# 機器學習系統設計實務

# HW1

姓名 ： 林凡皓

學號 : B103012002

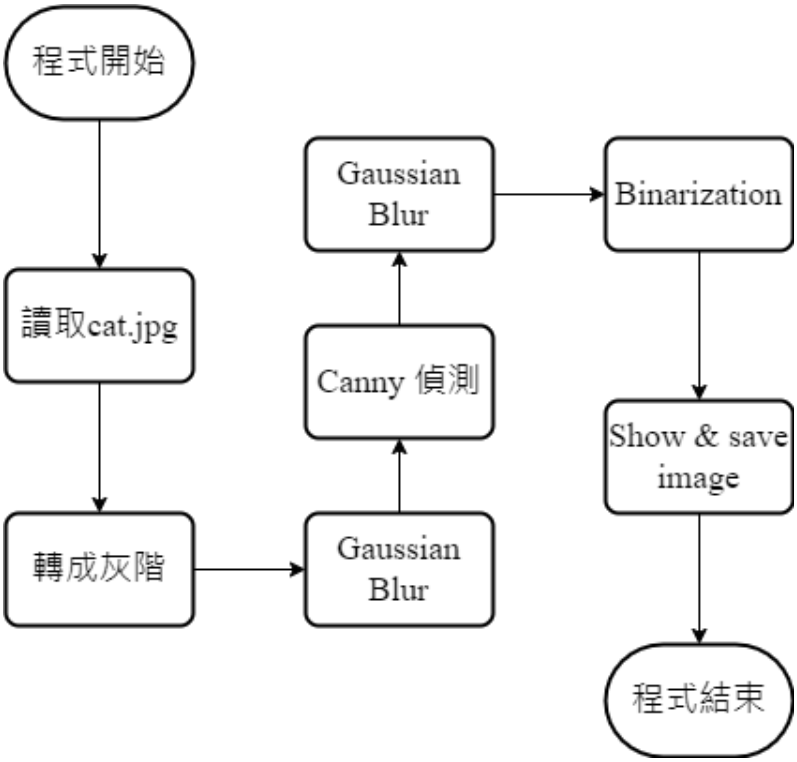
組別 ： 為什麼要醬組

# I. Edge Detection

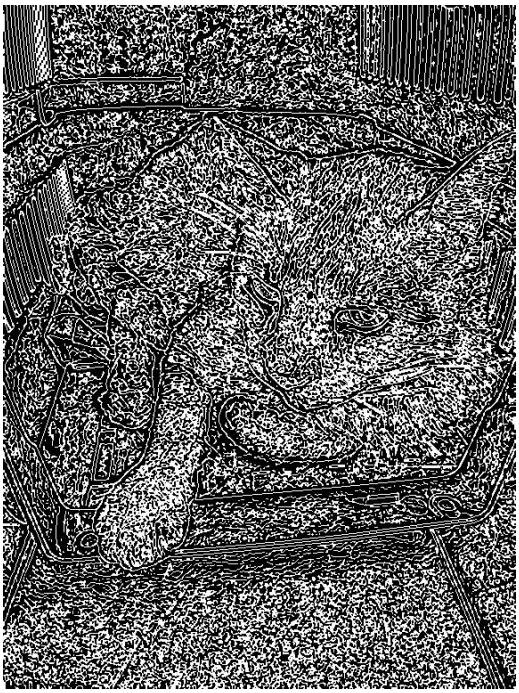
這次邊緣偵測作業中我嘗試四種方式，分別是 Canny、Sobel、Scharr 和 Laplacian。

