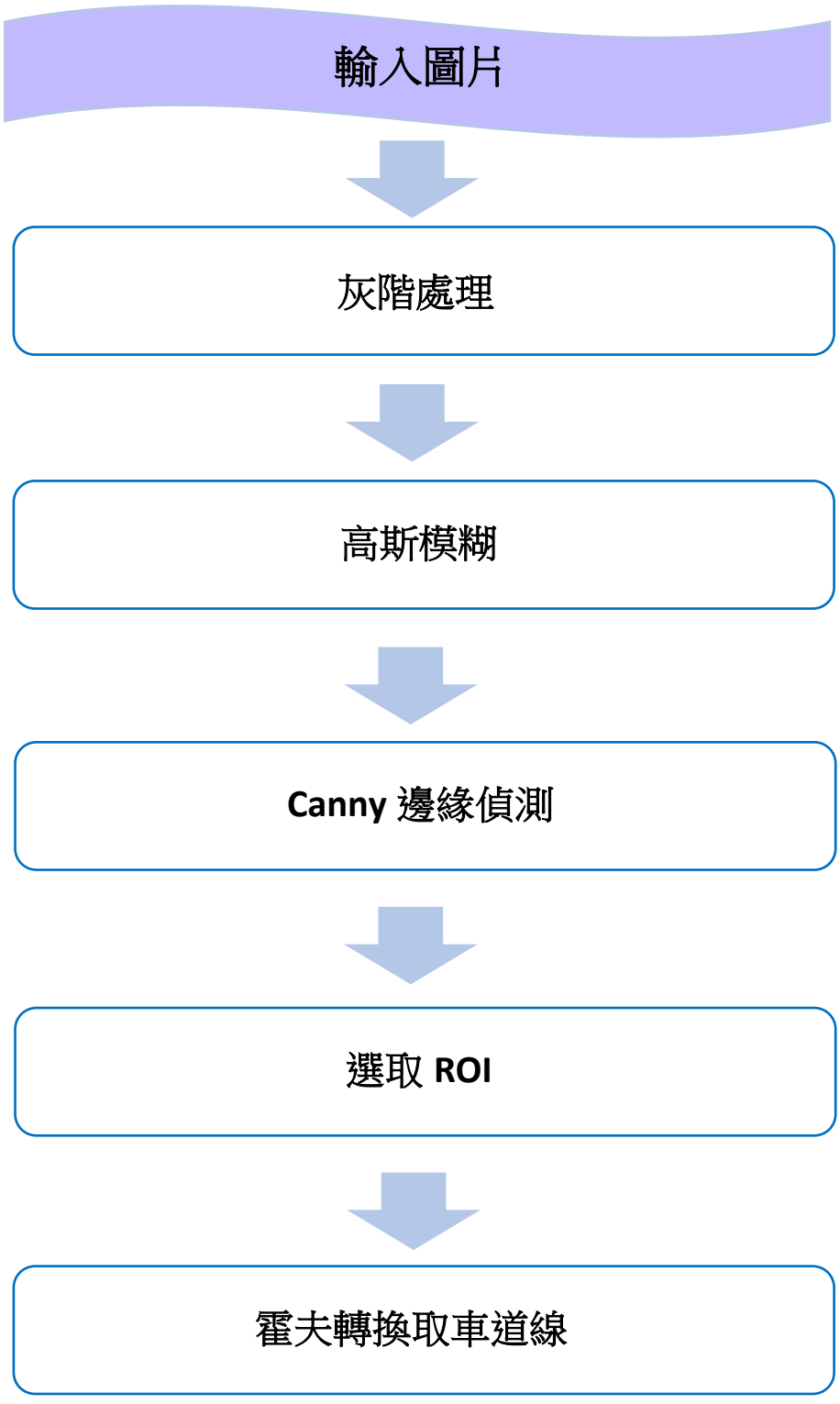
影像處理期末報告

S0961010 資管二 李誌軒

1. 作業一:偵測道路線 - 流程



2. 作業一程式碼、成果圖

a. 圖片一 (圖片來源:教授)

