

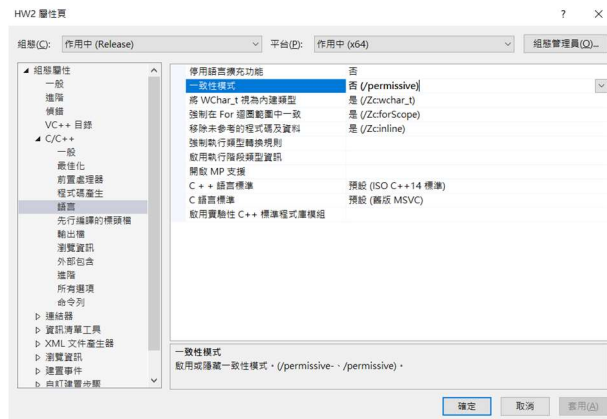
# 高等電腦視覺 作業三

姓名: 陳祐毅 學號:109618028

## 使用 C 語言

編譯環境 : Visual Studio 2019 (16.11.5)

如果遇到 `const char to char` 相關的 error，請在  
屬性 > C/C++ > 語言 > 一致性模式 改成否

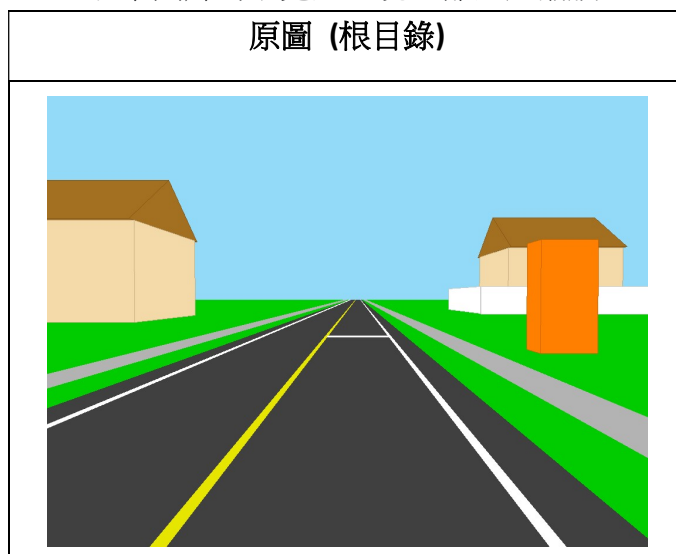


會自動建立 `./img_by_C/` 資料夾，存放所有照片

## 引用函式庫

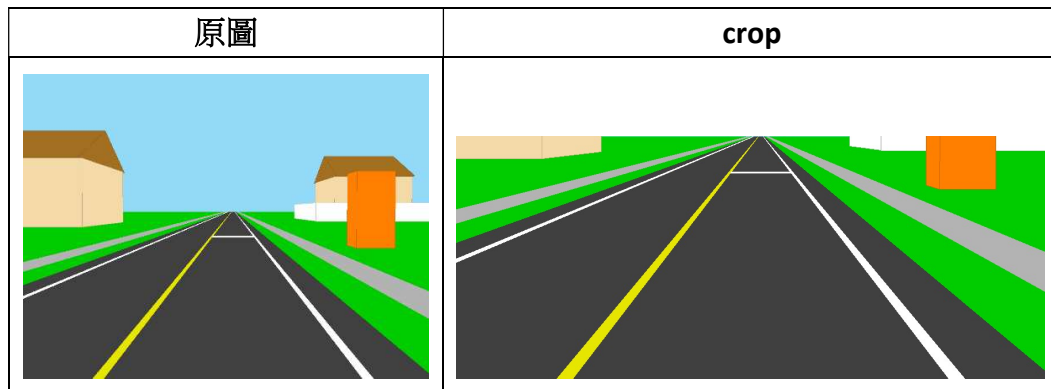
```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include <io.h> //創 img資料夾用
4 #include <direct.h> //創 img資料夾用
5 #include<math.h>
```

