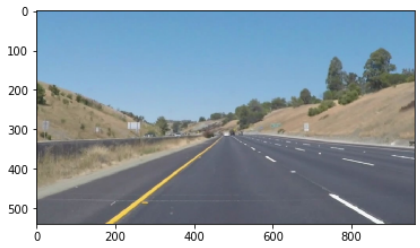
Nano degree Self Driving Car
Project 1: Finding Lane Lines

I worked on tackling the problem in 2 slightly different ways: Simple method as described in the lectures and the second method motivated to solve the challenge problem used HSV.

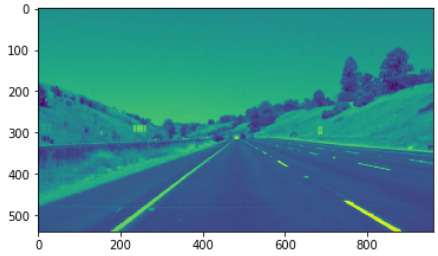
Method 1:

Method 1 involved 10 steps as under:

1. Gaussian blurring/smoothing to reduce the effects of noise.



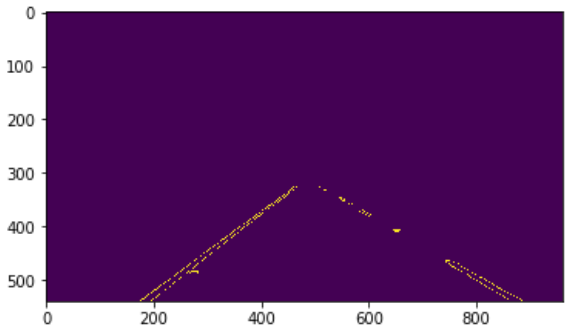
2. Conversion of image to grayscale



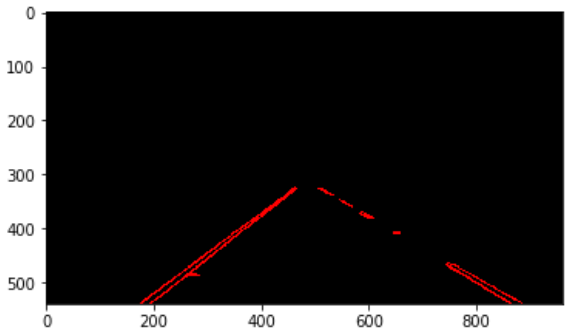
3. Determination of vertices identifying the region of interest. Since the camera is fixed to the car these will not change.

4. Canny edge detection function over the grayscale image

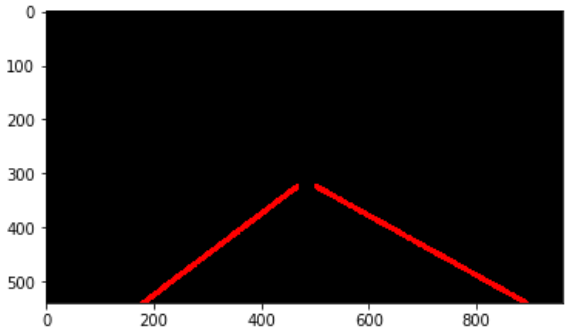
5. Masking the Canny edge detection image to isolate the ROI



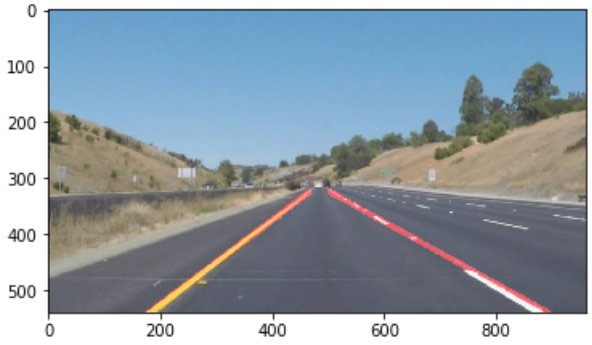
6. Hough lines function applied to the Canny edge detection image



7. The hough line function identifies all perspective lines which are part of the lane lines
8. I created a function average_lines() to find the mean slope and line from the hough lines.
9. Using the results of average_lines() and a modified version of draw_lines function called draw_lines_cus(), I created a image which has lane lines drawn out.



10. Finally, I merged the original image with the image generated above



The Caveat was tuning of the canny edge and hough lines parameters for the videos. I also put a restriction of the slopes allowable for the lane lines in my custom function average_lines().

Method 2:

Method 2 is similar to method 1 except the initial processing of the image. In method 2 I am not using a grayscale image. Instead, I converted my image to HSV format and attempted to extract the yellow and white lines individually. I merge the 2 images generated into a single image which makes up the lane line template. The impetus to do this was to solve the challenge problem.

Shortcomings:

Method 1 is not robust enough to deal with shadows in the challenge problem. I do need to clean up the code and make it concise. Another shortcoming might be that I am relying on too may real world parameters like the lane line slope ranges for the left and right lines.

Improvements:

Identify yellow and white lines separately and incorporate a snippet of code to handle shadows and lines which are noise (paint lines, reflections, road debris)