作品名稱：OpenCV 車牌辨識

一、說明

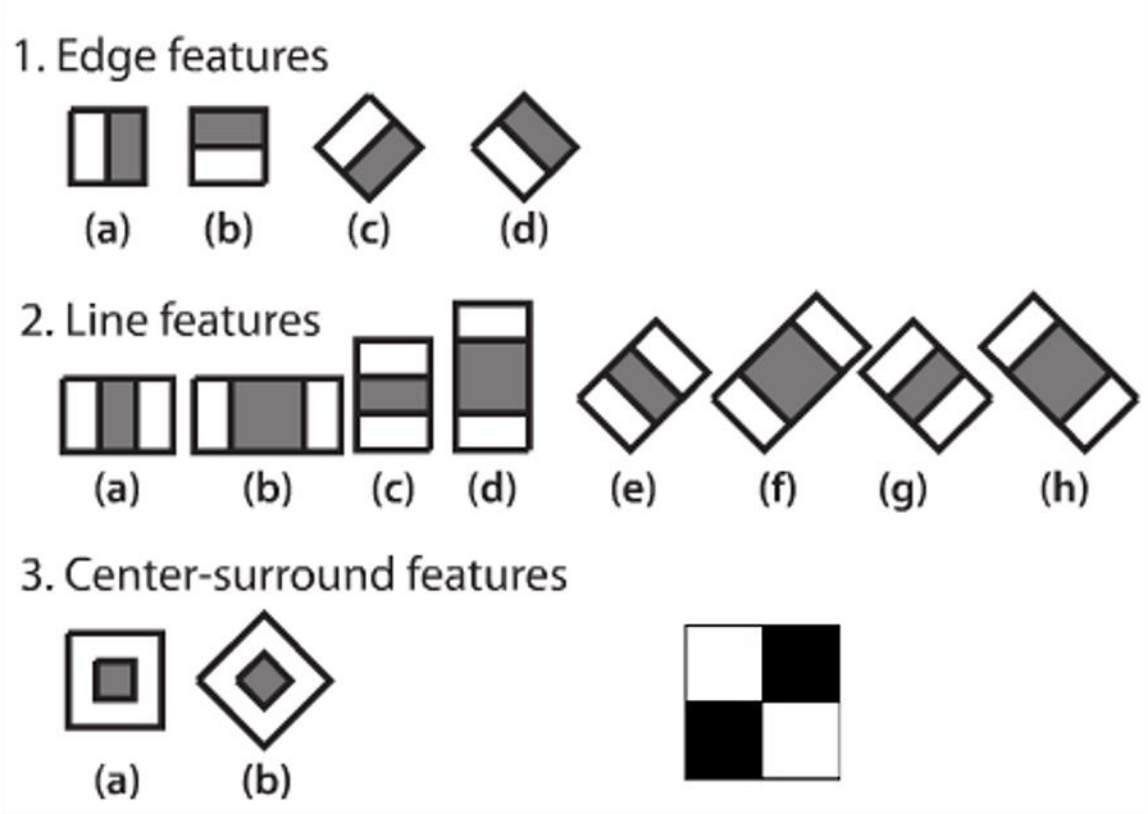
本專題主要使用 OpenCV 透過自行訓練並建立 Haar 特徵分類器「偵測車牌」模型，先偵測框選車牌及原始圖片轉換尺寸及偵測、然後將擷取車牌號碼產生圖形、再來是去除畸零地把邊緣的輪廓、雜訊、黑色部分去除、最後將用光學辨識軟體 OCR 辨識出車牌號碼，就會出現以下結果及結論。

二、論文

OpenCV 最為人稱道為「人臉辨識」。使用 OpenCV 提供的 Haar 特徵分類器人臉模型，即可輕鬆偵測人臉位置。是否有辦法偵測其他物件嗎？如果要偵測前請先建立 Haar 特徵分類器模型，然後再建立 Haar 特徵分類器偵測物件。