以下圖片為了完整呈現，都經過縮放



## 步驟一 crop

使用作業 pdf 給的 uHorizon 公式擷取地面部分的圖片



## 步驟二 轉換成 IPM img

這部分使用作業 pdf 的公式，

先從圖片座標轉世界座標尋找轉換後圖片的 size：

```
480     for (int u = 0; u < m; u++) {
481         for (int v = 0; v < n; v++) {
482             calculate1 = 1 / tan(th0 - alpha + (2 * alpha * u) / (m - 1));
483             calculate1 *= dz * sin(y0 - alpha + (2 * alpha * v) / (n - 1));
484
485             x = calculate1 + dx;
486             y = calculate1 + dy;
487
488             *x1 = (*x1 > x) ? ((int)x) : (int)(*x1); //左上座標 x
489             *y1 = (*y1 > y) ? ((int)y) : (int)(*y1); //左上座標 y
490             *x2 = (*x2 < x) ? ((int)x) : (int)(*x2); //右下座標 x
491             *y2 = (*y2 < y) ? ((int)y) : (int)(*y2); //右下座標 y
492         }
493     }
```

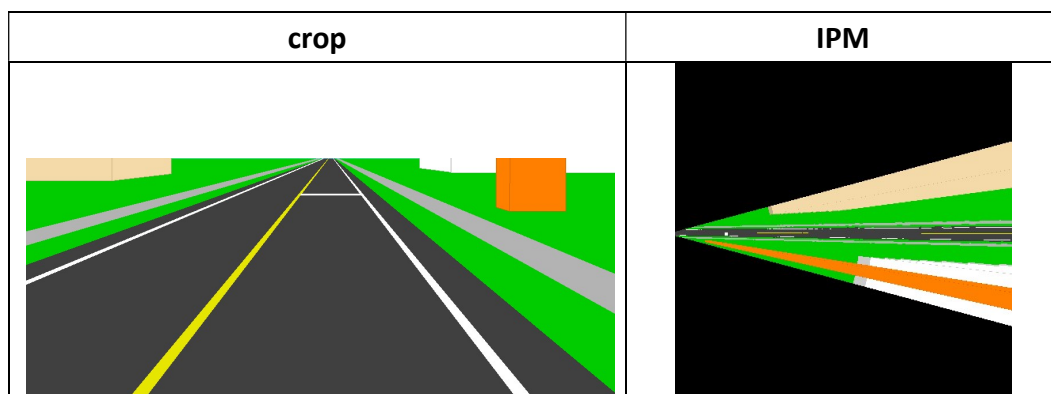
找到的兩個點(x1,y1)、(x2,y2) 中(x1, y1)為負整數，所以我們要將整個圖片往右移，由於整個圖片大小太大(4k\*4k)所以我將長寬都除 10：

```
536     Mat bird_img = src;
537     int bird_height = (x2 - x1) / 10;
538     int bird_width = (y2 - y1) / 10;
```

使用上面的長寬配置好記憶體後，使用世界座標轉圖片座標的公式，尋找要取得的索引值：

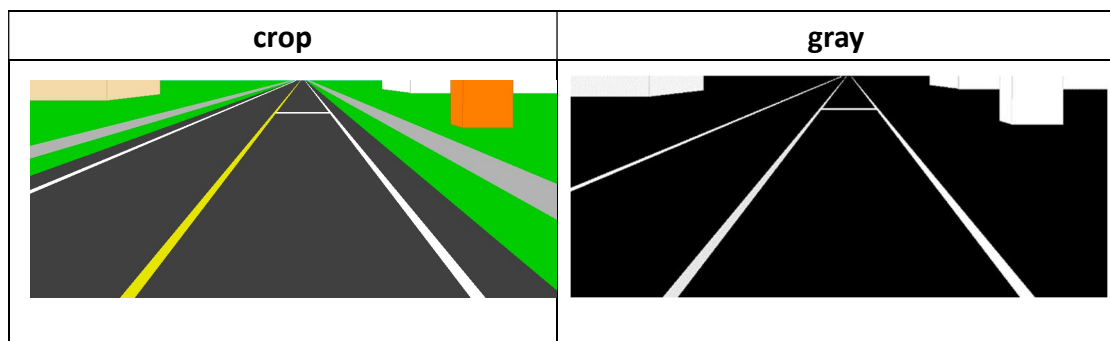
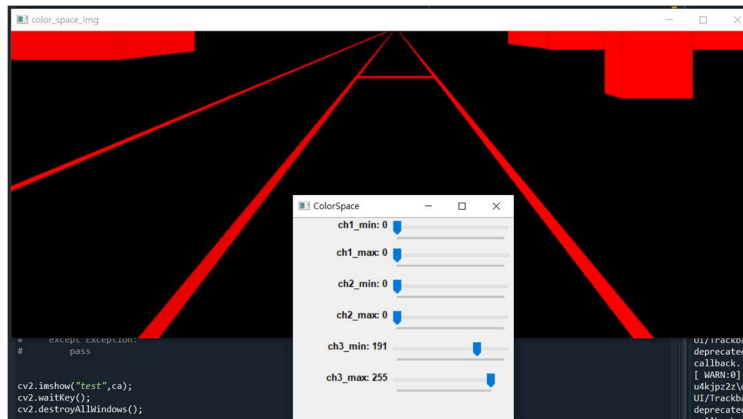
```
555 //bird轉換
556 int u, v;
557 double scale = 10;
558 double X, Y;
559 for (int x = 0; x < bird_img.rows; x++) {
560     for (int y = 0; y < bird_img.cols; y++) {
561         //找 原圖的轉換座標
562         X = (x * 10.0 + x1) / scale; //scale讓取的範圍在小一點
563         Y = y * 10.0 / scale;
564         word_to_img(src, &u, &v, X, Y);
565
566         if (u >= 0 && v >= 0) {
567             if (u < src.rows && v < src.cols) {
568                 for (int i = 0; i < 3; i++) {
569                     bird_img.image[i][x][y] = src.image[i][u][v];
570                 }
571             }
572         }
573     }
574 }
```