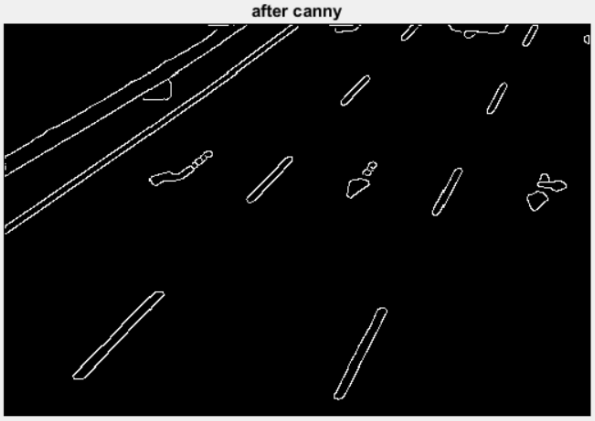
- 原圖

- 灰階 -> 高斯模糊 -> canny 邊緣偵測(圖一、二)

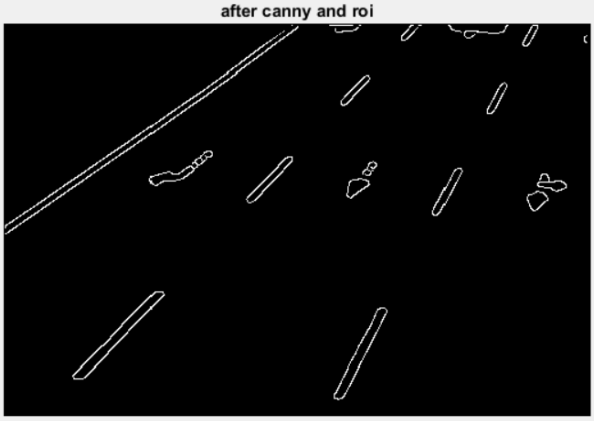
經過嘗試選出最適合的σ值，並擷取道路範圍的 ROI

```
m = imread('roadpic5.jpg');
m2 = rgb2gray(m);  %轉灰階
g = fspecial('gaussian',[5,5],5); %高斯模糊濾波
mg2 = imfilter(m2,g);             %高斯模糊
bw = edge(mg2,'canny',[0,0.4]);   %canny 邊緣偵測

      xi = [0 255 509 509 0];
      yi = [173 0 0 334 334]; %xi,yi: ROI區域
      roi = roipoly(m,xi,yi);
      roim = bw.*roi;
```



圖一: 做完模糊、邊緣偵測

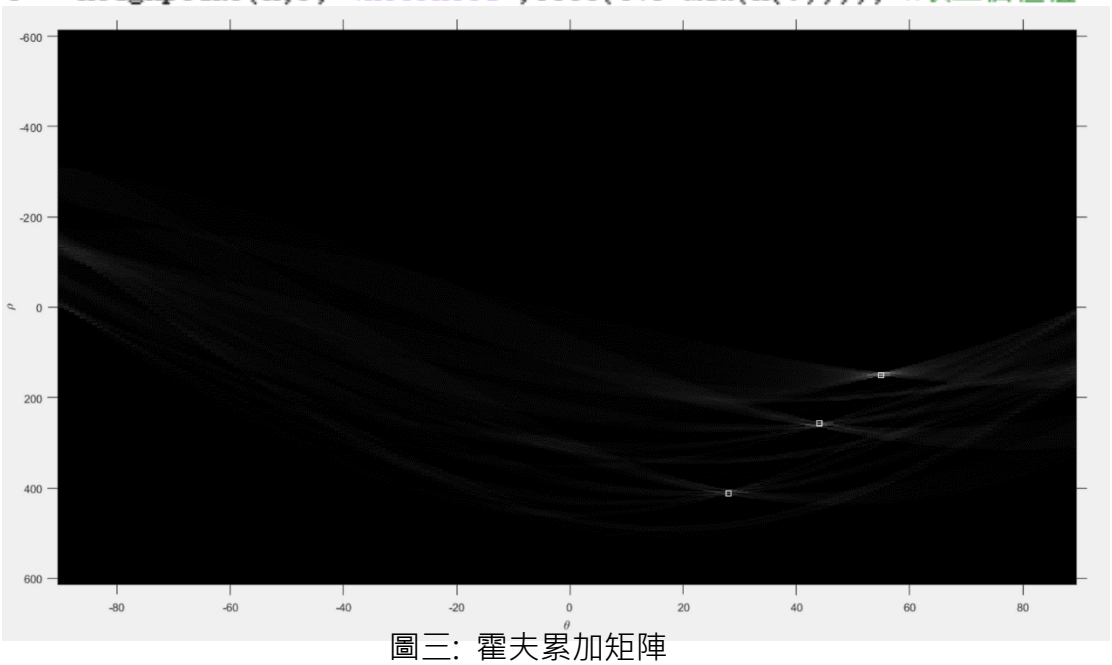


圖二: 做 ROI 區域

➢ 霍夫轉換取車道線

將邊緣偵測後，並擷取 ROI 的圖，做霍夫轉換取三個極值(閥值為 max 值 0.3 倍，並秀出累加矩陣(圖三)

```
[H,T,R] = hough(roim); %霍夫轉換
P = houghpeaks(H,3,'threshold',ceil(0.3*max(H(:)))); %取三個極值
```



