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 consisted of 5 steps. First, I converted the images to grayscale, then I use the Gaussian blur function with kernel size 3 to remove the noise of the picture, then I use the canny edge method to detect the edges of the blur image, to restrict the region of the detect area, the region of interest method is applied to find the polygon region, where the lanes lie in, after we have the image of the region of the interest, we need to connect the the edges inside to draw lines, then the inside the hough_line function the cv2.HoughLinesP method is to find the connected the lines first, then by calling the draw_line function, we add the line inside lines into the blank image, after we have the hough_line image after transformation, we call the weighed image to add the hough_line image back to the original image with correspondence weight.

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by separate the lines to left_lanes[] and right_lanes[], then I calculate the slope for all the lines and put them into the correspond lanes, after I have the collected slops for both left and right lane, then I select all the points for each lane and make the first order of the polyfit to get the estimate coefficients, by using the boundary vertices to get the minimum and maximum points of the line for each lane. In the end, I draw the line for each lane respect to the vertices and add back to the original image.

If you'd like to include images to show how the pipeline works, here is how to include an image:

Below is the output picture before modified the draw_line function:

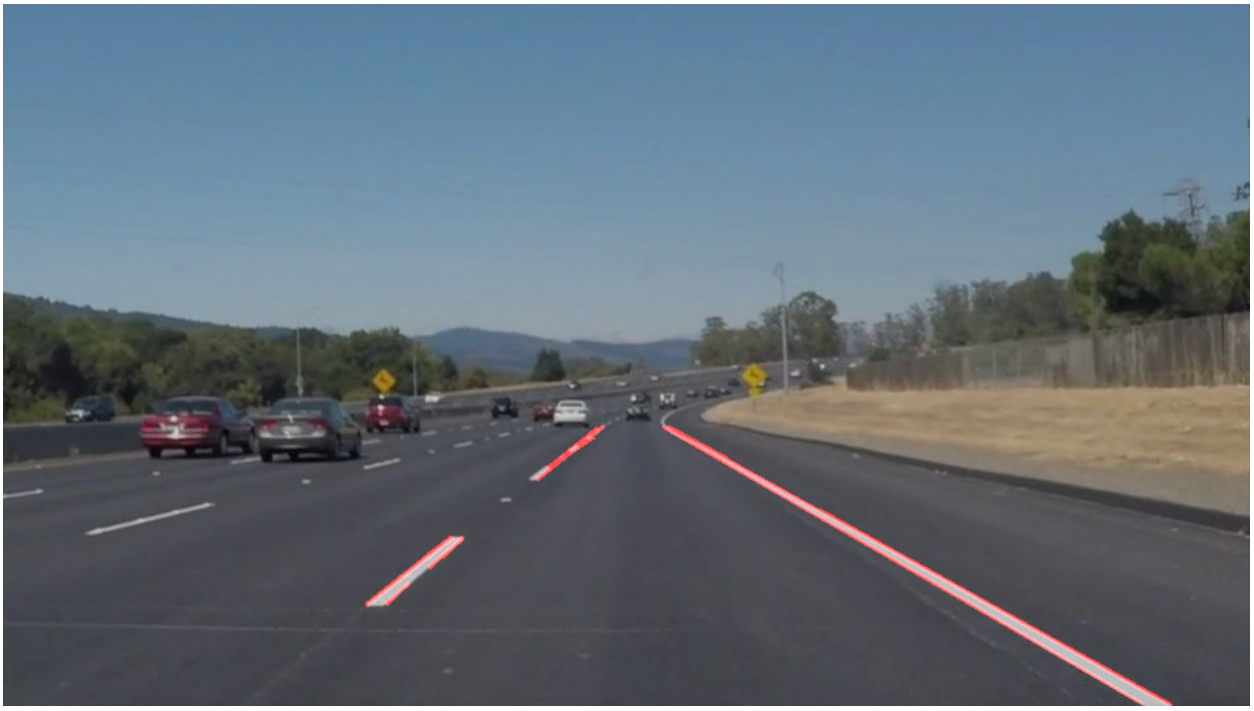

Figure: solidwhitecurve

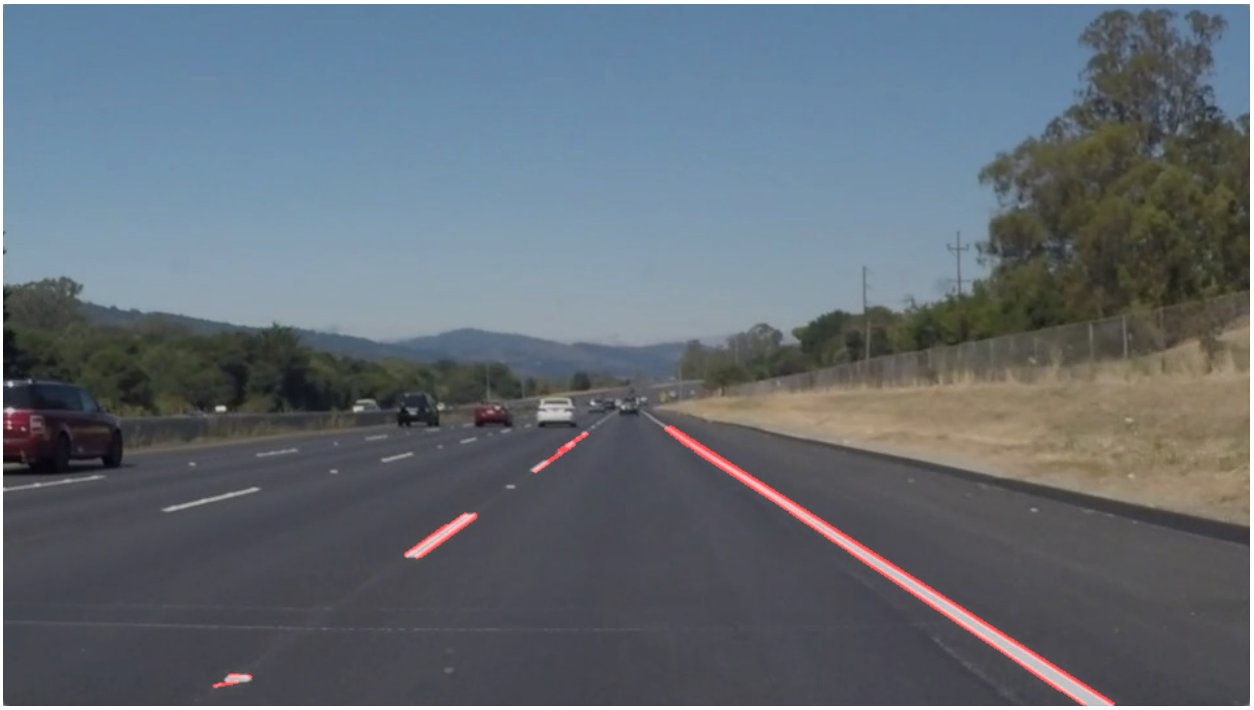

Figure: solidWhiteRight

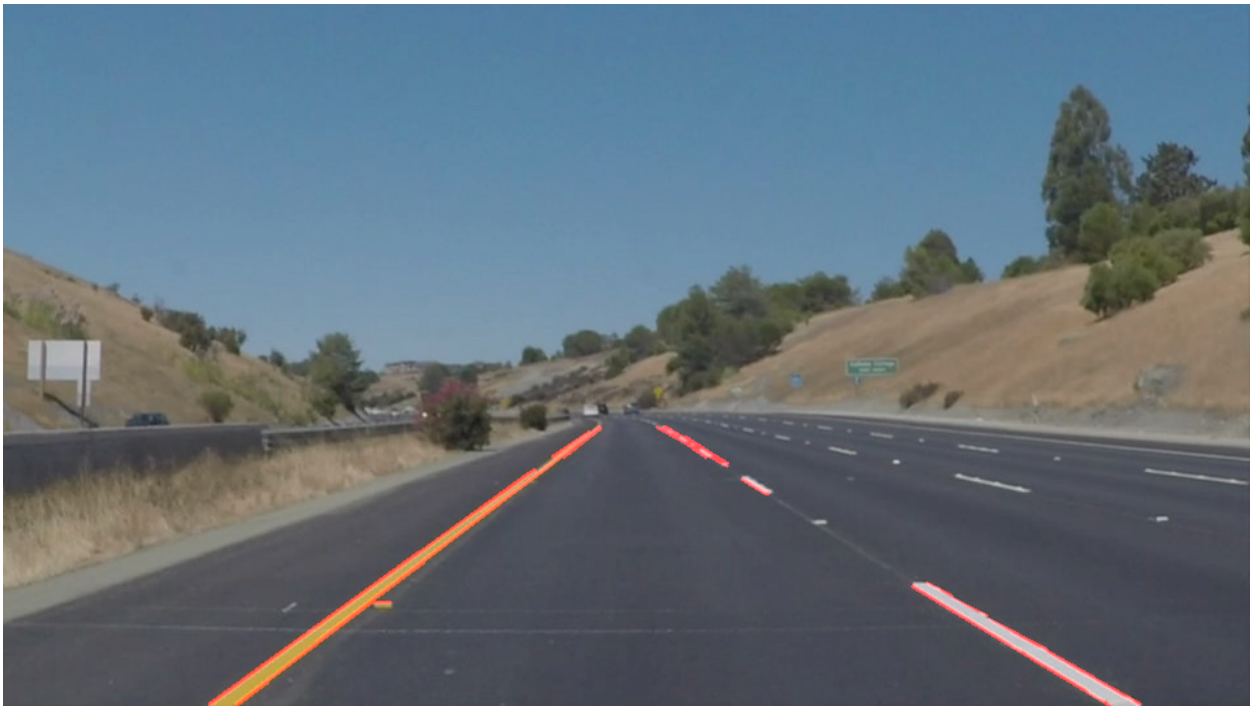

Figure: solidYellowCurve2

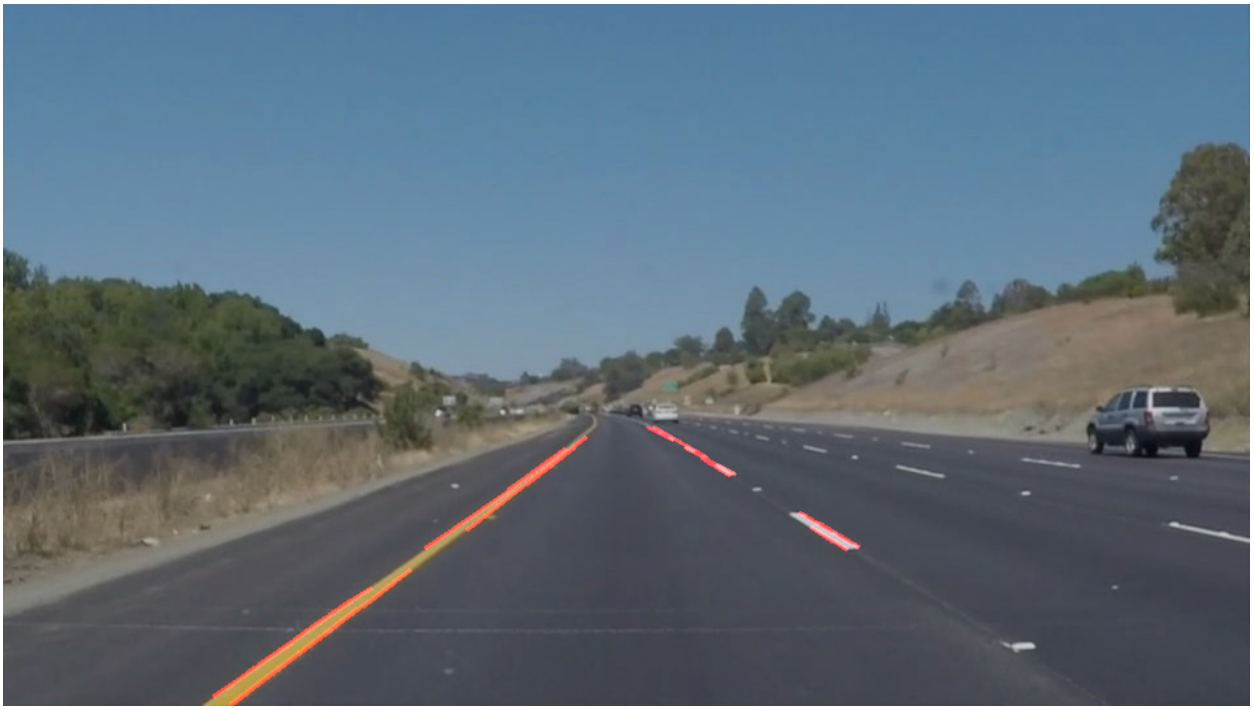

Figure: solidYellowCurve

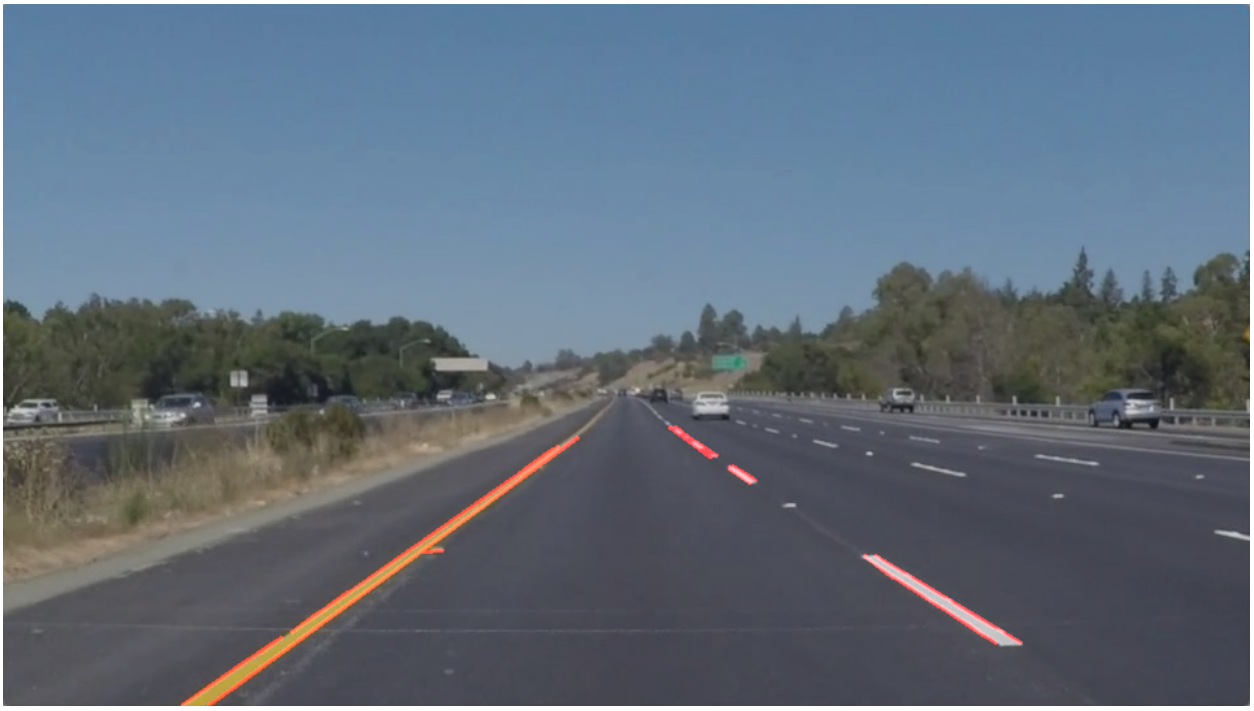

Fig :solidYellowleft

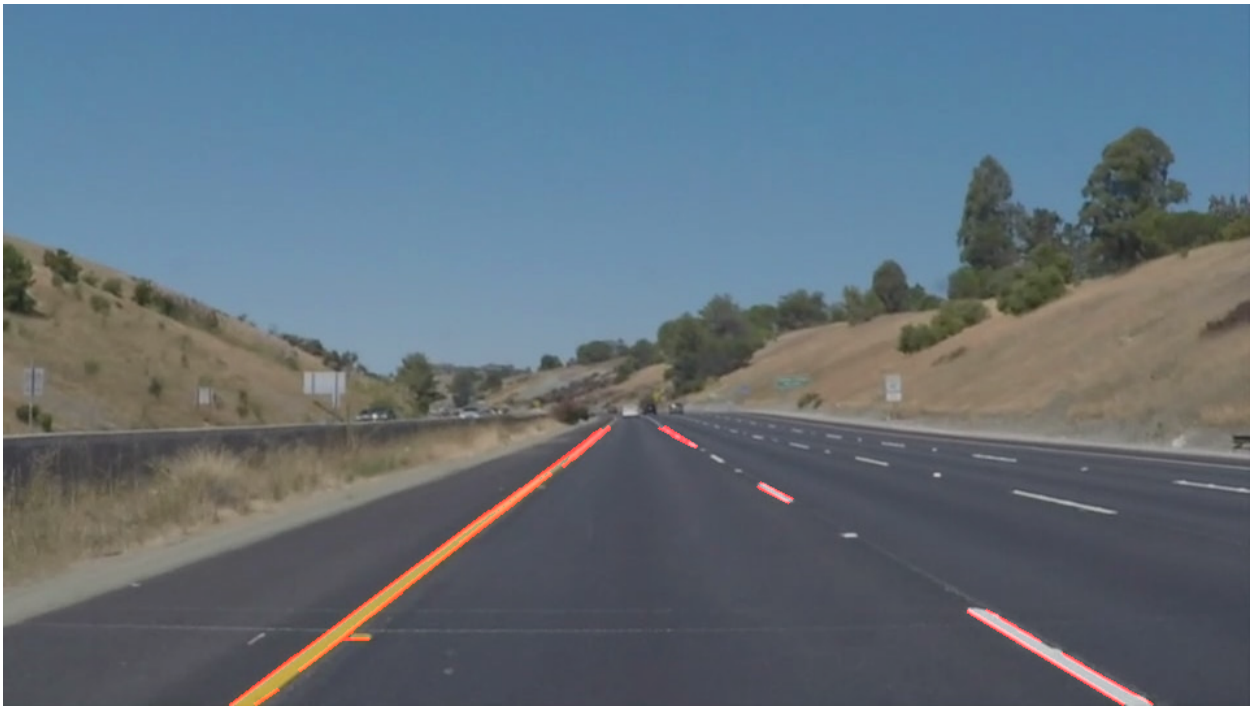

Fig: whitecarlaneswitch

After modified the draw_line function, the result is shown:

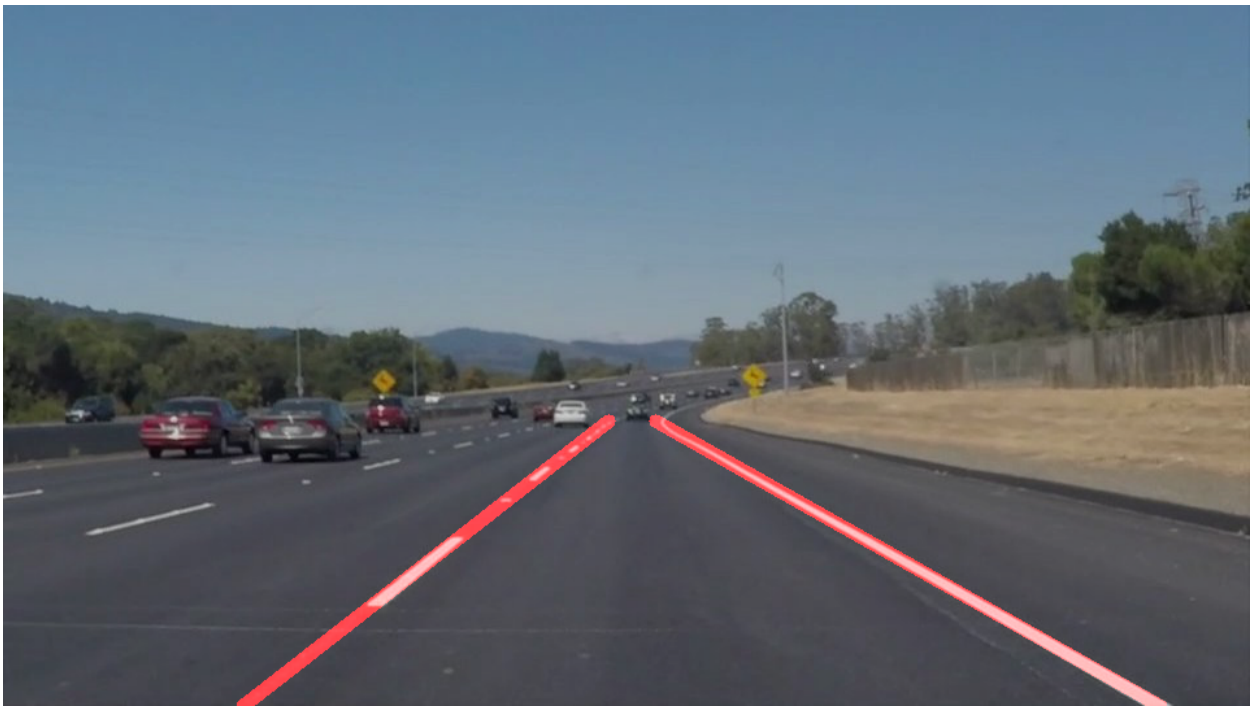

Fig: solidwhitecurve

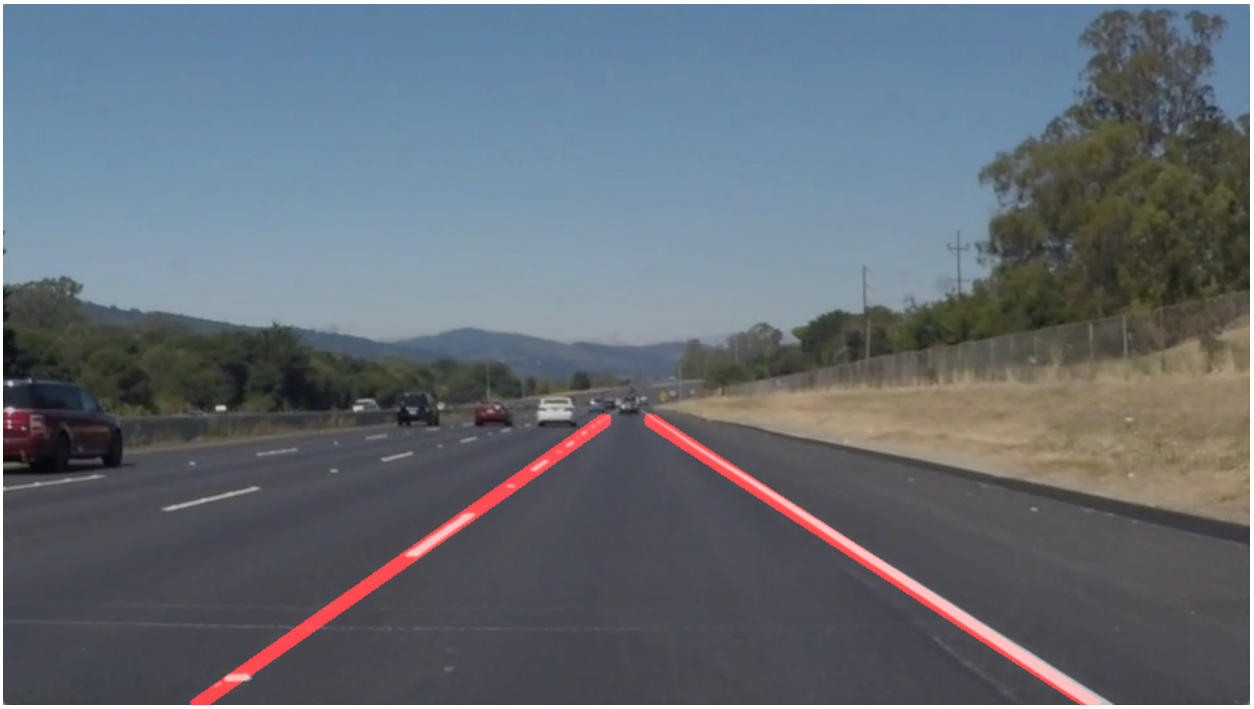

Figure: solidWhiteRight

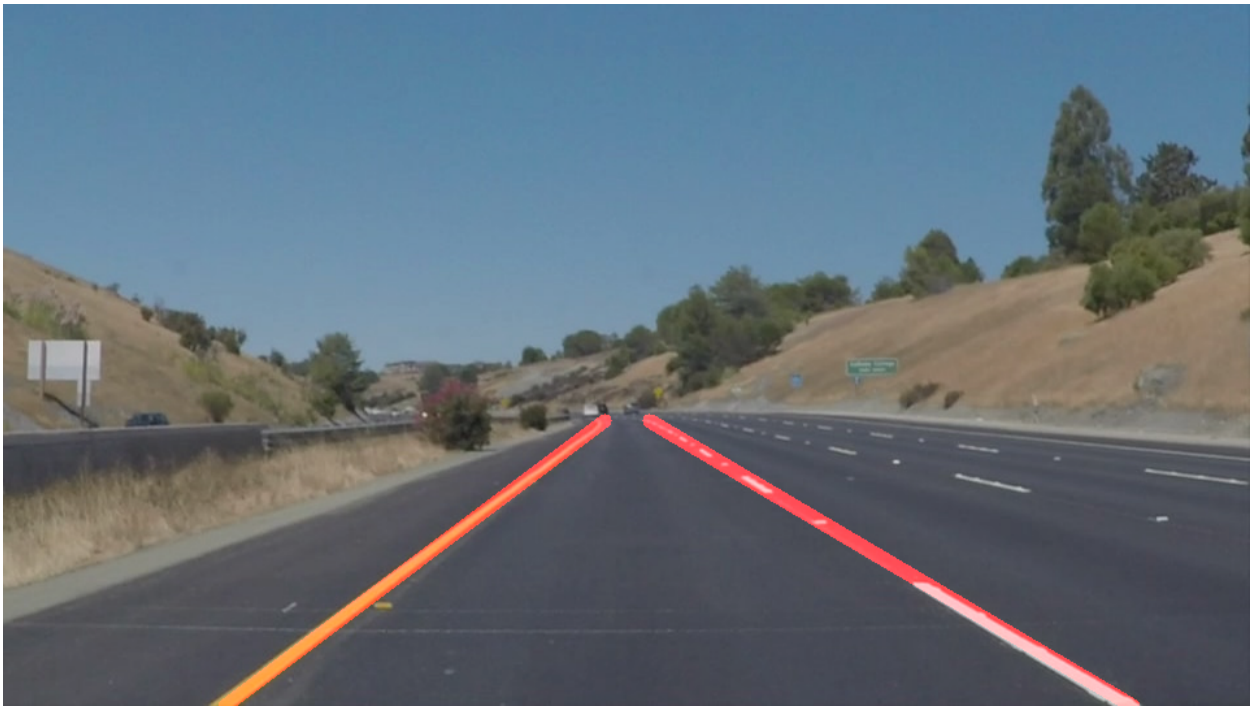

Figure: solidYellowCurve2

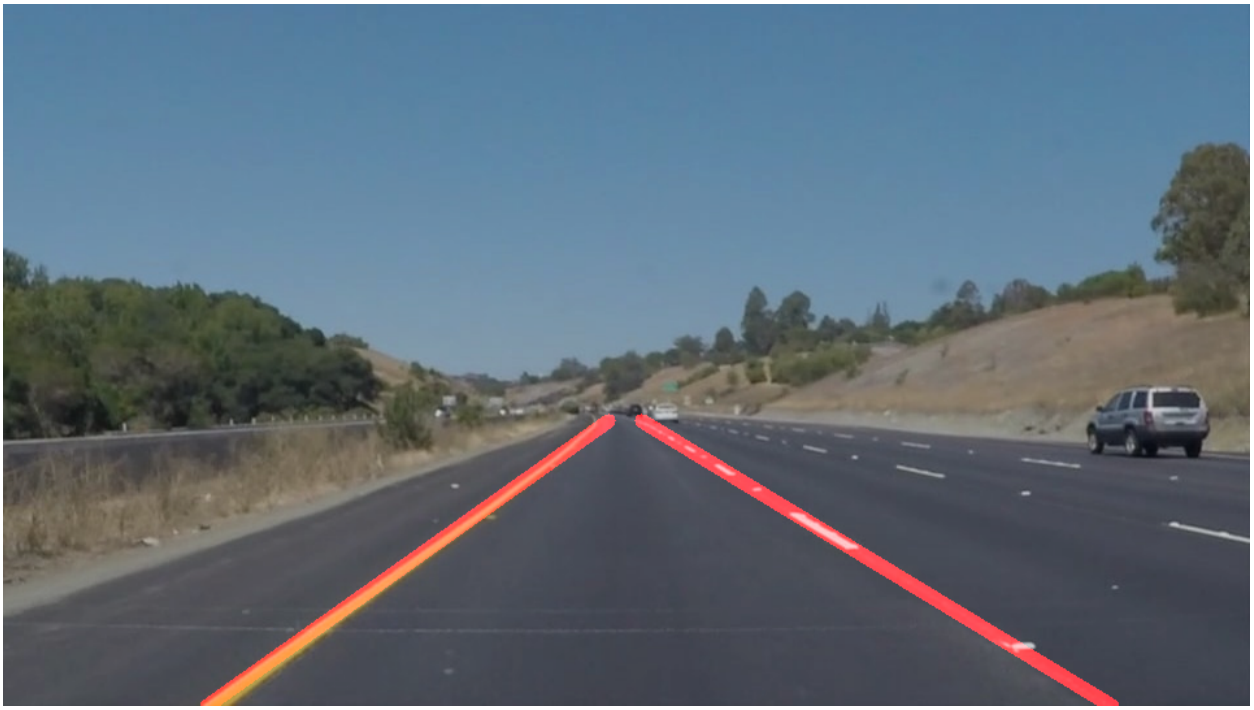

Figure: solidYellowCurve

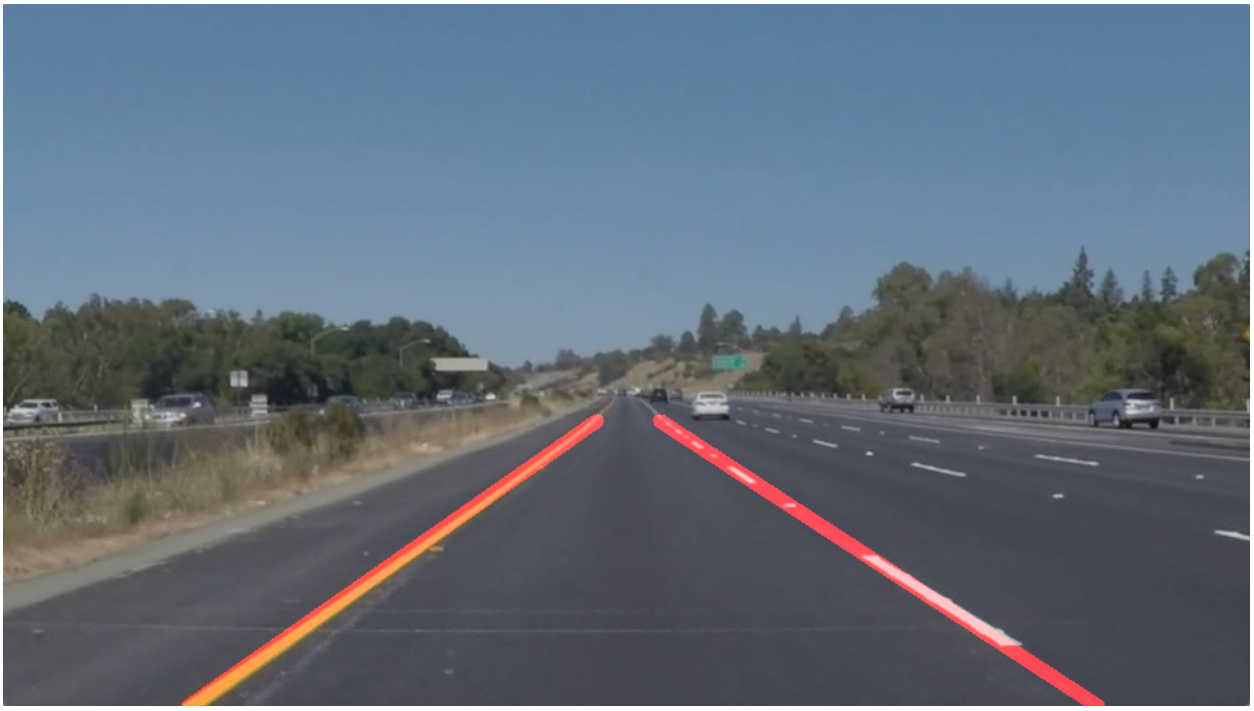

Figure :solidYellowleft

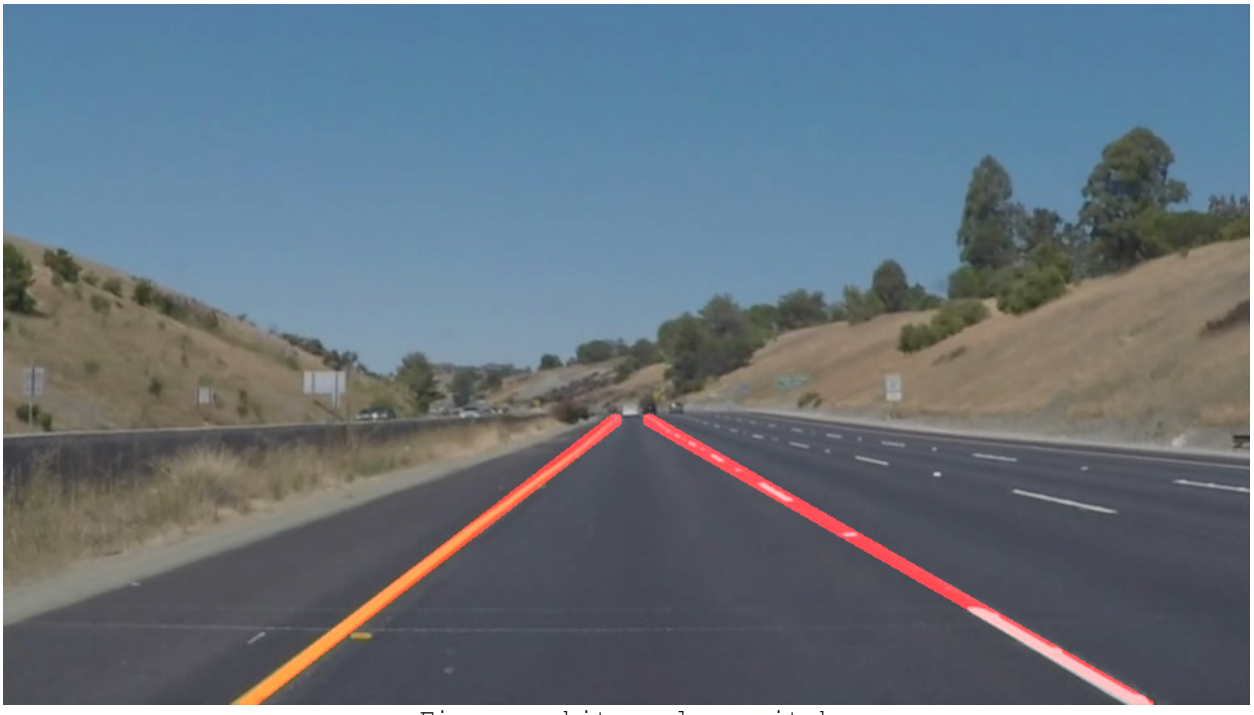

Figure: whitecarlaneswitch

2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be what would happen when the curvature of line is large, the straight hough_line method won't work, because the poly fit function is fist order.

Also, the region of the interest is another issue when the top vertices of the region is close to the boundary the identification may cause wrong identification for unexpected car switching lane, and if the top vertices are far away from the top boundary, which can't cover the majority of the region.

3. Suggest possible improvements to your pipeline

A possible improvement would be to change the hough_line and draw_line method, to extract point in these lines and try to make ployfit in multidimensional, so when the curvature of the line is constantly changings, it can adjust really quick accordingly, also we can improve the region of the interest function, to make the region more robust, for the last video challenge, my draw_line code doesn't work, which I think it is because of the straight line function, and I currently working on it with a better solution to fit the line with multi-dimension and more robust.