國立成功大學

自動駕駛實務報告

Project 1 : Finding Lane lines

學生：黃守維

學號：F14091148

中華民國113年4月

## Pipeline程式流程與說明

原圖：



最終設計出來的pipeline會依序經過以下七個步驟：

1. filter\_colors(image):

將原始圖片同時套上黃色及白色的濾鏡以取得下方圖片。

一張含有 黑暗, 黑色, 戶外, 路 的圖片

自動產生的描述

1. grayscale(img):

將前次輸出進行灰階處理，下圖為輸出結果。



1. gaussian\_blur(img, kernel\_size):

將前次輸出利用cv2內件函式cv2.GaussianBlur()，進行影像高斯模糊，下圖為輸出結果。

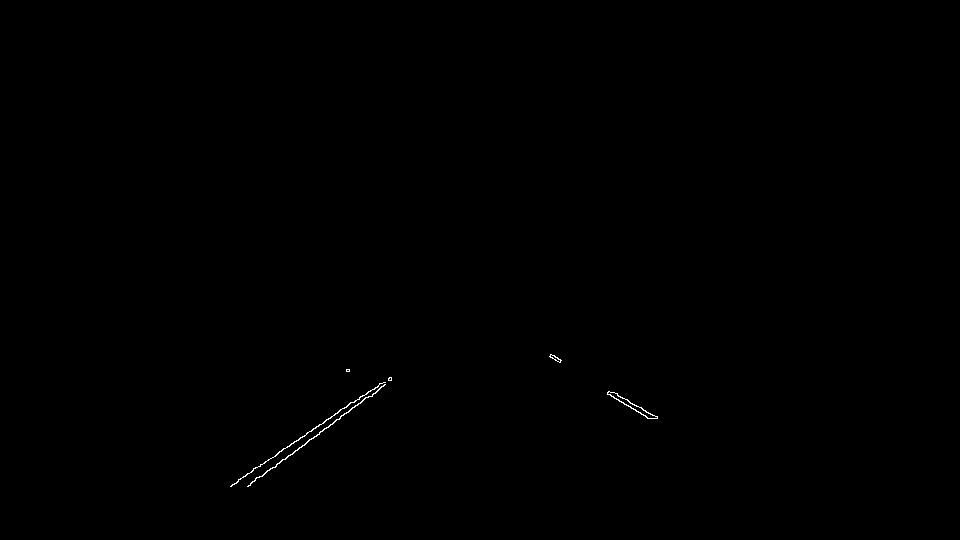
1. canny(img, low\_threshold, high\_threshold):

將前次輸出利用cv2內件函式cv2.Canny()，給定threshold來找出圖形邊緣，下圖為輸出結果。



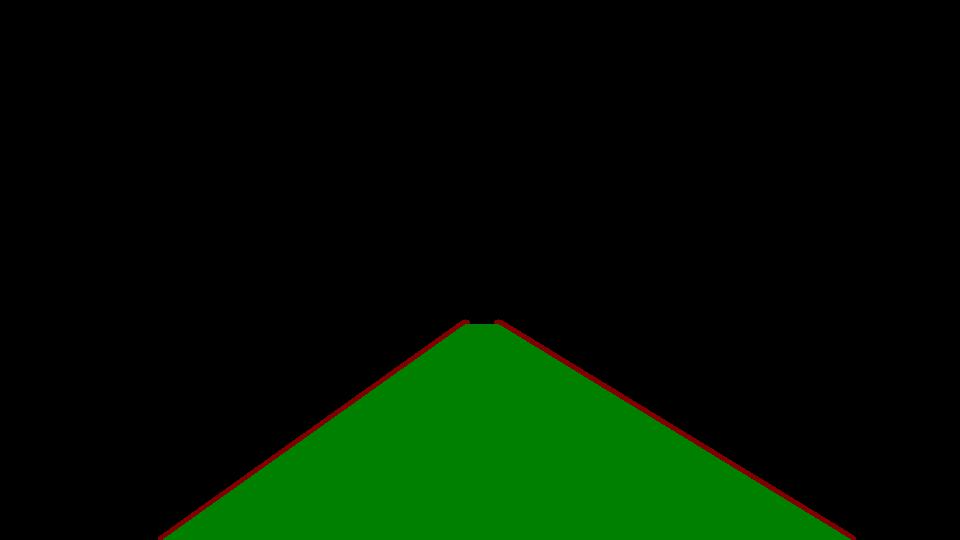
1. region\_of\_interest(img, vertices):

選出有興趣判斷的區域，為此我額外建立了一個函式get\_vertices(image): 用以建立梯形遮罩，下圖為輸出結果。



1. hough\_lines(img, rho, theta, threshold, min\_line\_len, max\_line\_gap):

利用cv2預設函式cv2.HoughLinesP() 檢測出圖像中的直線，為了方便觀察，我額外建立了一個函式 draw\_lines() 以清楚顯示車道範圍以及左右車道線位置，紅線代表左右車道線位置，綠色區域為車道範圍，下圖為輸出結果。



1. weighted\_img(img, initial\_img, α=0.8, β=1., λ=0.):

最後再將預測出來的車道線及車道範圍套用在原圖上，下圖為輸出結果。



## 辨識成果與設定參數

參數設定：

1. gaussian blur:

kernel\_size = 3

1. canny:

low\_threshold = 180

high\_threshold = 240

1. region\_of\_interest:

rows, cols = image.shape[:2]

bottom\_left = [cols\*0.10, rows\*0.9]

top\_left = [cols\*0.38, rows\*0.64]

bottom\_right = [cols\*0.90, rows\*0.9]

top\_right = [cols\*0.62, rows\*0.64]

1. hough\_lines:

rho = 2

theta = 1 \* np.pi/180

threshold = 20

min\_line\_length = 10

max\_line\_gap = 100

1. filter\_colors:

lower\_white = [200, 200, 200]

upper\_white = [255, 255, 255]

lower\_yellow = [90, 100, 100]

upper\_yellow = [110, 255, 255]

1. solidWhiteRight

影片連結：<https://youtu.be/Uw1rX-fVnWU>

1. solidYellowLeft

影片連結：<https://youtu.be/QCAmynzVY1Y>

1. challenge

影片連結：<https://youtu.be/afGteEFUtKA>

1. extraPointsQuestions

影片連結：<https://youtu.be/PpiGel0UyC4>

## 結果與討論

為解決challenge.mp4中淺色地板區域車道線的辨識問題，我在步驟的一開始使用filter\_colors()，給定白線的顏色判定範圍以及黃線的顏色判定範圍，將黃線以及白線的位置找出來，接著才套上灰階濾鏡，如此便能解決淺色地板區域車道線的辨識問題。

加分題extraPointsQuestions.mp4的部分主要是調整region\_of\_interest()中梯形遮罩的範圍，要跳過影片右下角的浮水印，同時不能導致solidWhiteRight.mp4、solidYellowLeft.mp4 和challenge.mp4車道辨識失敗，最終參數是經過多次調整後得出的結果。