Finding Lane Lines on the Road

1. Goal of this report/project:

- Make a pipeline that finds lane lines on the road
- Reflect on my work in a written report

2.Refection

2.1 Description of my pipeline

My pipeline consists of the following 7 steps:

Step 1: Grayscale

In this step, I use the *cv2.cvtColor ( )* function to convert all the images to grayscale images. Therefore, the images are composed only of shades of gray, where color black has the weakest density while color white has the strongest density.

Step 2: Canny Transform

In this step, I use the cv2.Canny function to conduct the edge detection. There are two inputs needed for this function. The first one is low threshold, below which are defined as non-edges. The second one is high threshold, above which are defined as edges. For all the values in between, if it is connected to a edge pixel, then it is classified also as edge, otherwise it would be classified as non-edge. As what is taught in the lecture, usually the value of high threshold is 2 to 3 times the value of the value of low threshold. After a number of trials, I set the low threshold as 100 and high threshold as 200.

Step 3: Apply Gaussian smoothing

In this step, I use cv2.GaussianBlur with a kernel size of 9 to smooth the grayscale images.

Step 4: Define region-of-interest

In this step, I apply the masking to define the region of interest. In my case, I define the region-of-interest as follows : [(0,image.shape[0]),(440,320),(540,320),(image.shape[1],image.shape[0])]

Step 5: Hough transform

In this step, I use cv2.HoughLinesP to proceed the processing with Hough transform to find all the straight lines in the edges. After a few trials, I set the input parameters of this API as follows: rho 5, theta pi/180, threshold 50, min_line_len 25, max_line_gap 25.

Step 6: Decide and draw the left and right lanes

In this step, I process all the lines found in step 5 to conclude them into 1 line for each side.

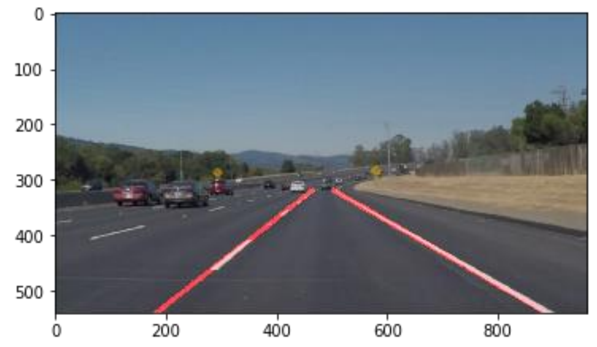
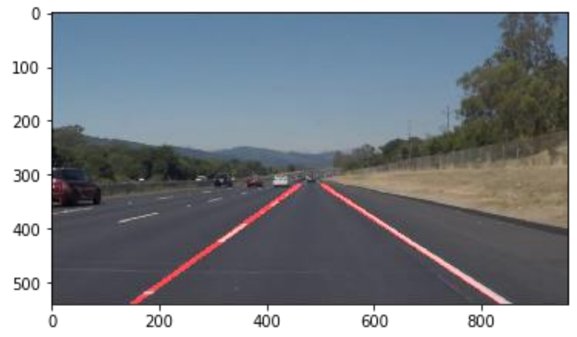
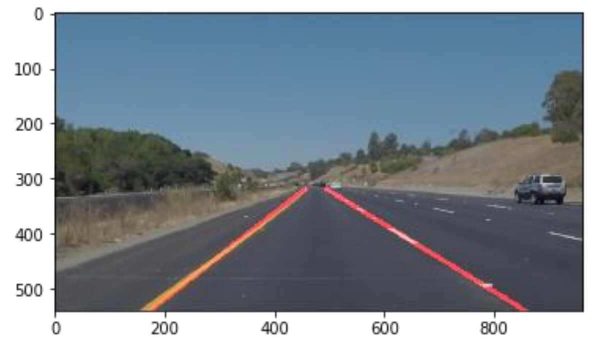
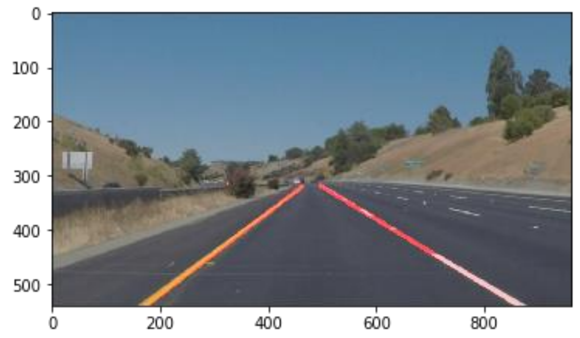
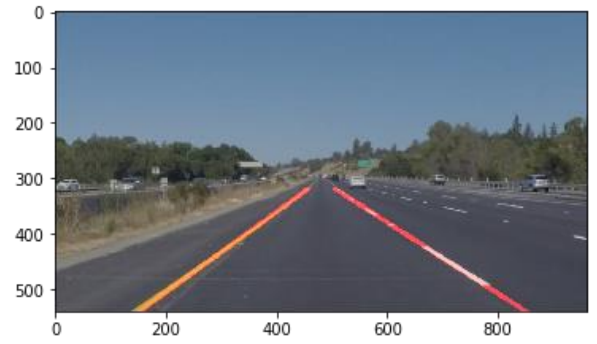
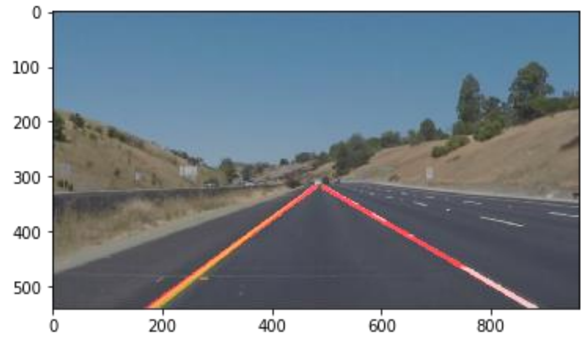
Firstly, I use the value of the slopes to classify all the lines in the left and right line groups. If the value of the slope is positive, then it belongs to the right lane. If the value of the slope is negative, it belongs to the left lane.

