課程名稱：自動駕駛實務

PRACTICES OF AUTONOMOUS DRIVING

作業名稱：Finding Lane Lines on the Road

姓名：黃子峻

系級：112 資訊四乙

學號：E14084117

# Project架構

此 Project 的流程架構主要分為三部分，如下流程圖中所示：

下方說明該三部分的子架構與內容：

## 建立車道辨識方法流程

下方流程圖顯示針對單一行駛圖片的**車道辨識方法流程：**

## 在測試圖片上檢視車道辨識效果

分別針對 /test-images 資料夾中的solidWhiteCurve.jpg、solidWhiteRight.jpg、solidYellowCurve.jpg、solidYellowCurve2.jpg、solidYellowLeft.jpg、whiteCarLaneSwitch.jpg 等車輛行駛圖片，執行 1.1 所描述的車道辨識方法，產生含有車道**辨識結果的圖片，檢視車道標示是否符合預期。**

## 在車輛行駛影片測試車道辨識效果

以 Python 的 moviepy 套件的editor. VideoFileClip() 函數分別讀入 /test-vedios資料夾中的 solidWhiteRight.mp4、solidYellowLeft.mp4、challenge.mp4、National\_Highway\_No\_1\_30s.mp4 等車輛行駛影片，並以moviepy 套件的 fl\_image() 方法將影片中的每一幀圖片以1.1 所描述的車道辨識方法替換為含有車道標註的圖片。最後用 moviepy 套件的write\_videofile()方法儲存含有車道辨識結果的影片。

National\_Highway\_No\_1\_30s.mp4 影片取得方式為：以此網站工具「https://www.freemake.com/tw/free\_video\_downloader/」下載「https://www.youtube.com/watch?v=0crwED4yhBA&t=4708s」影片的 mp4 格式影片檔案，擷取其中的 1:18:28 ～1:18:58 總長 30 秒的片段。

## Project程式碼檔案架構

* 專案程式碼的 GitHub Repository 連結：
* 備註：除了P1.ipynb、National\_Highway\_No\_1\_30s.mp4 以外檔案皆來源自<https://github.com/udacity/CarND-LaneLines-P1>
* environment.yml：紀錄 conda 環境要安裝的套件與版本
* P1.ipynb : 包含車道辨識程式碼
* /test-images 資料夾
* solidWhiteCurve.jpg
* solidWhiteRight.jpg
* solidYellowCurve.jpg
* solidYellowCurve2.jpg
* solidYellowLeft.jpg
* whiteCarLaneSwitch.jpg
* /test-vedios資料夾
  + solidWhiteRight.mp4
  + solidYellowLeft.mp4
  + challenge.mp4
  + National\_Highway\_No\_1\_30s.mp4
* /test-vedio-output資料夾
  + solidWhiteRight.mp4
  + solidYellowLeft.mp4
  + challenge.mp4
  + National\_Highway\_No\_1\_30s.mp4