在上面程式中，我將世界座標的座標點除上一個 `scale` 參數，目的是為了讓我們感興趣的區域可以放大，可以理解成一個飛行員從 500 公尺俯瞰地面變成從 50 公尺俯瞰地面，視野變小，但目標物變大了：



### 步驟三 灰階

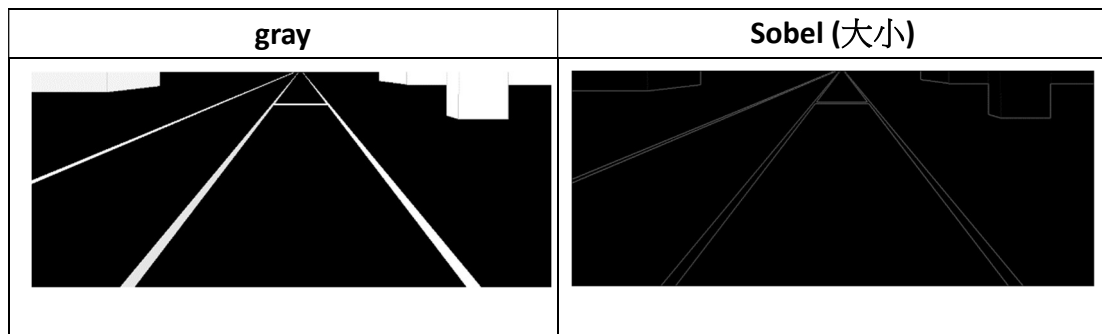
照理說，灰階已經用到爛了沒什麼好介紹的，但我這裡不是用普通的灰階，我寫程式到後面步驟時，發現只用 **sobel** 的 **direction** 真的不好取出道路線，一氣之下我用 **python**(最熟)找色彩空間，發現將 **Red channel** 的 **191~255** 的索引值都設為 **255** 最棒：



### 步驟四 sobel

我使用 **sobel** 做邊緣偵測，然後計算每個像素點的大小、方向

```
356 //filter X
357 double sobel_x[3][3] = { {-1, 0, 1},
358                           {-2, 0, 2},
359                           {-1, 0, 1} };
360
361 // filter Y
362 double sobel_y[3][3] = { { 1, 2, 1},
363                           { 0, 0, 0},
364                           {-1,-2,-1} };
365
```