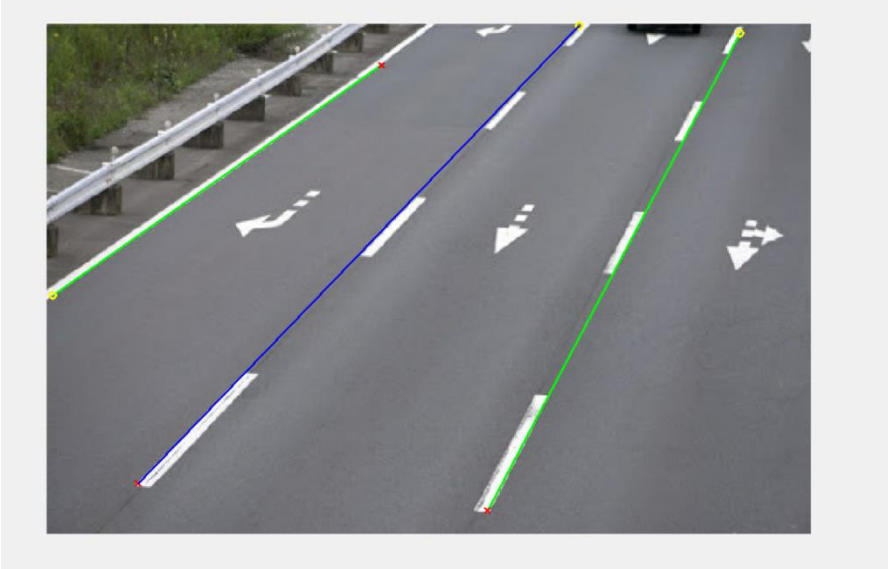
圖三: 霍夫累加矩陣

接著再依照 hough 函數得到的 T、R 以及三個極值在矩陣中的座標 P 獲得線段相關訊息，並設定 FillGap 以及 MinLength

```
lines = houghlines(roim,T,R,P,'FillGap',200,'MinLength',50);
```

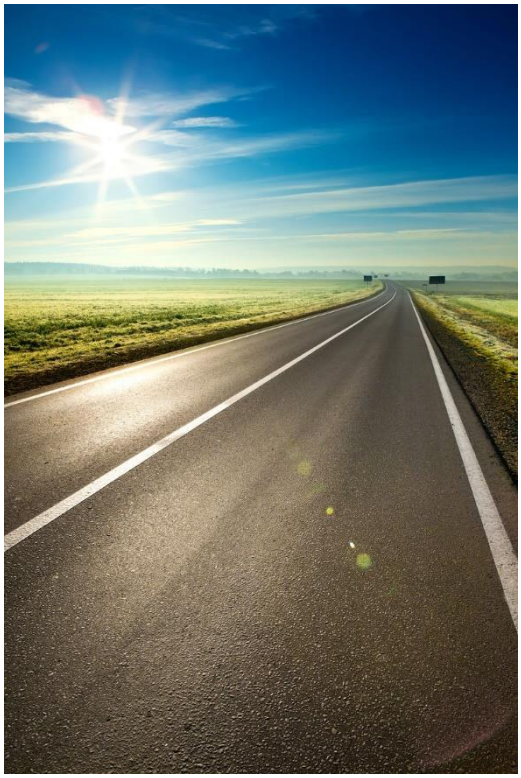
最後將 lines 的值，畫在原圖上及完成道路偵測

➢ 成果圖(藍線為最長)



b. 圖片二(圖片來源: https://sc.orsoon.com/material/600138.html)

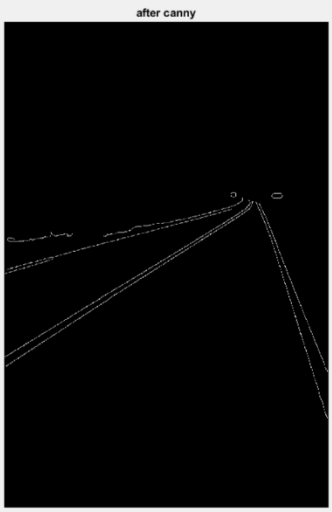
- 原圖



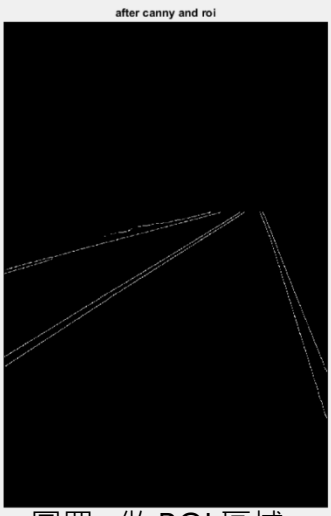
- 灰階 -> 高斯模糊 -> canny 邊緣偵測(圖三、四)