目前許多停車場已使用自動車牌辨識系統經營以節省人力成本，我們希望一步一步地探究影像辨識的技術並模擬建立停車場自動車牌辨識系統，從無到有自行訓練並建立 Haar 特徵分類器「偵測車牌」模型，先偵測框選車牌，接著將用光學辨識軟體 OCR 辨識出車牌號碼。

Haar 特徵是用來描繪一張圖片。Haar 特徵是一個矩形區域，可進行選轉、平移、縮放等，有 15 個類型。



Haar 特徵分類器可以幫我們在圖片或照片中偵測某特定物件是否存在，並可得知該物件的座標位置。這個套特定物件可以是人臉、交通標誌、動物等使用 Haar 特徵分類器模型分析。

三、實作

首先將要辨識的車牌圖片檔案更名為車牌號碼，使用 OCR 辨識車牌號碼後就可以將辨識結果與檔名名稱直接做比對，能不能辨識正確。


AHH9997


AKK7771


AKW6596


AXN6051

1. 原始圖片轉換尺寸及偵測：

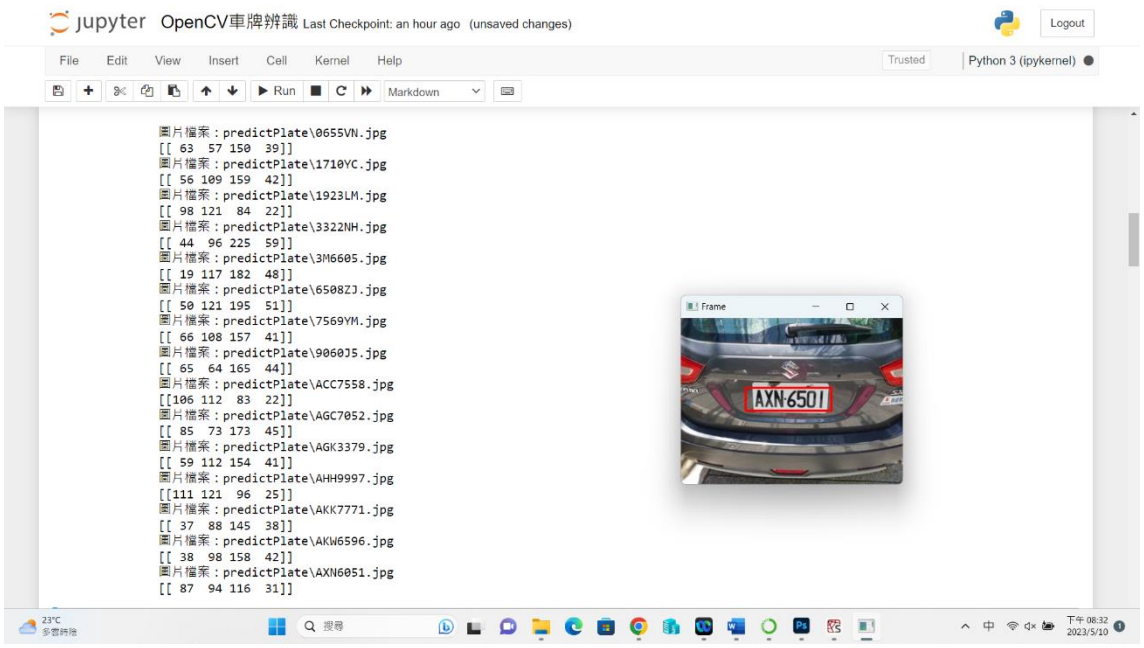
將所有數位相機拍攝或下載圖片尺寸轉換為 300x225 像素圖形，也使用 <haar_carplate.xml>模型做偵測。