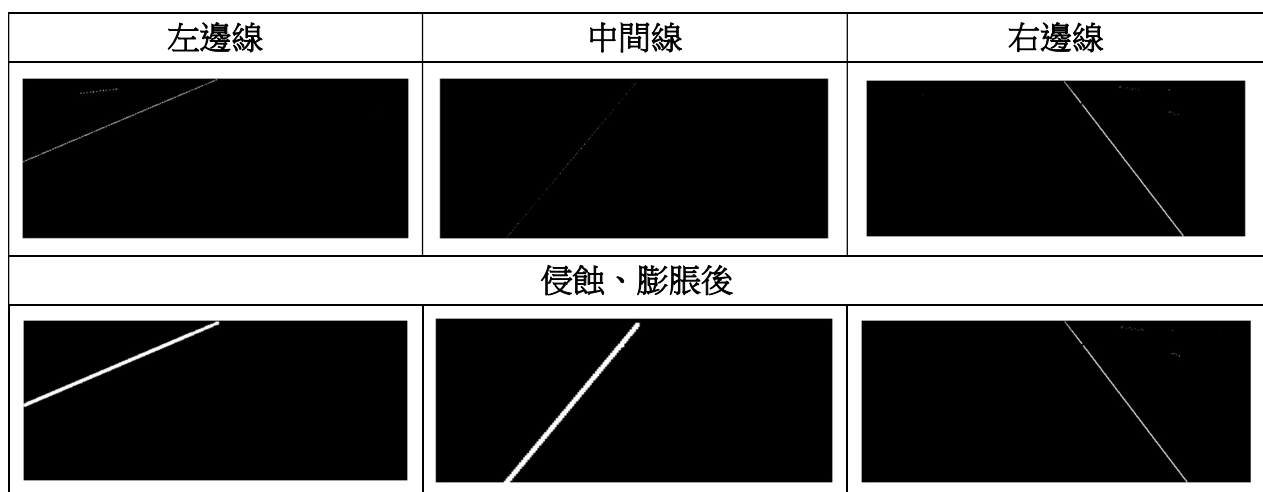
方向的計算，我使用了 atan2，他與 atan 不同的是，atan2 會分象限，角度橫跨 360 度：

```

399 //設置 (方向)
400 if (sum_x == 0 && sum_y == 0) {
401     direction = -1;
402     sobel_direct.direction[r][c] = (short)(direction); //沒有值得就設定 -1
403 }
404 else {
405     direction = atan2(sum_y, sum_x); //與正向x軸的角度
406     direction = direction * 180.0 / 3.14159; //改成角度
407     direction = (direction >= 0) ? direction : (360 + direction); //使角度位於0<= angle <360度
408     sobel_direct.direction[r][c] = (short)(direction);
409 }

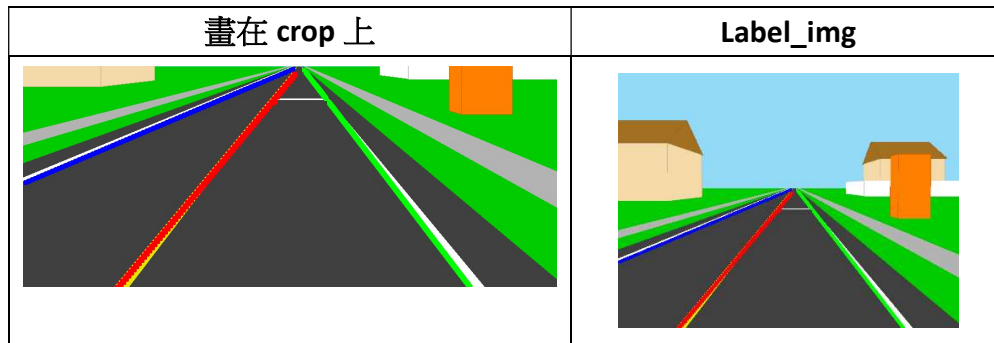
```

