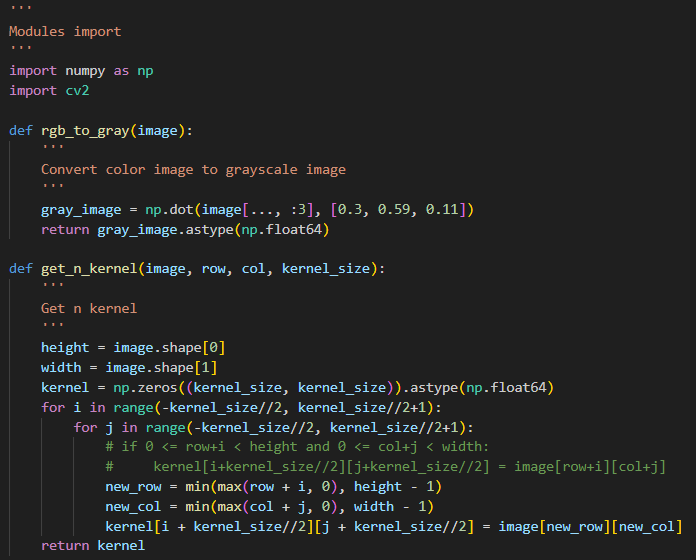
班級：資工三

學號：110590034

姓名：楊榮鈞



Function rgb\_to\_gray:

使用numpy的dot去實現。

rgb轉灰階是利用hw1的公式(0.3×𝑅)+(0.59×𝐺)+(0.11×𝐵)去轉成灰階。

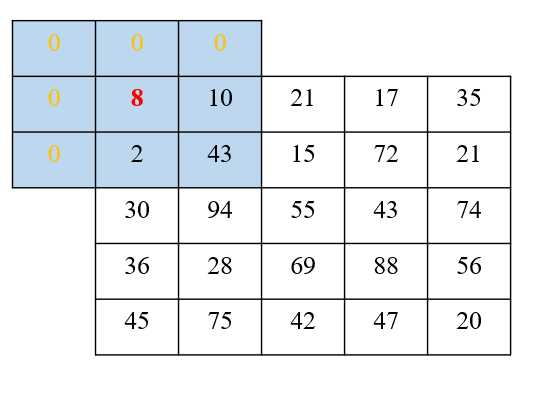
最後用astype(np.uint8)是因為現在的的範圍是0~255。

回傳gray\_image。

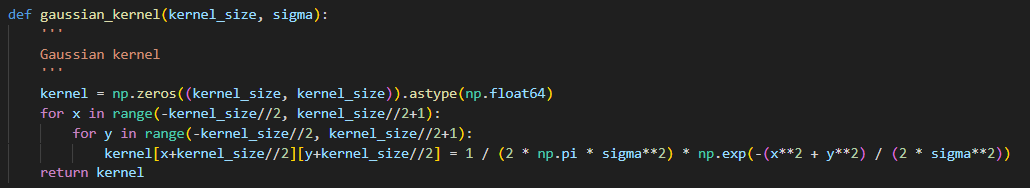
Function get\_n\_kernel:

先用image.shape找出height和width，再根據row和col(當前pixel的位置)抓取kernel\_size x kernel\_size大小的kernel。

