Udacity

Project 1: Finding Lane lines on the road

Udit Puri

The Goal of this project is to create a pipeline that can find the lane markings on a road – First in a series of images and eventually in a video stream. In a fully autonomous system, the vehicle should be able to identify the lane markings on the road along with many other variables to drive itself.

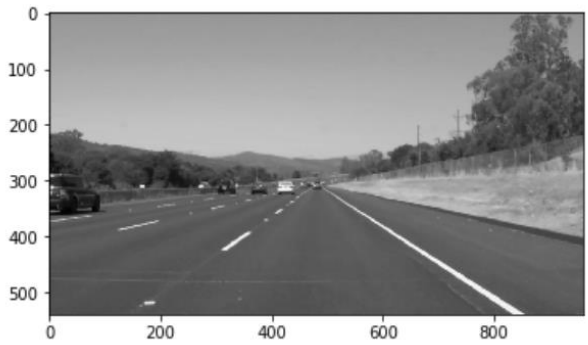
In this project, we try to achieve the same using computer vision tools and techniques.

1. Pipeline:

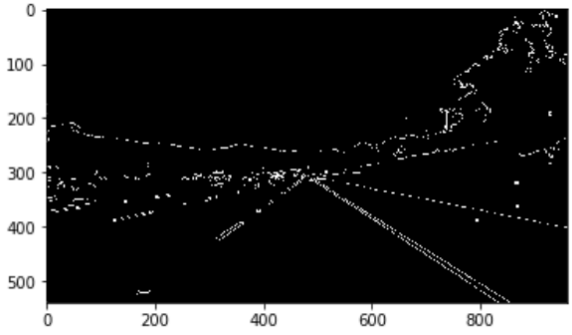
   There are various steps and stages in the pipeline that the image goes through:

   - Read the 960x540 image of the road

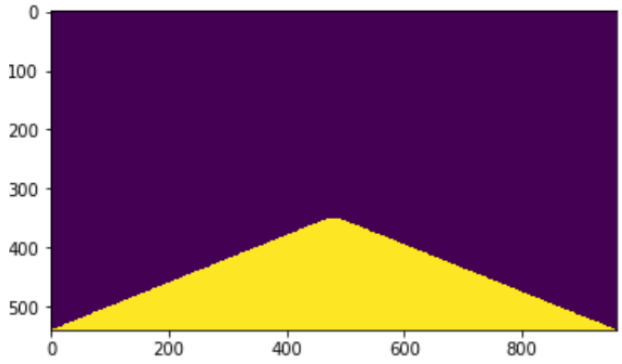
   - Convert the image to grayscale



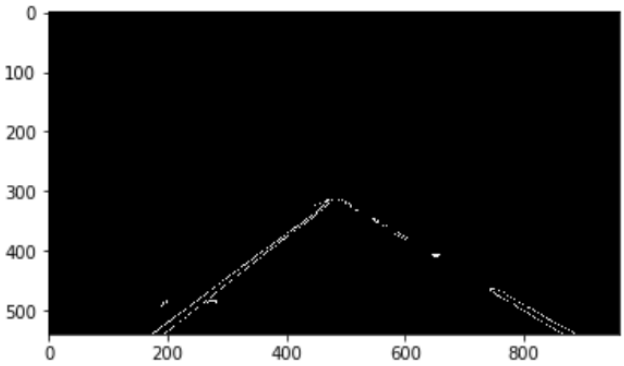
   - Apply Gaussian smoothening to blur the edges and CANNY edge detection



- Find the region of interest to make sure the algorithm is only focused at the region where the road and lane line exist.



- Overlay the output of canny edge detection and region of interest to only detect the lane lines



- Apply Hough transform to find the lane lines and draw solid red lines on both the left and right lane marking. Later sperate the right and left lane to interpolate between the points to draw smooth lines between the end points of both right and left lane.

2. Algorithm Details:

In order to detect the difference between the left lane and the right line, the concept of positive and negative slope is used. For the pair of x and y (pixel) values that give a negative slope – they are append to the left lane and if it gives a positive slope – they are append to the right lane. Moreover, we are rejecting any slope that are too horizontal and too vertical as such points would not lie on the lane line. Plus, the logic also looks into the fact that the left lane points should only exist in the left side of the image and vic-versa.

Now, the *draw_lines* function also calls another function called *drawline*. The *drawline* function just uses the equation of a line (y-y') = m(x-x') to draw line between the points. Point y1 is chosen to be at the bottom of the image since that is exactly where we would like to start the line from. x1 is calculated by rearranging the above equation of the line such that x1 = int((y1 - b)/m) [note: *m* and *b* are the slope and intercept of the line that can be found using the *polyfit* function. y2 is set as 340 (to be in the middle of the image which is the point where the lane sort of ends into the horizon) x2 can be found using y2. Using this mechanism the lines are drawn.

3. Shortcomings

The main short coming is that the straight lines do not work when there is a curve on the road. Using the equation of a straight line to draw line is not enough to trace on these curves. Sometimes there are small glitches in where the lane sort of veers slightly away from the road

4. Improvements

I would like to learn how to make the lines less shaky and more smooth. Applying some sort of averaging mechanism and more complex line equations can help with following the lines on a curve as well as making it less flicker,