在圖片二中，選擇設定σ、雙閥值較不一樣，並擷取所需之 ROI

```
m2 = rgb2gray(m);
g = fspecial('gaussian',[3,3],3);
mg2 = imfilter(m2,g);
bw = edge(mg2,'canny',[0,0.7],3);
xi = [0 460 670 800 800 0];
yi = [600 470 470 834 1198 1198];
roi = roipoly(m,xi,yi);
roim = bw.*roi;
```



圖三: 做完模糊、邊緣偵測

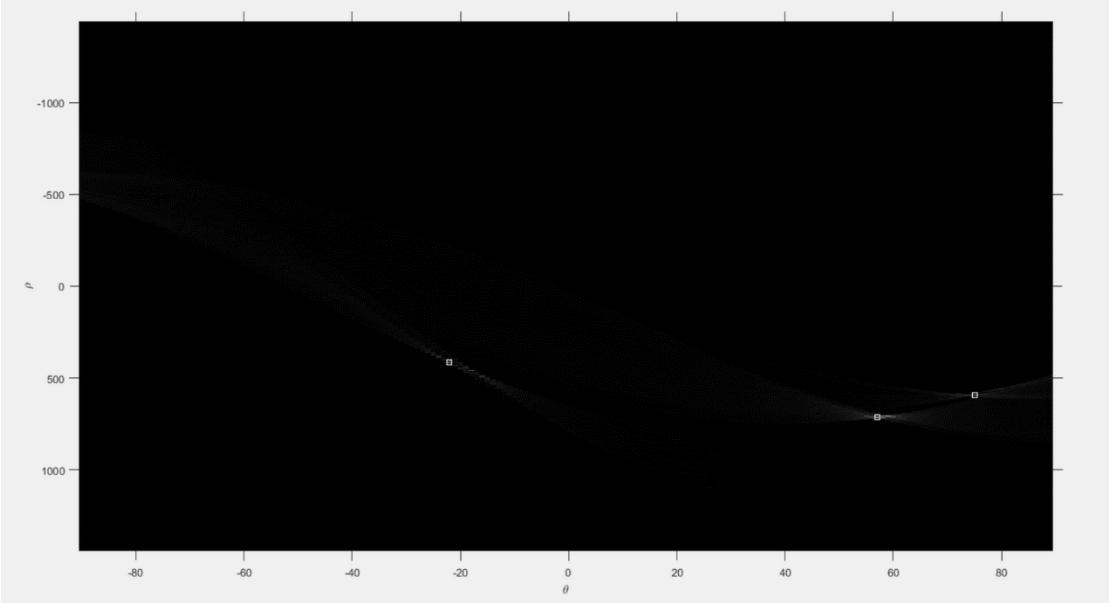


圖四: 做 ROI 區域

➢ 霍夫轉換取車道線

將邊緣偵測後，並擷取 ROI 的圖，做霍夫轉換取三個極值(閥值為 max 值 0.5 倍，並秀出累加矩陣(圖五)

```
P = houghpeaks(H,3,'threshold',ceil(0.5*max(H(:))));
```

