二、安裝與訓練車牌號碼 Harr 特徵分類器模型

01. 安裝與調整套件

- OpenCV Haar 特徵分類器模型訓練 https://github.com/sauhaardac/haar-training
- OpenCV 官網 https://opencv.org/releases/ 中，下載 opencv-3.4.9-vc14_vc.exe
- 安裝 opencv-3.4.9-vc14_vc.exe，並打開其資料夾，作以下處理：

  A. 將 opencv\build\x64\vc15\bin 中的三個檔案，複製到 haar-training\tarining：

  a. opencv_createsamples.exe

  b. opencv_traincascade.exe

  c. opencv_world349.dll

  B. 刪除 haar-training/tarining 中的兩個檔案：

  a. createsamples.exe

  b. traincascade.exe

- 加入正/負樣本的影像，刪除 Haar-Training-master\training\negative 與 Haar-Training-master\training\positive\rawdata 原先檔案

- 注意 1：Harr 分類器的訓練樣本只提供 bmp 的影像副檔名檔案
- 注意 2：刪除與複製是因為 Github 中的 Harr 分類器僅供 opencv 3.4 版本使用，以下程式指令可查看其版本

【09】
```
import cv2
print(cv2.__version__)
```

02. 開始使用 Haar 訓練模型

A．更正打包的批次檔：

a. 正樣本影像必須以 opencv_createsamples.exe 檔打包為向量檔才能訓練。將正樣本影像打包向量檔的批次檔為 training\samples_creation.bat，請使用記事本打開 bat 檔，並按照下面文字修改其內容（註：文字內容是同一列）：

【10】
```
opencv_createsamples.exe -info positive/info.txt -vec vector/facevector.vec -num 497 -w 76 -h 20
```

16. 參數意義如下：
    - -info：正樣本標記檔路徑
    - -vec：產生的向量檔路徑
    - -num：正樣本影像數量
    - -w：偵測物件的寬度
    - -h：偵測物件的高度

b. 在使用 Haar 特徵分類器模型時，偵測物件的寬度與寬度設定非常重要，小於此設定值的區域將無法偵測，而且偵測時會使用此寬高比進行偵測。因為車牌號碼寬高比為 3.8，所以我們將設定最小高度為 20 像素，再將寬度設為 20x3.8=76 像素

B．執行打包的批次檔，開始打包向量檔：

更正好後，點兩下執行 samples_creation.bat 會在 training\vector 的資料夾中產生 facevector.vec 向量檔

C．更正 Haar 特徵分類器訓練的批次檔：

訓練 Haar 特徵分類器的批次檔為 training\haarTraining.bat，請使用記事本打開 bat 檔，並按照下面文字修改其內容（註：文字內容是同一列）：

【10】

```
opencv_traincascade.exe -data cascades -vec vector/facevector.vec -bg negative/bg.txt -numPos 497 -numNeg 293 -numStages 15 -w 76 -h 20 -minHitRate 0.9999 -precalcValBufSize 512 -precalcIdxBufSize 512 -mode ALL
```

17. 參數意義如下：

- -data: 指定儲存訓練結果的資料夾
- -vec: 指定正樣本向量檔路徑
- -bg: 指定負樣本資料檔路徑
- -numPos: 指定正樣本影像數量
- -numNeg: 指定負樣本影像數量
- -numStage: 訓練級數，級數越多，模型的偵測準確率越高，但訓練花費的時間越長。通常級數設為 15 到 25 之間
- -w、-h: 偵測物件的寬度與高度
- -minHitRate: 每一級需要達到的準確率
- -precalcValBufSize 及-precalcIdxBufSize: 使用的記憶體，單位為 「M」，訓練及使用的記憶體大小，記憶體越大，訓練所花費的時間越短。如果訓練過程中出現 「記憶體不足」訊息時，可適度減小此數值
- -mode: 訓練模式，使用哪些 Haar 特徵類型來訓練。「ALL」是使用所有特徵類型，「BASIC」是使用線性 Haar 特徵類型，「CORE」是使用線性及中心 Haar 特徵類型

D．執行 Haar 特徵分類器訓練的批次檔，開始訓練：

請先清空 training\cascades 資料夾中所有資料，然後再點兩下執行 training\haarTraining.bat。訓練時間相當長（大約兩小時左右），請耐心等候。訓練完成後，訓練結果會存於 training\cascades 資料夾中。其中 cascades.xml 就是訓練完成的模型檔

