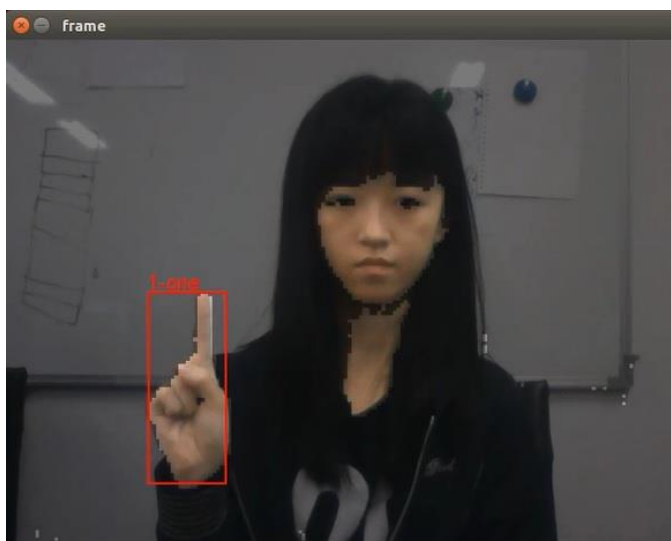


圖 2-1: 可以看到被框出的手部區塊還有左上角的結果

### 三、即時雙手手語判斷

有部分的手語需要兩隻手的配合，所以在一隻手的手型判斷成功以後，我們又做了一個可以判斷一隻手或是兩隻手的版本。

因為左右手在視覺上是**鏡像**的，而且理論上我們並不知道畫面中出現的是左手還是右手，所以我們是將左右手的 training data 共用，找到一個可以**同時對左右手判別手型的 model**。

大致上的方法與第二部分相同，因此下面只對**不同的部分**(標示為**橘色**)詳加說明。

#### 1. Training

##### (1) **翻轉照片**調整圖片大小 ([mirror\\_data.py](#), [resize.py](#))



因為要同時判斷左右手，所以將每個手型的 50 張圖片都水平翻轉，得到新的 50 張圖片(每個手型各 100 張)，並存入原本的資料夾中。

Database 中的圖片雖然都是正方形的，但大小並不固定，所以我們先將所有圖片大小調整為固定的 80\*80 並儲存。

## (2) 分割 training data 和 testing data ([train\\_mirror.py](#))

4/5 作為 training data，1/5 作為 testing data。

## (3) CNN(Convolutional neural network) model ([train\\_mirror.py](#))

## 2. Testing ([camera\\_rl2.py](#))

### (1) Train SVM model

### (2) Capture frame, resize

### (3) Color space transform, feature scaling, testing

### (4) Group the regions

### (5) Find the hand regions, classify, and show the result

相機畫面有三種可能的情況：沒有手、一隻手、兩隻手。我們假設影像中手和臉的 pixel 數皆會大於 400 (20\*20 是一個非常小的區塊)，所以我們先找到最大的三塊區塊，並去除小於 400 pixels 的區塊，得到下面三種情況。

