

National Sun Yat-sen University
Department of Electrical Engineering

112-1 機器學習系統設計實務與應用

HW1 OPENCV影像處理

姓名：胡庭翊

學號：B103015006

組別：你說的隊







1. 邊緣偵測

1.1. 運算流程圖



圖一、流程圖

1.2. 圖像輸出結果

讀取圖片	灰階處理	前濾波
		
邊緣偵測	後濾波	二值化處理(輸出)
		

表一、各階段輸出結果

1.3. 分析與討論

從路徑中讀取待處理的圖片後，圖片會經過以下步驟的處理：

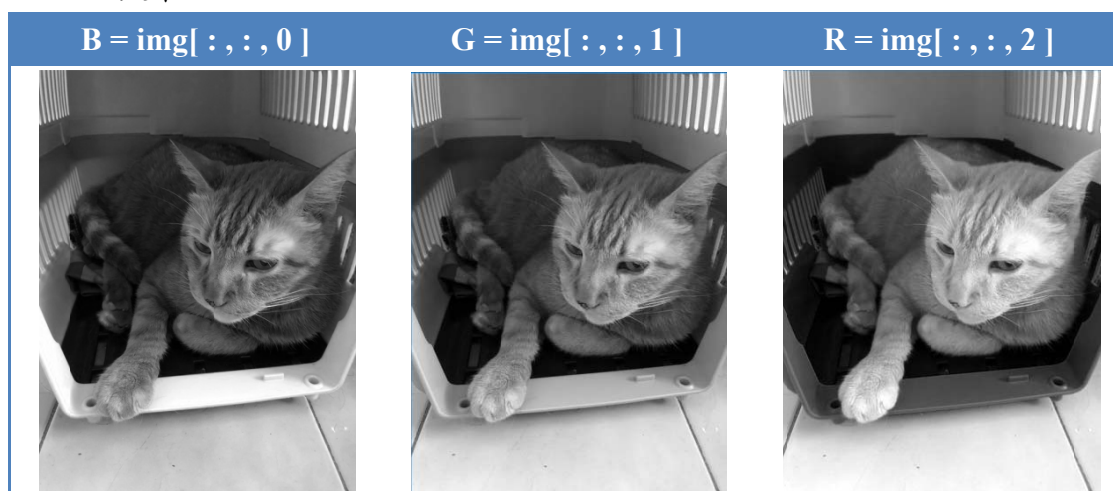
1.3.1. 灰階處理

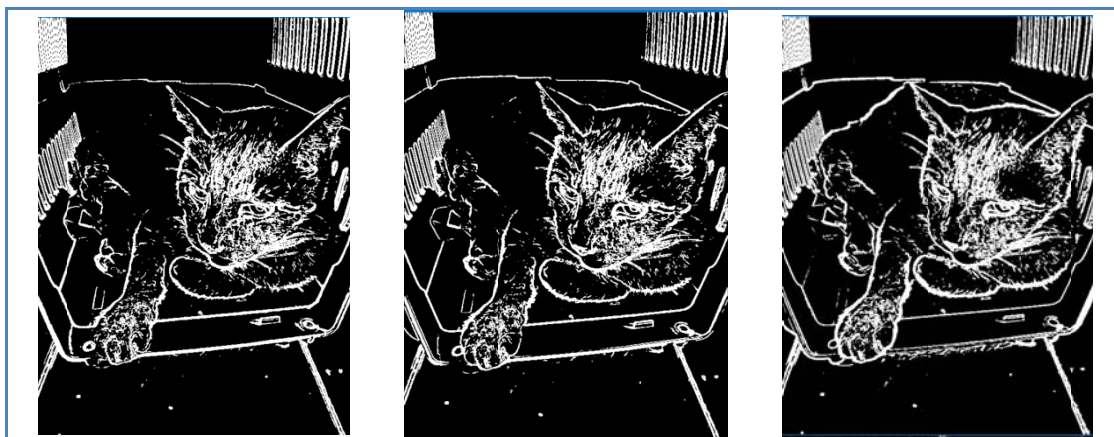
原先助教給的範本是調用 cv2 函式庫「cvtColor」內的其中一個功能，將圖片的 RGB 數值轉換成灰階。然而，在測試階段時我發現不管使用什麼方法去做處理，例如改變濾波及邊緣偵測的方式或調整閾值，貓咪的身體都沒辦法完整的展現出來。我想其原因是出在灰階處理後貓咪的身體本身和後面背景邊界不明顯，才會讓後續的影像處理沒辦法有效改善，造成身體輪廓不明顯。



圖二、原先的灰階處理效果與最終輸出結果

因此我直接提取圖像單個通道的像素去做處理，可以發現，當控制相同的濾波以及邊緣方法與二值化時，使用 `img[:, :, 2]` 時可以達到比先前更好的效果。





表二、分別提取 BGR 檢視邊緣處理的效果

1.3.2. 前濾波

濾波的目的是為了將雜訊清除，而範本內的方法分別有 blur、GaussianBlur、medianBlur、bilateralFilter 可以選擇。由於使用肉眼實在是無法比較出濾波效果，因此考量到影像處理中最常出現的雜訊是高斯雜訊 Gaussian 跟鹽與胡椒雜訊 Impulse，這裡我使用 GaussianBlur 作為我的前濾波器。

blur



GaussianBlur



medianBlur



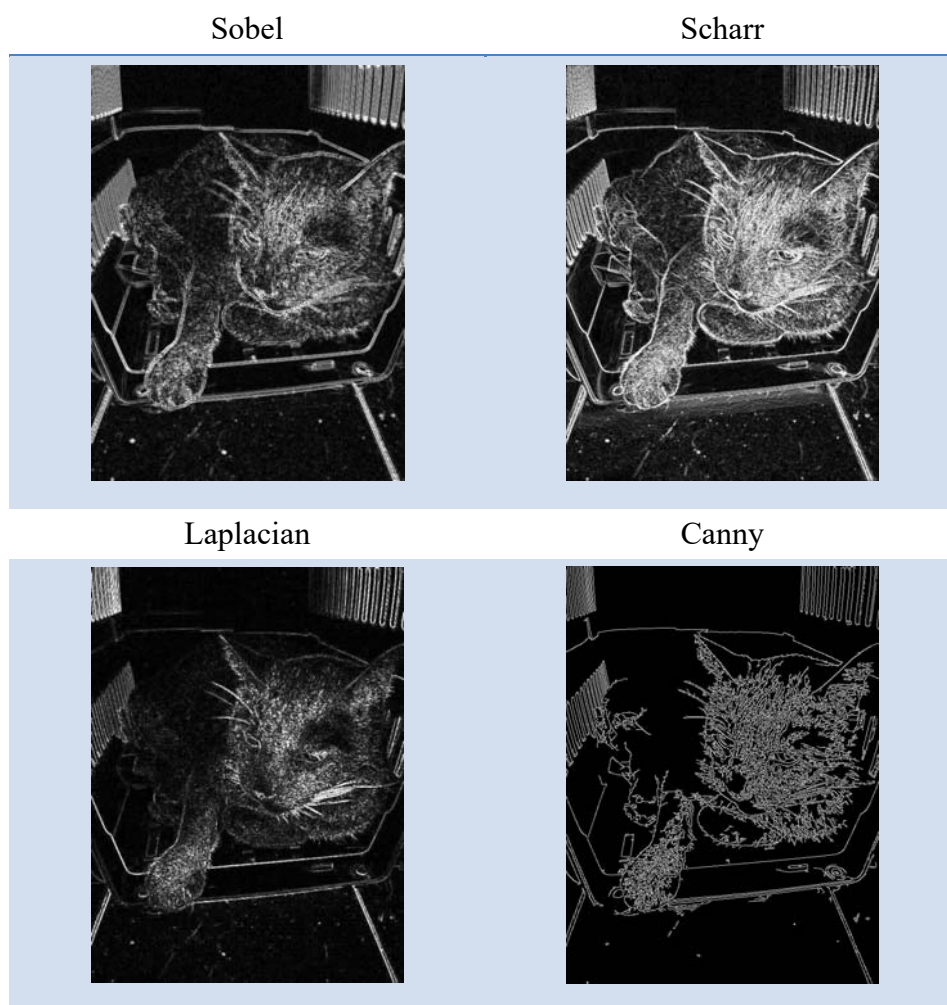
bilateralFilter



表三、四種濾波方法

1.3.3. 邊緣偵測

邊緣偵測的目的是標識圖像中亮度變化明顯的點，圖像屬性中的顯著變化通常反映了屬性的重要事件和變化。除此之外邊緣偵測也大幅度的減少了數據量，剔除了可以認為不相關的信息，保留了圖像重要的結構屬性。而範本內的方法分別有 Sobel、Scharr、Laplacian、Canny 可以選擇。經過測試處理，我認為 Sobel 與 Scharr 最能達到理想的結果，Laplacian 的話邊緣處理不太明顯，而 Canny 則是省略了過多要素。經測試，Sobel 與 Scharr 兩者二值化處理的閾值不同，Sobel 需要的閾值較小，而 Scharr 需要的較大，兩者皆可以達到相似的最終輸出結果。不過，就貓的後半身體邊緣偵測效果而言，還是 Scharr 略勝一籌。