1. Canny :

(1) 流程圖 ：

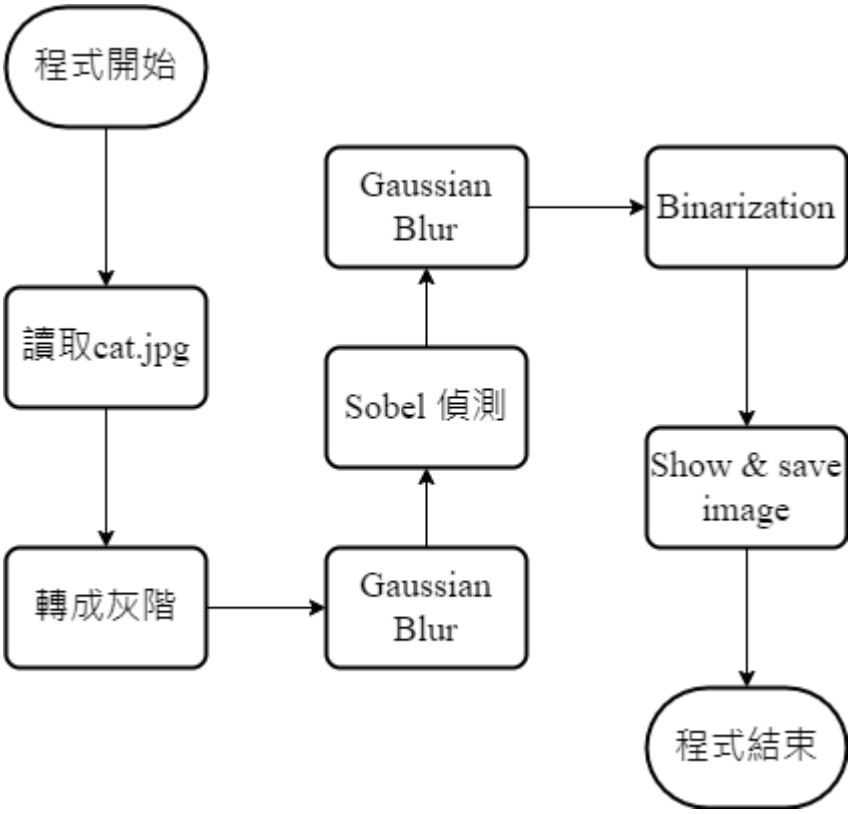



(2) 圖像輸出結果 ：



2. Sobel :

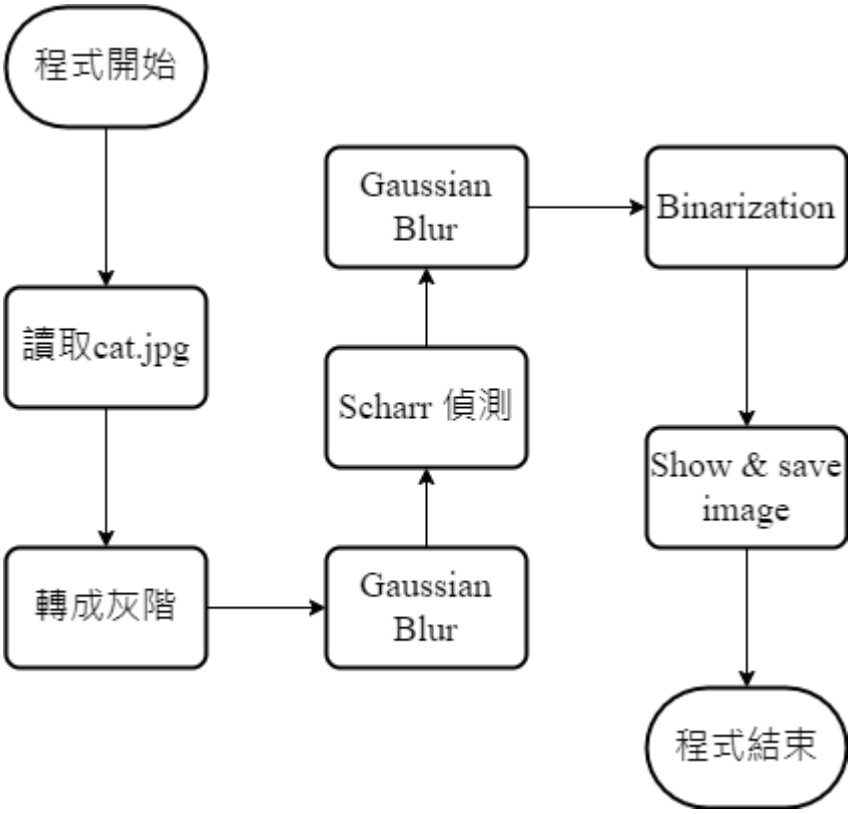
(1) 流程圖 ：




(2) 圖像輸出結果 ：

3. Scharr :

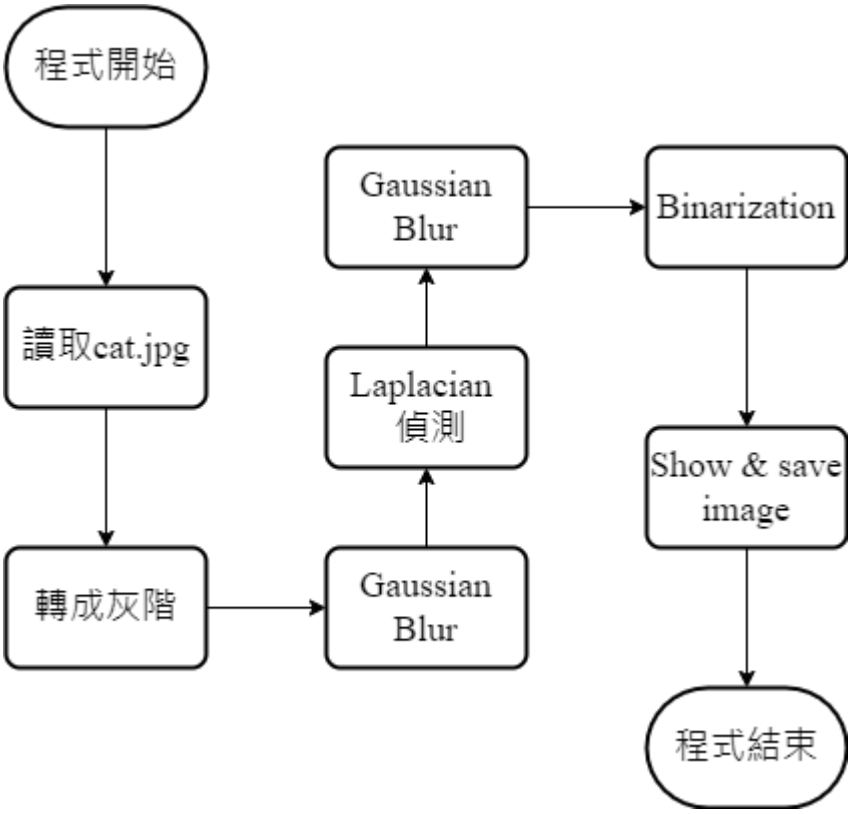
(1) 流程圖 ：



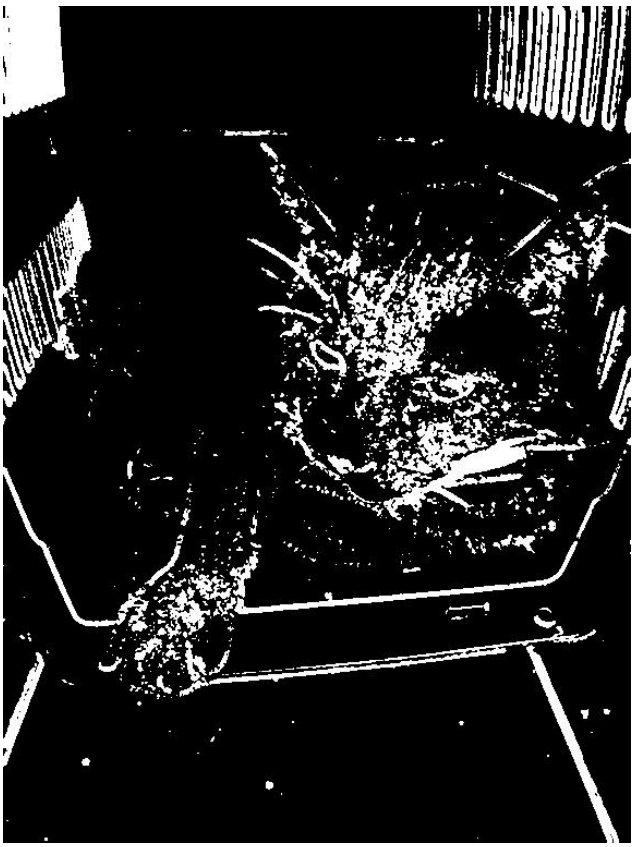

(2) 圖像輸出結果 ：

4. Laplacian :

(1) 流程圖 ：




(2) 圖像輸出結果 ：



5. 分析與討論：

(1) 圖像輸出結果：

由上面四張圖可以很明顯的看到，Canny 的輸出結果非常的細節，他連一些非常細微的小邊緣都能夠偵測出來。Sobel 是抓取一些大輪廓，將整張圖片用簡單的輪廓表現出來。Scharr 和 Sobel 一樣是抓取大輪廓，但是比起 Sobel，Scharr 又更加細節一點。Laplacian 看起來也是要抓輪廓，但是在這張圖片上的表現不如前面三種方式好。

