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.

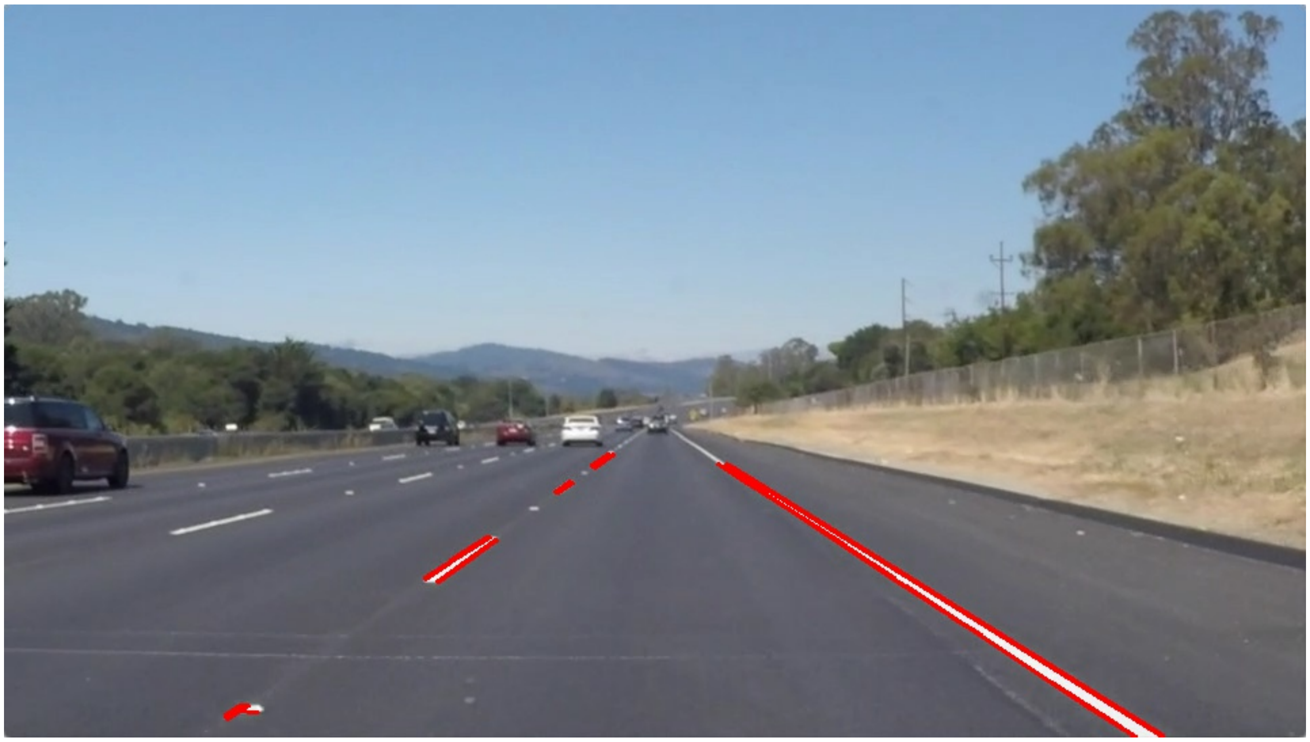
My pipeline consisted of the steps we took in the videos leading up to the project.