接著就是找三條線的角度啦，找到的線旁邊可能會有一些雜訊點，所以我使用侵蝕、膨脹將他們消除：



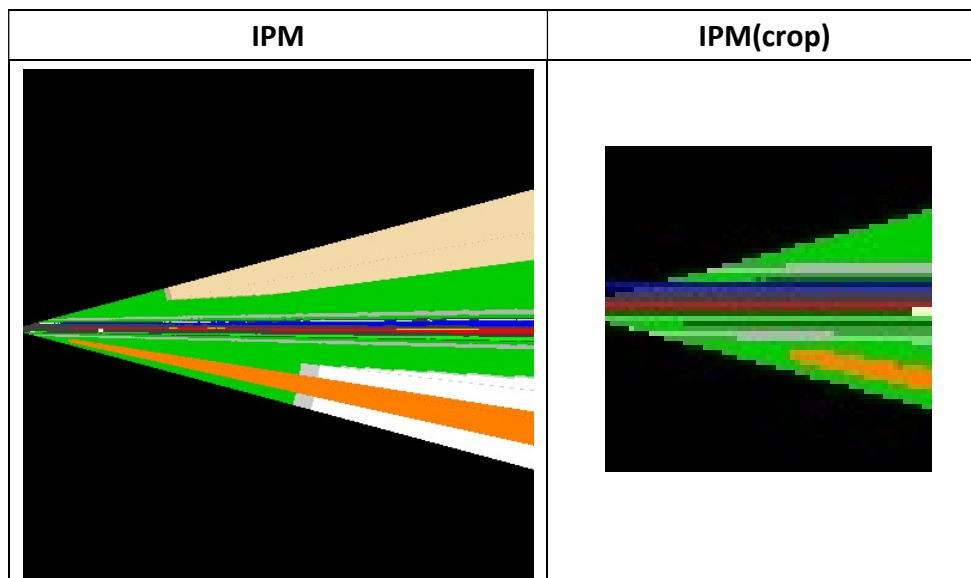
## 步驟五 畫線

我將上面得到的三張圖當作 **crop** 圖 的判斷式，  
如果為真(白色)>>畫上指定顏色，然後在把 **crop** 圖 縫在原圖上：



## 步驟六 轉 IPM 並 crop 需要的部分

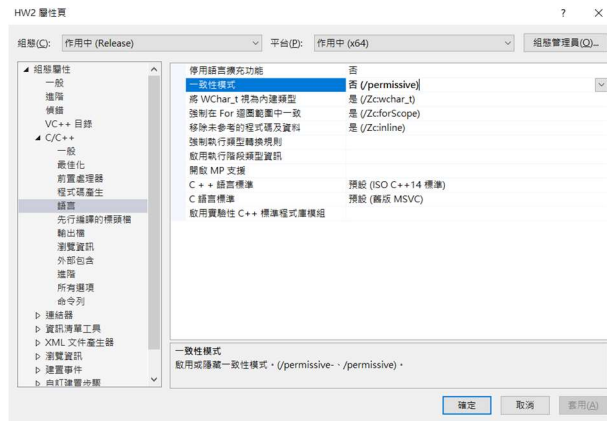
我將上述的 **Label\_img** 轉換成鳥瞰圖後，發現果然還是有些部分有缺洞，由於  
線的 label 像素質是我射的，所以我分別找到三條線的像素點並將其延伸，使關  
漏的地方補滿，然後 **crop** 出需要的部分：



## 使用 OpenCV

編譯環境：Visual Studio 2019 (16.11.5)、opencv (4.4.0)

如果遇到 `const char to char` 相關的 error，請在  
屬性 > C/C++ > 語言 > 一致性模式 改成否

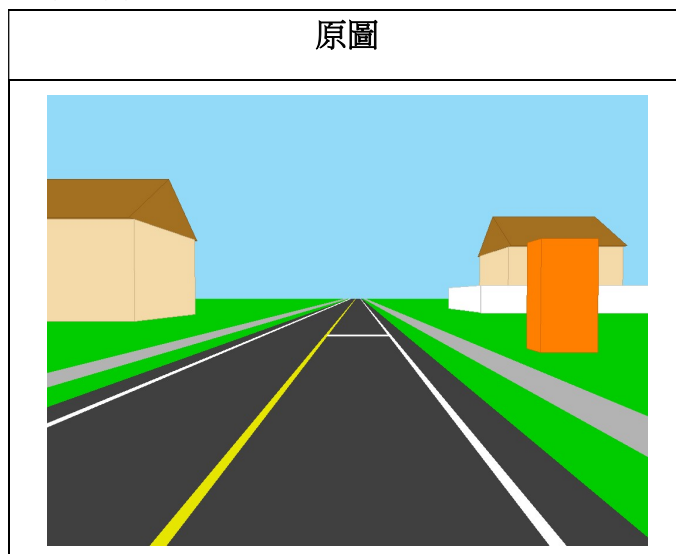


會自動建立 `./Image_cv2/` 資料夾，存放所有照片

## 引用函式庫

```
1  #include<opencv2/opencv.hpp>
2  #include<opencv2/highgui.hpp>
3  #include<iostream>
4  #include<string>
5  #include <io.h> //創 img資料夾用
6  #include <direct.h> //創 img資料夾用
```