其中我不是使用zero padding，而是超出size的範圍的話，則是取離此pixel範圍內離pixel最近的位置，如下圖所示。



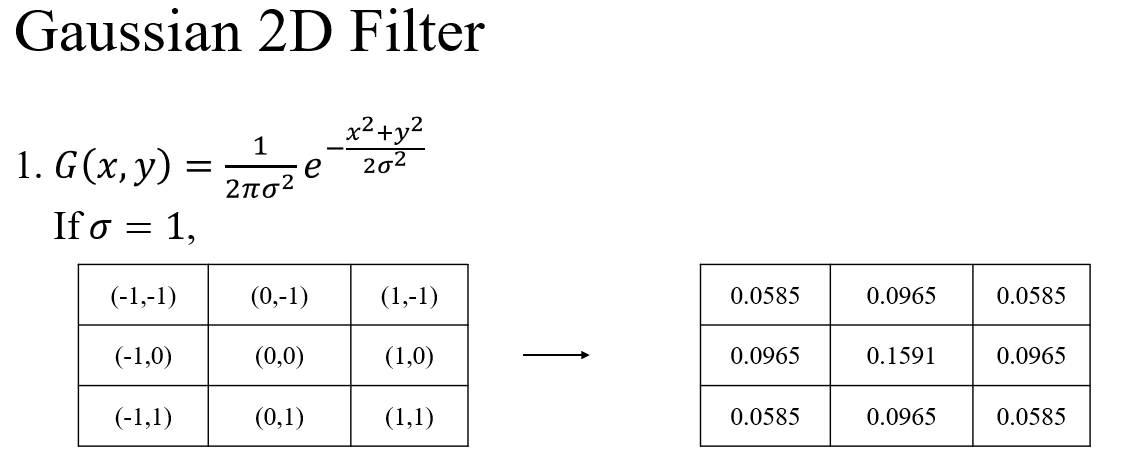


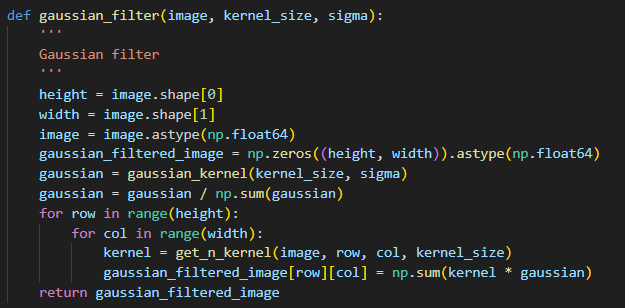


Function gaussian\_kernel:

計算gaussian的value，以Gaussian 2D filter的公式進行計算，如作業簡報的公式算法 G(x, y) = (1 / 2 \* pi \* sigma \*\* 2) \* exp \*\* (-(x \*\* 2 + y \*\* 2) / (2 \* sigma \*\* 2))。