First, the image is converted to grayscale

Next, a Gaussian Blur is applied to the image

Then we apply the Canny filter and a mask to select our region of interest

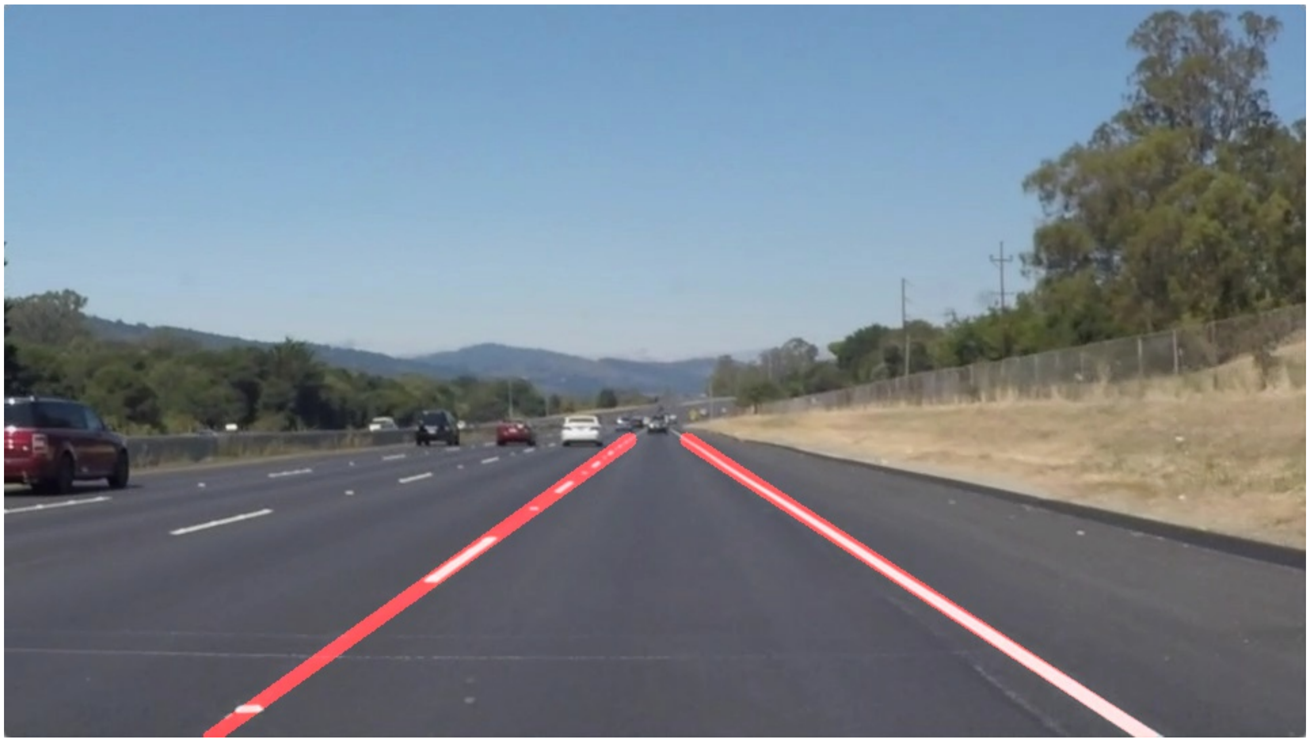
We then finally apply the Hough Transform, which includes the draw lines function.



The Hough Lines are passed to the draw_lines function which draws lines like we saw above.

To achieve the two distinct lines as seen in the video, I had to modify the `draw_lines` function. First the slopes of the hough lines are checked. We see that the two lines that define the lane are in opposite angles, and we see that since slope = tan (angle made by the line), the slope for one line is positive and the other line is negative. This helps us differentiate the lines. Since a lot of this project is based on assumptions, I took another step and assumed the lane ahead is in the center of the camera, which means that the slopes of both lines will always be in a certain range, and not just positive or negative. So I added a slope margin. This helps us when working with the Solid Yellow case.

The points of lines with satisfacrory slopes from hough transform are then sent to the `draw_line` function, which takes these coordinates and fits them to a linear curve `y = px + v` using `np.polyfit`. We can now draw the lines from all of this data.



2. Identify potential shortcomings with your current pipeline

The main shortcomings would be the extreme assumptions we made:

- The lane followed is straight (That why the first degree polyfit)
- The lane is in the center of the car's view. (This is why we could take select the region of interest manually.)
- The brightness of the road is consistent (We haven't seen many cases of different shadows or heavy traffic affecting our algorithm)

3. Suggest possible improvements to your pipeline

The shortcomings mentioned above were encountered in the challenge section. The brightness of the road changes and the lane is not straight, it is a constant turn. The algorithm, if applied directly as we did before, would result in two intersecting lines. If we want to stick to marking straight lines, we can get better results by increasing the `y_offset` parameter in the `draw_line` function, but that would just mean marking lesser of the lane. What we could also do would be to assume that given the distance of perception, we can assume that the car can see at most one bend in the lane, which means it's a second degree curve. I tried creating a function `lane_pipeline2` which just calls the hough function with degree set to 2. The implementation could be fine tuned much better, but I still included the result from the second degree curve fitting. Technically, the first degree implementation had a more consistent performance while the second degree had moments of higher accuracy performance.