Lane Finding Reflection

1.My pipeline consisted of 5 steps. First I blurred the image with a Gaussian filter with a kernel size of 3, and then I converted it to gray-scale. After that, I applied the Canny edge detector algorithm with a low threshold of 65 and high threshold of 195. The part of the pipeline was constructing a region of interest so as to remove the parts of the image that weren't part of our lane of the road. Next I applied the opencv hough_lines function to map the Hough transform and find the lines within the region of interest. I supplied it with the following parameters: rho = 1, theta = 1 radian, threshold = 12 votes, min_line_len = 12, and max_line_gap = 7. Finally I used the weighted_img function to superimpose the lane marker over the original image.

In order to draw a single line on the left and right lanes, I first removed all lines that had an absolute slope less than 0.4. These would represent horizontal lines in the image which would just skew the data. I then separated the coordinates corresponding to the right side of the lane and the left of the lane by putting the negative sloped lines in an array representing the left and putting the positive in an array representing the right. Then I took those data points and made use of the numpy polyfit function to find the line of best fit for the data points in each array. Using the poly1d function I mapped out the lines for both sides of the lane using custom inputs that would make the lines approximately the same length each time.

2. Possible shortcoming include rigid placement of the region of interest. The region of interest is based on specific coordinates and is not dynamic so lane finding wouldn't necessarily work on other video files. Another shortcoming is inability to properly model curved lanes. This makes the lane finding useless for practical purpose without that capability.

3. Improvements for the pipeline is to give it the ability to detect curved lanes and be able to dynamically produce a region of interest.