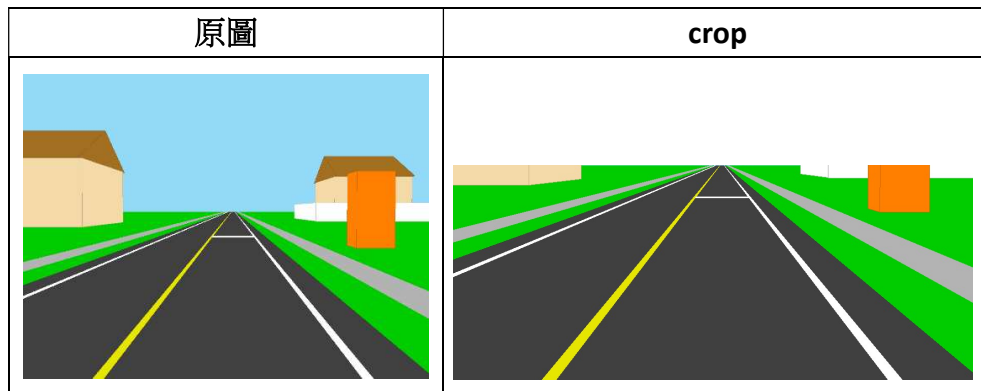
以下圖片為了完整呈現，都經過縮放



## 步驟一 crop

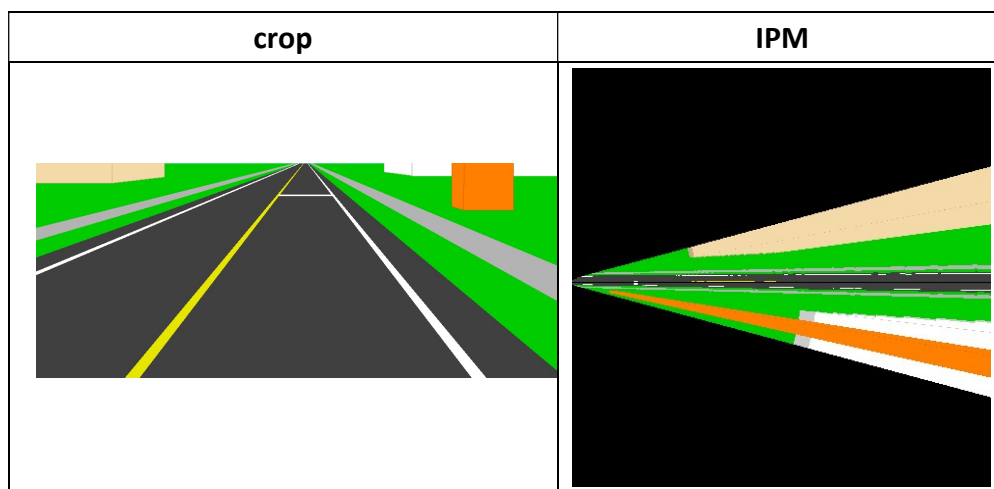
使用作業 pdf 給的 uHorizon 公式擷取地面部分的圖片

```
35 //crop
36 double alpha = 15 * 3.14159 / 180;
37 double calc = ((double)inputImg.rows - 1) * (-0.025 + alpha) / (2 * alpha);
38 int uHorizon = (int)calc;
39 cv::Mat roi(inputImg, cv::Rect(0, uHorizon, inputImg.cols, inputImg.rows-uHorizon));
40 cv::Mat cropImg;
41 roi.copyTo(cropImg);
42 cv::imwrite("./Image_cv2/1_crop.bmp", cropImg);
```



## 步驟二 轉換成 IPM img

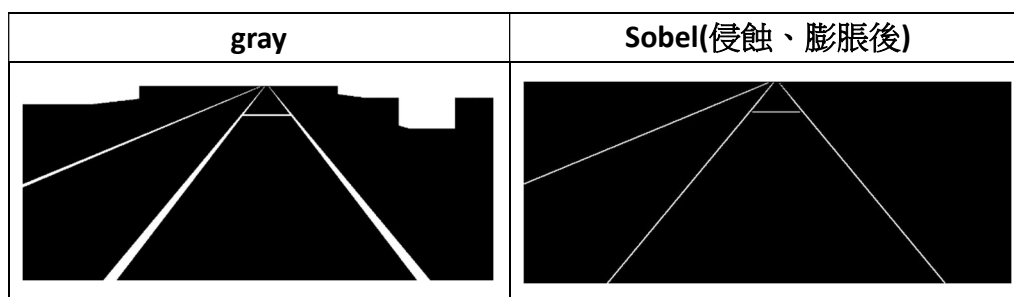
這部分跟 C 語言是一樣的，就不再贅述，我本來還很開心只要複製貼上就好，但實作突然才想到，語法不一樣...，不過幸好我 C 語言採用模仿 opencv 的方式，例如取 rows、cols 值就不用再從設，也不用到大改。多虧在 C 語言的跌跌撞撞，在 opencv 這部分的程式碼有變簡潔一點：





