Finding Lane Lines on the Road

Hyunjin Cho

March 2021

1 Pipeline and draw_lines() function modification

My pipeline consisted of 5 steps. First, I converted the images to grayscale, then I applied Gaussian blur function with a kernel size 7. After that, I applied canny edge detection. I applied an trapezoid image mask to select region of interest. At last, I applied Hough-Line transform .

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by choosing certain range of slope of the line. Most of jobs are one selecting range of slope between 25 degree to 40 degree. In order to improve more, I updated average slope that image with the range between 25 degree and 40 degree, so it can converge the average line that normally see in the region of the mask.

In Figure1 to Figure6 shows bottom, output of the lines of image in the test_images are created:

2 Identify potential shortcomings with your current pipeline

One potential shortcoming would be what would happen when line image appear in the video, it shows that line is not constantly continue to reach all the way to the bottom. For example, in Figure number 1 in left line, it cannot be appeared all the way down because there is no line appear.

3 Suggest possible improvements to your pipeline

A possible improvement would be to adding point where y-axis of image is max. Since we know the slope of the line and we know so if we us line equation y = mx + b, we can evaluate the point where locate bottom edges.

Figure 1: solidWhiteCurve

Figure 2: solidWhiteRight

Figure 3: solidYellowCurve

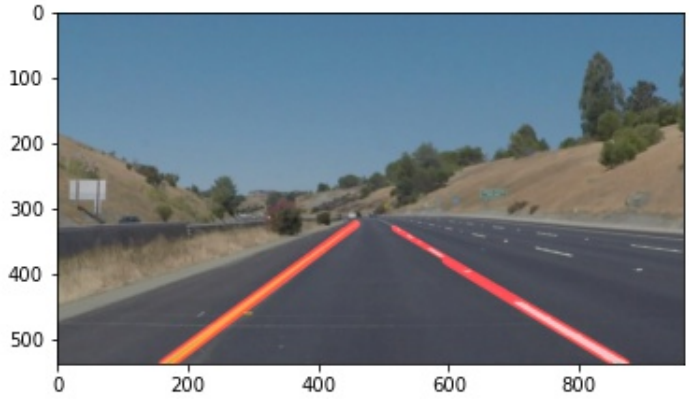

Figure 4: solidYellowCurve2

Figure 5: solidYellowLeft

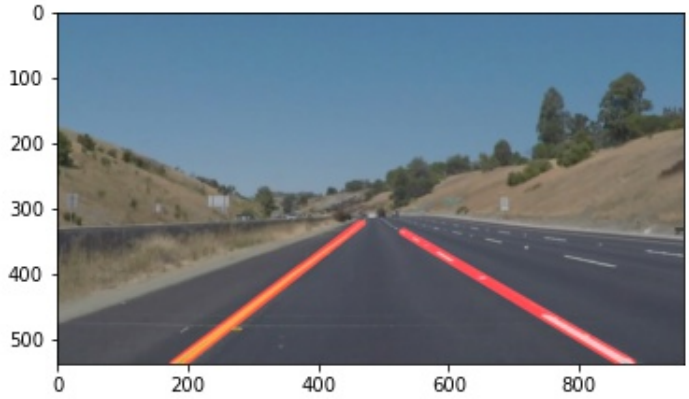

Figure 6: whiteCarLaneSwitch