(2) 四者間的優劣：

以這張貓咪圖片來說，Laplacian 的表現顯然不如其他三者。

如果是以"邊緣偵測"來判斷說哪一種方式比較好，那我覺得是 Canny，畢竟 Canny 基本上把所有邊界都抓出來了，甚至是細微到人類都不一定找的到的部分他也能找到。但是就"圖像偵測"來說，我覺得 Sobel 和 Scharr 比較實用，因為從它們的圖像輸出結果比較容易抓到一些特徵，像是貓咪的爪子、貓咪的眼睛、貓咪的頭……等。但是對於 Canny 的圖像輸出來說，不仔細去看的話只會覺得是一些雜亂的線條，而從中要抓出一些關鍵特徵便不像 Sobel 和 Scharr 的圖像輸出一樣容易。

如果更進一步拿 Sobel 和 Scharr 做比較，我認為兩者各有優缺。這邊舉兩個例子來說：

第一，要偵測整隻貓咪：

在這個範例中我認為 Scharr 比較好。我們可以注意到，在偵測貓咪後半身的邊界的時候，Scharr 是有成功偵測到的，但是 Sobel 沒有，這就會影響到偵測整隻貓咪的準確度。

第二，要偵測眼睛所在：

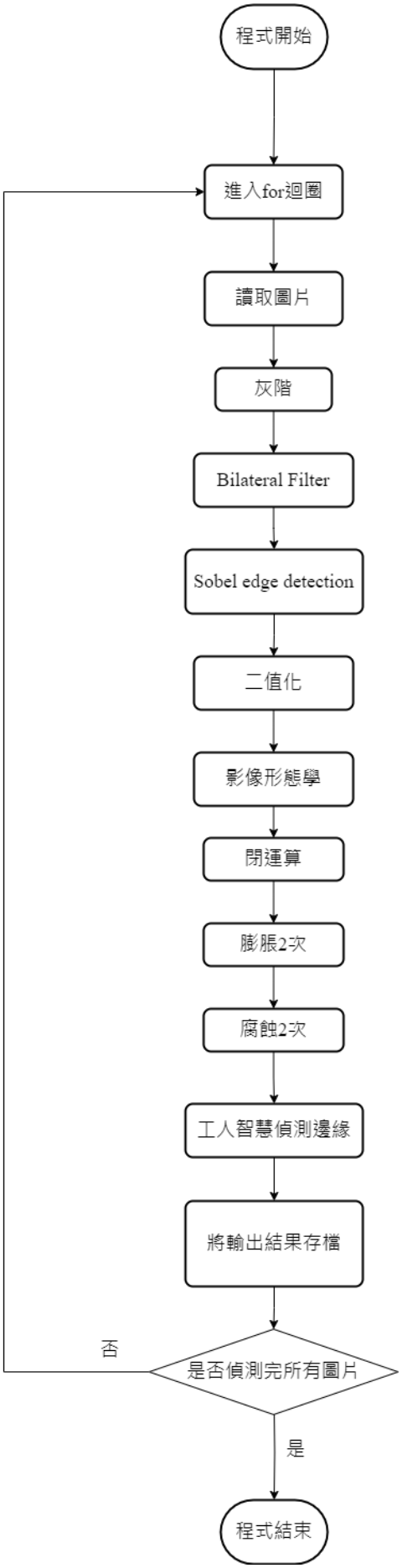
在這個範例中我認為 Sobel 比較好。我們可以看到在抓取貓咪臉部邊界時，會因為貓咪毛髮而使邊界非常多，在這種情況下如果抓取太細節，結果就會像 Scharr 的圖像輸出結果一樣有些雜亂，造成眼睛的位置不易辨別。但是如過是 Sobel，他會很剛好的忽略掉一些太過於細節的部份，讓眼睛的部分變得比較突出，造成更容易抓取眼睛的位置。

