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

Reflection

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

My pipeline includes 4 steps:

The first is to detect edges which is composed of 3 smaller steps: convert original image to grayscale, reduce noise through Gaussian function, and detect edges through Canny function.

The second is to detect lanes from the region of interest which has 2 smaller steps: define a region of interest using a mask, and then detect lanes inside this region through Hough Line function.

The third part is to combine lane image with the original image to show the colored lanes.

I modified draw_lines() function in this way:

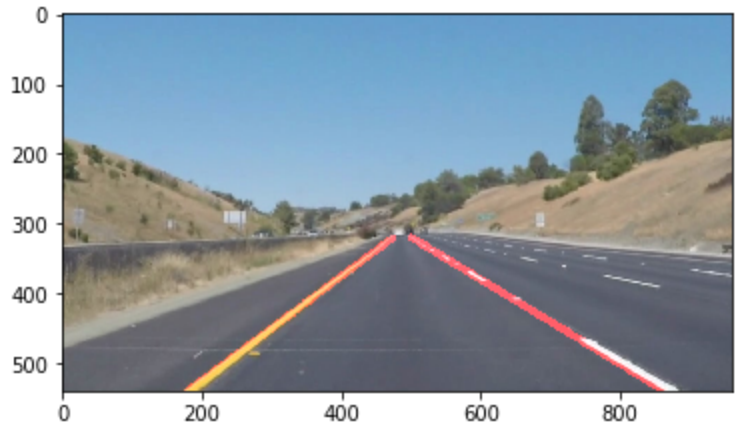
The idea is in each image of detected lanes, ideally all the points on the right lane should follow the same y=mx+b function, and this also applies to all the points on the left lane. Considering y1 and y2 can be constants across all the images, as long as we can figure out m and b for right lane and left lane, we can have x1 and x2 for both lanes, then a straight line for both right and left lanes can be drawn.

Firstly, we need to get all the points on the right lane and left lane respectively in order to figure out the m and b of y=mx+b in the image space for the right lane and left lane. As a matter of fact, the absolute value of the slope of the lanes should be within a reasonable range. Assuming the angle of the lane shouldn't be less than 30 degrees from the horizon, then the slope's absolute value should be greater than 0.5. Since (0,0) is at the left upper corner of the image, the left lane's slope should be negative and right lane's should be positive. For each line in the image space, if its slope's absolute value

is > 0.5, if it is negative, and both x1 and x2 are smaller than the x of the center of the image, it is the candidate line of the left lane. On the other hand, if the slope's absolute value is > 0.5 and positive, and both x1 and x2 are larger than the x of the center of the image, the line is a good candidate line of the right lane. After getting all the candidate points on the left lane and right lane respectively, we can use Numpy.polyfit() to compute the m and b of these 2 lanes.

Then rely on the fact that y1 and y2 might be constant across images since we only care the range between a certain point at the center of the image and the bottom of the image. We can compute x1 and x2 for both right lane and left lane based on m and b of these two lanes. Then we can easily draw two straight lines.

The image is:



2. Identify potential shortcomings with your current pipeline

There are two main shortcomings:

Firstly, all the parameter values are hardcoded so it might not be compatible with other images. I use hardcoded threshold values for Canny function, many hardcoded parameters for Hough transform, as well as hardcoded x and y boundaries for the region of interest. Those only work for the given test images.

Secondly, I assume all the lanes are straight in the region of concern so I use cv2.HoughLinesP() function but it won't work if the lanes have curve.

3. Suggest possible improvements to your pipeline

The improvement to the first shortcoming is to use dynamic way to adjust values for Canny and Hough transform functions.

The improvement to the second shortcoming is to use some threshold to decide whether the lane is straight or curvy.