# Project方法

此部分詳述章節「1.1 建立車道辨識方法流程」中的 7 個步驟中的方法細節：

## 定義車道標線顏色（黃、白）的過濾器，以找出特定顏色範圍的像素座標

由於各範例圖片、影片中的車道標線僅存在白色與黃色兩種顏色，因此可以透過保留白、黃兩種顏色的過濾器，在圖片或影片單幀中識別出白、黃兩種顏色的像素位置。

實作細節：

* + - 1. 輸入要處理的圖片或影片單幀圖片。
      2. 使用 OpenCV 的 cv2.inRange() 函數分別產生與輸入圖片長寬相同，黃色、白色部分別為數值 1 ，其餘部分數值為 0 的兩個numpy 陣列，做為遮罩。黃色的顏色範圍參數下界設為[20, 100, 100]，上界設為[30, 255, 255]；白色的顏色範圍參數下界設為[0, 0, 185]，上界設為[255, 70, 255]。
      3. 使用 OpenCV 的cv2.bitwise\_or()，以 OR 邏輯運算合併黃色、白色對應的numpy 陣列，作為新遮罩。
      4. 使用 OpenCV 的cv2.bitwise\_and()，將 C. 產生的遮罩以 AND 邏輯運算跟原圖合併，留下原圖中黃色與白色的部分，其餘部分為黑色。
      5. 建立一個與要處理的圖片或影片單幀圖片長寬相同，所有數值為 0 的 numpy 陣列，使用邏輯判斷讓與 D. 步驟產生的圖有顏色的像素座標相同座標的數值為 1 作為輸出。

下圖展示顏色（黃、白）的過濾器的效果：

一張含有 黑暗 的圖片

自動產生的描述

## 將 RGB 格式圖片轉為灰階格式，以 Gausion blur 模糊化，再以 Canny 方法偵測圖片中的邊緣(Edge)特徵，並得到其像素座標

車道標線的特徵除了顏色外，車道標線與深色路面交接處的邊緣也有助於判斷標線的走向，因此採用Canny 算法辨別出圖片中的邊緣特徵，該方法需要先進行將RGB 格式圖片轉為灰階格式，然後以 Gausion blur 模糊化消去雜訊的前處理。

* 實作細節：

1. 輸入要處理的圖片或影片單幀圖片。
2. 使用 OpenCV 的cv2.cvtColor() 將原影片單幀圖片的 RGB 格式 numpy 陣列，轉換為灰階格式。
3. 使用 OpenCV 的cv2.GaussianBlur() 將上一步的辨別結果模糊化。
4. 使用 OpenCV 的cv2.Canny() 辨別出圖片中道路上的邊緣特徵。
5. 建立一個與要處理的圖片或影片單幀圖片長寬相同，所有數值為 0 的 numpy 陣列，使用邏輯判斷讓與 D. 步驟產生的圖的非 0 像素座標相同座標的數值為 1 作為輸出。

下圖展示圖片轉為灰階並模糊化的效果：



下圖展示邊緣偵測的效果：

一張含有 文字 的圖片

自動產生的描述

## 以 AND 邏輯運算合併步驟 2.1、 2.2 的像素座標結果

由於希望找到圖片中顏色是黃色或白色的標線邊緣特徵，因此使用**AND 邏輯運算合併步驟** 2**.**1、 **2.**2 **的像素座標結果。**

* **實作細節：**

1. 使用 OpenCV 的cv2.bitwise\_and()，以AND 邏輯運算產生白色、黃色（2.1 輸出）以及邊緣特徵（2.2 輸出）的像素標記合併結果的numpy 陣列。

下圖展示2**.**1、 **2.**2 **的像素座標合併的效果（顏色僅為示意）**：



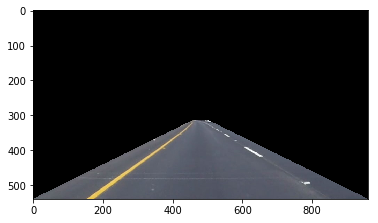
## 定義感興趣的車道範圍，並遮擋其餘部分在 2.3 合併結果中的像素座標

本專案中採用的梯形遮擋圖片中非車道的範圍，將2.**3 合併結果中**此梯形外的numpy 陣列數值設為 0 。

* **實作細節：**

1. 給定圖片中要保留之道路範圍（梯形）的四個頂點像素座標。
2. 建立一個與要處理的圖片或影片單幀圖片長寬相同，所有數值為 0 的 numpy 陣列。
3. 使用 OpenCV 的 cv2.fillPoly() 函數使得上一步建立的 numpy 陣列中，給定的梯形範圍內的數值為 255 ，其餘為 0。
4. 使用 OpenCV 的 cv2.bitwise\_and() 函數，產生**遮擋梯形外部分在** 2.**3 合併結果中的像素座標的結果作為輸出**。