程式碼:

```
def emptydir(dirname):          #清空資料夾
    if os.path.isdir(dirname): #資料夾存在就刪除
        shutil.rmtree(dirname)
        sleep(2)        #需延遲,否則會出錯
    os.mkdir(dirname)  #建立資料夾

def dirResize(src, dst):
    myfiles = glob.glob(src + '/*.JPG') #讀取資料夾全部 jpg 檔案
    emptydir(dst)
    print(src + ' 資料夾:')
    print('開始轉換圖形尺寸!')
    for f in myfiles:
        fname = f.split("\\")[-1]
        img = Image.open(f)
        img_new = img.resize((300, 225), PIL.Image.LANCZOS)  #尺寸
300x225
        img_new.save(dst + '/' + fname)
    print('轉換圖形尺寸完成!\n')

import PIL
from PIL import Image
import glob
import shutil, os
from time import sleep

dirResize('predictPlate_sr', 'predictPlate')
```

```
import cv2
import glob

files = glob.glob("predictPlate/*.jpg")
for file in files:
    print('圖片檔案：' + file)
    img = cv2.imread(file)
    detector = cv2.CascadeClassifier('haar_carplate.xml')
    signs = detector.detectMultiScale(img, minSize=(76, 20),
scaleFactor=1.1, minNeighbors=4)
    if len(signs) > 0 :
        for (x, y, w, h) in signs:
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
            print(signs)
    else:
        print('沒有偵測到車牌！')

    cv2.imshow('Frame', img)
    key = cv2.waitKey(0)
    cv2.destroyAllWindows()
    if key == 113 or key==81:  #按 q 鍵結束
        break
```

執行結果:



2. 擷取車牌號碼圖形：

使用 Haar 特徵分類器<haar_carplate.xml>模型框選出車牌號碼，並將車牌號碼圖形擷取下來。

程式碼:

```
def emptydir(dirname):          #清空資料夾
    if os.path.isdir(dirname): #資料夾存在就刪除
        shutil.rmtree(dirname)
        sleep(2)        #需延遲,否則會出錯
    os.mkdir(dirname)  #建立資料夾

import cv2
from PIL import Image
import glob
import shutil, os
from time import sleep
import numpy as np

print('開始擷取車牌！')
print('無法擷取車牌的圖片：')
dstdir = 'cropPlate'
emptydir(dstdir)
myfiles = glob.glob("predictPlate\*.JPG")
for imgname in myfiles:
    filename = (imgname.split('\\'))[-1]  #取得檔案名稱
    img = cv2.imread(imgname)  #讀入圖形
    detector = cv2.CascadeClassifier('haar_carplate.xml')
    signs = detector.detectMultiScale(img, scaleFactor=1.1,
minNeighbors=4, minSize=(20, 20))  #框出車牌
    #擷取車牌
    if len(signs) > 0 :
        for (x, y, w, h) in signs:
            image1 = Image.open(imgname)
            image2 = image1.crop((x, y, x+w, y+h))  #擷取車牌圖形
            image3 = image2.resize((140, 40), Image.LANCZOS) #轉換尺寸
為 140X40
            img_gray = np.array(image3.convert('L'))  #灰階
            _, img_thre = cv2.threshold(img_gray, 127, 255,
cv2.THRESH_BINARY) #黑白
```

```
            cv2.imwrite(dstdir + '/' + filename, img_thre) #存檔
    else:
        print(filename)

print('擷取車牌結束！')
```

執行結果:

3. 去除畸零地：

使用 ocr 辨識前請先將擷取好的車牌號碼圖形調整輪廓、雜訊、黑色部分去除就會變成乾淨完整的車牌號碼圖形，再用 ocr 模型進行完整的辨識提高效率。

程式碼:

```
def area(row, col):
    global nn
    if bg[row][col] != 255:
        return
    bg[row][col] = lifearea #記錄生命區的編號
    if col>1: #左方
        if bg[row][col-1]==255:
            nn +=1
            area(row,col-1)
    if col< w-1: #右方
        if bg[row][col+1]==255:
            nn +=1
            area(row,col+1)
    if row>1: #上方
        if bg[row-1][col]==255:
            nn+=1
            area(row-1,col)
    if row<h-1: #下方
        if bg[row+1][col]==255:
            nn+=1
            area(row+1,col)

import cv2
```

```
import numpy as np

image = cv2.imread('AHH9997.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  #灰階
_,thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV) #轉為黑白
contours1 = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE) #尋找輪廓
contours = contours1[0]   #取得輪廓

letter_image_regions = [] #文字圖形串列
for contour in contours:  #依序處理輪廓
    (x, y, w, h) = cv2.boundingRect(contour)  #單一輪廓資料
    letter_image_regions.append((x, y, w, h)) #輪廓資料加入串列
letter_image_regions = sorted(letter_image_regions, key=lambda x: x[0])
#按 X 坐標排序

#先計算可以擷取的字元數
count=0 #計算共擷取多少個字元
for box in letter_image_regions:  #依序處理輪廓資料
    x, y, w, h = box
    # x 必須介於 2~125 且寬度在 5~26、高度在 20~39 才是文字
    if x>=2 and x<=125 and w>=5 and w<=26 and h>=20 and h<40:
        count +=1

if count<6: #若字元數不足，可能是有兩個字元連在一起，將字元寬度放寬再重新擷取
    wmax=35
else:
    wmax=26 #正常字元寬度

nChar=0 #計算共擷取多少個字元
letterlist = [] #儲存擷取的字元坐標
for box in letter_image_regions:  #依序處理輪廓資料
    x, y, w, h = box
    # x 必須介於 2~125 且寬度在 5~wmax、高度在 20~39 才是文字
    if x>=2 and x<=125 and w>=5 and w<=wmax and h>=20 and h<40:
        nChar +=1
```

```
        letterlist.append((x, y, w, h)) #儲存擷取的字元

#去除雜點
for i in range(len(thresh)):  #i 為高度
    for j in range(len(thresh[i])): #j 為寬度
        if thresh[i][j] == 255:     #顏色為白色
            count = 0
            for k in range(-2, 3):
                for l in range(-2, 3):
                    try:
                        if thresh[i + k][j + l] == 255: #若是白點就將
count 加 1
                            count += 1
                    except IndexError:
                        pass
            if count <= 6:  #週圍少於等於 6 個白點
                thresh[i][j] = 0  #將白點去除

real_shape=[]
for i,box in enumerate(letterlist):  #依序擷取所有的字元
    x, y, w, h = box
    bg=thresh[y:y+h, x:x+w]

    # 去除崎鄰地
    if i==0 or i==nChar: # 只去除第一字元和最後字元的崎鄰地
        lifearea=0 # 生命區塊
        nn=0       # 每個生命區塊的生命數
        life=[]    # 記錄每個生命區塊的生命數串列
        for row in range(0,h):
            for col in range(0,w):
              if bg[row][col] == 255:
                  nn = 1  #生命起源
                  lifearea = lifearea + 1  #生命區塊數
                  area(row,col)  #以生命起源為起點探索每個生命區塊的總
生命數
                  life.append(nn)

        maxlife=max(life) #找到最大的生命數
```

```
        indexmaxlife=life.index(maxlife) #找到最大的生命數的區塊編號

        for row in range(0,h):
            for col in range(0,w):
              if bg[row][col] == indexmaxlife+1:
                  bg[row][col]=255
              else:
                  bg[row][col]=0

    real_shape.append(bg) #加入字元

#在圖片週圍加白色空白 OCR 才能辨識
newH, newW = thresh.shape

space = 8  #空白寬度
offset=2
bg = np.zeros((newH+space*2, newW+space*2+nChar*3, 1), np.uint8)  #建立
背景
bg.fill(0)  #背景黑色

# 將車牌文字加入黑色背景圖片中
for i,letter in enumerate(real_shape):
    h=letter.shape[0]   #原來文字圖形的高、寬
    w=letter.shape[1]
    x=letterlist[i][0]  #原來文字圖形的位置
    y=letterlist[i][1]
    for row in range(h):#將文字圖片加入背景
        for col in range(w):
            bg[space+y+row][space+x+col+i*offset] = letter[row][col] #
擷取圖形

_,bg = cv2.threshold(bg, 127, 255, cv2.THRESH_BINARY_INV) #轉為白色背
景、黑色文字
cv2.imwrite('assember0.jpg', bg) #存檔

cv2.imshow('image', image)      #顯示原始圖形
cv2.imshow('bg', bg)            #顯示組合的字元
cv2.moveWindow("image",500,250)#將視窗移到指定位置
```

```
cv2.moveWindow("bg",500,350)    #將視窗移到指定位置
key = cv2.waitKey(0)            #按任意鍵結束
cv2.destroyAllWindows()
```

