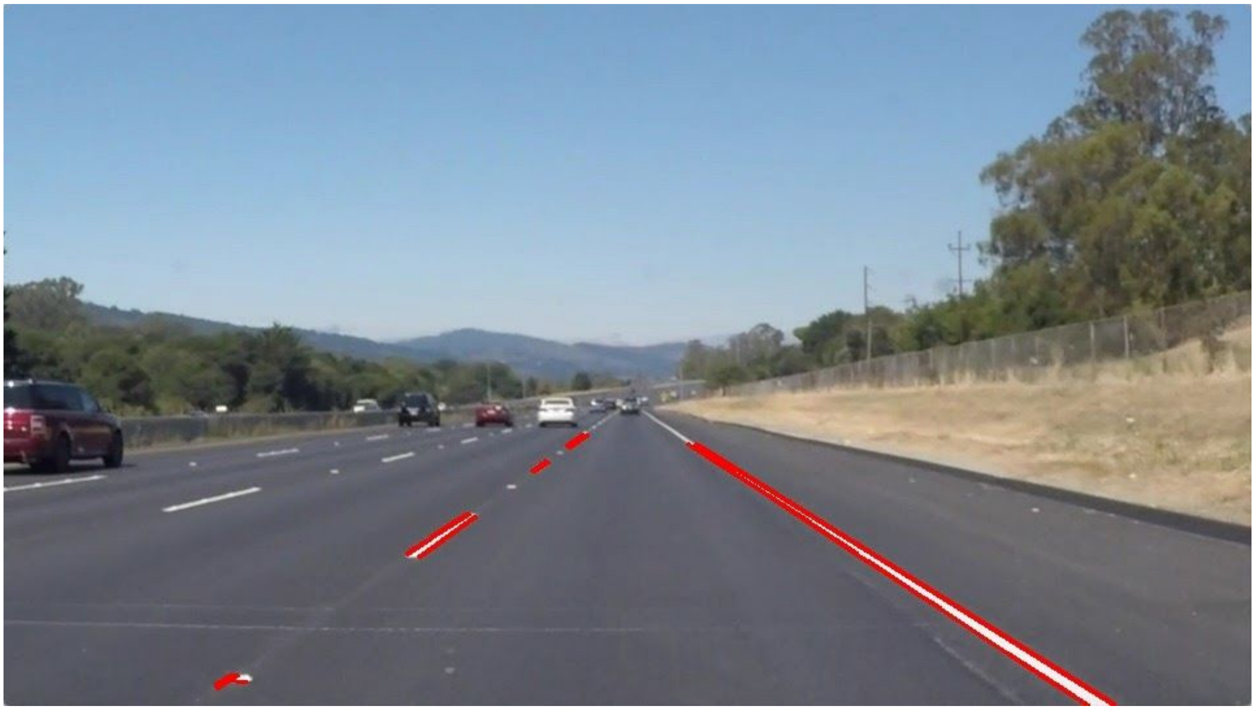
Pipeline Overview:

My project consisted of the following pipeline:

1. Convert image to grayscale.
2. Apply gaussian blur to remove unnecessary noise.
3. Apply Canny edge filtering. After some experimentation , I kept my two parameters values at 50 and 150 respectively.
4. Get the ROI.
5. Run hough lines algorithm (rho: 1, theta: pi/180, threshold: 50, min linelength: 2, maxlinegap: 200)
6. Draw lines.

Drawing continuous lines:



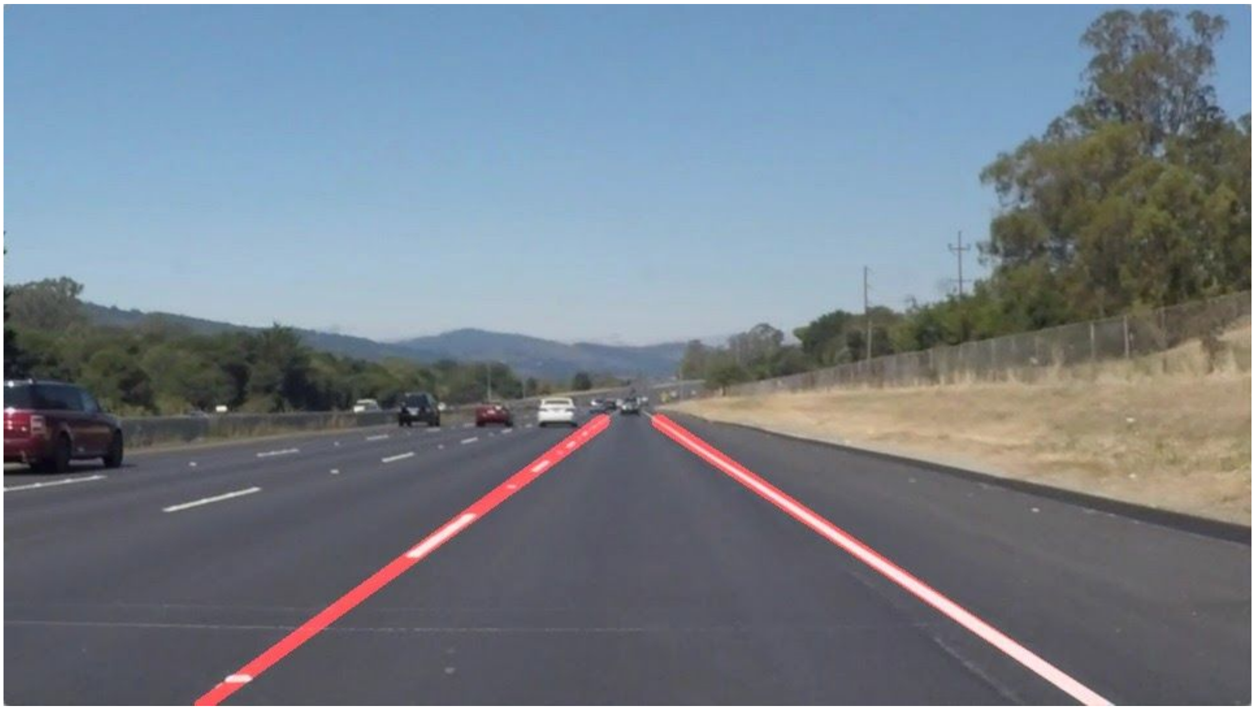
Initially the lines looked like this(basic line drawing code provided in the notebook).

After trying things like increasing max line gap property and such, I decided to do slope based extrapolation, as suggested in the notebook. My algorithm is as follows:

```
For each line in line_array:
       Draw line
       Determine slope
       If slope>0
            left_line_slope=slope
            left_intercept=calculated intercept
            Is the point closer to the bottom?
            If so, left_p=point
       If slope<0
           right_line_slope=slope
           right_intercept=calculated intercept
           Is the point closer to the bottom?
           If so, right_p=point

    Determine the bottom coordinates for left and right line
    Draw left and right line
```

Resulting image would be like this:



Shortcomings:

1. ROI is constant, which may not be good for all cases.
2. It depends on Canny Edge detection and such which are heavily dependent on lighting conditions.
3. Sometimes the slope might be slightly incorrect, resulting in a not so straight line.
4. It does not work well for curved lines. It failed to detect the right lane in the challenge video. Possible reason could be ROI or curve.
5. If the lanes are both curved with positive or negative slope, extrapolation will fail.

Possible Improvements:

1. A slightly sliding ROI which would search for the other line in case there is only one line.
2. Using Advanced deep learning based edge/lane detection.