03. 使用 Haar 特徵分類器偵測車牌

A．讀取模型後進行分類器的辨識，並將車牌用矩形標註出來：

【11】

```
import cv2
import matplotlib.pyplot as plt
img_path = 'carPlate/resize001.bmp'

img = cv2.imread(img_path)
detector = cv2.CascadeClassifier("License_Plate_Haar_cascade.xml")
signs = detector.detectMultiScale(img, minSize = (76, 20), scaleFactor = 1.1, minNeighbors=10)
```

```
if len(signs) > 0:
    for (sx, sy, sw, sh) in signs:
        cv2.rectangle(img, (sx, sy), (sx+sw, sy+sh), (0, 0, 255), 2)
        print(signs)
else:
    print('沒有辨識到車牌!')

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

18. cv2.CascadeClassifier(path)：讀取 OpenCV 模型，其中 path 為模型路徑。若想方便性的尋找 haar 的模型，可以使用 cv2.data.haarcascades 的方法快速查找，但是還是不建議使用這方式，例：cv2.CascadeClassifier(cv2.data.haarcascades+" License_Plate_Haar_cascade.xml ")
19. output = detector.detectMultiScale(img, minSize , maxSize, scaleFactor , minNeighbors, flags) 參數意義如下：
    - minSize：此參數設定最小偵測區塊
    - maxSize：此參數設定最大偵測區塊
    - scaleFactor：偵測原理是系統會以不同區塊大小對影像掃描進行特徵比對，以參數設定區塊的改變倍數。常設為「1.1」
    - minNeighbors：此為控制誤檢率參數。系統以不同區塊大小進行特徵比對時，在不同區塊中可能會多次成功取得特徵，成功取得特徵數需達到此參數設定值才算偵測成功。預設為「3」
    - flags：此參數設定檢測模式，常見的值有：
        - cv2.CASECADE_SCALE_IMAGE：按比例正常檢測
        - cv2.CASECADE_DO_CANNY_PRUNING：利用 Canny 邊緣檢測器來排除一些邊緣很少或很多的影像區域
        - cv2.CASECADE_FIND_BIGGEST_OBJECT：只檢測最大的物體
        - cv2.CASECADE_DO_ROUGH_SEARCH：只做初略檢測
20. cv2.rectangle(img, (x1, y1), (x2, y2), color, line size)：將 img 的影像繪製矩形，其中（x1, y1）與（x2, y2）為矩形左上角與右下角，color 為(B,G,R)的顏色，line size 為矩形線的寬度

圖一、Haar 特徵分類器偵測方式

圖二、車牌檢測結果

04. 車牌號碼各別擷取

A. 將車牌位置的區域轉為二值化的影像：

【12】

```
img = cv2.imread(img_path)
crop_img = img[sy:sy+sh, sx:sx+sw]
gray_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
_, binary_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY_INV)

plt.imshow(binary_img, cmap='gray')
plt.show()
```

21. ret, binary_img = cv2.threshold(img, threshold, max_value, type)，詳細如下：

回傳值：

- ret：回傳找到的最佳門檻值，一般情況都忽略此參數
- binary_img: 已二值化的影像

輸入參數：

- img：輸入的灰階影像
- threshold：對像素值進行分類的門檻值
- max_value：當像素值超過了門檻值（或是小於門檻值，根據 type 來決定），所賦予的值
- type：二值化操作的類型，有下列五種類型：
  - cv2.THRESH_BINARY：將大於門檻值的灰階值設為最大灰階值，小於門檻值設為 0
  - cv2.THRESH_BINARY_INV：將大於門檻值的灰階值設為 0，其他值設為最大灰階值
  - cv2.THRESH_TRUNC：將大於門檻值的灰階值設為門檻值，小於門檻值的值保持不變
  - cv2.THRESH_TOZERO：將小於門檻值的灰階值設為 0，大於門檻值的值保持不變
  - cv2.THRESH_ TOZERO_INV：將大於門檻值的灰階值設為 0，小於門檻值的值保持不變

圖三、車牌二值化後的影像

B．將二值化的影像進行輪廓偵測：

【13】

```
img = cv2.imread(img_path)
crop_img = img[sy:sy+sh, sx:sx+sw]
contours, hierarchy = cv2.findContours(binary_img, cv2.RETR_EXTERNAL, cv2.C
HAIN_APPROX_SIMPLE)
for i in range(len(contours)):
    (x, y, w, h) = cv2.boundingRect(contours[i])
    print((x, y, w, h))
    cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 1)
plt.imshow(crop_img)
plt.show()
```

22. contours, hierarchy = cv2.findContours(img，偵測模式，輪廓算法)

回傳值：contours 為輪廓 及 hierarchy 為階層

常見的偵測模式：

- cv2.RETR_EXTERNAL：只偵測輪廓外緣（常用）
- cv2.RETR_LIST：偵測輪廓時不建立等級關係
- cv2.RETR_CCOMP：偵測輪廓時建立兩個等級關係
- cv2.RETR_TREE：偵測輪廓時建立樹狀等級關係

常見的輪廓算法：

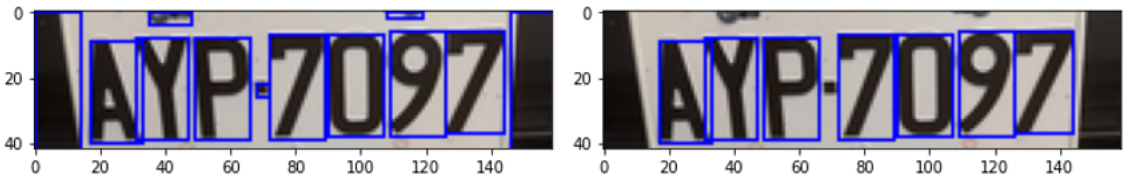
- cv2.CHAIN_APPROX_NONE：儲存所有輪廓點
- cv2. CHAIN_APPROX_SIMPLE：壓縮水平、垂直及對角線方向元素，只儲存該方向的終點，例如：舉行只儲存四個點，此算法速度較快（常用）

23. locations = cv2.boundingRect(輪廓資訊)：可將輪廓資訊以一個最小的矩形包圍起來，回傳該矩形的左上角位置與其寬高

C．設立條件式來移除非車牌號碼的輪廓區塊。若以像素的寬高來移除容易受到，鏡頭拍攝與車牌的距離而有落差，因此我們以比例關係來設立條件式：

【14】

```
img = cv2.imread(img_path)
crop_img = img[sy:sy+sh, sx:sx+sw]
contours, hierarchy = cv2.findContours(binary_img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
for i in range(len(contours)):
    (x, y, w, h) = cv2.boundingRect(contours[i])
    if sw*(3/20)>w>sw*(0.5/20) and sh*(19.5/20)>h>sh*(12/20):
        print((x, y, w, h))
        cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 1)
plt.imshow(crop_img)
plt.show()
```



圖四、最小矩形包圍後輪廓偵測的影像：（左圖）設條件式前 與（右圖）設條件式後