# Finding Lane Lines on the Road

---

**Finding Lane Lines on the Road**

The goals / steps of this project are the following:

- Make a pipeline that finds lane lines on the road
- Reflect on your work in a written report

---

## Reflection

### 1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consists of 5 steps. First, I convert the images to grayscale so that the image has a single color channel. Next, I apply gaussian blur with a kernal size of 9, which was determined from trial and error such that the resulting image is not overly blurred but general image noise is removed. The Canny edge is next used to determine the regions of largest gradient change within the image which ideally correspond to the general locations of lane lines (or other sharp changes). Canny edge detection is run with a low threshold value of 60, and a high threshold value of 130, which was determined from trial and error in order to detect edges as reliably as possible given the various test images. Line detection is then performed using Hough space transformation with the following parameters:

rho: 1 pixel resolution

theta = pi/180 pixel resolution

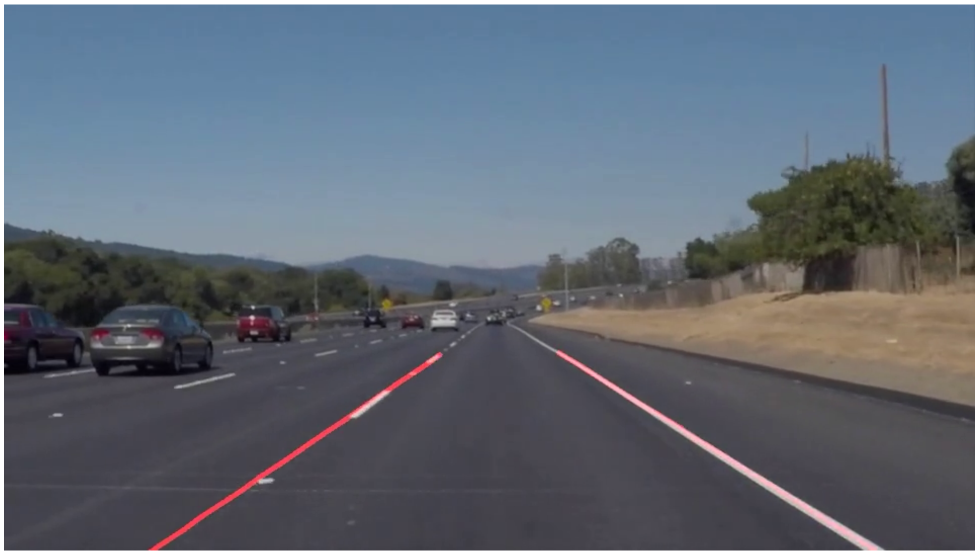
threshold = 12 intersections minimum for line recognition

min_line_len = 8 pixel length minimum for line recognition

max_line_gap = 8 pixel gap maximum for line recognition

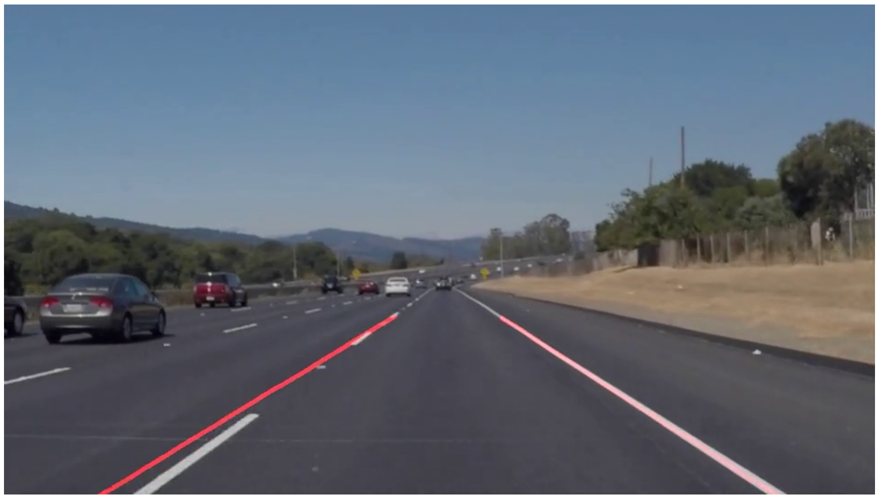
Finally, the resulting image is weighted and overlayed on top of the original image such that the detected lanes can be viewed.

