# Finding Lane Lines on the Road

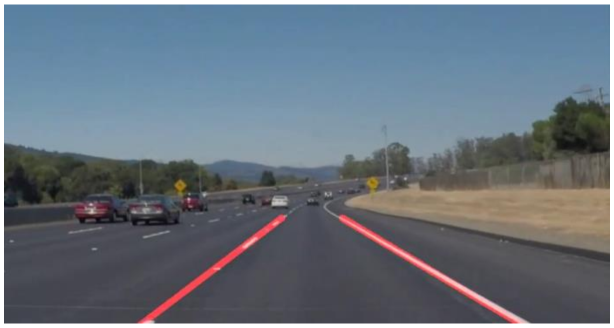
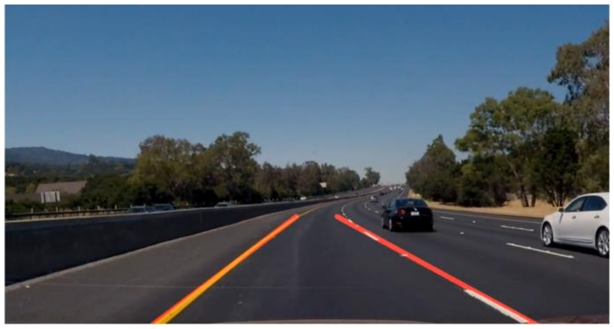
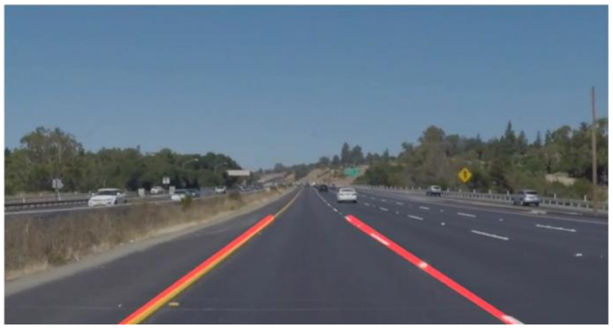
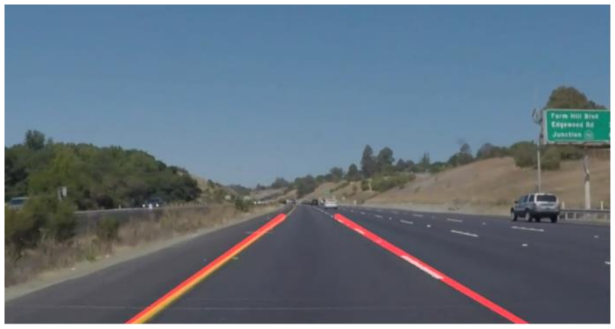
My pipeline consisted of 5 stages.

1. Conversion of image to grayscale
2. Applying Gaussian blur on the image to reduce the sharpness and the noise in the image.
3. Converting image to an image with only edges using Canny edge detection. This is done by checking adjacent cells. If high contrast in pixel value is found between them, It means and edge.
4. To remove unwanted area I created a triangular polygon of interest and masked the image to remove the unwanted portion.
5. Hough transform is used thereafter via Open CV method which helped connecting lines, we are interested in and in eliminating the rest. In this step we also used the draw_lines() method to add lines over the lanes in the image.
6. Last step was to merge both the modified image and the original one to produce the result.

Without any modification the Draw_lines() method yielded flickering lines over the lanes. Because it plotted all the lines returned by the Hough transform method. To remove flickering and select an almost constant length of line I created two dictionaries and iterated over all lines returned by Hough transform to find the longest. I stored its slope and intercept too in the dictionary after which I plotted this longest found line on the image.

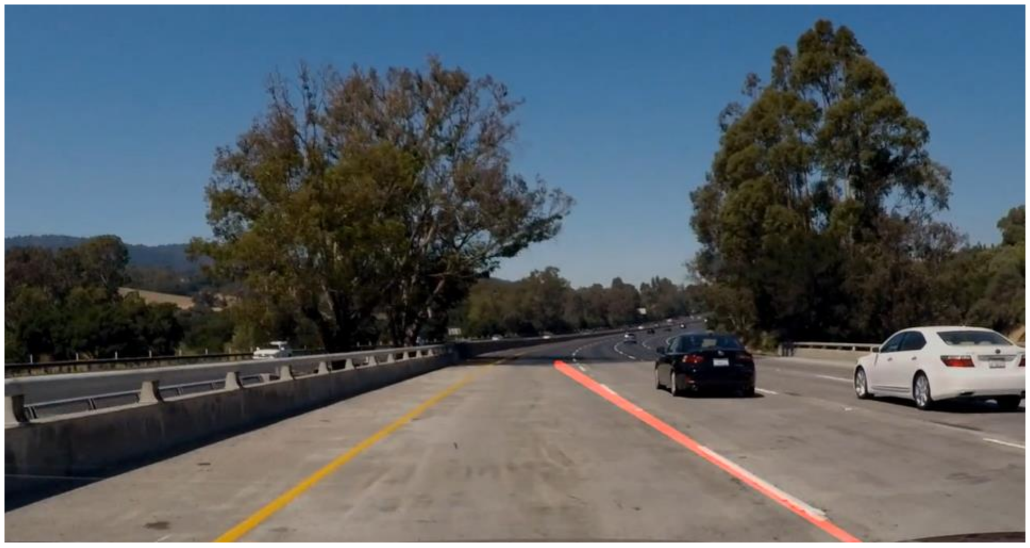
I did this separately for both the lines. I differentiated between the two lanes by their slope. Left lane must have a positive slope (slope>0) and right lane must be the negative sloped one (slope<0). However I also applied an arbitrary slope value to filter some unwanted slopes and tweaked it for better results. This reduced the flicker to a great extent.

I read about feedback loop on the net for getting data from past frames to apply an overall running average, but not able to apply it so far.

## Shortcomings:

1. I realized that the flicker is still visible although reduced to a large extent.
2. At some bright spots in the image the pipeline is not able to find the lanes. I believe the issue here is the smaller contrast between the adjacent pixels which is why the edge is not being detected. At one point of time the left lane was not detected by the pipeline at all.



3. I believe that getting a single line although reduced noise and flicker in the detected lane line, it might completely fail on turns and bends on the road, where lanes are not completely straight.
4. Also if lanes are not visible at all due to being too bright(over exposure) or dark light(under exposed), pipeline will fail to detect the road.

## Possible improvements to the pipeline:

1. Feedback (back propagation loop) can be implemented to get the previous frames data for averaging thresholds.
2. Also if it can be done, there should be some way to predict missing pixel values. Like at times there is some intense light which lightens up a certain spot, that spot should be interpreted with a predicted value.
3. In this scenario there was nothing hindering the vision of the car. It was able to view the lanes ahead of it and hence able to contemplate the route. However, in high traffic scenario car will not be able to see the lanes. So there should be some method better than just image processing to detect lanes or the movable area.