Project 1: 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

Your output should look something like this (above) after detecting line segments using the helper functions below

Your goal is to connect/average/extrapolate line segments to get output like this

Reflection

Description of My Pipeline

1. I converted the initial image(RGB) to grayscale.
2. Before using Canny Transform, I applied Gaussian smoothing, which is essentially a way of suppressing noise and spurious gradients by averaging.
3. Defined the region of interest, which is a polygon whose four points are defined.
4. Implemented Hough Transform to find the lane lines and then draw them with draw_lines() function.
5. Combined the image with Hough lines drawn and the initial image to get the resultant output image.

Modification of draw_lines() function:

In order to draw a single line on the left and right lanes separately, I modified the draw_lines() function as shown below:

1. Separated the Hough line segments within the region of interest by their slope (y2-y1)/(x2-x1) to determine which segments are parts of right lane(slope>0) versus the left lane(slope<0).
2. Averaging all the lines on each side by calculating the mean of (x1, y1, x2, y2) of all the line segments.
3. Based on the two averaging points on each side, using Numpy built-in function np.polyfit to obtain the coefficients of the linear line, which passes the averaging points.
4. Extrapolate the both lines by defining the y_max and y_min corresponding to the size of region of interest.

Potential Shortcomings

1. One potential shortcoming would be the less accurate detection when the road is a ramp instead of a relatively straight line.

2. Another shortcoming could be some existing noisy lines are detected from the shadow of the trees within the region of interest. In this case the modified draw_lines() can't work very well because the averaged points will also cover those noisy points.

Improvements

1. A possible improvement would be to grab more than two points when we are trying to average all the line segments, then we can use function np.polyfit() to fit those points with higher degree/order. Therefore, we can draw a curved line to perfectly fit the road, even though it is a ramp.

2. Another potential improvement could be to adjust the parameters of Hough Transform in order to get rid of those noisy lines.