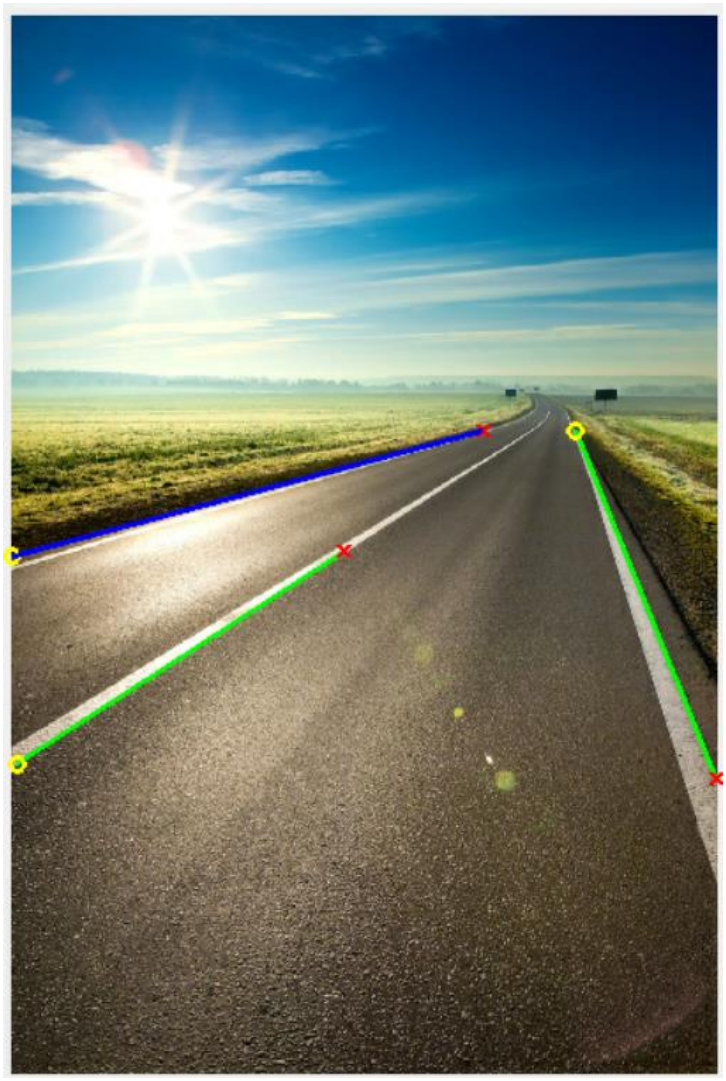


圖五: 霍夫累加矩陣

```
lines = houghlines(roim,T,R,P,'FillGap',100,'MinLength',50);
```

最後將 lines 的值，畫在原圖上及完成道路偵測，可本次效果較不如預期(中間的線不夠長)，認為是 lines 參數設定不如預期，

- 成果圖(藍線為最長)



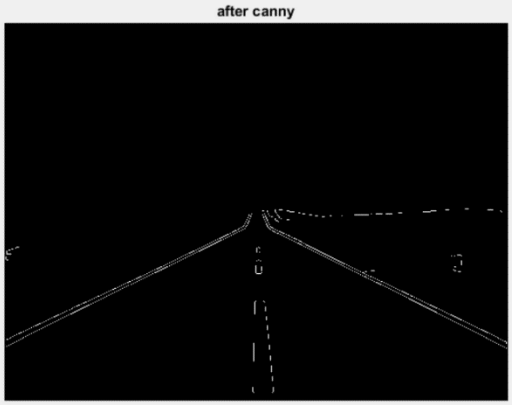
c. 圖片三(圖片來源: http://www.wall001.com/nature/road_wallpapers/html/image1.html)

- 原圖

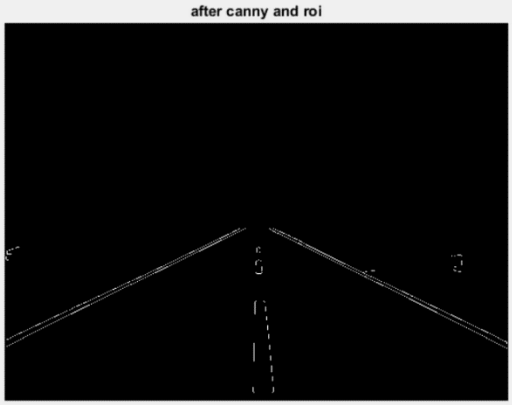
➢ 灰階 -> 高斯模糊 -> canny 邊緣偵測(圖六、七)

在圖片三中，路邊樹叢較多，因此選擇設定的模糊程度較大
並擷取所需之 ROI(下半部)

```
m2 = rgb2gray(m); %轉灰階
g = fspecial('gaussian',[5,5],11); %高斯模糊濾波
mg2 = imfilter(m2,g);              %高斯模糊
bw = edge(mg2,'canny',[0,0.7]);    %canny 邊緣偵測
            xi = [0 1024 1024 0];
            yi = [417 417 766 766];
            roi = roipoly(m,xi,yi);
            roim = bw.*roi;
```