#### (a) 僅一塊大於 400 pixels

判斷該區塊為臉，僅將膚色區塊標示出來，不對這個畫面進行判斷，如圖 2-2。



圖 2-2: 畫面中沒有手

**(b) 有兩塊大於 400 pixels**

判斷區塊為一張臉和一隻手，僅對位置較低的區塊進行判斷，並將判斷的結果標示於左上角，如圖 2-3。

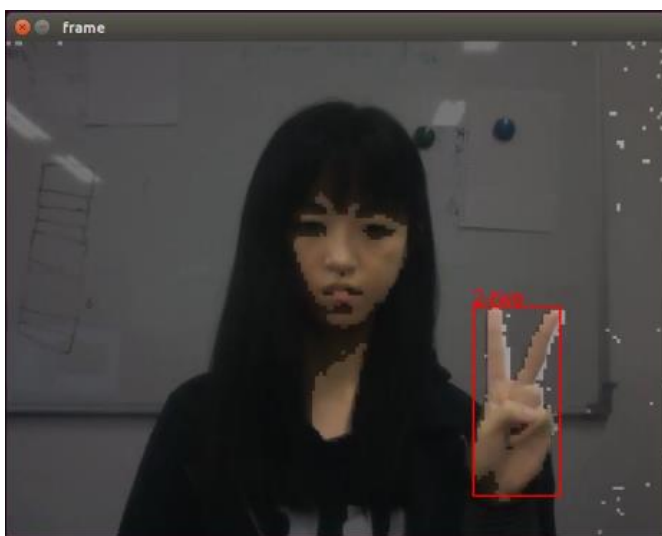


圖 2-3: 畫面中有一隻手

**(c) 三塊皆大於 400 pixels**

判斷區塊為一張臉和兩隻手，對位置較低的兩個區塊進行判斷，並將判斷結果標示於左上角，如圖 2-4。



圖 2-4: 畫面中有兩隻手

## ● 結果分析與未來展望

每當筆記型電腦的鏡頭讀取一張畫面後就會進行一次完整運算，每次讀取畫面到顯示結果這個過程會花費 0.048~0.056 秒，以手語的速度來說，我們認為這樣的即時性已經非常足夠了。

我們目前做出的手型辨識系統對一隻手的辨識正確率約為九成，對兩隻手的辨識率約為七成。主要是因為有很多手型本身非常的相似，在 train CNN 的 model 時，將左右手的 training data 放在一起會使不同手型間的差異性變小。因此我們列出了下列幾項未來可以努力的方向

### 1. 增加可以辨識的手型種類

因為台灣常用的手型並不只我們辨識的這 20 種，希望可以加入更多的手型，來增加這個手語辨識系統的實用性。

### 2. 增加 dataset 中照片的數量和多樣性

目前每個手型僅有 50 張照片，以機器學習來說是非常少的，多樣性也不夠。我們可以在現有的 dataset 中，加入更多不同人的手部照片，來增加多樣性，提高判斷的正確率。

### 3. 提升同時偵測雙手的正確率

因為 Python 對於 model 數量的限制導致目前左右手是共用同一個 CNN model，希望可以找到方法來解決這個問題，讓左右手可以有各自的 CNN model 來提升正確率。(可能可以透過一次運作多個 python 檔案來達成)

### 4. 辨識手語

我們現在辨識的是手型，雖然手型也有各自代表的意思，但更多的手語是由不同位置的手型所組成的，所以希望可以加入與身體相對位置的資訊來判斷更多的手語。

## ● 心得

在這次的 project 中，我們本來先試著將「擷取手部畫面」和「用 CNN 判斷手語種類」分別在 MATLAB 和 python 中實作，但為了達到 real time 的效果，便將 MATLAB 的內容改為 python，其中一個步驟轉換到 python 時，若膚色面積太大，會因為運算量過大而使程式終止，後來我們找到了一個 function，可以更有效地運算，避免這樣的情況，也減少了延遲時間。

我們在拍攝 training data 時，每個手語都會稍微轉動手部，但有些手語的角度變化太大，判斷時會容易將其他種類的手語誤判為該類手語，我們在重新調整過 training data 後改善了這個情況。

我們目前能夠即時判斷 20 種靜態手語，但手語的種類非常多，動作與位置的變化也會影響手語的意思，若要實際應用在手語翻譯，仍需再加強手部追蹤與辨識的手語種類，並提升可攜帶性，希望未來能夠再加強這個系統，將功能做得更好，幫助聾人與聽人之間的溝通。

## 參考資料

1. 維基百科。手語。  
<https://zh.wikipedia.org/wiki/%E6%89%8B%E8%AA%9E>
2. LIBSVM -- A Library for Support Vector Machines. Chih-Chung Chang and Chih-Jen Lin. 2016.  
<https://www.csie.ntu.edu.tw/~cjlin/libsvm/>
3. Wikipedia – Convolutional neural network  
[https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)
4. Real-time American Sign Language Recognition with Convolutional Neural Networks. Brandon Garcia and Sigberto Alarcon Viesca. 2016.  
[http://cs231n.stanford.edu/reports2016/214\\_Report.pdf](http://cs231n.stanford.edu/reports2016/214_Report.pdf)

