Finding Lane Lines on Roads

This first project of Udacity's self driving car Nanodegree program is all about detection Lanes on
roads(any color).

Pipeline Overview:

My lane detection pipeline consist of 5 steps:

Input image:

1) **Convert to grayscale:**

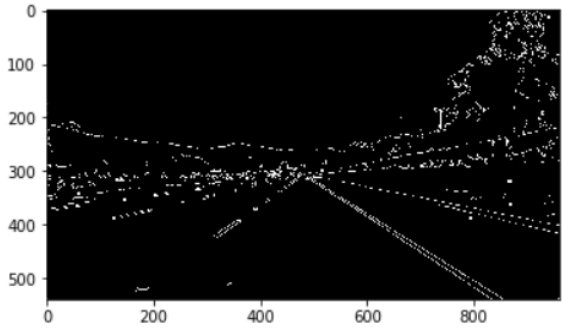
We process the image or frame within the video by changing colorful images to grayscale. Each pixel
can now be represented by a 8-bit integer number(0-255)

2) **Smoothing:**

Gaussian smoothing is used to removed unwanted noise in the image since in the next step we want to detect edges.We consider kernel size of 5x5.

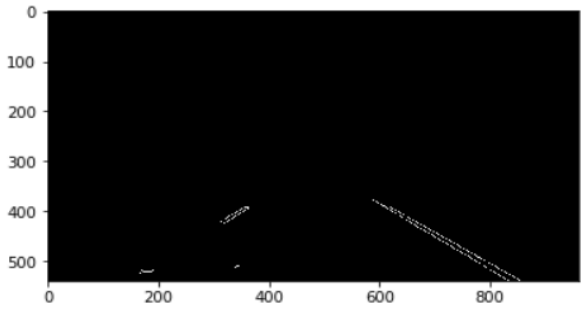
3) Canny Edge Detection:

Lanes have different color from its neighboring hence a state of art algorithm Canny edge detection is applied on the Gaussian smoothed image to detect relevant edges in the image.



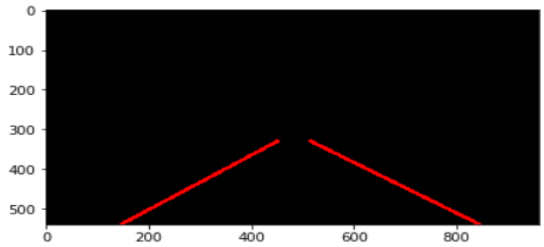
4) Region of interest:

In previous step we get a lot edges over the image but since we are interested only in the lanes in image hence we restrict ourselves over a small portion of images known as region of interest.A quadrilateral of interested width and height is selected to detect only the road lane marking.



5) **Hough transform:**

Previous steps gives us many points as edges of lane but we are more interested in continuous edge line. We find a line passing through all those points thus we use hough transformation to find a set of continuous lines .



Mapping of hough image on original image is done to achieve the final lane detected image.

Extrapolation of left and right lane lines

In order to find a single continuous left and right lane line I modified the draw_lines() function.

The max_y of the image would be image width(image.shape[1]) and y_min would be around 60% of the width as image mid region of lines

Min_y=image.shape[1]*0.6

Approach I'll show only for left lane and same applies to right lane as well.

Approach:

- Since the image origin is at top left corner hence the left lane line has a negative slope and right lane line has positive slope.
- Gather all the hough lines with negative slope
    - Save the intercept and the slope.
- Average the slope(average_slope) and intercept(average_intercept)
- Find the x_left_bottom and x_left_top as:
    - x_left_bottom=(y_max-avg_intercept)/average_slope
    - x_left_top=(y_min-avg_intercept)/average_slope
- Draw line from(x_left_bottom,y_max) to (x_left_top,y_min)

Note:Apply the same for right lane.

Potential Shortcomings of current pipeline: