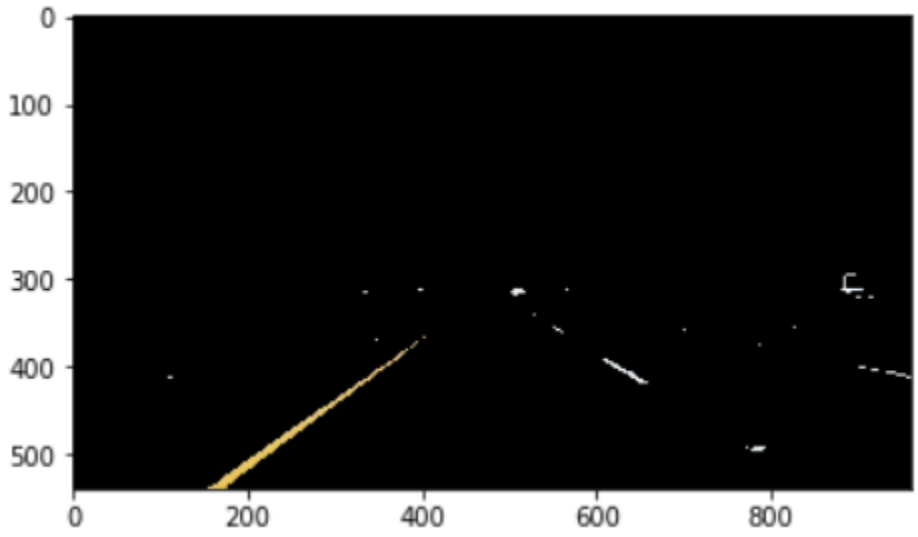
Reflection

1. Pipeline description

My pipeline consisted of the following steps:

I. First, I converted the image to the HSV and the HSL color space to evaluate the effectiveness of extracting the lane markings from the images. We know that lane markings can be of two colors – white and yellow. So, we set thresholds for each one of them separately and fuse the two images to obtain the lanes as shown below:

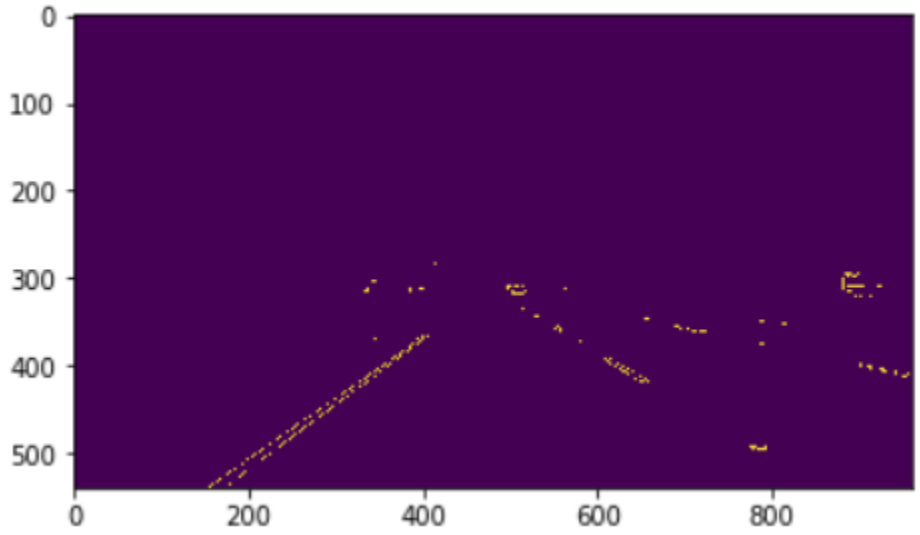


As you can see, with the right threshold chosen for the image the white and yellow lanes are isolated.

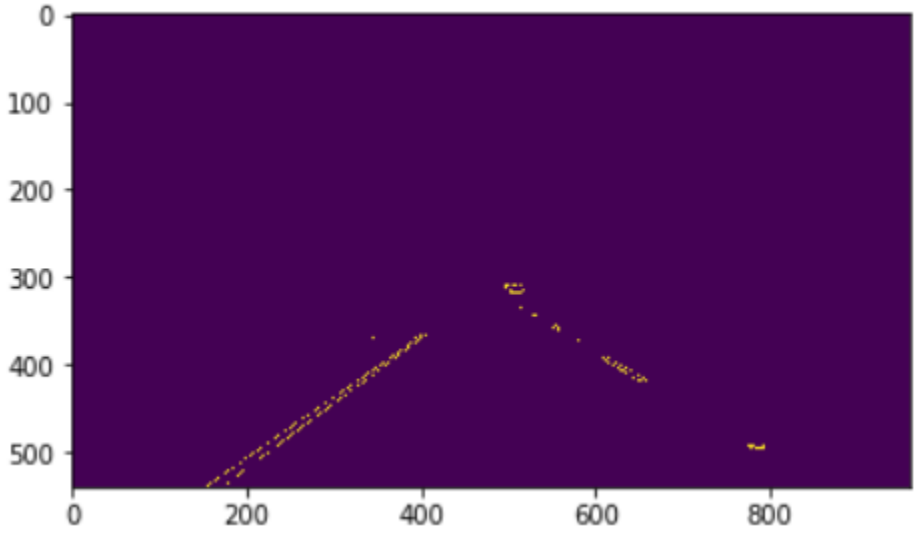
II. The image is then converted to grayscale to convert the image to a 1-dimensional array of pixel intensities ranging from 0-255.

III. The image is gaussian blurred with a kernel size of 5. This is done to remove any noise that might reside in the image.

IV. Canny edge detection is applied on the blurred image using thresholds 50, 190. The edge extracted image is as shown below:



V. Created a mask for the image which allows us to focus on the lanes while eliminating noise. This is done because we know the position of the camera. Hence, we can focus on the important aspects such as the lower half of the image, and the center as shown below:



VI. Now that we have the edges extracted as pixels based on the change in gradients, we apply the Hough transform to check for the number of points that lie on a straight line. This gives us line segments that connects the points to form lines. The output of the hough transformed image is as shown below:

VII. The key aspect of the project was to identify and separate the left and right lanes. This was done in two steps. First, we separate the line segments as left and right, based on the line inclination. Next, we polyfit the (x, y) coordinates on either side to generate a single solid line. These lines are then plotted on the original image as shown below:

2. Identify potential shortcomings with your current pipeline

I. One of the major shortcomings of the pipeline was that the lines were a little wiggly at times. This was mainly due to the varying conditions in the image and the influence of noisy elements. I tried applying a simple time domain filter to accept the previous line data. However, I was unable to store data from the previous time step because of the way the video is being read.

II. The pipeline does not take into account, the curvature of the road and fits a curve with degree 1 (basically a straight line). The pipeline could be extended to fit polynomials of higher order.

3. Suggest possible improvements to your pipeline

I. Wiggly lines:

A. The first method of dealing with this is to apply a Kalman filter to the detected lines. Based on the confidence in prediction, the filter would fuse the previous measurement and current data to predict the actual position of the line. This would eliminate the wiggly lines.

B. The second method is to apply a simple time domain filter that uses the previous line data if there is no data generated in the current time step.

II. Curvature of the road:

A. The curvature of the road can be fitted with a higher polynomial using a cost function that minimizes the deviation from as many points as possible.