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 the work in a written report

Reflection

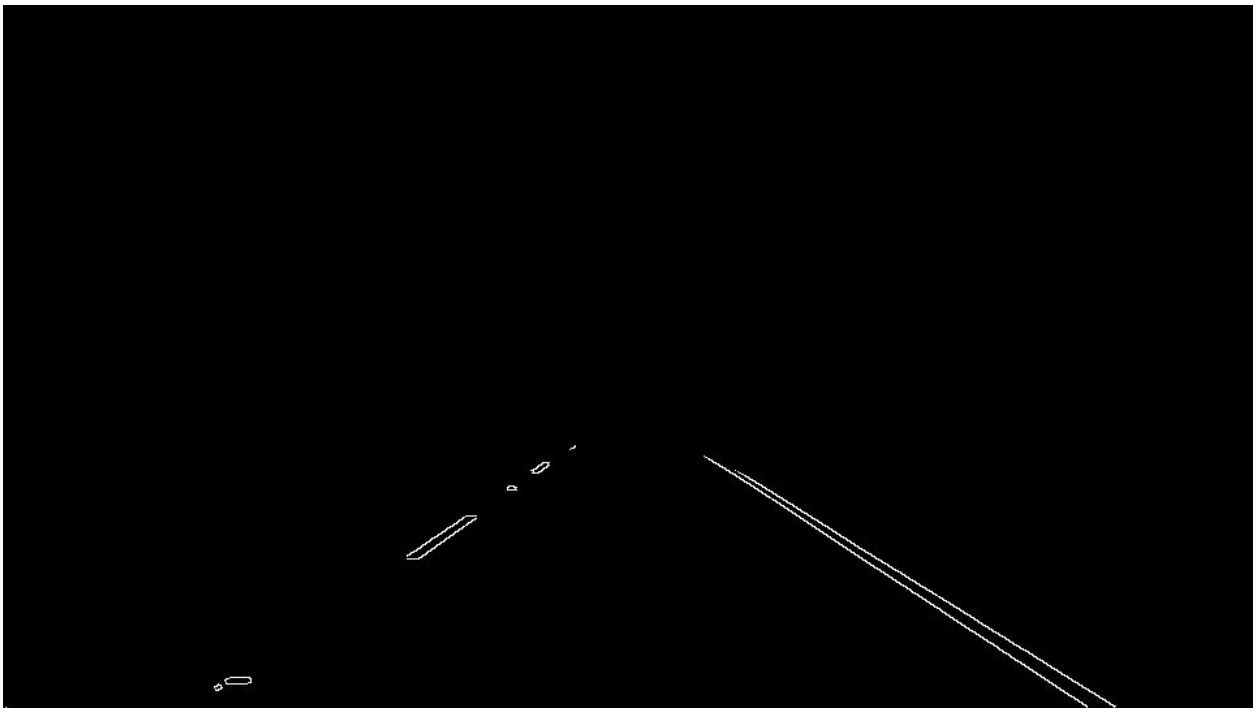
1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My initial pipeline consisted of 5 steps (When used without color range enhancement step in final solution).

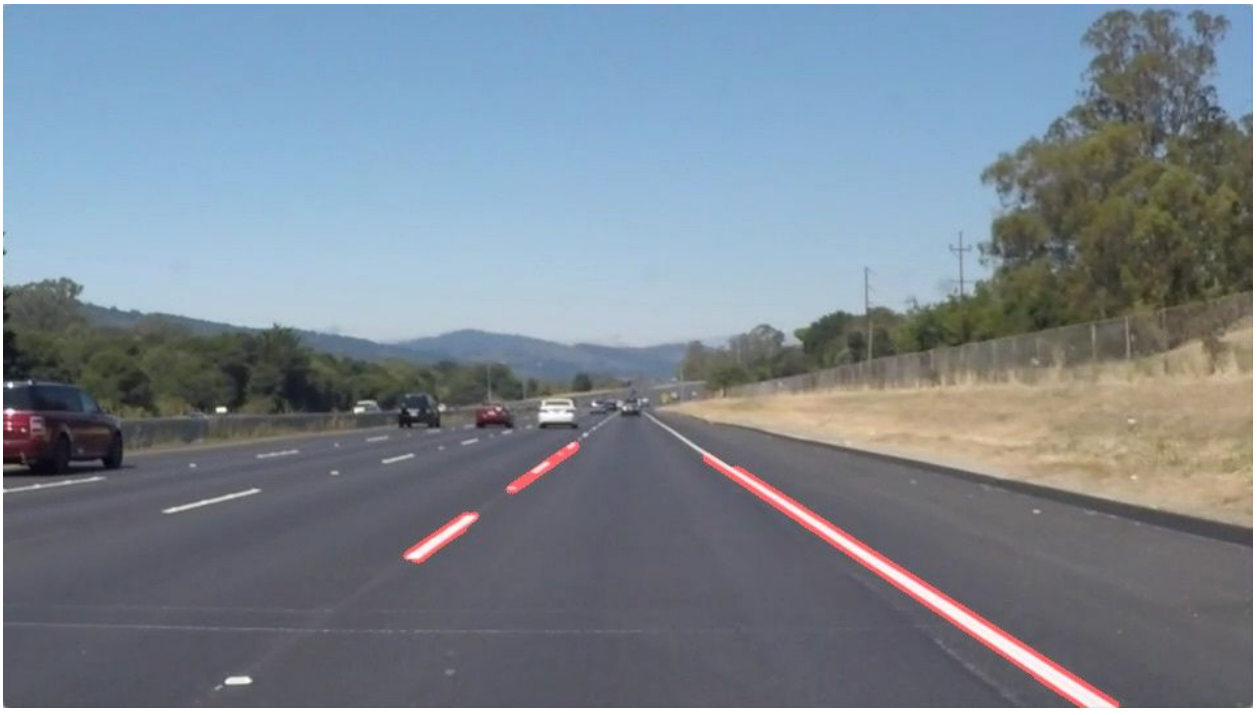
First, I converted the images to grayscale and I applied Gaussian blur to the grayscale image. (`kernel_size = 5`)

