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.

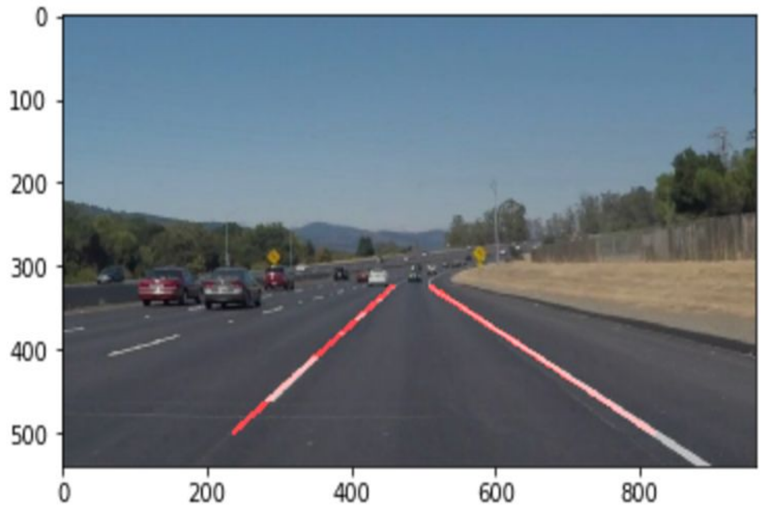
Following are the steps followed in my pipeline fn() 'process_image_for_lane_detection()' which receives an **input image**:

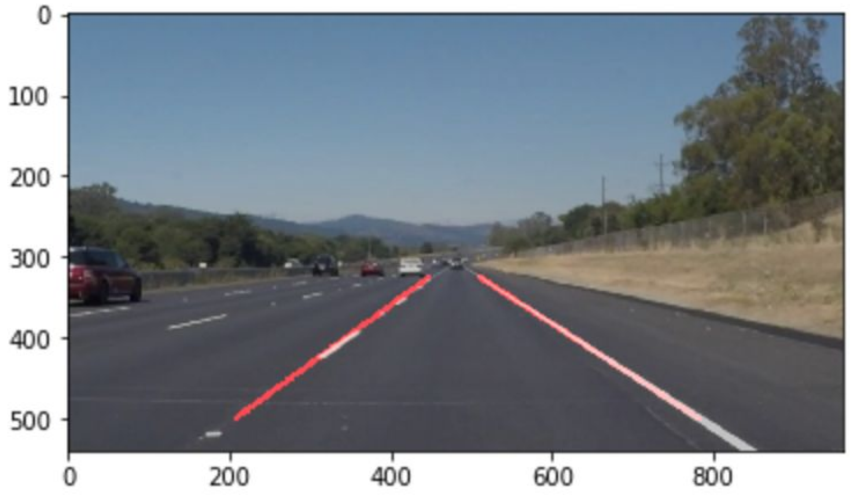
A. Converted the **input image** to **gray scale** image.
B. Applied *Gaussian smoothing* on the **input image** from step A.
C. Applied *Canny Edge Detection* on image from Step B.
D. Extracted *region of interest* from Step C image.
E. Applied *Hough Transform* on Image from Step D.
F. Applied *weighted image* on the : (1). input image and the Hough lines got from step 5.

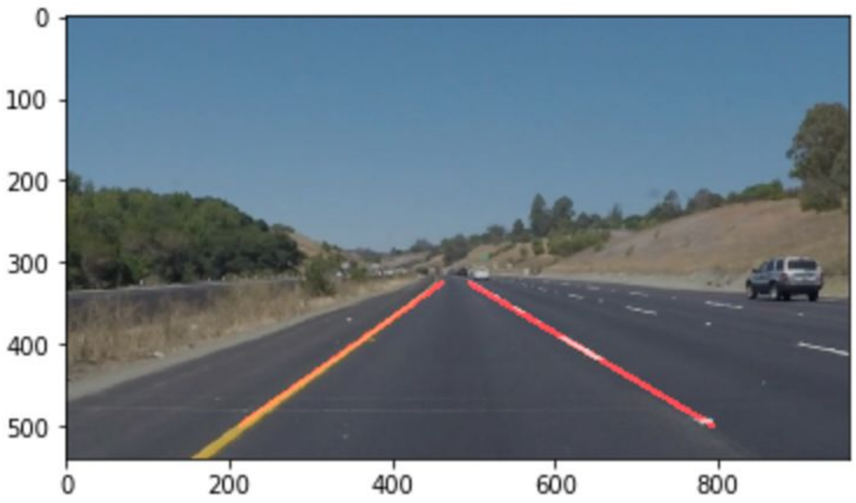
I modified the **draw_lines()** function as follows:

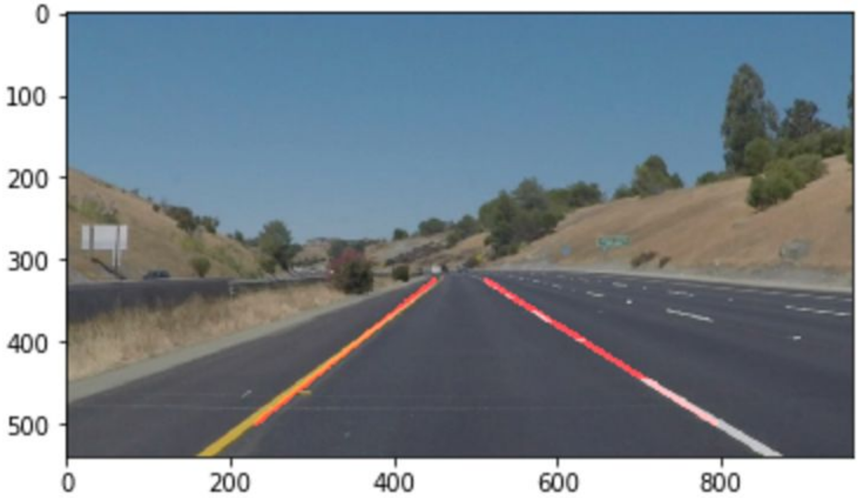
Iterated over all the lines passed and did the following on each line [x1,y1,x2,y2]:
- Calculated the slope and the centroid for [x1, x2] and [y1 and y2]
- Treated each line as either a part of left line or right line, depending on it slope value.
    * For the **left** line, I accepted the lines which had slope > 0.5
    * For the **right** line, I accepted the line points with slope < - 0.5
    * All other points were left, taken as outliers for data cleansing.
- All the left points in the slope, x and y list, I averaged them out for smoothing purposes and used them to draw the left and right lines for lane detection.

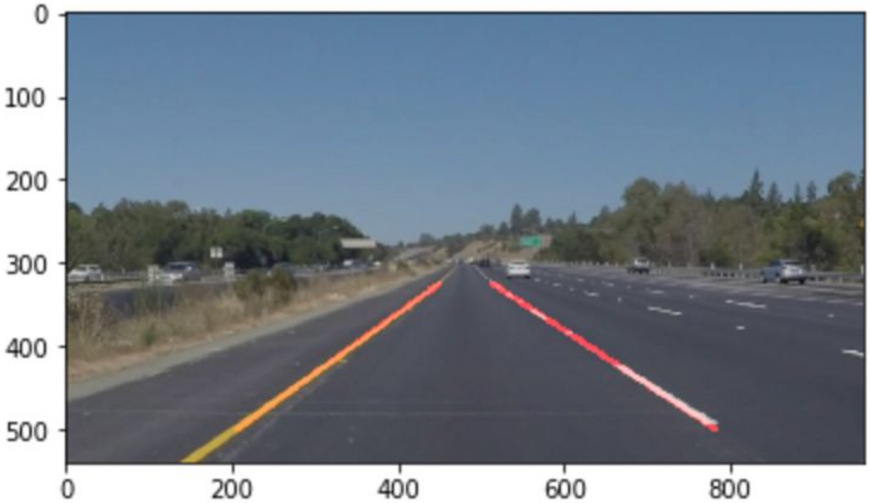
If you'd like to include images to show how the pipeline works, here is how to include an image:

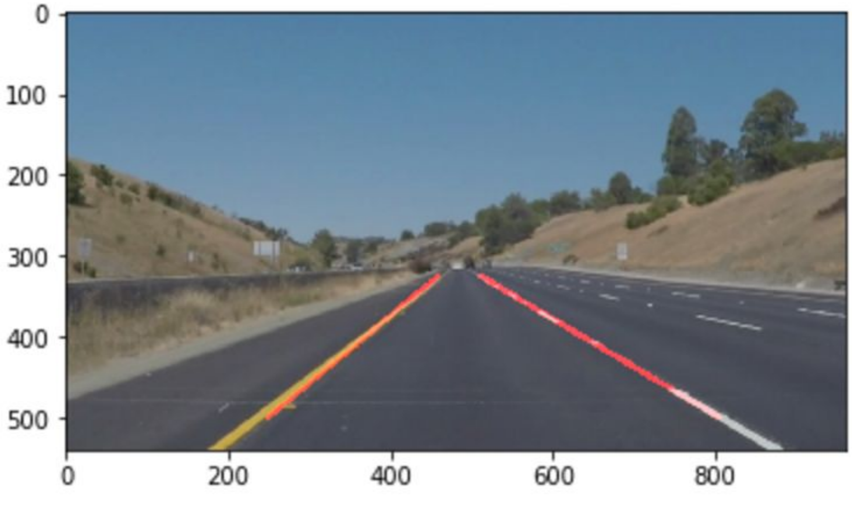

0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800

2. Identify potential shortcomings with your current pipeline

- In **draw_lines()** fn, as of now, I have used the slope cutoffs (0.5 and -0.5) and averaging as way of removing the noise. There is still noise, which is making the lines jittery and shaky.
- Hard coding for x,y points for the region of interest.
- Angle of vision and road curviness while taking the photo can make the lines detection unreliable.

3. Suggest possible improvements to your pipeline

Further instrumentation can be done to make them smooth and curvy, thus avoiding jitteriness.

- Inability to adapt based on the :

  (1). Angle at which image was taken for the road lines (slope)

  (2). Road curviness in picture, and

  (3). 'region of interest' based on image dimensions, so have better lane lines detection.

This way outliers lines/points can be reduced.