此外，如果考慮到運算時間跟複雜度，我認為 Canny 會需要較長的時間，畢竟要那麼詳細的抓到邊界肯定需要更複雜的演算法來支撐。以實用性來說，在這次作業中我傾向使用 Sobel 或 Scharr。

## II. License Plate

1. 方法說明：一開始做作業的時候，我一直以為要將車牌變得很清晰，像是車牌上的數字、文字等，但是經過一番嚐試之後，我發現到並不是要讓車牌變得很清楚，應該是要讓車牌的邊界變得很清楚，因此我就開始利用腐蝕和膨脹讓邊緣變得清晰，結果會如下面三張圖所示，這樣電腦才能夠更容易抓取到我們要的部分。

2. 流程圖：


程式開始
進入for迴圈
讀取圖片
灰階
Bilateral Filter
Sobel edge detection
二值化
影像形態學
閉運算
膨脹2次
腐蝕2次
工人智慧偵測邊緣
將輸出結果存檔
是否偵測完所有圖片
否
是
程式結束

3. 圖像輸出結果：

- Image 01：

這張圖片的困難處在於很容易誤抓到照片下方的計程車標誌，後來我是在工人智慧選擇方框的地方加上長要大於寬 40 的限制才成功避免掉抓到計程車標誌的問題。

- Image 02：

這張圖的困難出在於很容易抓到車窗以及背景樹木那一區塊。最終我是透過限制方框的大小才成功解決。

- Image 03：

這張是我認為最難的一張圖，因為車子是黑色的，而且車牌上方的銀色框框很容易讓程式誤判。對於這張圖，我選擇比較少次的膨脹與腐蝕，不然會變成一大塊黑色的。在二值化的地方我使用較大的 threshold，邊緣偵測演算法的部分也有特意挑選。

4. 額外說明：這次的車牌偵測作業中，我認為比較重要的參數為邊緣偵測的演算法選擇、二值化的 threshold、膨脹次數和腐蝕次數。我之所以選擇使用 Sobel 作為邊緣偵測演算法是因為在邊緣偵測作業中，我得出的結論為 Sobel 和 Scharr 為比較適合抓取輪廓的演算法，再考慮到第三張圖的車牌很小且車身為黑色，使用 Scharr 再加上膨脹會很容易把邊界模糊掉，因此我做中選擇用 Sobel。二值化的 threshold、膨脹和腐蝕次數我都是以第三張圖為基準做微調，最終 threshold 為 110、膨脹和腐蝕次數皆為 2。
5. 實際應用的考慮：雖然說這次的參數成功將這次作業的車牌都抓到了，但是要將這個模型應到現實世界是不可能的，我隨意到網路上找幾張車子的圖片然後做偵測，結果全部失敗，如下面三張圖。




圖片出處：

下圖：https://www.techbang.com/posts/80290-which-brand-is-the-most-popular-car-of-taiwanese