## ● 附錄列表

附錄一、svm.m .....	13
附錄二、resize.py .....	15
附錄三、train.py .....	16
附錄四、camera_right.py .....	17
附錄五、mirror_data.py .....	20
附錄六、train_mirror.py .....	21
附錄七、camera_rl2.py .....	22

## 附錄一、 svm.m

```
1  clear
2  close all
3  addpath('C:\Users\Wang\Desktop\libsvm-3.21\matlab');
4
5  %% training data and labels^M
6  training_data=[];
7
8  for pic_num=1:25
9      if pic_num<10
10         training_data=[training_data;['tr/tr0' num2str(pic_num) '.jpg']];
11     else
12         training_data=[training_data;['tr/tr' num2str(pic_num) '.jpg']];
13     end
14 end
15 label_a=[1;1;1;1;1;0;0;0;0;0;1;1;1;0;0;0;1;1;1;0;1;0;1;0]; % 0: nonface, 1: face
16
17 features_a=zeros(size(training_data,1),3);
18 for pic_num=1:size(training_data,1)
19     im1=double(imread(training_data(pic_num,:)));
20     im1=rgb2ycbcr(im1);
21     x=mean(mean(im1(:,:)));
22     features_a(pic_num,:)=[x(1) x(2) x(3)];
23 end
24
25 %% scaling^M
26 [m1,N]=size(features_a);
27 mf=mean(features_a);
28 nrm=diag(1./std(features_a,1));
29 features_1=(features_a-ones(m1,1)*mf)*nrm;
30
31 %% SVM^M
32 model=svmtrain(label_a,features_1);
33 start_num=[3952,4014,4083,4159,4215,4269,4323,4384,4458,4522,4575,4676,4733,...
34           4792,4844,4896,4948,5003,5065,4627];
35
36 %% test data and labels^M
37 for sign=1:1
38     for pic_num=10:10
39         test_data=['data/' num2str(sign) '/IMG_' num2str(start_num(sign)+pic_num-1) '.jpg'];
40         for m=1:size(test_data,1)
41             im1=double(imread(test_data(m,:)));
42             im1=im1(1:2:479,1:2:639,:);
43             im2=rgb2ycbcr(im1);
44             features_b=reshape(im2,size(im2,1)*size(im2,2),3);
45             label_b=floor(rand(size(im2,1)*size(im2,2),1)+0.5);
46             % scaling
47             [m2,N]=size(features_b);
```

```

48     features_2=(features_b-ones(m2,1)*mf)*nrm;
49     % test % predicted: the SVM output of the test data
50     [predicted,accuracy,d_values]=svmpredict(label_b,features_2,model);
51     % group the regions
52     predicted=reshape(predicted,size(im2,1),size(im2,2));
53     % skin picture
54     figure
55     imshow(predicted)
56     colormap(gray(256))
57     % bwlabel: group the regions
58     A1=bwlabel(predicted);
59     size1=zeros(max(max(A1)),1);
60     for n=1:max(max(A1))
61         [fr,fc]=find(A1==n);
62         size1(n,1)=size(fr,1);
63     end
64     % find the 1st region and 2nd region
65     [M1,I1]=max(size1);
66     size1(I1)=0;
67     [M2,I2]=max(size1);
68     %find the lower region
69     Y_temp1=im2(:, :, 1).*(A1==I1);
70     Y_temp2=im2(:, :, 1).*(A1==I2);
71     [x,y]=find(Y_temp1~=0);
72     mean1=mean(x);
73     [x,y]=find(Y_temp2~=0);
74     mean2=mean(x);
75     if mean1>mean2
76         Y1=Y_temp1;
77     else
78         Y1=Y_temp2;
79     end
80     [fr,fc]=find(Y1~=0);
81     Y1=Y1(min(fr):max(fr),min(fc):max(fc));
82     [height, width]=size(Y1);
83     max_num=max(height,width);
84     max_num=ceil(max_num/2)*2;
85     Y2=zeros(max_num,max_num);
86     Y2(((max_num/2)-ceil(height/2)+1):((max_num/2)-ceil(height/2)+height),...
87         ((max_num/2)-ceil(width/2)+1):((max_num/2)-ceil(width/2)+width))=Y1;
88     % saved hand picture
89     figure
90     imshow(Y2/255)
91     % save data
92     imwrite(Y2/255,['result/' num2str(sign) '/' num2str(sign) '_' num2str(pic_num) '.jpg'],'Quality',100)
93     % hand picture
94     R1=im1(:, :, 1).*(A1==I2);
95     G1=im1(:, :, 2).*(A1==I2);
96     B1=im1(:, :, 3).*(A1==I2);
97     figure
98     imshow(cat(3,R1/255,G1/255,B1/255))
99     colormap(gray(256))
100 end
101 end
102 end