Secondly, I calculate the average X position and Y position of the all the lines in the left lane group and the right lane group accordingly. I assume the left lane will pass the average left x and y position and the right lane will pass the average right X and Y position. Meanwhile, I also calculate the average slope for left lane and right lane. With one x y position point and the value of the slope known, I calculate the value of b for each lane.

Now I have both the m and b values for each lane. According to step 4, I can know the Y values of the 2 end points of the lanes. With the m,b,y values all known, I can easily calculate the value of X for the end points. With both the X and Y values of the end point known, I can draw the left and right lanes accordingly.

Step 7: Set lane to transparent

In this step, I change the drawn lanes to transparent to have a better view of both the real and drawn lanes. The image below is the output of my pipeline.

2.2 Potential shortcomings with my current pipeline

- My solution of defining the region-of interest can be a shortcoming., because some of my corners are fixed values. The accuracy is quite ok when the road is straight and there is no width change of the road. However, it may cause problem when the road is not straight any more.
- In the current scenario, the road is straight and flat. If the road is bumpy or there are a lot of sharp turnings, the solution of using only 0 as the threshold to classify left and right lanes might not be enough.

2.3 Possible improvements to my pipeline

- For the first shortcoming, a more intelligent region-of-interest should be considered. Now I am using the same region of interest for the whole video. It can't not be enough in real situation. An improvement can be: the average of the slope is calculating in a time window (I'm not sure about the length of the time window, maybe 5 second). At every end of the time window, we compare the slope value with the previous one, if the deviation is beyond a certain threshold, which indicates a big change in the road condition, we need to define a new region-of interest.
- For the second shortcoming, a smarter way of classifying the right and left lane should be considered. As, the left lane and right lane are symmetric, we can use the symmetry to think about a more accurate threshold to recognize left and right lanes. Also, when the road is bumpy, there will be distortion on the lanes. We need to diagnose distortion and remove the negative effect of the distortion.