左上：https://c.8891.com.tw/feature/2881

右上：https://incar.tw/post/chinese-evs-are-strong-selling-in-euro

對於這樣的結果我認為是合理的，原因如下：

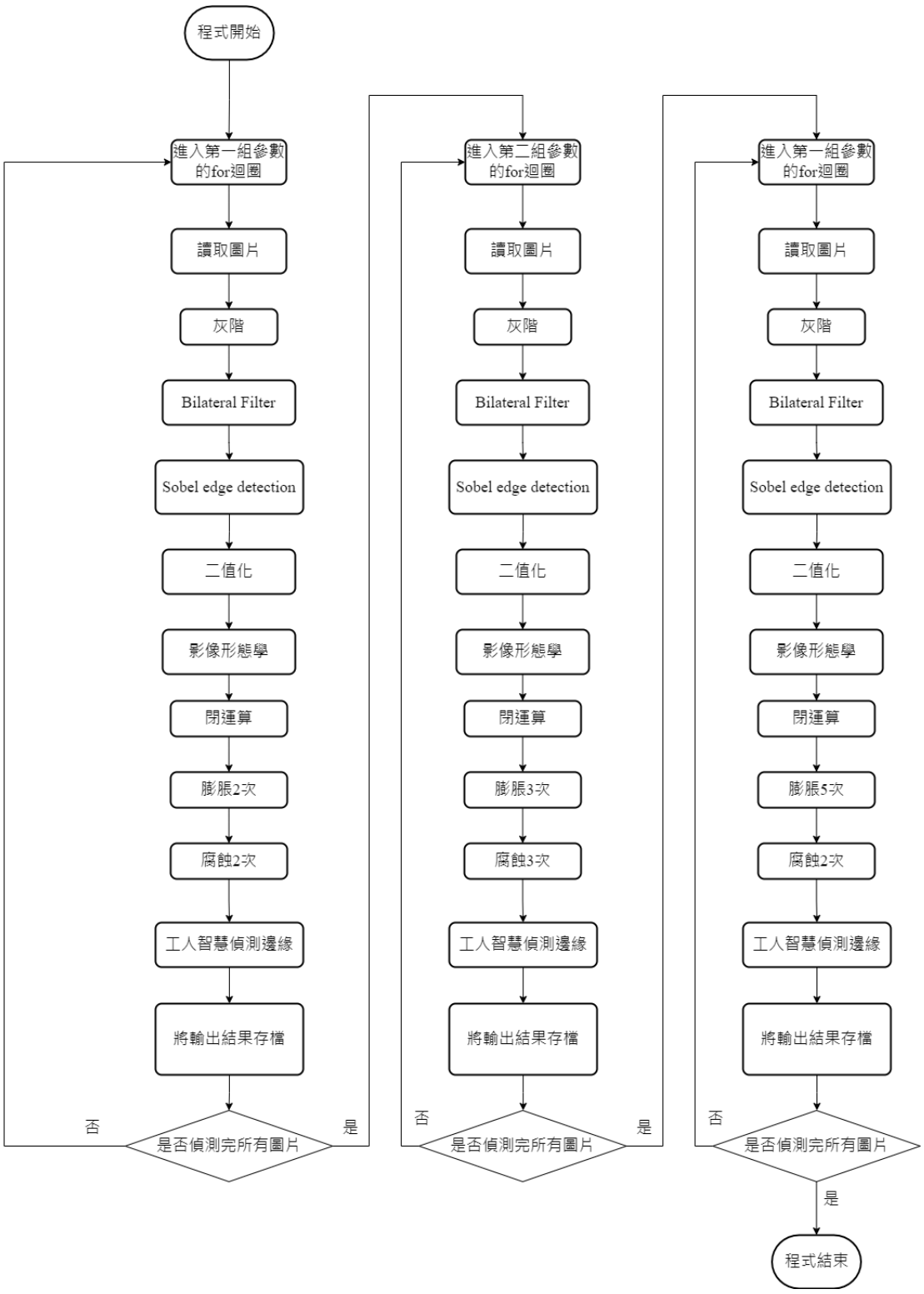
- 每張圖片的車牌大小都不一樣，在現實世界中不太可能會有每張圖車牌大小都差不多的情況。
- 車子顏色、車子造型、車子裝飾等都是影響抓取邊緣的結果的因素。

因此我認為想要用一組參數就完成一個車牌偵測的系統是不可能的，對於現實生活中的應用還有非常多的因素需要考慮，因此如果要設計一個能夠應用在生活上的車牌偵測系統，應該要制定多組參數來應對多種情況，而這也是為甚麼網路上車牌偵測系統都要使用神經網路這種複雜的模型，而不是這次作業這種肩擔的系統。

接下來會展示與說明握簡單改良的模型。

## III. 改良版 License Plate

1. 設計理念：為了修改第一版本對於我自己找的三張圖片偵測錯誤的問題，我採用多組參數來強化我的模型。會有這樣的想法來自於神經網路。神經網路的主要概念就是它能夠針對不同的情況去尋找到合適的參數，而這次改良版本就是仿效針對不同情況去找參數的概念。
2. 實現方式：主要是針對第一版本的模型做擴大，增加一些參數的組合來讓模型可以對多樣的情況做出合理的判斷。我主要是針對我自己找的三張圖片再去尋找兩組參數。
3. 流程圖：