```

## 附錄二、resize.py

```
1  from PIL import Image
2  import os
3
4  sign_num = 20
5  data_num = 50
6  width = 80
7  height = 80
8
9  path='./result_'+str(width)+'_'+str(height)
10 if not os.path.isdir(path):
11     os.mkdir(path)
12
13 for i in range(sign_num):
14     path2='./result_'+str(width)+'_'+str(height)+'/'+str(i+1)
15     if not os.path.isdir(path2):
16         os.mkdir(path2)
17     for j in range(data_num):
18         img=Image.open('./result/'+str(i+1)+'/'+str(i+1)+'_'+str(j+1)+'.jpg')
19         img2=img.resize((width,height))
20         img2.save(path+'/'+str(i+1)+'/'+str(i+1)+'_'+str(j+1)+'.jpg')
```



### 附錄三、 train.py

```
1  import sys
2  import numpy as np
3  from PIL import Image
4  from keras.models import Sequential
5  from keras.layers.core import Dense, Activation
6  from keras.layers import Convolution2D, MaxPooling2D, Flatten
7
8  sign_num=20
9  data_num=50
10 height=80
11 width=80
12
13 #load training data
14 train=np.zeros((sign_num*data_num, height*width+sign_num))
15 for i in range(sign_num):
16     for j in range(data_num):
17         img=Image.open('result_80_80/'+str(i+1)+'/'+str(i+1)+'_'+str(j+1)+'.jpg')
18         train[i*data_num+j, :height*width]=np.array(img).reshape(1, height*width)
19         train[i*data_num+i*data_num+data_num, height*width+i]=1
20
21 np.random.shuffle(train)
22 x_train=train[:, :height*width].reshape(sign_num*data_num, 1, height, width)
23 y_train=train[:, height*width:]
24 x_train=x_train.astype('float32')
25 x_train=x_train/255
26
27 #convolutional neural network
28 #define model
29 model=Sequential()
30 model.add(Convolution2D(25, 3, 3, input_shape=(1, height, width)))
31 model.add(MaxPooling2D((2, 2)))
32 model.add(Convolution2D(50, 3, 3))
33 model.add(MaxPooling2D((2, 2)))
34 model.add(Flatten())
35 model.add(Dense(100))
36 model.add(Activation('sigmoid'))
37 model.add(Dense(sign_num))
38 model.add(Activation('softmax'))
39
40 model.summary()
41
42 #compile model
43 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
44
45 #fit model
46 model.fit(x_train, y_train, batch_size=100, nb_epoch=10, validation_split=0.2)
47
48 #evaluate model
49 score=model.evaluate(x_train, y_train)
50 print('Total loss on Training Set:', score[0])
51 print('Accuracy of Training Set:', score[1])
52
53 #save model
54 model.save('model10.h5')
```