Then I applied canny edge detection to detect the edges of the image. (low_threshold = 50, high_threshold = 150)

Then I used a polygon mask to filter out non relevant portion of the image.



Next I applied Hough transform to detect the line segments of image. (rho = 2, theta = pi/180, threshold = 15, min_line_length = 35, max_line_gap = 20)



In order to draw a single line on the left and right lanes, I modified the draw_lines() function by first calling a function to separate the line segments within specified

absolute minimum and maximum slopes (min_line_slope = 0.5, max_line_slope = 0.8) into left and right buckets according to their position and slope.

Then I calculated the weighted average of extrapolated points of these set of lines between top and bottom of polygon region that was used to filter the edges, and used these values to draw left and right lane line on the image.

Note: Here for the averaging, I used extrapolated x position of lines top and bottom edge within the region mask top and bottom, instead of using middle position of lines and slope. One reason is if you need to calculate the average slope you need to use angle instead of slope since slope is non proportional through the direction of lane lines. Also it is easier to calculate weighted average when you use extrapolated x coordinates instead middle point and angle pairs.

Furthermore it's worth noticing that I have put the pipeline into a class so that it's easier to tweak parameters of the pipeline steps from a single entry point. This also provides ability to turn on or off extrapolating the lines, to produce images with raw lines drawn or to produce solid lines over the images.

Changes required for challenge video.

Above pipeline produced fairly good results with solidWhiteRight.mp4 and solidYellowLeft.mp4 with moderate but acceptable amount of jitter on lanes with dashed lines. But it's performance on challenge.mp4 was poor with higher amount of jitter and occasionally failing to detect lines during road segments with lighter road color.

I did following changes to the pipeline to make it more robust with challenge.mp4 video

- Added a color enhancement step to the pipeline that reduces the color values to a factor of 0.8, of ranges falling out side the specified ranges for the lane lines in HSV color space. This will enhance the contrast between road and lane lines on lighter areas as well as reduce the contrast of shadows on the road. Here I have used HSV color space instead RGB color space for the ease of defining ranges for a particular color. Also I choose to enhance the colors instead of filtering out since it's harder to specify color boundaries for a wide range of illumination conditions without introducing unwanted regions of road surface on the image.
- Increased Gaussian `kernel_size` to 7 and Hough `rho` to 3 and used a `min_line_length` of 25.

These changes produced a fairly acceptable result with moderate jitter and a couple of brief lapses in detected lane lines due to dashed lane being little bit sparse with shorter lane segments than other two videos. It should be possible to smooth the result further with a temporal smoothing approach.

2. Identify potential shortcomings with your current pipeline

One potential shortcoming in the current pipeline is that it only takes average of line segments of each side as a linear line. This will usually represent rough approximation of tangent of the lines during sharp curves on the road and do not

represent the curved nature of the road, which is important for functions like determining target steering angle for functions like Lane Keeping Assist. Furthermore curved lines tend to be filtered out due to extreme slope angles.

Another shortcoming would be what would happen when there is lot of vehicles in the road like bellow in city driving conditions. Also additional markings on the road like pedestrian crossings etc. would also cause problems.

3. Suggest possible improvements to your pipeline

A possible improvement would be to use a polygon fit on either detected line segments or on the pixel coordinates of the individual pixels that belonging to the color range of lane lines. Furthermore we could dynamically set the `min_line_slope`and `max_line_slope` and the `vertices` for region mask based on the parameters of the last detected lane_lines.

Another potential improvement could be to combine the lane detector with a vehicle detector to filter out bounding boxes of the vehicles from edge image during the region masking. We can extend the same to remove additional markings like pedestrian crossings etc.

It is also worth exploring the possibility of temporal smoothing the lane lines to remove high frequency jitter and this would also help out in cases where Hough transform fails to detect the lane lines for brief periods.