圖六: 做完模糊、邊緣偵測

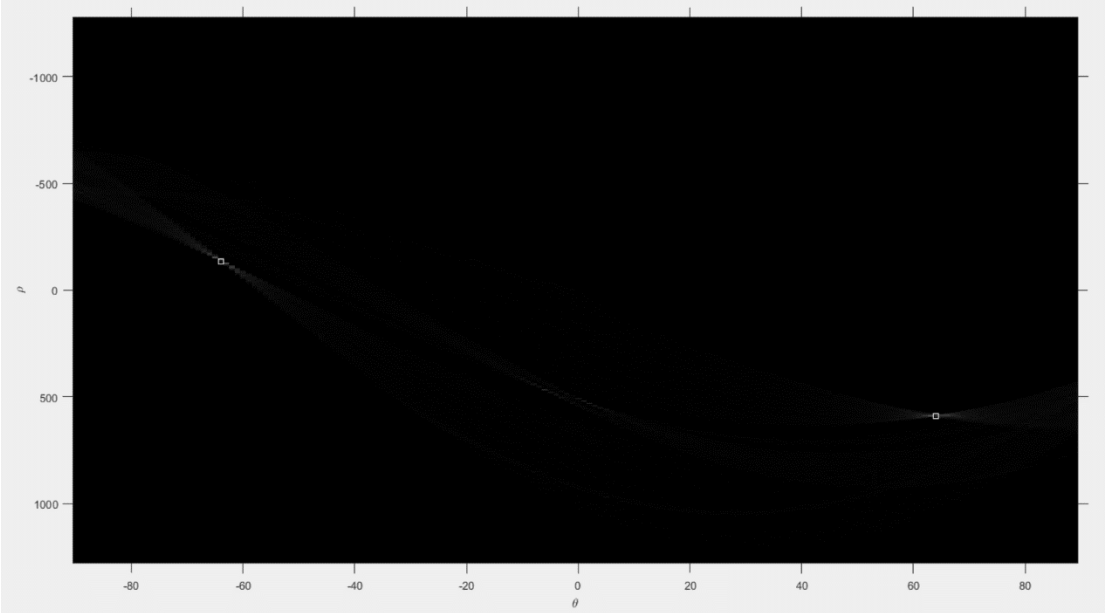


圖七: 做 ROI 區域

➢ 霍夫轉換取車道線

將邊緣偵測後，並擷取 ROI 的圖，做霍夫轉換取兩個極值(閥值為 max 值 0.5 倍，並秀出累加矩陣(圖八)

```
P = houghpeaks(H,2,'threshold',ceil(0.5*max(H(:))));
```

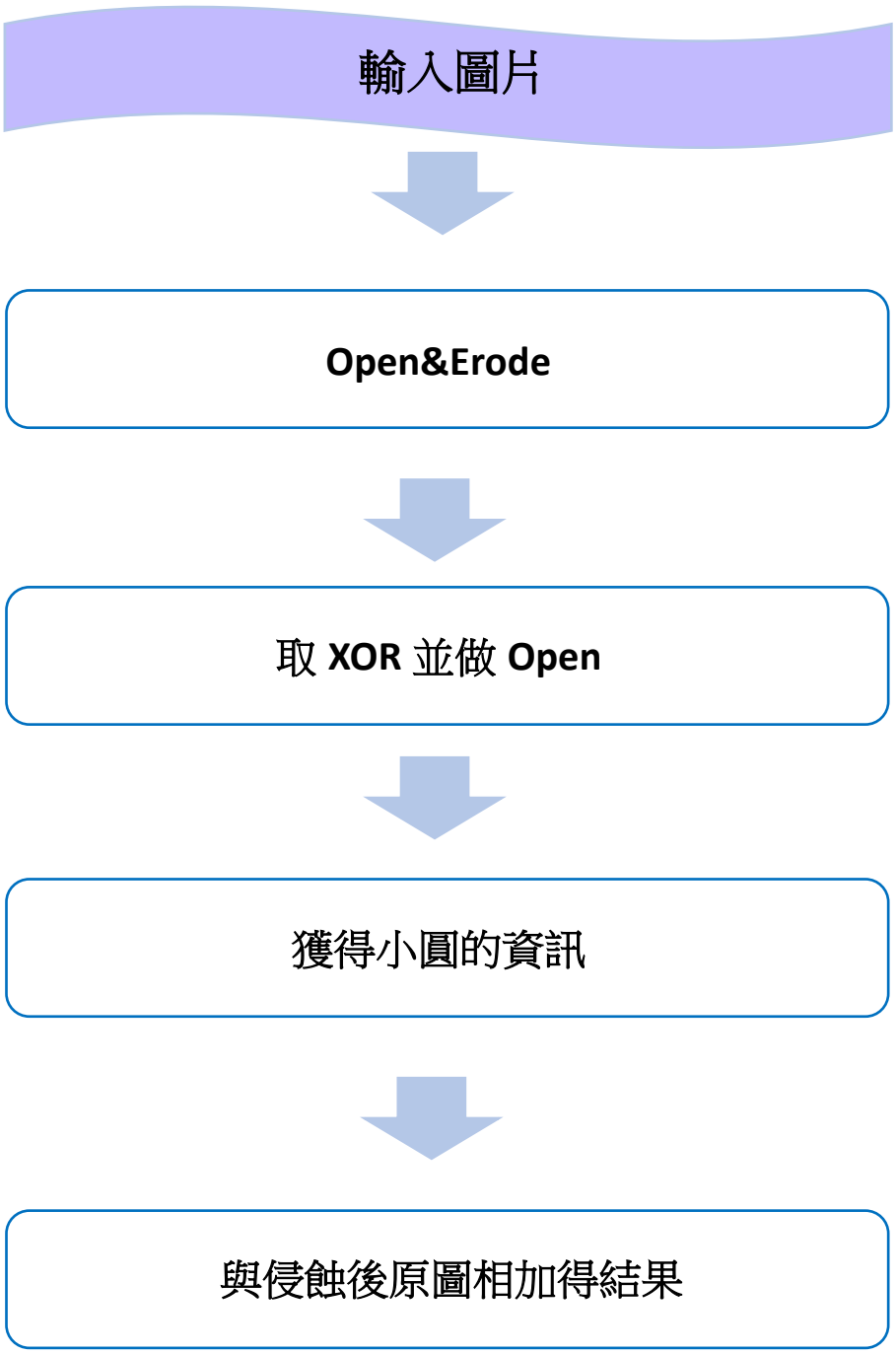



```
lines = houghlines(roim,T,R,P,'FillGap',50,'MinLength',50);
```