## 附錄四、camera\_right.py

```
1  import cv2
2  import time
3  import numpy as np
4  from PIL import Image, ImageDraw, ImageFont
5  from sklearn import svm
6  from keras.models import load_model
7  from scipy.ndimage import label
8
9  model=load_model('model10.h5')
10 sign_type=['1-one', '2-two', '3-three', '4-four', '5-five',
11           '6-six', '7-seven', '8-eight', '9-nine', '10-hand',
12           '11-square', '12-child', '13-money', '14-dragon', '15-wc',
13           '16-wait', '17-hundred', '18-fourty', '19-zero', '20-borrow'];
14 font=ImageFont.truetype("arial.ttf",20)
15
16 def myArray2Image(A, dataType):
17     A=A.astype('uint8')
18     A=Image.fromarray(A, dataType)
19     return A
20
21
22 def main(A1,img5,Y1,fr,fc):
23     Y1=Y1[np.amin(fr):np.amax(fr),np.amin(fc):np.amax(fc)]
24     [height, width]=Y1.shape
25
26     max_num=max(height,width)
27     max_num=(np.ceil(max_num/2.0)*2).astype('uint8')
28     Y2=np.zeros((max_num,max_num))
29
30     temp1=(np.ceil(height/2)).astype('uint8')
31     temp2=(np.ceil(width/2)).astype('uint8')
32
33     Y2[(max_num/2-temp1):(max_num/2-temp1+height),
34        (max_num/2-temp2):(max_num/2-temp2+width)]=Y1
35
36     Y2=myArray2Image(Y2, 'L')
37     Y2=Y2.resize((80,80))
38
39     Y2=np.array(Y2)
40     data=Y2
41     result=model.predict(data.reshape(1, 1, 80, 80))
42     result=np.argmax(result, axis=1)+1
43
44     A1=myArray2Image(A1, 'L')
45     A1=np.array(A1.resize((640,480)))
46     tmp=(A1>0)
47
48     tmp2=np.ones(shape=(480,640))
49     img5=img5.astype('float64')
50     img5[:, :, 0]=(tmp2*0.5+tmp*0.5)
51     img5[:, :, 1]=(tmp2*0.5+tmp*0.5)
52     img5[:, :, 2]=(tmp2*0.5+tmp*0.5)
53     fr_min=np.amin(fr*480/135)
54     fr_max=np.amax(fr*480/135)
55     fc_min=np.amin(fc*480/135)
56     fc_max=np.amax(fc*480/135)
57
58     img5[fr_min:fr_min+2,fc_min:fc_max+1,:]=[0,0,255]
59     img5[fr_max-1:fr_max+1,fc_min:fc_max+1,:]=[0,0,255]
```

```

60     img5[fr_min:fr_max+1,fc_min:fc_min+2,:]=[0,0,255]
61     img5[fr_min:fr_max+1,fc_max-1:fc_max+1,:]=[0,0,255]
62
63     img5=myArray2Image(img5,'RGB')
64     draw=ImageDraw.Draw(img5)
65     draw.text((fc_min,fr_min-20),sign_type[result-1],(0,0,255),font=font)
66     draw=ImageDraw.Draw(img5)
67     img5=np.array(img5)
68     print sign_type[result-1]
69     return img5
70
71
72 #train
73 feature_1=np.zeros((26,3))
74 for i in range(26):
75     if i<9:
76         img=Image.open('tr/tr0'+str(i+1)+'.jpg')
77     else:
78         img=Image.open('tr/tr'+str(i+1)+'.jpg')
79     img=np.array(img.convert('YCbCr'))
80     feature_1[i,0]=np.mean(img[:, :,0])
81     feature_1[i,1]=np.mean(img[:, :,1])
82     feature_1[i,2]=np.mean(img[:, :,2])
83
84 label_1=np.array([1,1,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,1,1,0,1,1])
85
86 #feature scaling
87 mean=np.mean(feature_1,axis=0)
88 std=np.std(feature_1,axis=0)
89 feature_1-=mean
90 feature_1/=std
91
92 #svm
93 clf=svm.SVC()
94 clf.fit(feature_1, label_1)
95
96 #capture frame
97 cap=cv2.VideoCapture(0)
98
99
100 #test
101 while(True):
102     #capture frame-by-frame
103     t_start=time.time()
104     ret, frame=cap.read()
105     img2=np.array(frame)
106     img5=img2
107     img2=cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)
108
109     img2=myArray2Image(img2,'RGB')
110     img2=img2.resize((180,135))
111     img2=np.array(img2.convert('YCbCr'))
112
113     #downsampling
114     Y=img2[:, :,0]
115
116     M=np.size(img2,0)