執行結果:

4. 進行車牌辨識結果：

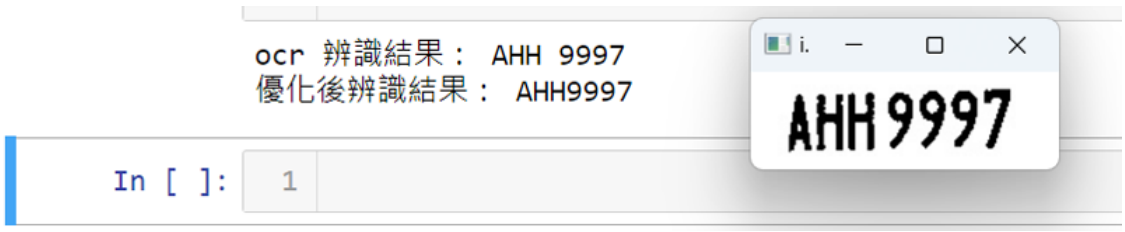
前面的程式已將車牌號碼去除輪廓、雜訊、黑色部分去除就會變成乾淨完整的車牌號碼圖形後，現在使用 ocr 來辨識車牌。

程式碼:

```
import cv2
from PIL import Image
import sys
import pyocr
import pyocr.builders
import re

image = cv2.imread('assember0.jpg')
#OCR 辨識車牌
tools = pyocr.get_available_tools()
if len(tools) == 0:
    print("No OCR tool found")
    sys.exit(1)
tool = tools[0]  #取得可用工具

result = tool.image_to_string(
```

```
    Image.open('assember0.jpg'),
    builder=pyocr.builders.TextBuilder()
)
# 將 ocr 辨識結果優化
txt=result.replace("!","1") # 如果是 ! 字元，更改為字元 1
real_txt=re.findall(r'[A-Z]+|[\d]+',txt) #只取數字和大寫英文字母

#組合真正的車牌
txt_Plate=""
for char in real_txt:
    txt_Plate += char
print("ocr 辨識結果：", result)
print("優化後辨識結果：",txt_Plate)

cv2.imshow('image', image)      #顯示原始圖形
cv2.moveWindow("image",500,250)#將視窗移到指定位置
key = cv2.waitKey(0)            #按任意鍵結束
cv2.destroyAllWindows()
```

執行結果:



四、結論

使用 OCR 辨識車牌號碼時就必須要有個完整乾淨的圖形，就能將車牌號碼辨識的很準確，要正確的擷取到車牌號碼和訓練的模型有很大的影響，要爭增加模型的準確性，就必須要有大量的車牌作訓練。即使正確的擷取車牌號碼，車牌大小、歪斜、不清晰、光線落差、反光以及拍攝角度，都會影響辨識的結果，因此，OCR 辨識比較適合在特地的場合，攝影機位置調整好，車牌大小、可控制角度以及光線好，這樣就可以達到很高的辨識率。

生活上除了車牌號碼可以透過影像辨識來管理，其他如通行證、身分證等，亦或是更複雜的文件上有文字、數字及特殊符號都可以透過人工智慧的影像辨識來處理，讓人們的生活更加便利，也更加安全。

五、參考資料

1. Python 機器學習超進化：AI 影像辨識跨界應用實戰
2. 巨匠電腦 Python 人工智慧整合開發課程