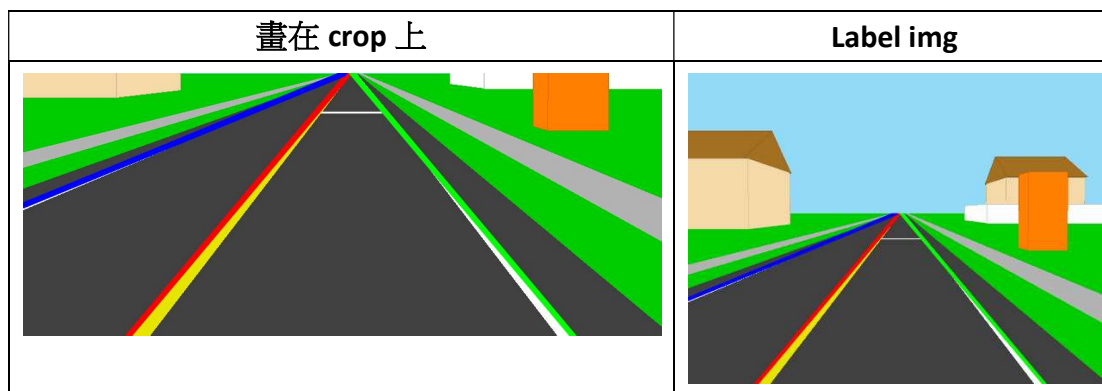
### 步驟三 灰階、sobel

這裡的灰階一樣是取 red channel 191~255 的值，然後採用 opencv 的 Sobel 函式，並對 Sobel 過後的圖做侵蝕膨脹去除不必要的部分：



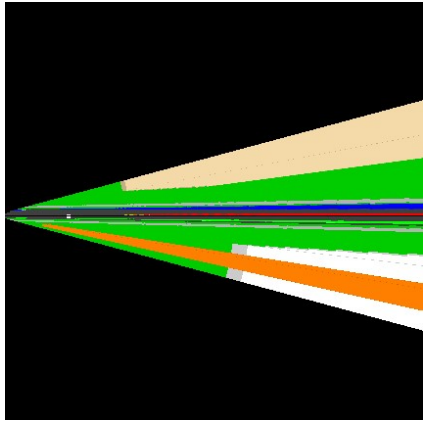
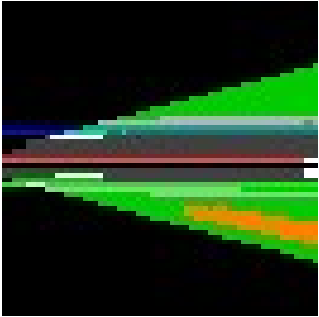
### 步驟四 HoughLines

由於老師說可以用 opencv 的直線偵測，所以我使用了 HoughLines() 找三條線，然後使用 line() 函式將線依需求顏色畫在 crop 圖上，最後縫合到原圖：





## 步驟五 轉 IPM

這部分的方法也與 C 語言一樣，就不再贅述僅顯現成果

IPM(有缺漏的部分)	IPM(crop)
	

## OpenCV bonus

原圖(左)	原圖(右)
	