下圖展示梯形**遮擋在原圖與**2.**3 合併像素的效果（顏色僅為示意）：**

一張含有 夜晚 的圖片

自動產生的描述

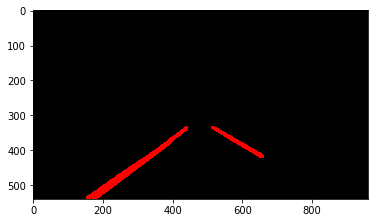
## 以 Hough Transform方法找到 2.4 結果中的各直線兩點座標

由於目標影片、圖片中的車道接近直線，但 2.4 產生結果中的邊緣特徵不一定構成直線，所以再對 2.4 結果執行**Hough** Transform**演算法，識別出其中的**直線特徵。

* **實作細節：**

1. 將 2.4 的結果作為輸入
2. 使用 OpenCV 的**cv2.HoughLinesP() 函數，找出數個構成直線特徵的兩點像素座標作為輸出。**

**下圖展示將 B. 所得直線特徵的兩點像素座標的直線繪製結果：**

****

## 以步驟 2.5 結果計算車道左右兩側標線的直線方程式

因為車道的左、右兩標線分別具有不同的直線方程式，所以對 2.5 找到的直線特徵進行左、或右標線特徵的分類，同時檢查每個直線特徵的斜率與位置是否有異常。

* **實作細節：**

1. **將 2.5** 的結果（每個**直線特徵的兩點像素座標**）作為輸入
2. **找到圖片橫向的中點座標 x（圖片寬度的一半）**
3. **檢查當：**
   1. **直線特徵的兩點像素座標皆小於x，將該直線特徵分類為左側標線的直線特徵。**
   2. **直線特徵的兩點像素座標皆大於x，將該直線特徵分類為右側標線的直線特徵。**
   3. 其他情況則捨棄該直線特徵。
4. **分別將構成左側標線的直線特徵的每個像素座標作為輸入，以 sklearn 套件的linear\_model.LinearRegression() 函數，求出左側標線的直線方程式係數。右側標線方程式也以同樣方式求得。**

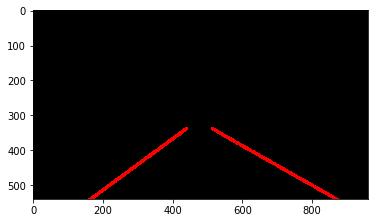
## 以步驟 2.6 所得之車道標線方程式，繪製車道左右兩側標線於原圖

此步驟將車道的辨識結果，以畫線方式標示於原圖片中。

* **實作細節：**

1. **取得2.5 所得所有直線特徵的兩點像素座標中最接近圖片上緣的 y 值。**
2. **取得圖片下緣的座標y 值。**
3. **將 A. B. 所得的兩個 y 值分別帶入 2.6 所得之左、右側標線的直線方程式，得到特定y對應的 x 值。**
4. **將 C. 以左側標線的直線方程式所得的兩點座標帶入 OpenCV 的 cv2.line() 函數，在原圖片上繪製直線。右側標線以同樣方式繪製。**

**下圖展示D. 的繪製結果：**

****

# 結果

## Youtube 車道辨識結果影片連結

1. solidWhiteRight.mp4：https://youtu.be/WvKc\_PZ1868
2. solidYellowLeft.mp4：https://youtu.be/ljX3qiaRrt8
3. challenge.mp4：https://youtu.be/clMibiFHxWc
4. National\_Highway\_No\_1\_30s.mp4：https://youtu.be/0lcTfawvjGg

## 測試圖片的車道辨識結果：

1. solidYellowCurve.jpg車道辨識結果

* 從圖片中可以看到，方法正確的辨別實心黃色標線與分段的白色標線，且不受地平線附近道路微彎的影響。



1. solidYellowLeft.jpg車道辨識結果

* 從圖片中可以看到，方法正確的辨別實心黃色標線與分段的白色標線。



1. solidYellowCurve2.jpg車道辨識結果
   * 從圖片中可以看到，方法正確的辨別實心黃色標線與分段的白色標線，且不受地平線附近道路微彎的影響。

