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.

Reflections

Description

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

The pipeline consisted of 5 steps. First read the image then converted it to the grayscale and applied Guassian Blur for Canny Edge Detection.

After canny edge Detection, I have applied region_of_interest function for masking, then applied hough lines by tweaking with the parameters of HoughLines which will be used to generate the line segments which are needed.

Code Snippets

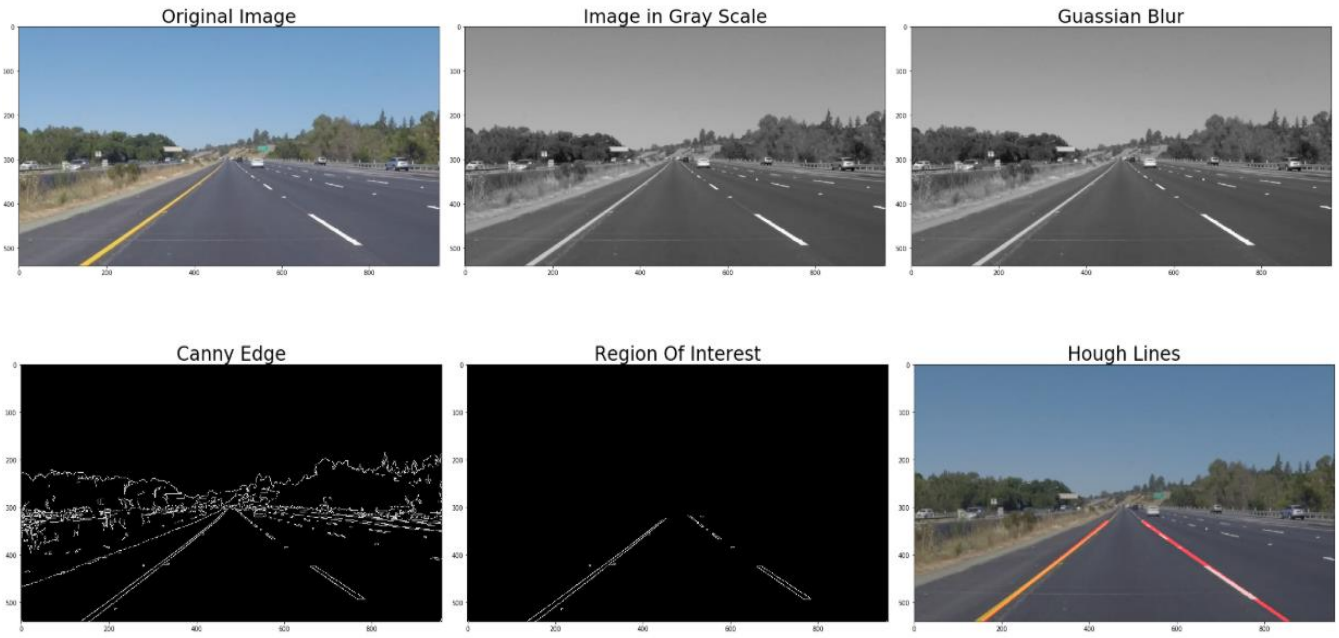
```
# Reading Image
image =  mpimg.imread("test_images/"+i)
initial_img = np.copy(image)
line_image = np.copy(image)*0

# Converting to gray scale
gray = grayscale(image)

# Applying Guassian Blur
blur = gaussian_blur(gray, 1)

# Defining Canny Edge
edges = canny(blur, low_thresh, high_thresh)

# Using Polygon for Masking
roi = region_of_interest(edges, vertices)
```



Now to join all the line segments, I have made few changes in draw_line() function they are:

- Separated the Right side(both line and slope) and left side(both line and slope).
- Then averaged Right Side and Left Side (for both line and slope).
- Then extrapolated to our mask boundaries for both the sides.

Code Snippet for extrapolation

```
avg_right_line = []
for x1,y1,x2,y2 in avg_right_pos:
    x = int(np.mean([x1, x2]))
    y = int(np.mean([y1, y2]))
    slope = avg_right_slope
    b = -(slope * x) + y
    avg_right_line = [int((330-b)/slope), 330, int((540-b)/slope), 540]

lines = [[avg_left_line], [avg_right_line]]
```

Identify potential shortcomings with your current pipeline.

Potential shortcomings are:

When lanes are not straight, instead they are very curvy throughout the road. Here we can use polynomial fit, to get the curvy lines, also few vision functions like perspective transform, sobel filters etc, to get an accurate output.

Output from the pipeline may struggle, if there are shadows, glare and other weather conditions. For this shortcoming we can use convolutional neural networks to get accurate results.

In the project, I cannot identify yellow lines properly, so in that case using convolution or computer vision filters like sobel will be useful.

Suggest possible improvements to your pipeline

For the second video in P1.ipynb notebook lines detected by the hough lines are intersecting in between of the video. So that can be improved.

Challenge video output can be improved by using polynomial fit/Curve fit