**Udacity Self-Driving Car Engineer**

**Report**

* **Project:** Finding Lane Lines
* **Name:** Shao Hongxu

**Let me just use some brief sentence to explain the thing I did, basically these are 3 big parts here:**

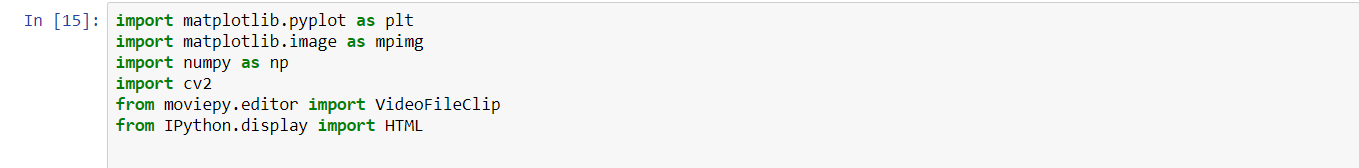
**Part 1:** import necessary lib

**Part 2:** identify process\_image function

**Part 3:** convert clips into video. (only for video)

**To be specific as below, you will find some note on Script, key take away and Area could be improved:**

**Part 1:** import necessary lib:

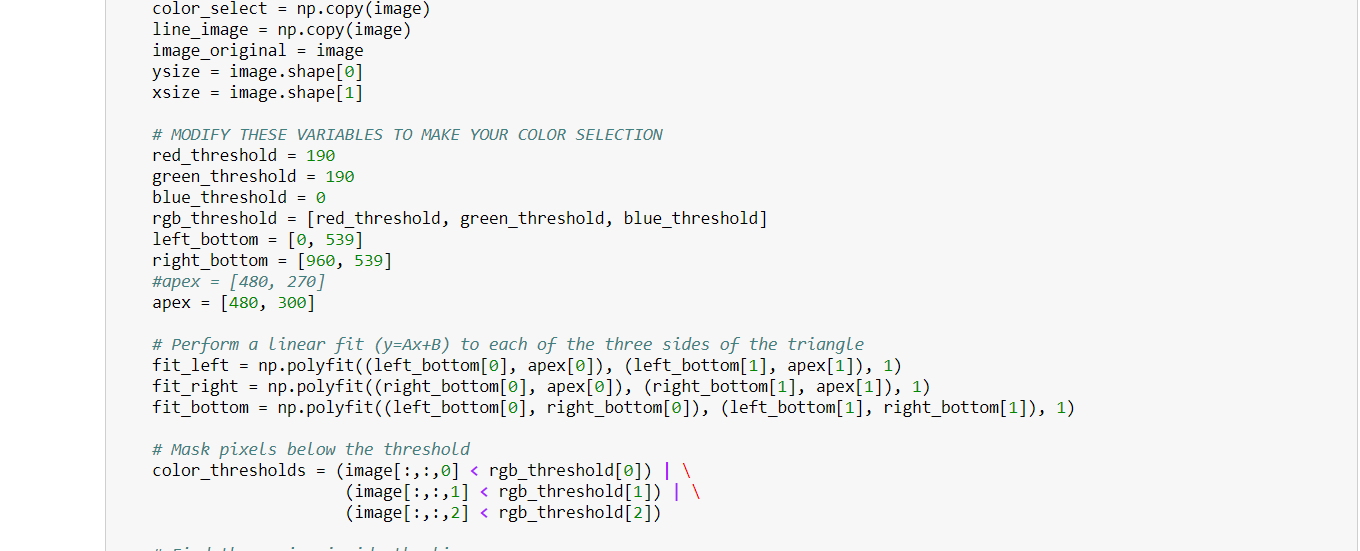


**Part 2:** identify process\_image function:



To archive the goal of detecting lane Lines, below is what I have done under process\_image function:

**2. 1:** Using colour selection function to decouple image area which is not lane lines.



**-> Key take away:** it has to be a very basic colour selection process considering different colour of line detection, like wine, yellow etc.