In order to draw a single line on the left and right lanes, I modified the draw_lines() function to include linear best fit approximation of all line start and end points detected via Hough space transformation, which extrapolates the detected lines.  Detected lines are separated based on positive and negative slopes in order to designate left and right lane lines.  Additionally, lines which do not have a slope within a specific range of [0.55 0.85] for positive slopes or [-0.85 -0.55] for negative slopes are removed from consideration, as these ranges should encompass a reasonable lane line slope.

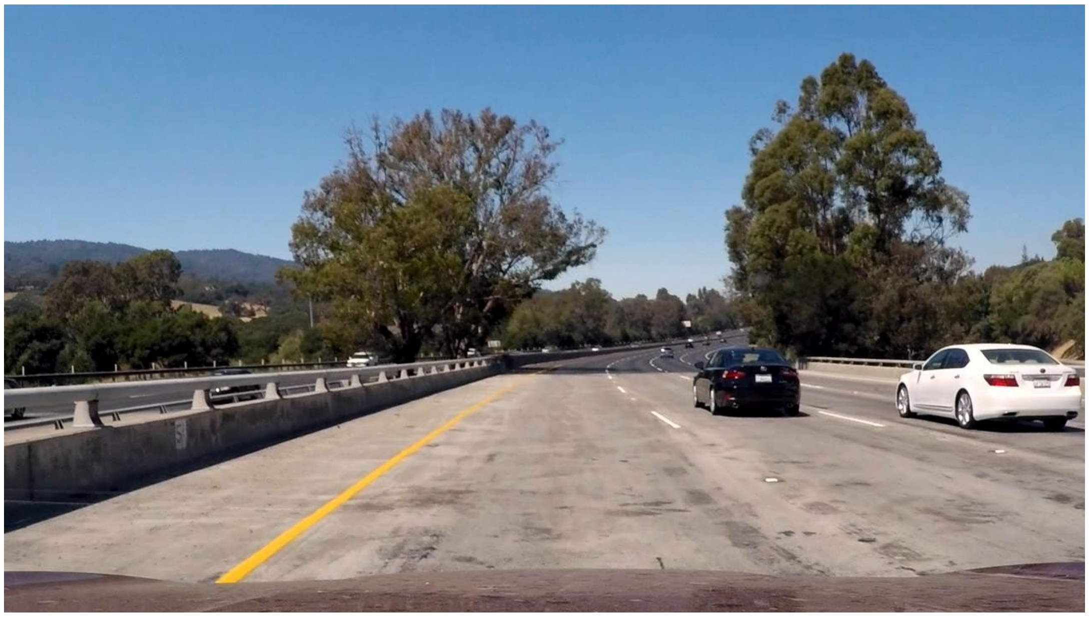


## 2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be if lines that are detected by Hough transformation are not actual lane lines, the resulting points will still be included into the best fit approximation.  This can cause the lane line extrapolation to be skewed slightly from the actual lane line.



Another shortcoming is that in regions of the road or where debris/dirt may closely resemble the color of the lane line, line detection may break down due to the canny edge detection algorithm not being able to distinguish sharp gradients. An example of this could be seen in the challenge.mp4 during the section of the video in which the road is a lighter color (no lanes could be determined reliably)



## 3. Suggest possible improvements to your pipeline

A possible improvement I noticed would be to first distinguish a range of acceptable lane line colors (yellow/white) within a certain range of 3 color channels, and then color all other pixels within the background as black.  I implemented this improvement into the lane detection algorithm, such that lane pixels colored yellow within an RGB range of [180, 160, 70] to [245, 220, 130] or lane pixels colored white within an RGB range of [235, 235, 235] to [255, 255, 255] were specifically isolated from the remainder of the image.  This change improved lane detection during the challenge.mp4 video, as well as other instances in which the lane detection position would appear erratic from the actual lane location, effectively eliminating a large number of false positive Hough lines which would have otherwise been identified.

Jesse Turner

Project 1

Self Driving Car Engineer Term 1