其中會以kernel\_size的大小去做整個kernel的處理如圖所示(圖片是3x3的kernel大小)。



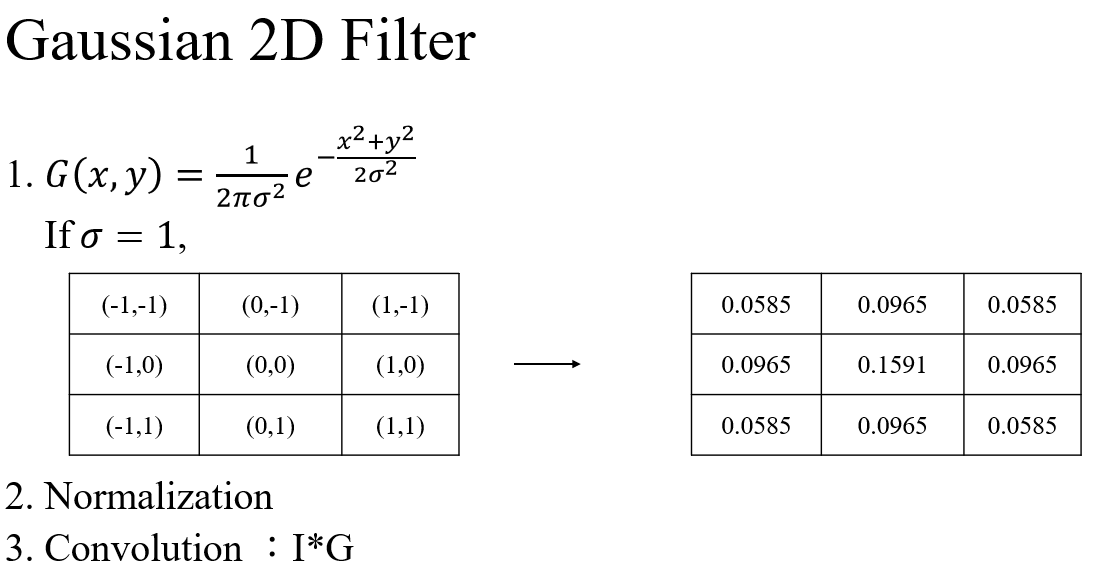


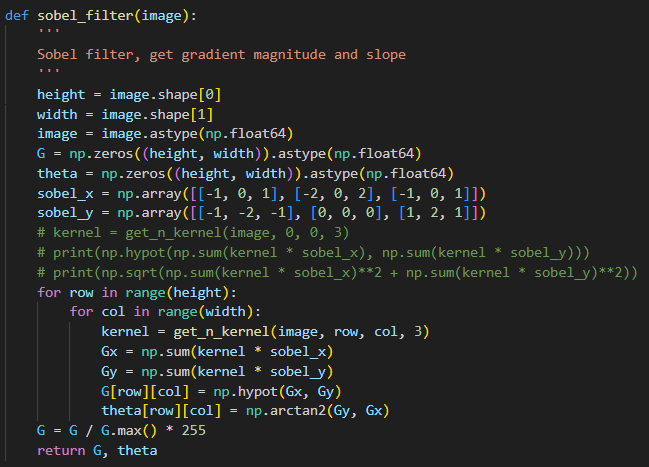
Function gaussian\_filter:

利用shape找出image的height和width，並利用astype設定image的type為np.float64，然後利用np.zeros創建gaussian\_filtered\_image的array。

利用gaussian\_kernel找出kernel的gaussian value，然後將gaussian除gaussian的總和並存在gaussian中。

利用for迴圈計算每個pixel的value，其中會先用get\_n\_kernel抓取當前pixel的kernel，並利用np.sum(kernel \* gaussian)得出kernel \* gaussian的總和，再存入當前pixel的value中。





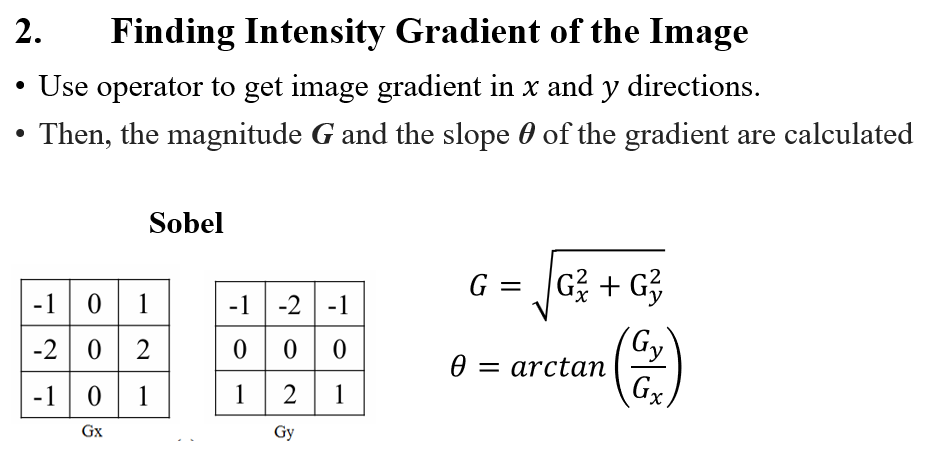
Function sobel\_filter:

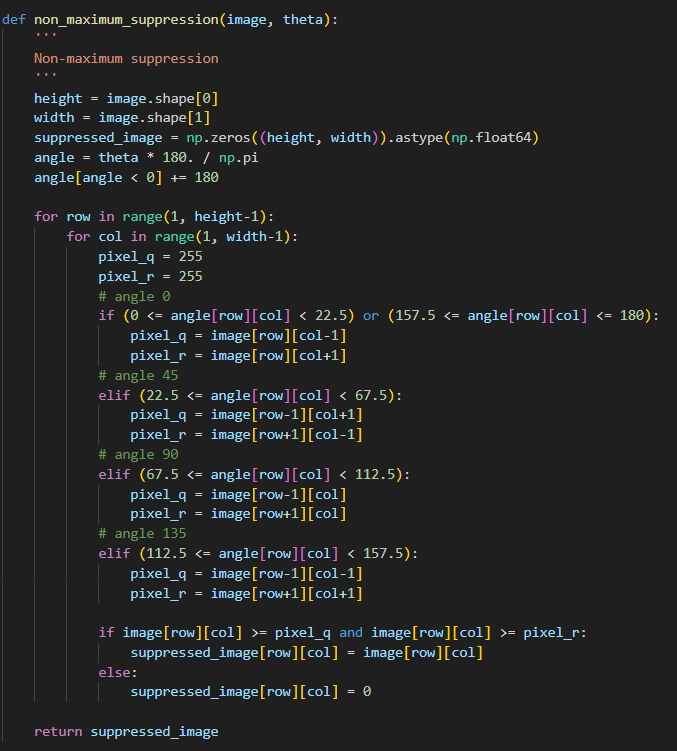
利用shape找出height和width，並使用astype把image的type設成np.float64，然後利用np.zeros創建G和theta分別存magnitude和slope，接著建立sobel\_x和sobel\_y的array。