red\_threshold = 190

green\_threshold = 190

blue\_threshold = 0 (as yellow is made by red & green only)

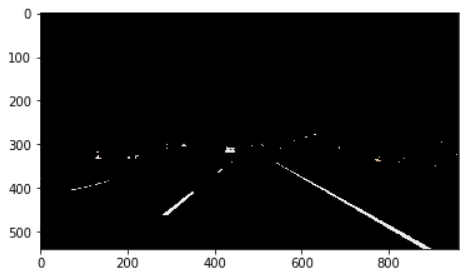
**-> Area could be improved:** Here we can do a research on how many colours could be for line, and use an "or" logic to cover all different colour conditions. And parameter here not working for all conditions, like in chanllenge.mp4, few scenconds the road is in white and line is in light yellow, in that case, this parameter can’t detect the line. This is an important area has to be improved but not yet have any idea. Will update later in case some thought/ idea coming out.

**-> Result:**

*Original image:*



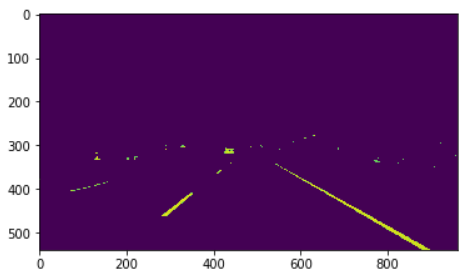
*Image after using colour selection function:*



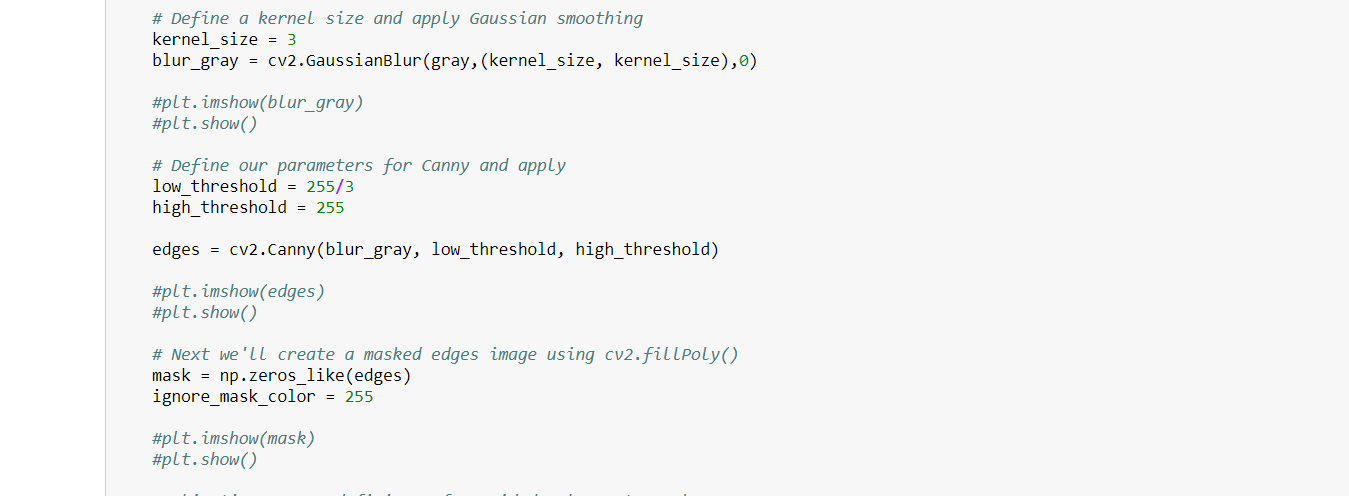
**2. 2:** Gray the image

**-> Result:**

*Image after using Gray function:*



**2. 3:** using Canny function to find edges between low\_threshold and high\_threshold



And draw the line in Red:

**-> Key take away:** low\_threshold (Research told me on practice experience say low could be 1/3 of high, even author of function says so, but i need to research in deep especially explanation from math angle)

high\_threshold (we should make it as highest 255)

low\_threshold = 255/3

high\_threshold = 255

**-> Area could be improved:** how to do it better on selecting below parameters:

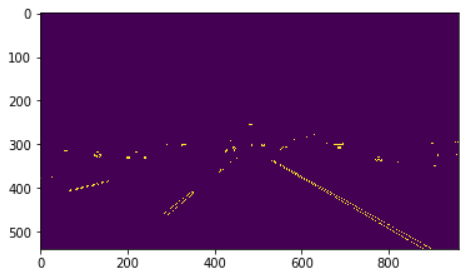
low\_threshold

high\_threshold

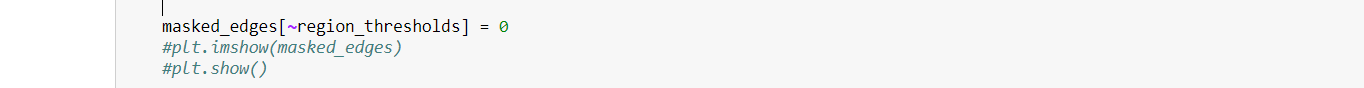
But can we rely on computer vision to choose parameter by itself? like we think about a rule to tell the quality of the result given from different parameter. or machine learning to handle that.

**-> Result:**

*Image after using Canny function:*

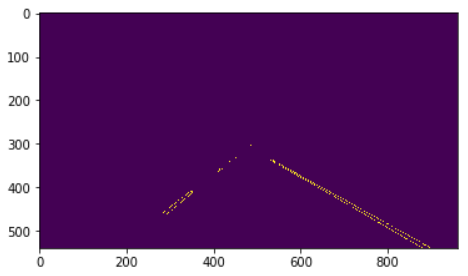


**2. 4:** to restrict analysis area into a triangle



**-> Result:**

*Image after using restrict function:*



**2. 5:** using HoughLinesP to find target line.



**-> Key take away:** again, parameter...

rho = 1 # distance resolution in pixels of the Hough grid

theta = np.pi/180 # angular resolution in radians of the Hough grid

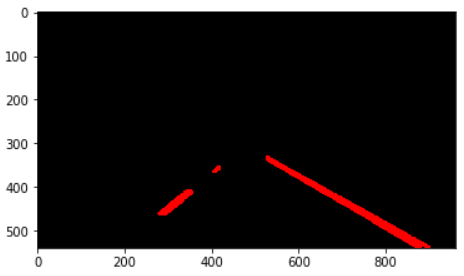
threshold = 15 # minimum number of votes (intersections in Hough grid cell)

min\_line\_length = 30 #minimum number of pixels making up a line

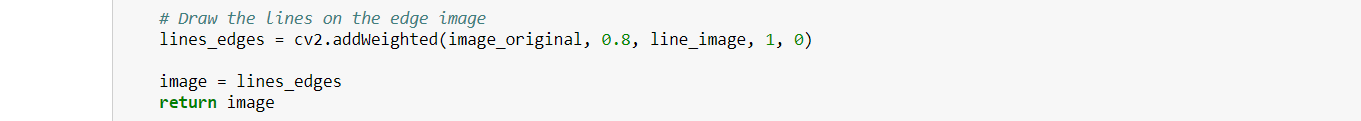
max\_line\_gap = 10 # maximum gap in pixels between connectable line segments

**-> Result:**

*Image after using HoughLinesP function:*



**2. 6:** Club /addWeighted together with original image



**-> Result:**

*Image after using HoughLinesP function:*



**Part 3:** convert clips into video. (only for video)



**Summary:**

These are the steps I could think of for project 1, as a base line. Still a lot of improvement needed to achieve a better accuracy & more effective exceptional handling model.

The key area has to be improved is few second inside challenge.mp4 video, the road changed from black into white, and a lot of dirty line made by car wheel, and system detected them as line as well, will continue thinking about the solution on it, in case any new idea, will update here accordingly.