程式開始
進入第一組參數的for迴圈
讀取圖片
灰階
Bilateral Filter
Sobel edge detection
二值化
影像形態學
閉運算
膨脹2次
腐蝕2次
工人智慧偵測邊緣
將輸出結果存檔
是否偵測完所有圖片
否
是
進入第二組參數的for迴圈
讀取圖片
灰階
Bilateral Filter
Sobel edge detection
二值化
影像形態學
閉運算
膨脹3次
腐蝕3次
工人智慧偵測邊緣
將輸出結果存檔
是否偵測完所有圖片
否
是
進入第一組參數的for迴圈
讀取圖片
灰階
Bilateral Filter
Sobel edge detection
二值化
影像形態學
閉運算
膨脹5次
腐蝕2次
工人智慧偵測邊緣
將輸出結果存檔
是否偵測完所有圖片
否
是
程式結束

4. 圖像輸出結果：

作業提供的三張圖：




可以看到對於作業提供的三張圖，改良後的版本依樣能夠成功抓取車牌。

自己找的圖：




對於額外找的三張圖也能夠成功偵測車牌。

其他圖：

可以看到，改良後模型對於一些新的圖片也能夠成功偵測。

圖片出處：

左圖：https://technews.tw/2022/03/11/kia-ev6-taiwan/

右圖：https://www.mitsubishi-motors.com.tw/coltplus

5. 實際應用上的取捨：首先，我認為只用一組參數的模型是無法在現實生活中應用的，畢竟在實際應用中有很多種情況，在只有一組參數的條件下是無法有效的處理各式各樣的情況。接著就是使用大型神經網路的模型，像是 YOLO，以及像我的第二個版本這種自己建立少數幾組參數的模型之間的取捨。舉例來說，如果今天是要做在高速公路上超速車輛的車台辨識，那我覺得使用神經網路的模型會比較好。在高速公路上，車輛在不同車道、車速的不同造成車子模糊、多輛車自同時出現在圖像中等情況都是小型模型無法有效處理的因素，因此我們應該要犧牲一點硬體上的表現，像是使用更多記憶體空間、計算量加大來幫助我們處理這些外在因素。但是如果我們只是做停車場進出口的車牌辨識，那我認為使用小型模型會是一個很好的選擇。在這種應用情況拍攝到的圖像都是比較固定的，如這次作業使用的圖象，因此我們可以藉由使用較少的參數來達到節省記憶體空間、節省運算複雜度等好處。

## IV. 心得總結

從邊緣偵測的作業中，我最大的收穫就是對於四種邊緣偵測的算法有了大致的了解以及粗略的選擇判斷。對於這四種算法，我利用偵測的細節程度將這四種做法做排序，結果為 Canny > Scharr > Sobel > Laplacian。在應用上，我認為這四種做法會需要依據使用場合做取捨，例如在做車牌偵測時，我們更需要的是能夠抓取大致輪廓的演算法，因此 Sobel 或是 Scharr 會是不錯的選擇。如果使用場合會需要抓取到很多細節，那 Canny 就會是

一個不錯的選擇。

我花了非常多的時間在調整車牌偵測作業的參數，從一開始試圖讓車牌顯示得很清晰，到後來了解到應該是要將車牌邊緣便清晰，我很明確地感覺到我的進步以及學習到的事情。除次之外，我覺得很重要的收穫是了解到現實生活的應用要考慮到很多事情。調整二值化的 threshold、腐蝕和膨脹次數等都會對結果造成很大的影響，因此如果只是想要用一組參數來完成車牌偵測系統，雖然在硬體考量方面是很不錯的選擇，畢竟一組參數的系統需要的記憶體空間很小，在運算上也很容易，但是準確度非常差。我覺得這樣的準確度拿到現實應用是不可能的，我們應該要在硬體成本與系統表現上取得平衡。像是 YOLO 之類的演算法，雖然會需要更多的記憶體空間，計算複雜度也會提升很多，但是 YOLO 做出來的準確度拿到現實中應用才是有意義的。

# V. Reference


[1] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.openai.com/.

[2] pyimagesearch “OpenCV:Automatic License/Number Plate Recognition (ANPR) with Pyhton” https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/ .

[3] ShungYU Talk (2020-03-18) “Python OpenCV 影像邊緣偵測 Canny Edge Detection” https://shengyu7697.github.io/python-opencv-canny/.

[4] allenchen “Day3-當自駕車駕駛遇見 AI-Canny 邊緣檢測(Canny edge detection)” https://ithelp.ithome.com.tw/articles/10202295?sc=rss.iron

[5] STEAM 教學網 “影像邊緣偵測” https://steam.oxxostudio.tw/category/python/ai/opencv-edge-detection.html