註解的部分是觀察np.hypot(np.sum(kernel \* sobel\_x), np.sum(kernel \* sobel\_y))是否跟np.sqrt(np.sum(kernel \* sobel\_x)\*\*2 + np.sum(kernel \* sobel\_y)\*\*2)相等。

利用for迴圈計算每個pixel的magnitude和slope，首先會利用get\_n\_kernel找到kernel，然後找到pixel的Gx和Gy，計算magnitude((Gx2+Gy2)1/2)和slope(arctan(Gy/Gx))，並將結果分別存入G和theta。

最後計算完後，將G進行normalize使其的值在0-255間，這樣可以輸出圖片方便觀察。

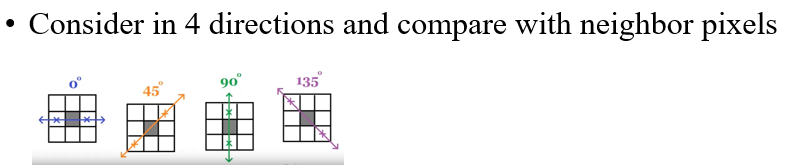


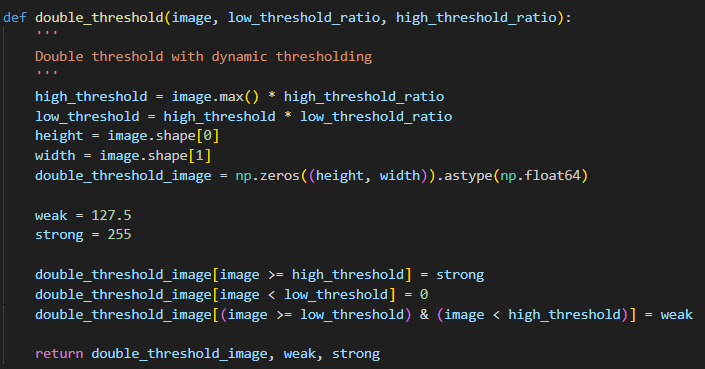


Function non\_maximum\_suppression:

利用shape找出height和width，利用np.zeros創建suppressed\_image的array，然後將theta轉成角度，並存入angle，接著在將angle中<0的值都加上180度。

使用for迴圈根據pixel的angle去判斷pixel在direction的neighbor中是否為最大的點，如果是的話會保留pixel的值，不是的話則設成0，如同下圖所示。





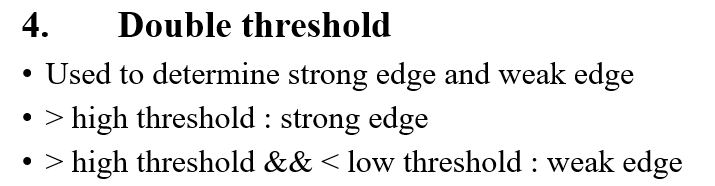
Function double\_threshold:

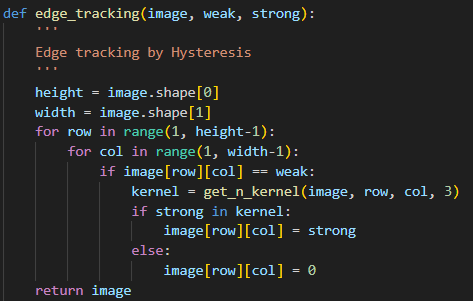
這次的threshold是利用ratio的方式(動態)去尋找high\_threshold和low\_threshold，其中high\_threshold會是image的最大值乘上high\_threshold\_ratio，low\_threshold是high\_threshold乘上low\_threshold\_ratio。