一張含有 文字, 場景, 天空, 路 的圖片

自動產生的描述

1. solidWhiteRight.jpg車道辨識結果
   * 從圖片中可以看到，方法正確的辨別實心白色標線與分段的白色標線。



1. whiteCarLaneSwitch.jpg車道辨識結果
   * 從圖片中可以看到，方法正確的辨別實心黃色標線與分段的白色標線，且不受右側車道白色分段標線的影響。



1. solidWhiteCurve.jpg車道辨識結果
   * 從圖片中可以看到，方法正確的辨別實心白色標線與分段的白色標線，且不受圖片中心道路右彎的影響。



# 心得與討論

過去我一直誤以為辨識車道標線一定要使用經過複雜訓練、需要大量運算資源的模型，但經歷本次 Project 結合許多經典電腦視覺方法，就能在筆直路段的上良好的辨識出車道標線，讓我感到十分驚艷。我認為本次 Project 當中最難的部分在於顏色過濾器與 Hough Transform 的參數調整。有時候為了讓標線更加清晰而去調整顏色過濾器，但稍微沒調整好參數就很容易讓車道旁的黃褐色草地與較白的天空無法被遮擋。另外由於短短幾秒的車輛行駛影片就包含許多幀圖片，要想對車道標線判斷的每幀圖片去調整Hough Transform 的參數是不小的討戰，而且也需要擔心調整後原先沒問題的幀數會因為更動而出問題。

本次 Project 的測試影片都屬於白天且視野開闊與前車距離遠的情況，讓我不禁好奇如果在充斥著各色燈光且忽明忽暗的晚上接到上行時使，需要哪些更進階的電腦視覺技術輔助車道標線的辨識，還有如果離前車很近，且左右或前方車輛遮擋住大部分的標線的時候，自駕車是否還能以電腦視覺辨別正在行駛的車道？

在本次Project 的加分題中，發現在國道實際行駛影片中辨識車道標線比起測試影片困難了一個檔次，期許我能在期末專題實作過程中，補足技術不足之處，對自駕車的視覺機制有更深入的了解。