```

```

117     N=np.size(img2,1)
118     img2=img2.reshape(M*N,3)
119
120     #feature scaling
121     img2=img2.astype('float64')
122     img2-=mean
123     img2/=std
124
125     #test
126     img2=clf.predict(img2).reshape(M,N)
127
128     #group the regions
129     s=np.array([[1,1,1],
130                [1,1,1],
131                [1,1,1]])
132     A1, num=label(img2,structure=s)
133
134     #find the size of each region
135     size1=np.zeros((num,1))
136     for i in range(num):
137         size1[i]=np.where(A1==i+1)[0].shape
138
139     #find the largest two regions
140     if size1.shape[0]==0:
141         continue
142     I1=np.argmax(size1)
143     size_max1=np.amax(size1)
144     size1[I1]=0
145     I2=np.argmax(size1)
146     size_max2=np.amax(size1)
147
148     [fr1, fc1]=np.where(A1==I1+1)
149     [fr2, fc2]=np.where(A1==I2+1)
150
151     #choose the lower one
152     if np.mean(fr1)>np.mean(fr2):
153         I2=I1
154         fr2=fr1
155         fc2=fc1
156         size_max2=size_max1
157
158     Y1=Y*(A1==I2+1)
159     img5=main(A1,img5,Y1,fr2,fc2)
160
161     cv2.imshow('frame',img5)
162
163     t_end=time.time()
164     print t_end-t_start
165
166     if cv2.waitKey(1) & 0xFF == ord('q'):
167         break
168

```

## 附錄五、 mirror\_data.py

```
1  import cv2
2  import time
3  import numpy as np
4  from PIL import Image
5
6  sign_num=20
7  data_num=50
8
9  for i in range(sign_num):
10     for j in range(data_num):
11         img=Image.open('result_80_80/'+str(i+1)+'/'+str(i+1)+'_'+str(j+1)+'.jpg')
12         img2=img.transpose(Image.FLIP_LEFT_RIGHT)
13         cv2.imwrite('result_mirror/'+str(i+1)+'/'+str(i+1)+'_'+str(j+1)+'.jpg',np.array(img))
14         cv2.imwrite('result_mirror/'+str(i+1)+'/'+str(i+1)+'_'+str(j+51)+'.jpg',np.array(img2))
```

## 附錄六、 train\_mirror.py

```
1  import sys
2  import numpy as np
3  from PIL import Image
4  from keras.models import Sequential
5  from keras.layers.core import Dense, Activation
6  from keras.layers import Convolution2D, MaxPooling2D, Flatten
7
8  sign_num=20
9  data_num=100
10 height=80
11 width=80
12
13 #load training data
14 train=np.zeros((sign_num*data_num, height*width+sign_num))
15 for i in range(sign_num):
16     for j in range(data_num):
17         img=Image.open('result_mirror/'+str(i+1)+'/'+str(i+1)+'_'+str(j+1)+'.jpg')
18         train[i*data_num+j, :height*width]=np.array(img).reshape(1, height*width)
19         train[i*data_num:i*data_num+data_num, height*width+i]=1
20
21 np.random.shuffle(train)
22 x_train=train[:, :height*width].reshape(sign_num*data_num, 1, height, width)
23 y_train=train[:, height*width:]
24 x_train=x_train.astype('float32')
25 x_train=x_train/255
26
27 #convolutional neural network
28 #define model
29 model=Sequential()
30 model.add(Convolution2D(25, 3, 3, input_shape=(1, height, width)))
31 model.add(MaxPooling2D((2, 2)))
32 model.add(Convolution2D(50, 3, 3))
33 model.add(MaxPooling2D((2, 2)))
34 model.add(Flatten())
35 model.add(Dense(100))
36 model.add(Activation('sigmoid'))
37 model.add(Dense(sign_num))
38 model.add(Activation('softmax'))
39
40 model.summary()
41
42 #compile model
43 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
44
45 #fit model
46 model.fit(x_train, y_train, batch_size=100, nb_epoch=10, validation_split=0.2)
47
48 #evaluate model
49 score=model.evaluate(x_train, y_train)
50 print('Total loss on Training Set:', score[0])
51 print('Accuracy of Training Set:', score[1])
52
53 #save model
54 model.save('model_m.h5')
```