最後將 lines 的值，畫在原圖上及完成道路偵測，本次效果極佳!

- **成果圖(藍線為最長)**

3. 作業二:數量計算 - 流程

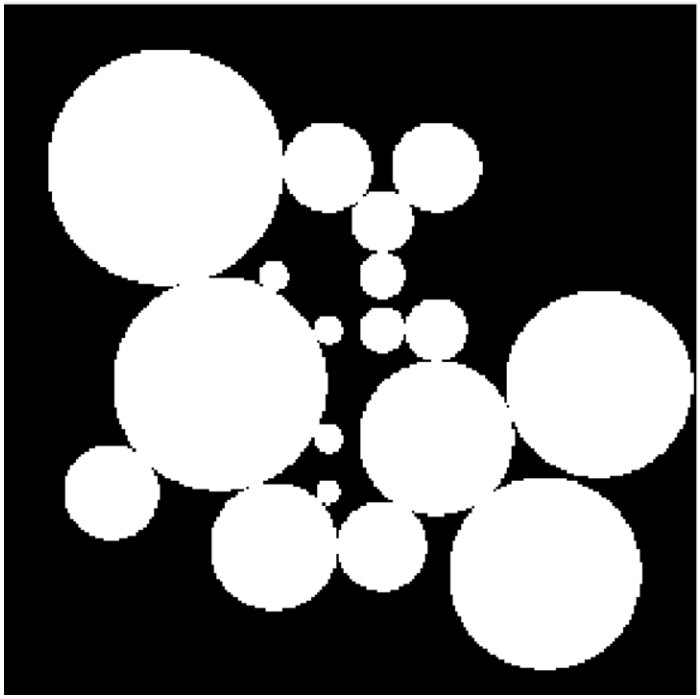


4. 作業二程式碼、成果圖

因經過多次嘗試，發現不論做什麼動作(侵蝕、開啟…)，其中的某幾個小圓都會被消除不見，因此我決定分別處理大圓以及小圓。

- **將圖片轉成 logical 以方便後面計算**

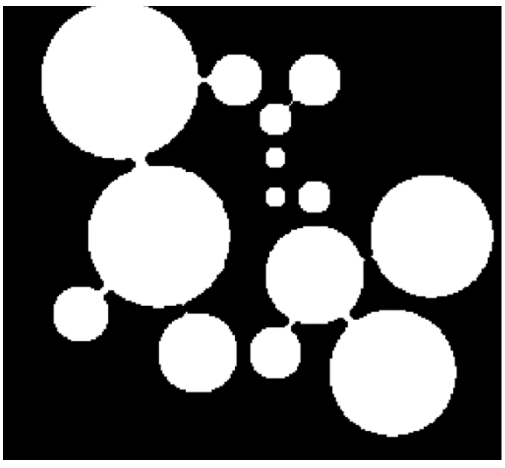
```
m = imread('circles2.png');
m = im2bw(m,0.5);
```



