Finding Lane Lines on the Road

Writeup

Reflection

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

My pipeline consisted of 6 stages.

Stage 1: Convert the input image to grayscale.

Stage 2: Do Gaussian smoothing on the grayscale image. The gaussian kernel size is set to 5.

Stage 3: Perform Canny on the image output from stage 2. The parameters for low and high threshold are set to 50 and 150.

Stage 4: Create the mask region of interest. It is a quadrilateral region mask. The four vertices are calculated based on the image shape (540x960) and setup parameters. Stage 4 will output a masked image after Canny.

Stage 5: The Hough Transform and draw the lines. The output from “HoughLinesP” will be lines, which will simply be an array containing the endpoints (x1, y1, x2, y2) of all line segments detected by the transform operation. Then the updated draw_lines() function will draw two single lines in red to indicate the left lane and right lane in the image.

The idea of draw_lines() is in following steps:

1. Loop through all the points from Hough Transform and only focus on the valid points (in the range of the image shape), for a single input of points (x1, y1, x2, y2), calculate the slope by (y2-y1)/(x2-x1).

2. If the slope is less than 0, this is for the left lane. Store the slope value in a list, calculate the intercpt and store in a list. Store the minimum y (for left lane, it should be y2) value in a list. For slope greater than 0, it is the right lane and do the same steps as left lane did.

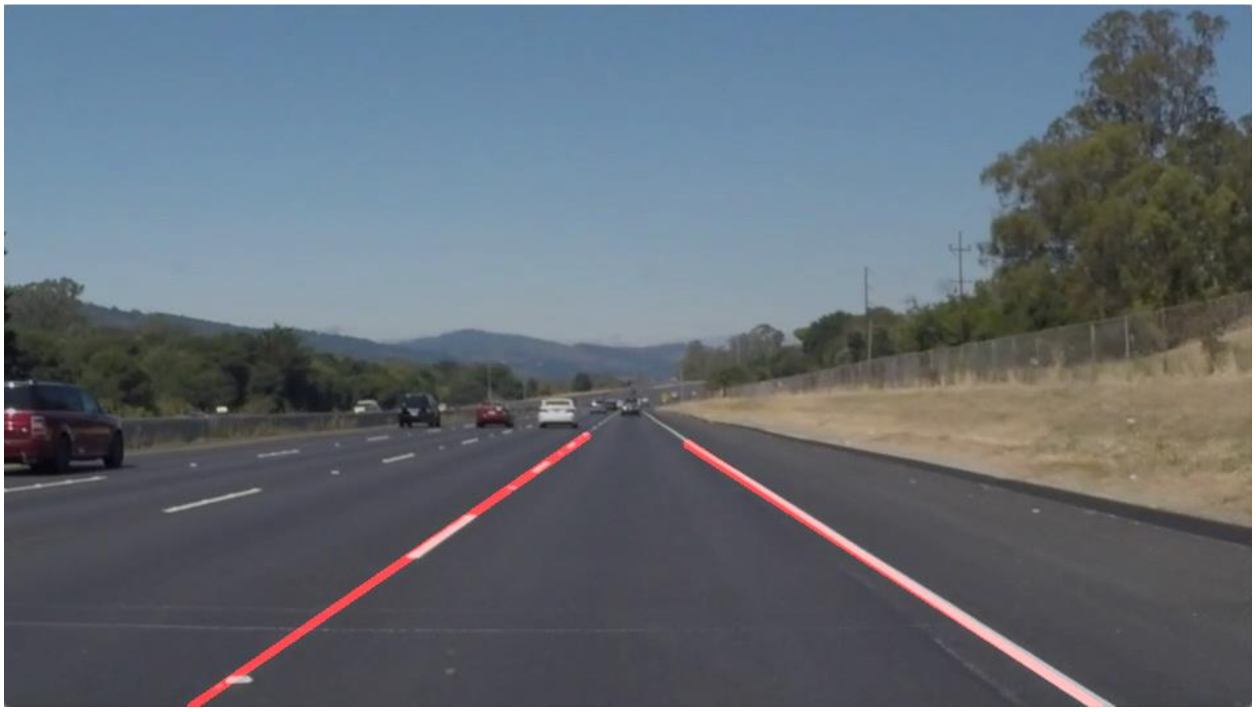
3. After the loop, there are list of left and right slope, intercept and y points. Calculate the average slope and intercept value for left and right lane.

4. Find the minimum y from the list and calculate the relevant x by using the average slope and intercept. The maximum y is the maximum size of image shape which is 540 in this case. Then calculate the relevant x by using the average slope and intercept. As a result, find the two points (x1, y1, x2, y2) for both left and right lane.

5. Draw the left and right lane by using the points from step 4.

Stage 6: Merge the drew lines image back to the original image. So the left and right lane are highlighted in red in the input image.

This is the result image after applying the image processing pipeline described above:



2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be what would happen when the road lane is not a straight line but curved line. Because the draw line function is using the linear line equation y=mx+b. When the lane is curved, the draw line will offset from the curve lane.

A second potential shortcoming is when the lane is dash lines other than a solid line, the draw line is hard to make a complete line. This is due to the Canny cannot detect a full line from the dash lines. If the external environment is not ideal, the draw line will not well.

3. Suggest possible improvements to your pipeline

For the curved line, a linear model will not work well. It is possible to go to nonlinear model, quadratic function or even higher degree. But this will over weight the processing performance especially in real time driving. Meanwhile, besides the parameters in the quadratic formula, there are several other parameters need to setup in the pipeline. It is hard to try all the combinations blindly. A better way to get the optimized parameters is through a neural network to train a large amount of driving data.

Another potential improvement could be to predict points of each frame. In the video output, the draw lines are not moving smoothly compare with the example video output. This is because the points are calculated at each frame and the line are drew at each frame. The points vary in a certain range causes the lines are vary at each frame. If the pipeline can do prediction of the next point in next frame, and make sure it is not very different from the current one, then the lines drew should stable and it will play smoothly from frame to frame.