自動駕駛實務

車道辨識作業

一、實做過程中遇到的問題

起初，solidWhiteRight 和 solidsolidYellowLeft 是用下方程式碼做出來的

```
def Color_select(img,rgb_threshold):
    color_thresholds = (img[:,:,0] < rgb_threshold[0])    \
                | (img[:,:,1] < rgb_threshold[1])  \
                | (img[:,:,2] < rgb_threshold[2])
    color_select[color_thresholds] = [0,0,0]
    return color_select,color_thresholds
def Mask(img,left_bottom ,right_bottom ,left_top ,right_top ):
    region_select = np.copy(img)
    fit_left = np.polyfit((left_bottom[0], left_top[0]), (left_bottom[1], left_top[1]), 1)
    fit_right = np.polyfit((right_bottom[0], right_top[0]), (right_bottom[1], right_top[1]), 1)
    fit_top = np.polyfit((left_top[0], right_top[0]), (left_top[1], right_top[1]), 1)
    fit_bottom = np.polyfit((left_bottom[0], right_bottom[0]), (left_bottom[1], right_bottom[1]), 1)

# Find the region inside the lines
    XX, YY = np.meshgrid(np.arange(0, xsize), np.arange(0, ysize))
    region_thresholds = (YY > (XX*fit_left[0] + fit_left[1])) & \
                        (YY > (XX*fit_right[0] + fit_right[1])) & \
                        (YY < (XX*fit_bottom[0] + fit_bottom[1]))& \
                        (YY > (XX*fit_top[0] + fit_top[1]))

# Color pixels red which are inside the region of interest
    region_select[region_thresholds] = [255, 0, 0]

    return region_select, region_thresholds
```

```
#------------------Color and Region Selection-------------------#
##parameter for solidYellowLeft ##
rgb_threshold = [190,170,80]
left_bottom = [130, 540]
right_bottom = [870, 540]
left_top = [450, 320]
right_top = [510, 320]
#----------------------------------------------------------------#
color_select,color_thresholds=Color_select(image6,rgb_threshold)
region_select,region_thresholds=Mask(image6,left_bottom,right_bottom,left_top,right_top)
line_image[~color_thresholds & region_thresholds] = [255,0,0]

#------------------Canny edges and Hough lines-------------------#
gray = grayscale(image6)
blur_gray = gaussian_blur(image6, 5)
edges=canny(blur_gray, 50, 200)

###parameter for solidsolidYellowLeft ###
vertices = np.array([[(100, 540),(900, 540), (450, 320), (520, 320)]], dtype=np.int32) # Can't be too fitness!!!!
masked_edges=region_of_interest(edges, vertices)
lines=hough_lines(masked_edges, 1, np.pi/180, 20, 60, 50)
process_image6=weighted_img(lines, image6, α=0.8, β=1., γ=0.)
```

```
#-------------------------Overlapping-------------------------#
row,col,depth=image6.shape
image6_output=np.zeros([row,col,depth])
for i in range(0,row):
    for j in range(0,col):
        if line_image[i,j,0]==255 or process_image6[i,j,0]==255:
            image6_output[i,j,0]=255
image6_output=image6_output.astype('uint8')
image6_output=weighted_img(image6_output, image6, α=0.8, β=1., γ=0.)
```

上述方法是先將老師所提供的程式碼做包裝(Color_select 和 Mask)，並且 Mask 做了梯形的遮罩，**先將圖片做 rgb_threshold**，成功將車道線塗紅後，**再做 Canny edges 和 Hough Transfrom**，將**兩者得到的結果做疊加 (Overlapping)**，以下是圖像結果

solidWhiteRight

solidsolidYellowLeft

solidsolidYellowLeft 將 rgb_threshold 調大

做的過程有跟同學做的影片比較，發現**道路畫線相對穩定**，可是畫出來的線會有以上結果，**一是毛邊的問題，二是會發現線越來越粗**(因為影片中的

線不斷做疊加，導致線條越疊越寬的結果)。

解決方案

毛邊結果:
可以透過將 rgb_threshold 調大些，濾掉毛邊的那些點。
線條越來越粗:
不知為何使用疊加紅線的處理方式，經 MoviePy 轉換出來的結果會不斷累加每幀，**最終選擇換個方式做，少掉第一個將車道線塗紅的動作。**

注意，換個方法做，我**依舊選擇使用梯形作為遮罩**，去做以上兩個影片的處理，經過一系列的參數調整，發現常常會有以下情形產生，像是我的**Hough lines 經常只出現一小段**，以及**前方容易出現橫的紅線**，又或者是**Hough lines 一閃一滅，無法持續顯示，甚至有線條偏移的狀況產生**。這些問題也在調整參數的同時慢慢有些想法以及經驗，以下是我的解決辦法

解決方案

線只出現一小段:
跟做的遮罩太剛好(too fit)有關，在我將遮罩調寬些後，狀況改善許多。
橫的紅線:
跟 Hough lines 的 max_line_gap 選太大有關，容易將左右連起，也跟遮罩

梯形上底設太寬有關，容易將兩側車子未過濾的的點連起來。

Hough lines 一閃一滅:

跟 Hough lines 的 threshold 選太大有關，將影像上方相對少數的點濾掉了，導致跟上下底的連線時常斷掉

線條偏移的狀況:

跟梯形的遮罩有關，因為其上底有兩個點，下底向上連線時有時會向另一邊去，這個問題經**自我思考後得到的解決辦法有兩種，一種是選用三角形的遮罩，另一種是做斜率的計算，給定一個斜率的閥值，就比較不會有偏移狀況產生。**

由於遇到上述的問題，這次我選用三角形的遮罩，中心最高點選擇影像中道路的盡頭偏上一些(避免 too fit)，左右遮罩也不選太剛好的範圍，再根據上方的經驗去做參數的調整校正，皇天不負苦心人，初步得到了相對乾淨、連續的影片，接下來，就是盡量將影片線條抓準抓穩的時候了。

二、針對畫線做穩定和準確度上的處理

我處理方法的靈感主要來自於**線條偏移的狀況的解決辦法二**，如果在畫線時，**依據兩點的斜率將得到的線做分類，並記錄點的位置，不僅能框住線的偏移，還能分類左跟右的線。分出了左線和右線，也有點的位置，可以針對某一邊的點再做分類，舉左邊為例，算出最左下，最左上的點，注意，這邊的點是在遮罩後以及框住線偏移的清況下產生，因此可以將點連線並繪出**，此方法不僅能讓線條更穩定，也能維持一定的準度。

給定線的斜率閥值和分左右線 code

```
def draw_lines(img, lines, color=[255, 0, 0], thickness=5, cutoff_slope1 = 0.35 , cutoff_slope2 = 0.65):

    line_left_x = []
    line_left_y = []
    line_right_x = []
    line_right_y = []

    for line in lines:
        for x1,y1,x2,y2 in line:
            #
            slope = (y2-y1)/(x2-x1)
            if cutoff_slope2>slope > cutoff_slope1:
                line_left_x += [x1, x2]
                line_left_y += [y1, y2]
            elif -cutoff_slope2<slope < -cutoff_slope1:
                line_right_x += [x1, x2]
                line_right_y += [y1, y2]
```

以下便是經此方法處理後的影片成果展現:

solidWhiteRight

solidYellowLeft

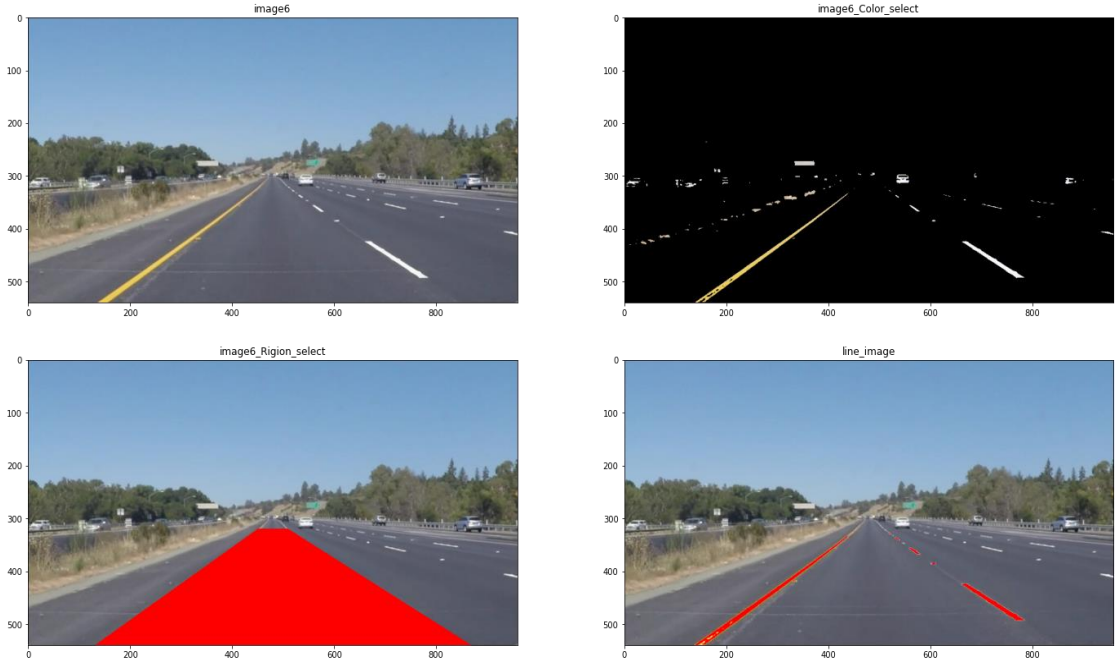
加分題-台灣高速公路

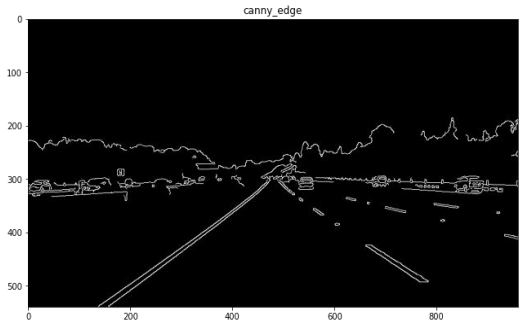
三、討論

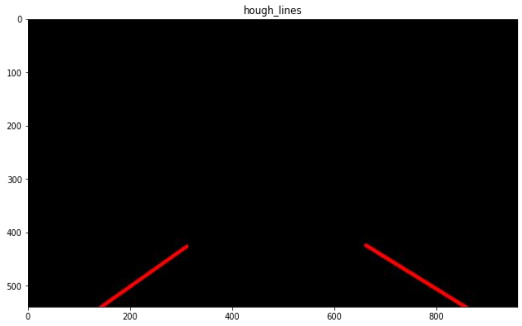
使用這種方法去畫圖雖然相對穩定許多，但也**隱藏一些問題**，由於斜率是一個範圍，所以線難免還是會有些許偏差，且在調程式參數時有時會遇到跑不出來的清況，估計跟斜率閥值範圍設太小有關，因此，也不能將範圍設定的十分精準，個人猜測這也是為何跑挑戰題常常調不好參數的原因，調太大，線條會傾來傾去，調太小，程式又常常跑不出來，最終只好先選擇放棄，去做加分題的題目。

四、工具使用

程式上我使用的是 google colab 去跑，圖像色彩閥值設定則使用老師推薦的 Color_Picker，透過將影像分析結果做子圖去看，方便自己調整參數。







五、調整參數

前面基本上都提過了，這邊稍微做總結

rgb_threshold:

透過 Color_Picker 去設定閥值

region_select:

透過將老師給的程式碼稍作修改，變成梯形遮罩，再通過子圖的結果去做調整

```
def Mask(img,left_bottom ,right_bottom ,left_top ,right_top ):
    region_select = np.copy(img)
    fit_left = np.polyfit((left_bottom[0], left_top[0]), (left_bottom[1], left_top[1]), 1)
    fit_right = np.polyfit((right_bottom[0], right_top[0]), (right_bottom[1], right_top[1]), 1)
    fit_top = np.polyfit((left_top[0], right_top[0]), (left_top[1], right_top[1]), 1)
    fit_bottom = np.polyfit((left_bottom[0], right_bottom[0]), (left_bottom[1], right_bottom[1]), 1)

# Find the region inside the lines
    XX, YY = np.meshgrid(np.arange(0, xsize), np.arange(0, ysize))
    region_thresholds = (YY > (XX*fit_left[0] + fit_left[1])) & \
                        (YY > (XX*fit_right[0] + fit_right[1])) & \
                        (YY < (XX*fit_bottom[0] + fit_bottom[1]))& \
                        (YY > (XX*fit_top[0] + fit_top[1]))

# Color pixels red which are inside the region of interest
    region_select[region_thresholds] = [255, 0, 0]

    return region_select, region_thresholds
```

```
#--------------------Color and Region Selection--------------------#
##parameter for solidYellowLeft ##
rgb_threshold = [190,170,80]
left_bottom = [130, 540]
right_bottom = [870, 540]
left_top = [450, 320]
right_top = [510, 320]
#------------------------------------------------------------------#
color_select,color_thresholds=Color_select(image6,rgb_threshold)
region_select,region_thresholds=Mask(image6,left_bottom,right_bottom,left_top,right_top)
line_image[~color_thresholds & region_thresholds] = [255,0,0]
```

Canny edges:

若是想將線條變少，調高較高的閥值，這樣可以使得強邊緣的量變少，因為其演算法是透過強邊緣去尋找線條，找出從強邊緣出發，連結弱邊緣形成線條，沒連到的弱邊緣則捨去。

弱邊緣：介於兩閥值間找出的 edges

強邊緣：大於較高閥值找出的 edges。

Hough lines:

```
def hough_lines(img, rho, theta, threshold, min_line_len, max_line_gap):
    """
    `img` should be the output of a Canny transform.

    Returns an image with hough lines drawn.
    """
    lines = cv2.HoughLinesP(img, rho, theta, threshold, np.array([]), minLineLength=min_line_len, maxLineGap=max_line_gap)
    line_img = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)
    draw_lines(line_img, lines)
    return line_img
```

threshold:

用來調最少點組成的線，調太高有時會出現 Hough lines 偏短的現象

min_line_len:

用來調最小線長，調太小時，在左右線上可能跑出橫線，不小心將毛邊連起來

max_line_gap:

用來調兩線最大間距，若是使用梯形遮罩時，調太高容易出現梯形的上底

六、心得

這次 Project 主要是訓練我們的耐心和毅力，不僅是要將道路辨識做出即可，真正的細節處理其實是在於該如何使線條更穩定，更準確。個人很慶幸自己有先將 Hough Transform Quiz 做出來和玩過老師給的 sample code 後才去做影片處理，因為在了解每項參數的意義後再去調整才會更有方向，調起來也不會像在茫茫大海中撈針，不懂自己在做些甚麼，最後附上自己在做道路辨識前調整的 sample code 成果以示證明。

Finding Lane Lines (最底下 code for 加分題使用，可以略過)