## 影片單一幀圖片中的車道辨識流程

本專案中所使用的自駕車影片是由多幀圖片依順序排列組成，因此要想辨識出影片中的車道須先建立從每幀圖片中辨識出車道的方法流程。本專案中的方法如下描述。

### 配合 Canny 算法辨別出圖片中道路上的邊緣特徵像素座標

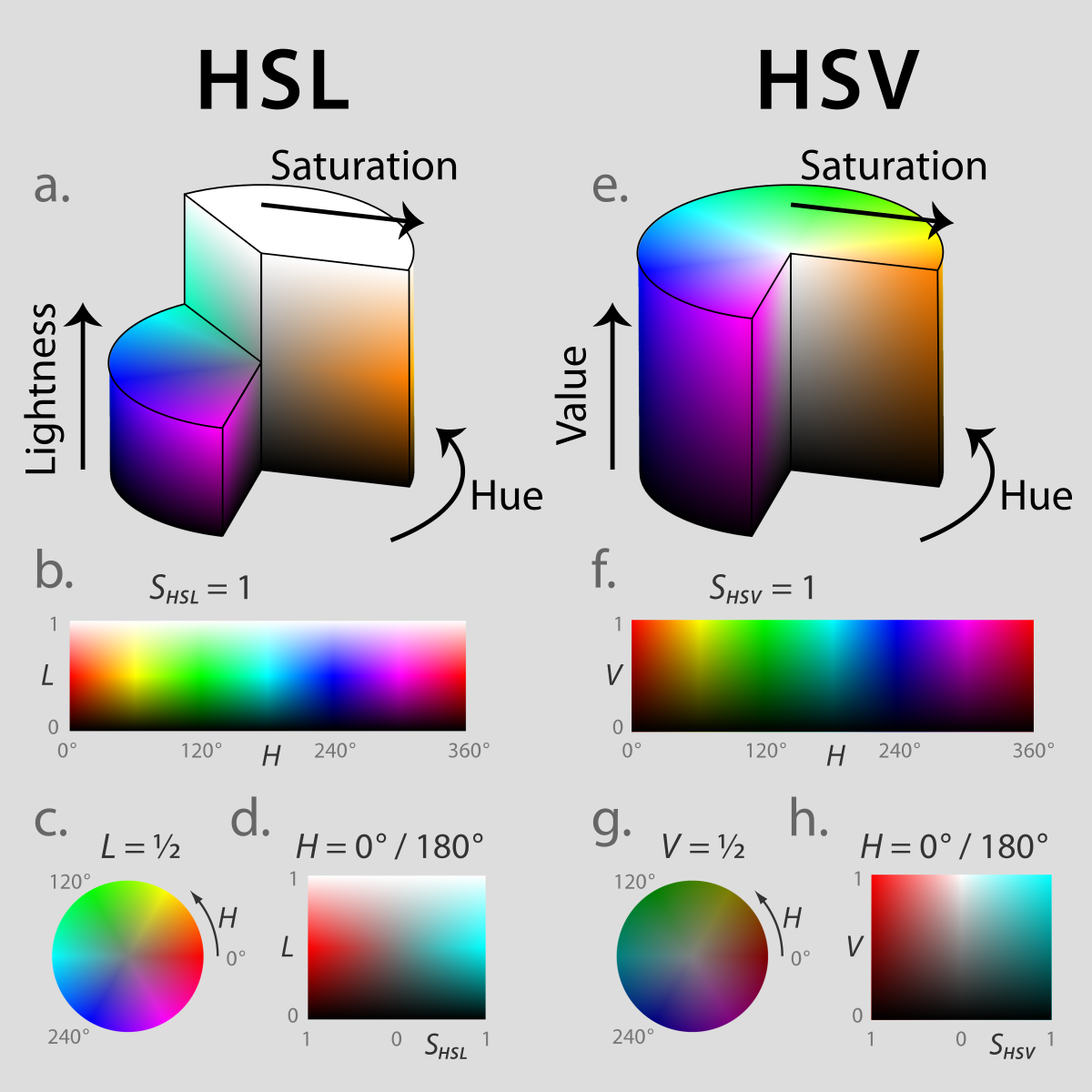
遮擋圖片中非車道的範圍（1.1）

### 結合白色與黃色車道標線以及道路上的邊緣特徵的像素座標

### 以 hough 算法辨識出合併像素結果中的直線兩點像素座標

### 將車道辨識結果標線顯示於影片中

* + - 1. 使用 OpenCV 的cv2.bitwise\_and()，產生原影片單幀圖片與含梯形為0 數值之numpy 陣列的 AND 邏輯運算結果，也就是產生只有保留道路上白色與黃色像素，其餘部分為黑色的圖片。

1. **且兩點構成斜率介於 30 ~ 70 度**
2. 本專案中採用HSV 色域的白、黃兩種顏色的過濾器設計的顏色範圍如表 1 5中描述。
3. 
4. 圖 1 HSV 色彩表示法示意圖（來源：<https://zh.wikipedia.org/zh-tw/HSL%E5%92%8CHSV%E8%89%B2%E5%BD%A9%E7%A9%BA%E9%97%B4>）

|  |  |  |  |
| --- | --- | --- | --- |
| 顏色 | 色相範圍 | 飽和度範圍 | 明度範圍 |
| 白色 |  |  |  |
| 黃色 |  |  |  |

1. 表 1