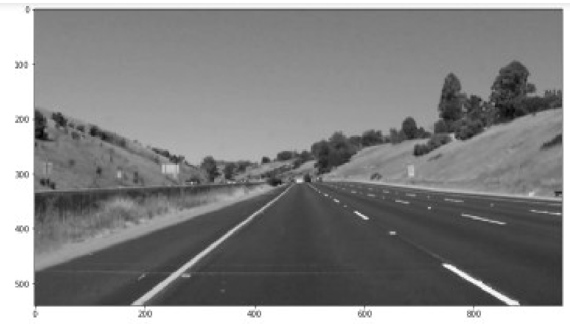
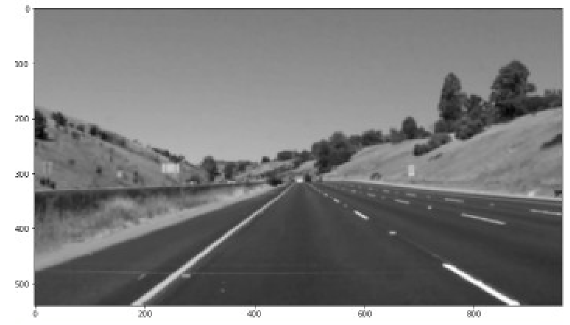
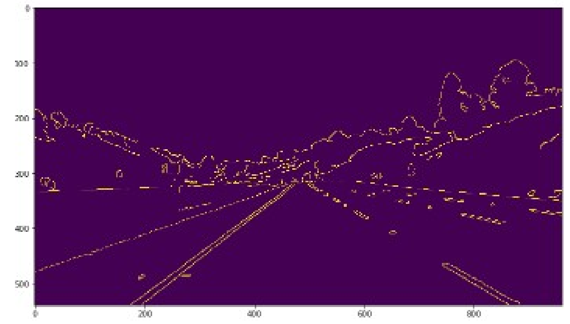
Ajay Curam
Udacity Self Driving Nano Degree
Project 1: Finding Lane Lines on the Road



Goals:
The goal of this project was to successfully identify lane lines within a video utilizing canny edge detection and hough transforms. We were to create this pipeline through Python code, and utilizing Open-CV.

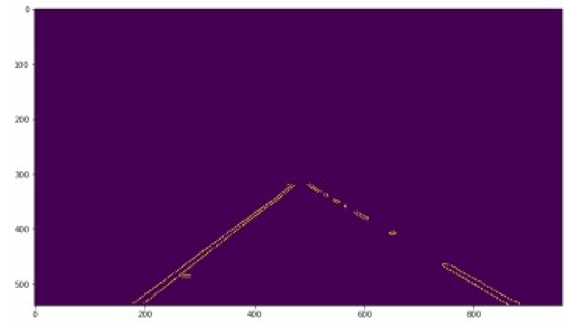


Reflection:
1. My pipelines consisted of 5 steps. The first was to convert the image to gray scale. This allowed the image to be parsed for either black or white pixels, and I identified that pixels close to white in coloration were the lane lines on the road. A Gaussian blur filter was applied to the image in order to blur the image to skew any pixels that could interfere with canny edge detection. Canny edge detection was applied in order to arrange the image in terms of gradient, and shows pixels where colors vary a lot. Then a mask was applied to the image in order to isolate where the lane lines were located in the image. In this mask, hough lines were detected which indicated where out lane lines were on the image, and this was overlayed on top of the image. The result was multiple lines that went through were the lane lines where, but did not give a definite trace of where the lane lines really were. The pipleline my images went through are shown to the right:



2. I needed to modify the draw lane lines to take an overall average of the lines that were found after the hough transformation was done. I first split up each line that was created into either the left or right lane depending on its slope value. Positive slopes indicated the left lane, while negative slopes indicated the right lane. In order to plot definite lines on the image, I reverse engineered the y = mx+b formula. I found the overall average slope for both the left and right lines independently. Then I found their b-intercept by rearranging the two equations:

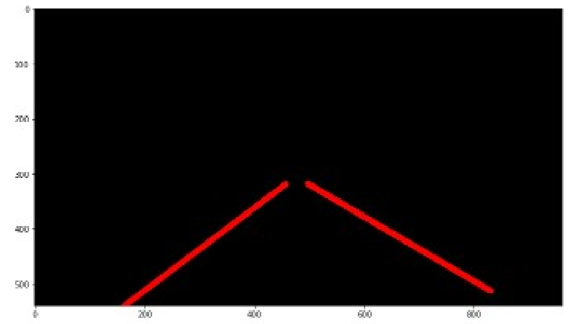
A) y1 = x1m+b
B) y2 = x2m+b



I multiplied equation A with x2, and B with x1, then subtracted them to yield:

C) b = (((y1*x2)-(y2*x1))/(x2-x1))

I then utilized y = mx + b with a constant x to solve for two y values that were used to plot the left and right lane lines. Each lane line found by the Hough transformation was averaged for both slope and b-intercept, and then the average line was plotted. The final result wasn't as accurate as I would have liked, but I feel that my methodology approaching the problem was proper. The resultant line needed to be offset a bit, as the coordinate plane for an image is not the plane one uses to plot mathematical functions.



3. I believe that my pipeline needs quite a bit of work. First off, it is not robust enough to find lane lines in images of different resolutions, as it needs to be given a X value to find a Y for. In addition the right lane isn't as long as it needs to be. The pipeline needs to be refined much more in order for something like this to be implemented in an actual driverless car.