表四、四種邊緣偵測方法

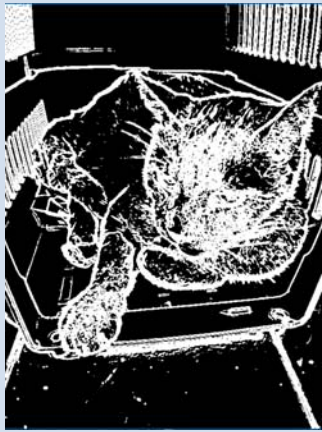





1.3.4. 後濾波

後濾波的處理與前濾波相仿，皆是為了消除雜訊，將影像平滑化處理。由於高斯低通濾波器 GaussianBlur 通常會造成影像模糊，故我的後濾波處理選擇使用雙邊濾波器 bilateralFilter。雙邊濾波器能偵測保留影像之線條，只對位於線條同一邊的像素執行平均平滑作用，我認為是做完邊緣偵測後的後濾波器最佳選擇。

1.3.5. 二值化處理

二值化又稱為灰度分割，是圖像分割的一種最簡單的方法。二值化可以把灰度圖像轉換成二值圖像。把大於某個臨界灰度值的像素灰度設為灰度極大值，把小於這個值的像素灰度設為灰度極小值，從而實現二值化。它的目標是將灰度圖像中的每個像素點分為兩個類別：一個是前景（通常是物體或感興趣的區域），另一個是背景。

這裡，由於我最終選擇的是 Schar 邊緣偵測方法，所以經過測試，將閾值設為 110 能達到最佳的效果，閾值再往上會造成過多的主體被省略，而往下則是造成過多的背景被提取。

