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$\frac{380}{160} = 2.375$，
   因此我麼設定$minAR = 2$，$maxAR = 3$。
   若是白色色塊長寬比介於$minAR \sim maxAR$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」。
10. 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psm = 11，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

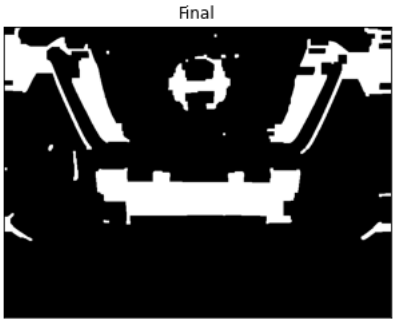
Case 2:

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$minAR = 2 \sim maxAR = 3$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:

Original

License Plate

預測結果：PTG3

可能成因：
解析度為600 × 496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1920 × 1080，可能可以有比較好的判斷，需要再測試。

Case 5:

Original

License Plate

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。