使用shape找出image的height和width，並利用np.zeros創建double\_threshold\_image的array。

將weak的值設成127.5，strong的值設成255

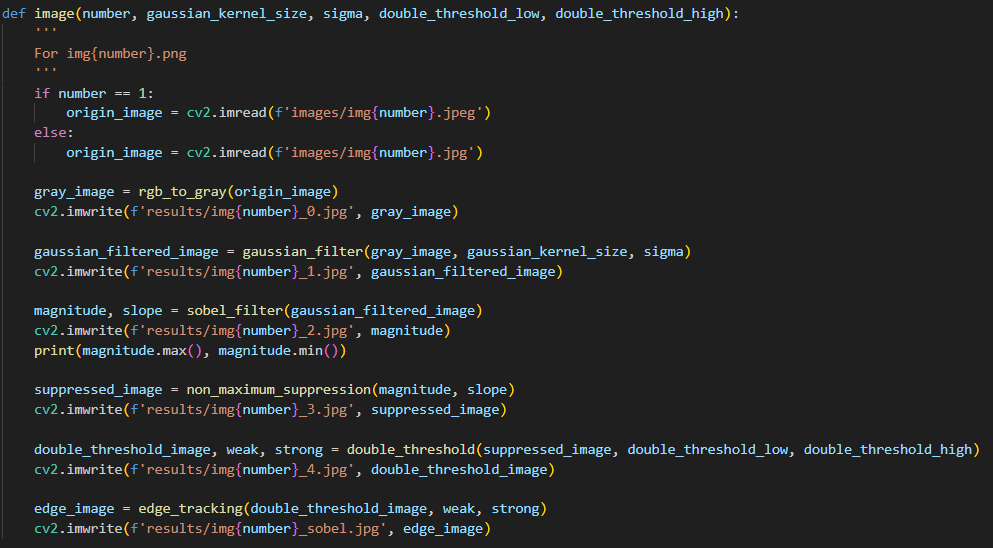
然後將image中pixel >= high\_threshold的value都設成strong，image中pixel < low\_threshold的value都設成0，image中low\_threshold <= pixel < high\_threshold的value都設成weak，如作業簡報上所示。





Function edge\_tracking:

利用shape找出image的height和width，然後使用for迴圈判斷所有weak edge是否為strong edge，如果是的話會將weak edge轉成strong edge，如果不是的話，則將此weak edge刪除(設成0)。做法是如果其pixel的kernel內含有strong的value，則將此pixel設成strong，如果沒有strong的value，則設成0。



Function image:

使用cv2的imread讀取圖片到origin\_image。

使用rgb\_to\_gray(origin\_image)得出gray\_image，然後使用cv2.imwrite存檔。

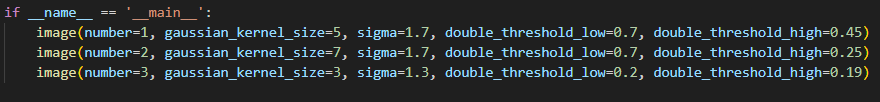
使用gaussian\_filter(gray\_image, gaussian\_kernel\_size, sigma)得出gaussian\_filtered\_image，然後使用cv2.imwrite存檔。

使用sobel\_filter(gaussian\_filtered\_image)得出magnitude和slope，然後使用cv2.imwrite將magnitude(sobel\_filtered\_image)存檔，並print出magnitude的最大值和最小值用來觀察。

使用non\_maximum\_suppression(magnitude, slope)得出suppressed\_image，然後使用cv2.imwrite存檔。

使用double\_threshold(suppressed\_image, double\_threshold\_low, double\_threshold\_high)得出double\_threshold\_image、weak和strong，然後使用cv2.imwrite將double\_threshold\_image存檔。

使用edge\_tracking(double\_threshold\_image, weak, strong)得出edge\_image，然後使用cv2.imwrite存檔。



執行image，產生所有助教給的圖片的edge\_image。

Result images

img1.jpg

(gray image)



(gaussian\_filter)