閾值 輸出 圖像	70	90	110
			
閾值 輸出 圖像	130	150	170
			

表五、閾值大小對輸出結果的影響

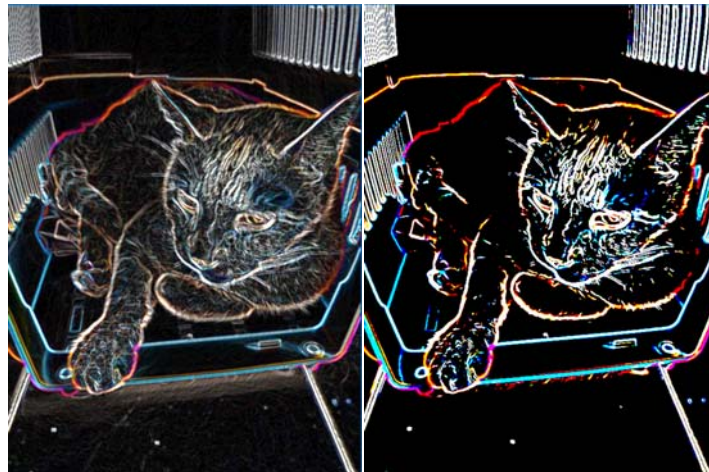
1.3.6. 輸出結果

二值化處理後再將圖片透過 imwrite 寫入指定路徑的資料夾內便完成圖像的輸出。



圖三、最終輸出結果

另外，我意外發現若不小心忘記做灰階處理，邊緣偵測後的圖像輸出結果會在邊緣處顯現比原圖更鮮豔的色彩線條，還蠻有藝術感的。



圖四、未經過灰階處理的邊緣偵測與二值化處理後的書出結果













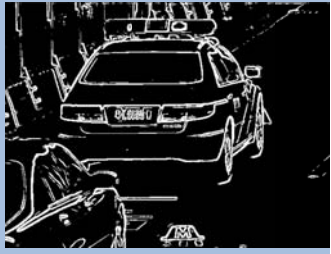


2. 車牌辨識

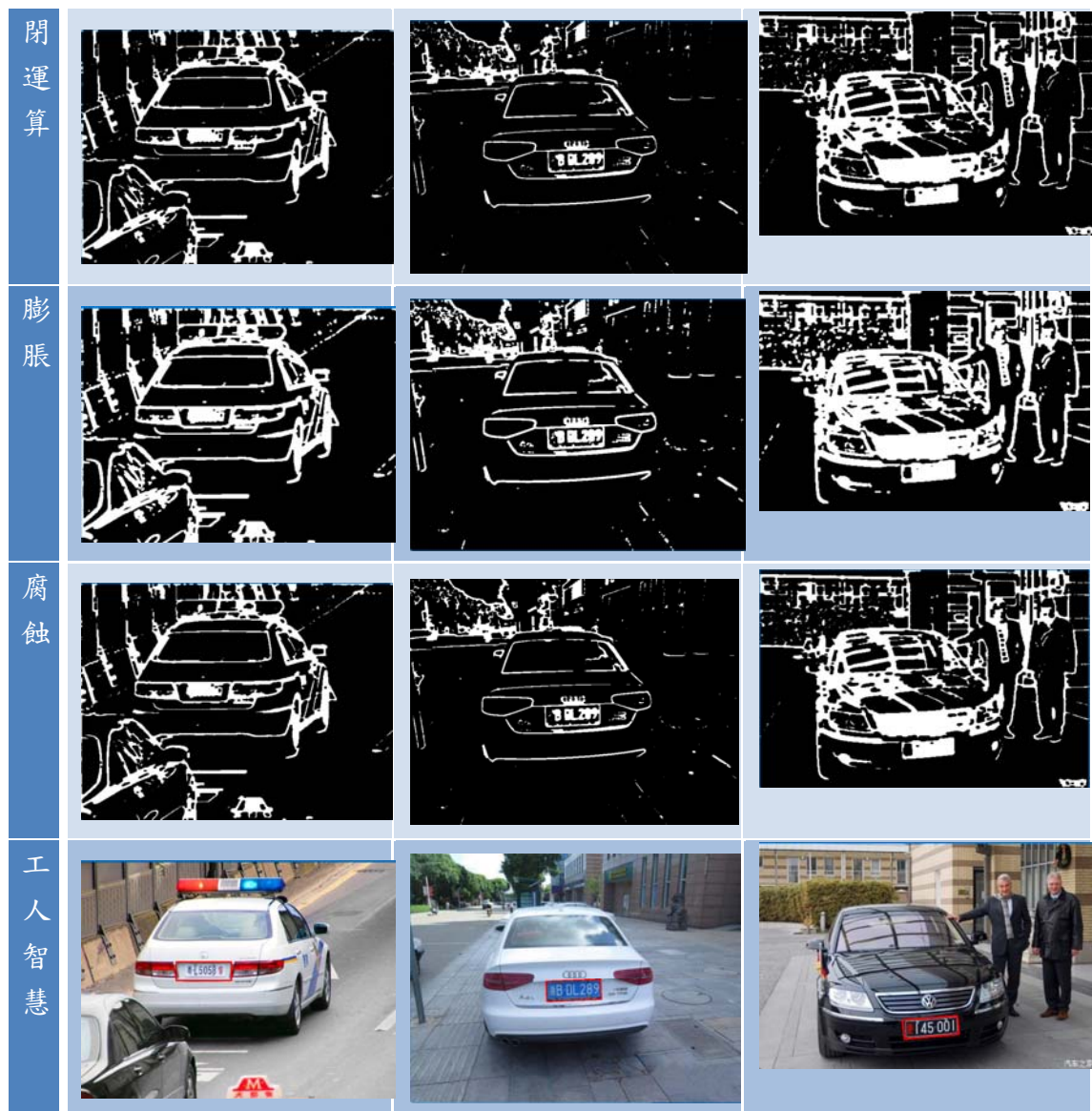
2.1. 運算流程圖



圖五、流程圖

2.2. 圖像輸出結果

	Picture 1	Picture 2	Picture 3
讀取圖片			
灰階處理			
濾波			
邊緣偵測			
二值化			



表六、各個階段輸出結果

2.3. 分析與討論

從路徑中讀取待處理的圖片後，圖片會經過以下步驟的處理：

2.3.1. 灰階處理

由於沒有像第一個任務時一樣需要加強灰階處理的部分，故此步驟我使用的是助教給的範例代碼，利用 cv2 內建函式將圖像 BGR 轉換成灰階。

2.3.2. 濾波

因為處理過後的圖像還需進行腐蝕膨脹等處理，所以我省略了後濾波的動作，單純在邊緣偵測前做 GaussianBlur 濾波，將雜訊過濾同時也不免的將圖像模糊化。

2.3.3. 邊緣偵測

我在這個階段使用的是 Sobel 邊緣偵測方法，因為我認為車牌辨識不需要像 Scharr 方法一樣將所有圖像主體的邊緣完整提出，因此 Sobel 成為了我

的首選方法。

2.3.4. 二值化

二值化的閾值以及膨脹腐蝕的次數都會影響車牌辨識的結果，所以我使用了 for 迴圈一個一個組合去嘗試求得最佳解。

經測試，我發現當 GaussianBlur 濾波器的大小等於 5 時，二值化閾值設為 70，並經過一次腐蝕一次膨脹能成功將車牌選取出來。

2.3.5. 訂定核心結構

在形態學的操作中，需要訂定一個核結構 kernel，這裡我依循範本的方法，使用 OPENCV 內的函數 `getStructuringElement(shape, ksize, anchor=None)` 進行核結構的定義，將其設為橢圓型態。其中，`shape` 表示內核的形狀，`ksize` 表示內核的尺寸 (n, n)，除了橢圓外，我們還有另外兩種形狀可以選擇：

橢圓：`cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))`

十字形：`cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))`

矩形：`cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))`

element_ellipse	element_cross	element_rect