## 附錄七、 camera\_rl2.py

```
1  import cv2
2  import time
3  import numpy as np
4  from PIL import Image, ImageDraw, ImageFont
5  from sklearn import svm
6  from keras.models import load_model
7  from scipy.ndimage import label
8
9  model=load_model('model_m.h5')
10 sign_type=['1-one','2-two','3-three','4-four','5-five',
11            '6-six','7-seven','8-eight','9-nine','10-hand',
12            '11-square','12-child','13-money','14-dragon','15-wc',
13            '16-wait','17-hundred','18-fourty','19-zero','20-borrow'];
14 font=ImageFont.truetype("arial.ttf",20)
15
16 def myArray2Image(A, dataType):
17     A=A.astype('uint8')
18     A=Image.fromarray(A, dataType)
19     return A
20
21
22 def main(A1,img5,Y1,fr,fc,run):
23     Y1=Y1[np.amin(fr):np.amax(fr),np.amin(fc):np.amax(fc)]
24     [height, width]=Y1.shape
25
26     max_num=max(height,width)
27     max_num=(np.ceil(max_num/2.0)*2).astype('uint8')
28     Y2=np.zeros((max_num,max_num))
29
30     temp1=(np.ceil(height/2)).astype('uint8')
31     temp2=(np.ceil(width/2)).astype('uint8')
32
33     Y2[(max_num/2-temp1):(max_num/2-temp1+height),
34        (max_num/2-temp2):(max_num/2-temp2+width)]=Y1
35
36     Y2=myArray2Image(Y2, 'L')
37     Y2=Y2.resize((80,80))
38
39     Y2=np.array(Y2)
40     data=Y2
41     result=model.predict(data.reshape(1, 1, 80, 80))
42     result=np.argmax(result, axis=1)+1
43
44     A1=myArray2Image(A1, 'L')
45     A1=np.array(A1.resize((640,480)))
46     tmp=(A1>0)
47
48     tmp2=np.ones(shape=(480,640))
49     img5=img5.astype('float64')
50     if run==0:
51         img5[:, :, 0]*=(tmp2*0.5+tmp*0.5)
52         img5[:, :, 1]*=(tmp2*0.5+tmp*0.5)
53         img5[:, :, 2]*=(tmp2*0.5+tmp*0.5)
```

```

54     else:
55         img5[:, :, 0] *= tmp2
56         img5[:, :, 1] *= tmp2
57         img5[:, :, 2] *= tmp2
58         fr_min = np.amin(fr * 480 / 135)
59         fr_max = np.amax(fr * 480 / 135)
60         fc_min = np.amin(fc * 480 / 135)
61         fc_max = np.amax(fc * 480 / 135)
62
63         img5[fr_min:fr_min+2, fc_min:fc_max+1, :] = [0, 0, 255]
64         img5[fr_max-1:fr_max+1, fc_min:fc_max+1, :] = [0, 0, 255]
65         img5[fr_min:fr_max+1, fc_min:fc_min+2, :] = [0, 0, 255]
66         img5[fr_min:fr_max+1, fc_max-1:fc_max+1, :] = [0, 0, 255]
67
68         img5 = myArray2Image(img5, 'RGB')
69         draw = ImageDraw.Draw(img5)
70         draw.text((fc_min, fr_min-20), sign_type[result-1], (0, 0, 255), font=font)
71         draw = ImageDraw.Draw(img5)
72         img5 = np.array(img5)
73         print sign_type[result-1]
74         return img5
75
76
77 #train
78 feature_1 = np.zeros((24, 3))
79 for i in range(24):
80     if i < 9:
81         img = Image.open('tr/tr0'+str(i+1)+'.jpg')
82     else:
83         img = Image.open('tr/tr'+str(i+1)+'.jpg')
84     img = np.array(img.convert('YCbCr'))
85     feature_1[i, 0] = np.mean(img[:, :, 0])
86     feature_1[i, 1] = np.mean(img[:, :, 1])
87     feature_1[i, 2] = np.mean(img[:, :, 2])
88
89 label_1 = np.array([1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1])
90
91 #feature scaling
92 mean = np.mean(feature_1, axis=0)
93 std = np.std(feature_1, axis=0)
94 feature_1 -= mean
95 feature_1 /= std
96
97 #svm
98 clf = svm.SVC()
99 clf.fit(feature_1, label_1)
100
101 #capture frame
102 cap = cv2.VideoCapture(0)
103
104
105 #test
106 while(True):
107     #capture frame-by-frame
108     t_start = time.time()
109     ret, frame = cap.read()
110     img2 = np.array(frame)
111     img5 = img2
112     img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