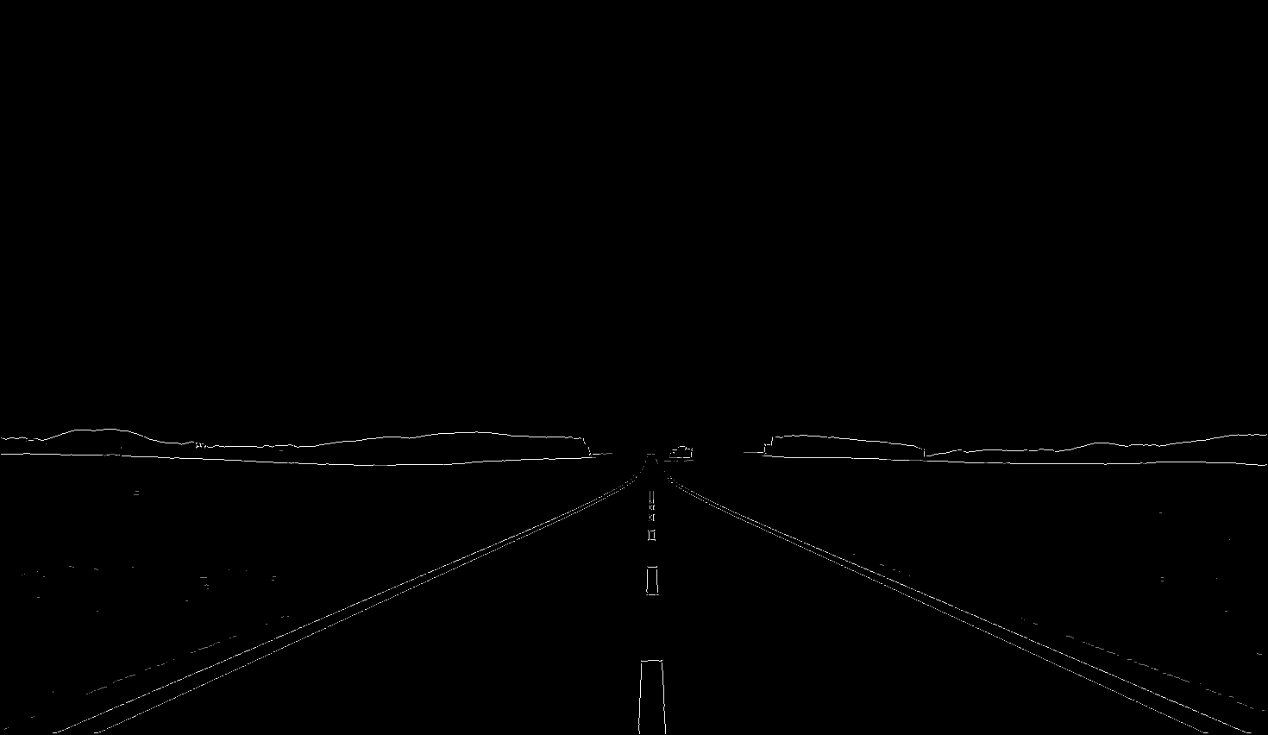
(sobel filter/magnitude)



(non-maximum suppression)

