License Plate Recognition

使用Python 3.6的環境。

建立license\_plate\_recognition.py為判斷的主程式檔，

讀取同樣目錄中儲存車牌的資料夾license\_plates中的圖片。

license\_plate\_recognition.py中只有一個class ImageSearchANPR，實體化後來判斷資料。

作業程序：

1. 先利用cv2讀取圖片檔（jpeg）
2. 將其轉換成grayscale image，數入給find\_ocr\_and\_contour()
3. 把圖片中的暗處以白色顯示出來（blackhat）。
4. 找出圖片中的亮處，以白色顯示（ligth\_img）。
5. 將圖片呈現出x方向的梯度，把他們的x方向的亮度分界線找出來，並重新調整圖片的亮度。
6. 利用cv2.GaussianBlur模糊畫面，再將色塊內部增強一致性，最後利用cv2.threshold二元值化。另外也利用先膨脹後侵蝕的作用，將圖片鄰近的白色色塊變成大色塊，以助構築車牌的色塊。
7. 利用blackhat最後生成出的車牌白色色塊和light\_img的亮色處的重疊區，來判斷真正的車牌位置。
8. 利用cv2.findContours()找出白色色塊的輪廓，我抓出前5大面積的輪廓輸出。
9. 將輸出的輪廓，利用中華民國新式車牌號碼的規定：

長380 mm，高為160 mm，現標準車牌的長寬比（Aspect ratio）為，

因此我麼設定，。

若是白色色塊長寬比介於之間，即判定此輪廓為ROI (region of interest)，並由locate\_license\_plate()輸出此「輪廓」與「判定後的車牌區域圖片」。

1. 最後利用pytesseract轉換圖像為文字，輸出車牌號碼。

**Case 1:**





預測結果：ARC5678

可能成因：

此案例為B判斷成R，可能是左下角，太過貼齊邊緣，而Tesseract這個框架可能訓練的資料，都是有保留邊緣的，造成判斷錯誤。

解決辦法：

1. 在藉由locate\_license\_plate\_candidates()時，可以對gradX做cv2.morphologyEx()的iterations次數調高；或者是對thresh\_img做cv2.dilate()的iterations調高一點；或者將GaussianBlur()的ksize大一點，讓車牌的白色元素能夠足夠延展開。
2. 經嘗試後發現，Pytesseract中的若設定，預測結果為ABC5678，判斷正確，可能是此種模式對於圖片的文字處理，能達到邊緣最大化的關係。

EX:

psm=7 Treat the image as a single text line.

psm=11 Sparse text. Find as much text as possible in no particular order.

**Case 2:**





預測結果：PAN

**Case 2:**

可能成因：由於此畫面仍然有noise在判斷目標數字與字母中（即字體內部的黑點），而Pytesseract的識別極為敏銳，故造成判斷錯誤。



解決辦法：

可以利用cv2.morphologyEx(op= cv2.MORPH\_CLOSE)增加iterations次數，或將kernel的大小變大一點，讓內部的塗色完整，減少雜訊。

**Case 3:**





預測結果：無

可能成因：

車牌本身最後和周圍的亮處合在一起，造成車牌周圍的白色矩形的長寬比，不在標準車牌的長寬比裡，因此無法收集為contour。

解決辦法：

由locate\_license\_plate\_candidates()數入圖片後，將圖片做一個亮度的初始化，再利用cv2.threshold調整到一個適當的值來減少車牌周圍的白色處。

**Case 4:**

****



預測結果：PTG3

可能成因：

解析度為的圖片，車牌經放大後，會太過模糊，導致判斷失效。

解決辦法：

1. 若是輸入圖片都改為vector image，則能避免此問題。
2. 解析度為，可能可以有比較好的判斷，需要再測試。

**Case 5:**





預測結果：YVR

**Case 5:**

可能成因：

此圖的ROI的雜訊偏多，可能因此讓Pytesseract判斷失效。



解決辦法：

可以先利用cv2.GaussianBlur()，在利用cv2.threshold(blur,0,255,cv.THRESH\_BINARY+cv.THRESH\_OTSU)能將邊緣清楚分開及塗色處顏色一致。