- **做第一次的開啟&侵蝕(圖八)**

此步驟主要是先對圖像做初步的切斷連結處以及侵蝕，以下 kernel 的大小則是多次嘗試得出的結果

```
mopen = imopen(m,strel('disk',6));
mopenerode = imerode(mopen,strel('disk',4));
```

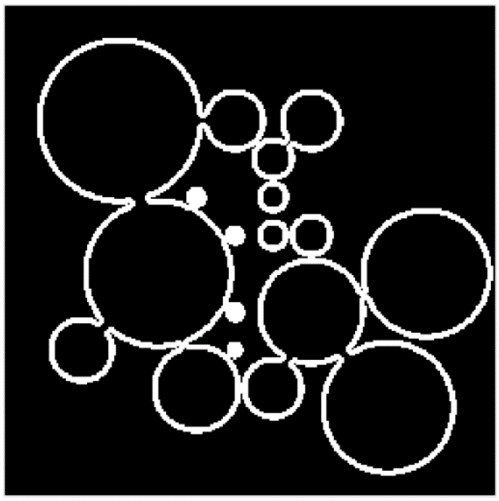


圖八：開啟、侵蝕後結果

➢ 做 XOR 並且侵蝕

接著將上圖八與原圖做 XOR 可得圖九

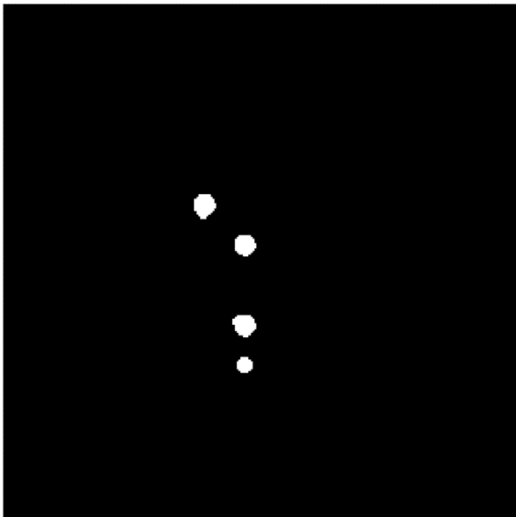
```
x = xor(mopenerode,m);
```



圖九

可看到有許多不必要的空心圓，因此再做一次開啟再確保幾個小圈存在的情況刪除空心圓(圖十)

```
f = imopen(x,strel('disk',4));
```

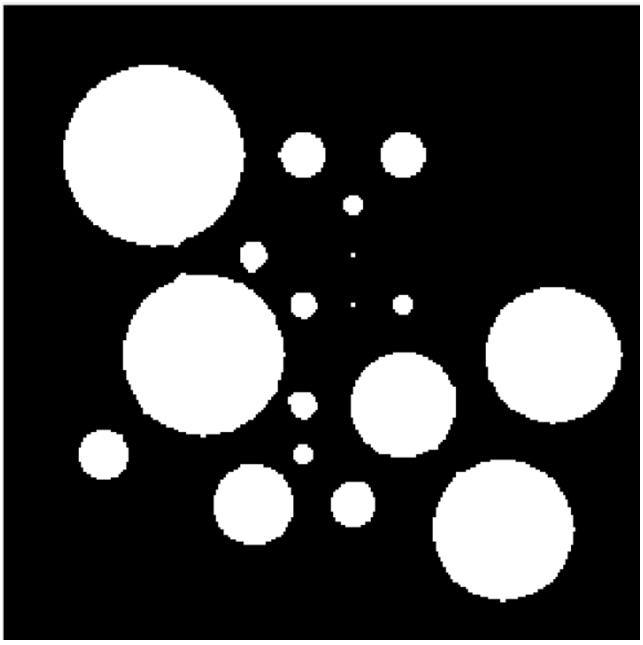


圖十

➢ 相加圖形

最後將原圖做比前面更強的侵蝕，此時會失去幾個小圓，但可以有效的刪除連結處，並且加上圖十的小圓，最後獲得成果圖(下頁圖十一)

```
final = imerode(m,strel('disk',8)) + f;
```



圖十一：成果圖

- 最後使用 **label** 去做計算圓數量的動作

```
final = bwlabel(final);
count = max(final(:));
display(count);
```

```
count =

    18
```