```



```

113
114     img2=myArray2Image(img2,'RGB')
115     img2=img2.resize((180,135))
116     img2=np.array(img2.convert('YCbCr'))
117
118     #downsampling
119     Y=img2[:, :, 0]
120
121     M=np.size(img2,0)
122     N=np.size(img2,1)
123     img2=img2.reshape(M*N,3)
124
125     #feature scaling
126     img2=img2.astype('float64')
127     img2-=mean
128     img2/=std
129
130     #test
131     img2=c1f.predict(img2).reshape(M,N)
132
133     #group the regions
134     s=np.array([[1,1,1],
135                [1,1,1],
136                [1,1,1]])
137     A1, num=label(img2,structure=s)
138
139     #find the size of each region
140     size1=np.zeros((num,1))
141     for i in range(num):
142         size1[i]=np.where(A1==i+1)[0].shape
143
144     #find the largest three regions
145     if size1.shape[0]==0:
146         continue
147     I1=np.argmax(size1)
148     size_max1=np.amax(size1)
149     size1[I1]=0
150     I2=np.argmax(size1)
151     size_max2=np.amax(size1)
152     size1[I2]=0
153     I3=np.argmax(size1)
154     size_max3=np.amax(size1)
155
156     [fr1, fc1]=np.where(A1==I1+1)
157     [fr2, fc2]=np.where(A1==I2+1)
158     [fr3, fc3]=np.where(A1==I3+1)
159
160     BB=np.array([np.mean(fr1),np.mean(fr2),np.mean(fr3)])
161     BB2=np.array([np.mean(fr1),np.mean(fr2),np.mean(fr3)])
162     fr_array=np.array([fr1,fr2,fr3])
163     fc_array=np.array([fc1,fc2,fc3])
164     I_array=np.array([I1,I2,I3])
165
166     big=np.argmax(BB)
167     BB[big]=0
168     middle=np.argmax(BB)
169     BB[middle]=0
170     small=np.argmax(BB)

```

```

171
172 #three regions are larger than 400: choose the lower two
173 if size_max3>=400:
174     if np.mean(fr1)==BB2[small]:
175         AA1=Y*(A1==I2+1)
176         AA2=Y*(A1==I3+1)
177         img5=main(A1,img5,AA1,fr2,fc2,0)
178         img5=main(A1,img5,AA2,fr3,fc3,1)
179     elif np.mean(fr2)==BB2[small]:
180         AA1=Y*(A1==I1+1)
181         AA2=Y*(A1==I3+1)
182         img5=main(A1,img5,AA1,fr1,fc1,0)
183         img5=main(A1,img5,AA2,fr3,fc3,1)
184     else:
185         AA1=Y*(A1==I1+1)
186         AA2=Y*(A1==I2+1)
187         img5=main(A1,img5,AA1,fr1,fc1,0)
188         img5=main(A1,img5,AA2,fr2,fc2,1)
189
190 #two regions are larger than 400: choose the lower one
191 elif size_max3<400 and size_max2>=400:
192     if BB2[middle]<400:
193         AA1=Y*(A1==I_array[big]+1)
194         img5=main(A1,img5,AA1,fr_array[big],fc_array[big],0)
195     else:
196         AA1=Y*(A1==I_array[middle]+1)
197         img5=main(A1,img5,AA1,fr_array[middle],fc_array[middle],0)
198
199 #one or no region is larger than 400
200 else:
201     A1=myArray2Image(A1,'L')
202     A1=np.array(A1.resize((640,480)))
203     tmp=(A1>0)
204     tmp2=np.ones(shape=(480,640))
205     img5=img5.astype('float64')
206     img5[:, :, 0]*=(0.5*tmp+0.5*tmp2)
207     img5[:, :, 1]*=(0.5*tmp+0.5*tmp2)
208     img5[:, :, 2]*=(0.5*tmp+0.5*tmp2)
209     img5=myArray2Image(img5,'RGB')
210     img5=np.array(img5)
211
212 cv2.imshow('frame',img5)
213
214 t_end=time.time()
215 print t_end-t_start
216
217 if cv2.waitKey(1) & 0xFF == ord('q'):
218     break
219

```