Finding Lane Lines on the Road

Reflection

1. Pipeline:

My pipeline currently is as follows:

a. I would Keep only yellow and white pixels and black out all other pixels: filter_colors()

- This would remove any unwanted edges from shadows, cracks on the road.

b. I would nextg convert the image into grayscale with the function grayscale()

c. I would apply Gaussian smoothing with the function gaussian_blur() which reduces details from the image. This is required to get rid of details that are unwanted to focus on bigger object boundaries. The amount of blurriness is controlled using specific kernel size

d. Run Canny Edge Detector:canny(). This works on the difference of intensities with respect to the neighboring pixels. The intensity difference is achieved through low and high threshold parameters

e. Now we need to focus on a particular region of an image where lane lines are most likely to occur. Rest of the image is blurred and focus on edges belonging to the part which most likely are to be lane lines. Created a trapezoidal region-of-interest in the center lower-half of the image:region_of_interest().

f. Now we need to focus on edges and try to find a pattern between the line segments which can be joined to form a lane. We run Hough Line Detector:hough_lines() which transforms each edge x-y plane to slope intercept m-c plane to find out the parts

of the same line. We get a list of pair of points where pair of points signify a line

g. Next we have to filter Hough lines by slope and endpoint location, and separate them into candidate right/left lane line segments. Then we would need to run linear regression on candidate right/left lane line segment endpoints, to create right/left lane line equations. From these equations, we can draw right/left lane lines on the image:draw_lines()

i. first the slope of the input line coming from hough transformation is examined. If the slope is positive, it is considered to be a potential right lane and for a negative slope, it is considered to be a potential left lane

ii. Now we have two candidate lines for the right and left lane which are averaged out and two lanes are hypothetically drawn on the image

iii. To have a clear and consistent display, the lines are extended till the bottom

iv. The lane lines are superimposed on the original image

2. Potential shortcomings

The algorithm might fail during night time, in bad weather or in conditions of inadequate light. It may fail in roads where there are sharp bends, since I am trying to extrapolate straight lane lines.

3. Possible improvements

I would test my algorithm on highways with different weather conditions, different times and different weather. I would like to test my logic during heavy traffic. That is when I would be able to figure out ways to improve the algorithm to make it more robust.

I might also try to test my code on roads that have faint or no markings at all. It would be a good idea to test the logic on rough roads as well