<pre>[[0 0 1 0 0] [1 1 1 1 1] [1 1 1 1 1] [1 1 1 1 1] [0 0 1 0 0]]</pre>	<pre>[[0 0 1 0 0] [0 0 1 0 0] [1 1 1 1 1] [0 0 1 0 0] [0 0 1 0 0]]</pre>	<pre>[[1 1 1 1 1] [1 1 1 1 1] [1 1 1 1 1] [1 1 1 1 1] [1 1 1 1 1]]</pre>

圖六、以尺寸為(5,5)為例的核心型態

2.3.6. 閉運算

閉運算是先膨脹、後腐蝕的運算，有助於關閉前景物體的小孔，或去除物體上的小黑點，還可以將不同的前景影像進行連接。這裡，我使用的是範本的方法，透過 OPENCV 中的函數 `cv2.morphologyEx(th1, cv2.MORPH_CLOSE, kernel)` 進行運算。

經過閉運算後其實車牌的辨識已經有了一定的效果，然而尚有改善的空間，故我後續還有再進行膨脹以及腐蝕的個別處理。



圖七、閉運算後的車牌辨識成效

2.3.7. 膨脹

膨脹操作是形態學的其中一種操作，能對影像的邊界進行擴張，用以平滑對象邊緣並減少與填充對象之間的距離。使用內建函式，經過迴圈測試，這裡我選擇進行一次的膨脹。

2.3.8. 腐蝕

腐蝕操作與膨脹操作的作用是相反的，它能夠將影像的邊界點消除，使影像沿邊界內收縮。腐蝕用來「收縮」或「細化」二值影像的前景，借此實現去除雜訊、元素分割等功能，這裡我同樣參考範本使用內建函式，經過迴圈測試，選擇進行一次的腐蝕。

2.3.9. 工人智慧

在使用 OPENCV 內的 `cv2.findContours()` 函數來將物體的輪廓標示出來後，我們使用經驗法則，也就是工人智慧輸出車牌辨識的最佳結果。

這裡，我參考先前曾經成功過的例子，打開 Box 觀察框定點的四個邊角座標之間有什麼關係。

Picture 1			Picture 2			Picture 3		
	0	1		0	1		0	1
0	182	189	0	326	307	0	234	287
1	286	189	1	457	302	1	323	280
2	286	224	2	459	351	2	326	310
3	182	224	3	328	356	3	237	318

表七、Box 的四個框定點理想座標

我發現，當框定點的長寬比值在 5 到 1.5 這個範圍內，且長在 80 到 145 之間、寬在 25 到 70 之間時，該框定點極為可能是我們的目標物車牌。

```
if (5 > (Box[1][0] - Box[0][0]) / (Box[3][1] - Box[0][1]) > 1.5) and 80 < (Box[1][0] - Box[0][0]) < 145 and 25 < (Box[3][1] - Box[0][1]) < 70 :  
    possible_img = cv2.drawContours(possible_img, [Box], -1, (0, 0, 255), 2)  
    cv2.imshow('possible_img', possible_img)  
    break
```

圖八、工人智慧判斷式

2.3.10. 輸出結果

最後的結果如下，可以觀察到方框精準的框定在車牌上。