## 步驟一 將左圖放進要輸出的圖片



## 步驟二 將右圖進行仿射轉換

```
41
42 //原始座標點
43 origP[0] = cv::Point2f(0, 0);
44 origP[1] = cv::Point2f(0, src_img_R.rows);
45 origP[2] = cv::Point2f(src_img_R.cols, 0);
46 origP[3] = cv::Point2f(src_img_R.cols, src_img_R.rows);
47
48 //轉換後座標點
49 cvtP[0] = cv::Point2f(250, 130);
50 cvtP[1] = cv::Point2f(310, 470);
51 cvtP[2] = cv::Point2f(770, 0);
52 cvtP[3] = cv::Point2f(900, 475);

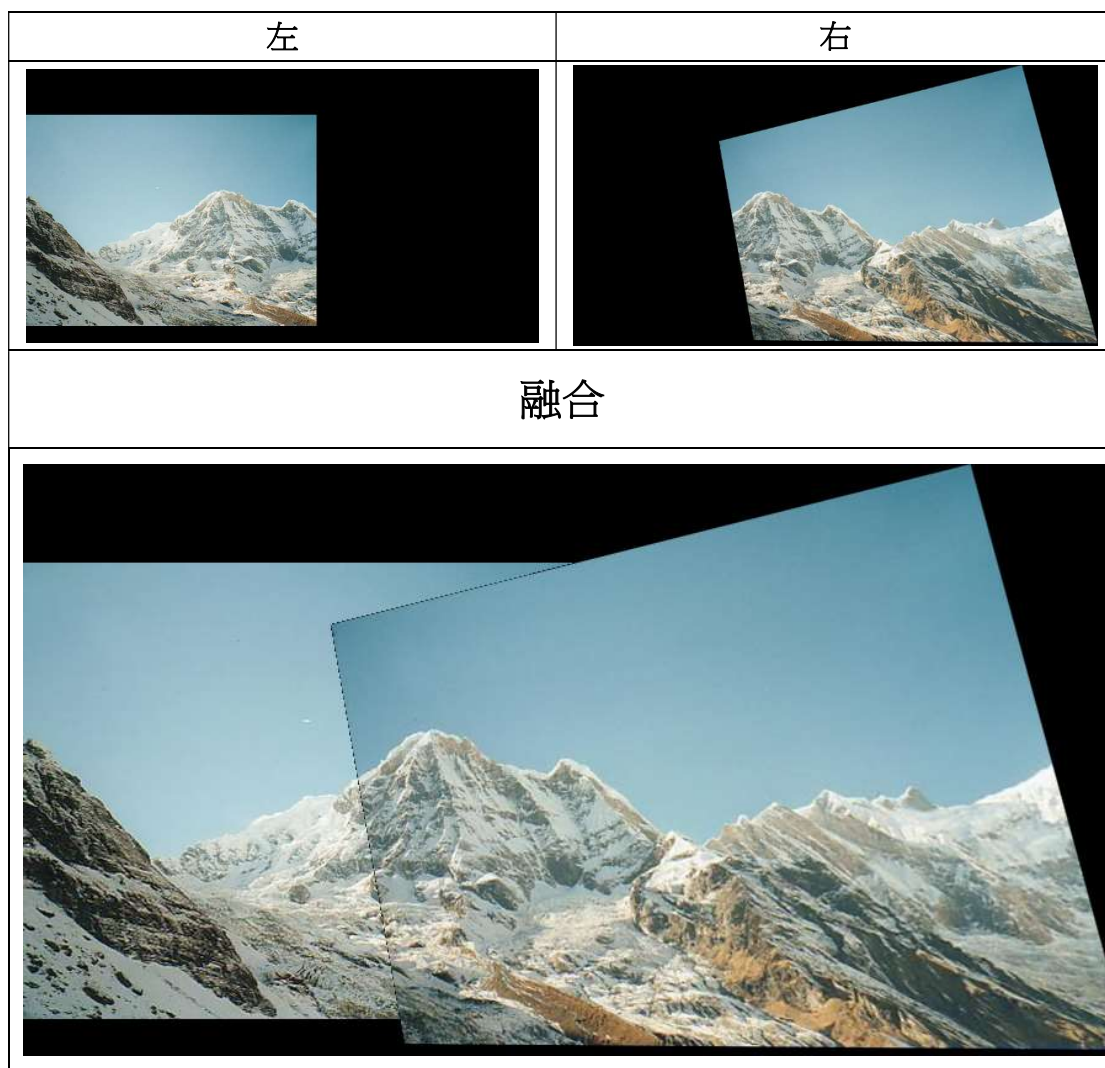
```

```
60 cv::Mat Matri = cv::getPerspectiveTransform(origP, cvtP);
61 warpPerspective(new_R, new_R, Matri, stitchImg.size()); //仿射轉換
62 cv::imwrite("2_onlyRightImg.bmp", new_R);

```



### 步驟三 融合上面兩張圖



這次寫程式花很多時間在修改參數值，尤其在尋找線段的部分，努力地尋覓線條在哪裡，有時找到了需要的線條旁邊又有許多雜訊干擾，找到心力交瘁 XD  
然後由於用過 python 的 opencv，我就一直覺得 C++ 的 opencv 好難用，其實主要還是在於我對 C++ 的熟悉度不夠吧，不過學了兩種語言的 opencv 後我漸漸的學會先用 python 做看看方向再用 C++ 做一次。不過我還是想抱怨一下，使用 opencv 的 houghline 明明很好找到需要的線條，怎麼 C++ 這麼難找...。最後，果然需要作業的練習，我對 C 語言的熟悉度也上升了不少，應該吧 XD。