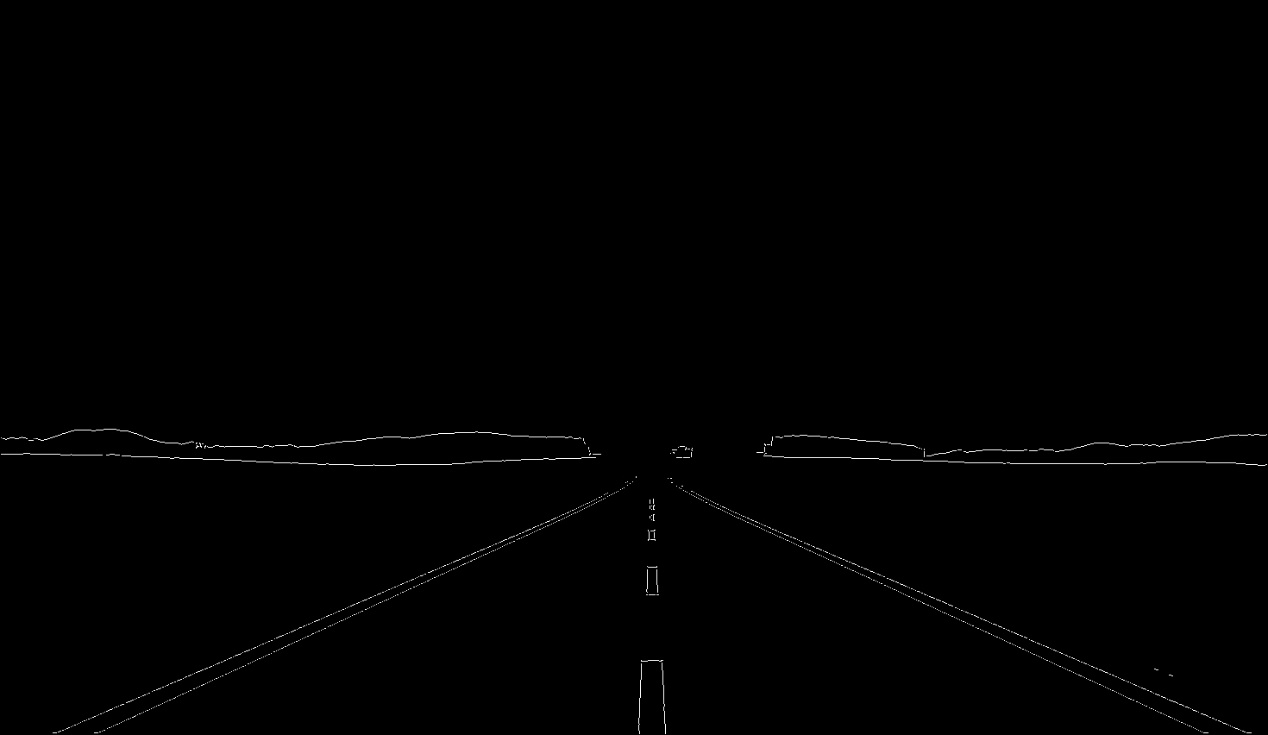


(double threshold)



(edge tracking by Hysteresis)

img1\_sobel.jpg



我預期的結果與最後的輸出相同，只有右下角的部分有一點雜訊。

其中在sobel filter的圖片中可以明顯地看到edge，可以看出最後的輸出的雛型，在non-maximum suppression中有明顯的將sobel filter的圖片細化edge，在double threshold中有消除了一些雜訊，到edge tracking by Hysteresis中有成功將連到strong edge的weak edge接起來，沒有連到strong edge的weak edge當雜訊消除。

img2.jpg

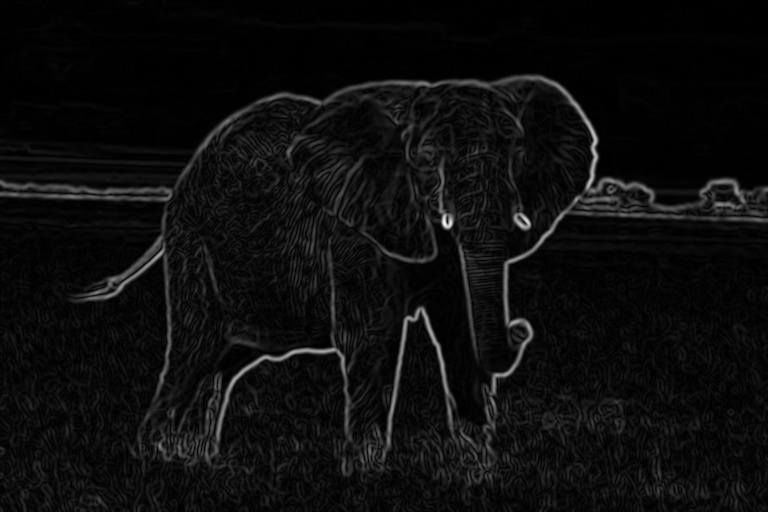
(gray image)



(gaussian\_filter)



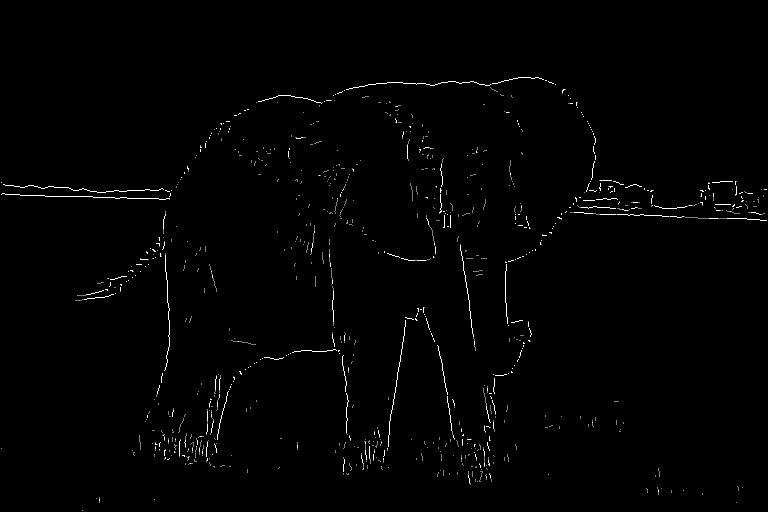
(sobel filter/magnitude)



(non-maximum suppression)

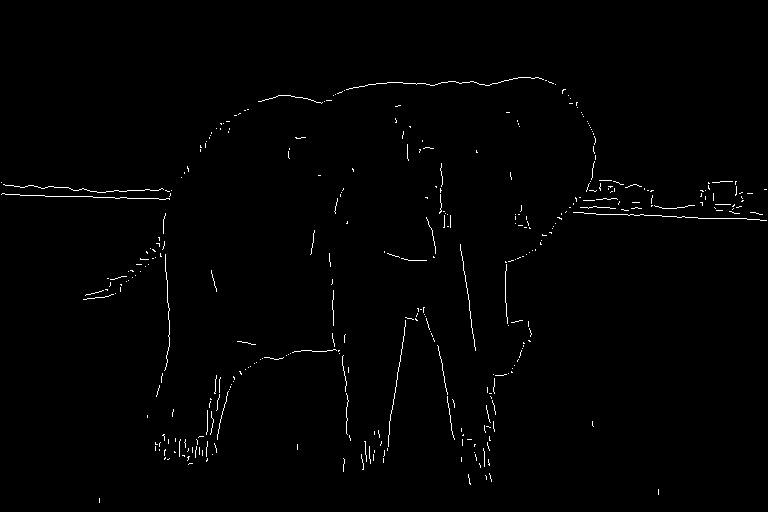


(double threshold)



(edge tracking by Hysteresis)

img2\_sobel.jpg



我預期的結果與最後的輸出大致上相同，仍然有一些草的雜訊沒有成功消除，大象的左半部分也有些edge沒有接起來。

其中在sobel filter的圖片中可以明顯地看到edge，可以看出最後的輸出的雛型，在non-maximum suppression中有明顯的將sobel filter的圖片細化edge，在double threshold中有消除了一些雜訊，到edge tracking by Hysteresis中有成功將大部分連到strong edge的weak edge接起來，沒有連到strong edge的weak edge當雜訊消除。

img3.jpg

(gray image)



(gaussian\_filter)



(sobel filter/magnitude)



(non-maximum suppression)



(double threshold)



(edge tracking by Hysteresis)

img3\_sobel.jpg



我預期的結果與最後的輸出大致上相同，左上的帽子沒有很好的連接起來，我覺得可能是因為在sobel filter的結果中帽子的左上角的gradient magnitude比較小一點。

其中在sobel filter的圖片中可以明顯地看到edge，可以看出最後的輸出的雛型，在non-maximum suppression中有明顯的將sobel filter的圖片細化edge，在double threshold中有消除了一些雜訊，到edge tracking by Hysteresis中有成功將連到strong edge的weak edge接起來，沒有連到strong edge的weak edge當雜訊消除。

最後的結果總結：

這次hw6的作業比較要注意的地方是找出kernel時，不要使用zero padding，因為這樣圖片的邊框要是在經過RGB image轉成gray image和gaussian filter後pixel的value不是黑色的話，就會讓zero padding的結果將邊框設成edge，這樣會導致圖片出來的結果會出現白色的邊框。我的解決方式是在找kernel時，讓超出image範圍的pixel的value設為image對應邊框的pixel。

然後大象的圖和Lenna的圖我覺得如果gaussian filter時，消除的noise和smooth的程度在多一點的話，我覺得最後出來的image的edge效果會更好。其中我覺得這兩張圖的gaussian filter再加上median filter後，最後的image的edge會比較好的呈現，只是相對應的會少一些圖片的feature。