UDACITY:Self Driving Nanodegree

Project 1: Finding Lane Lines on the Road

Prepared By: Rakesh Paul

18-04-2018

Project Goal:

The goals / steps of this project are the following:

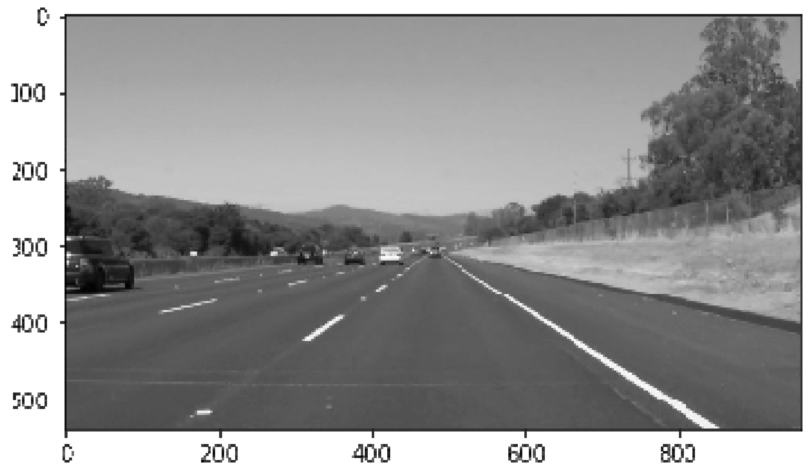
- Make a pipeline that finds lane lines on the road
- Reflect on your work in a written report

Project Execution Pipeline:

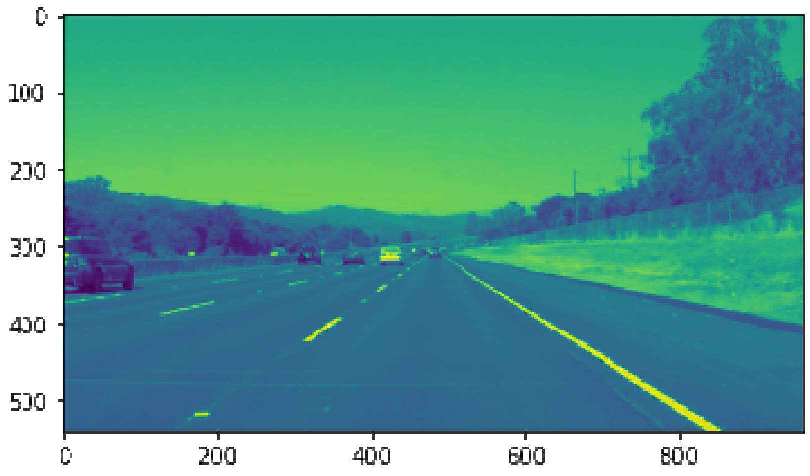
The Project pipeline consists of 5 steps:

- Conversion to Grayscale
- Noise reduction
- Finding Edges using Canny
- Extracting ROI
- Hough Transform

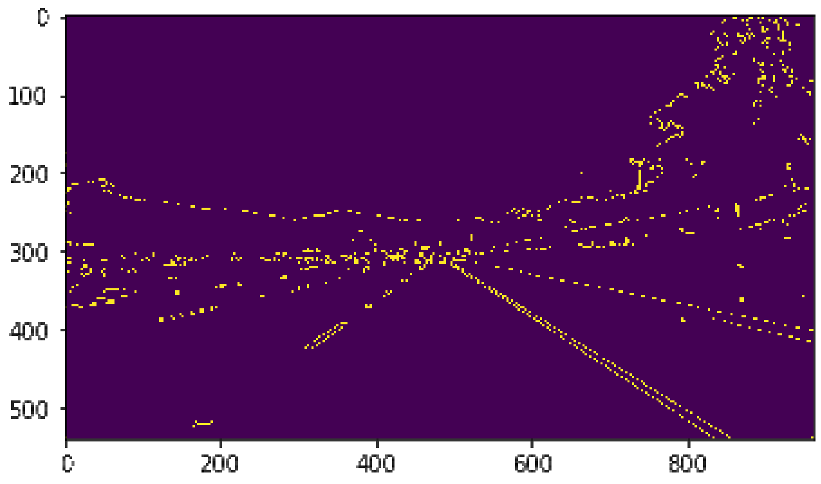
Conversion to Grayscale: The input image is converted from RGB to Grayscale representation for simplified processing of image data. It allows for easier processing of image data for find edges due to single channel representation.



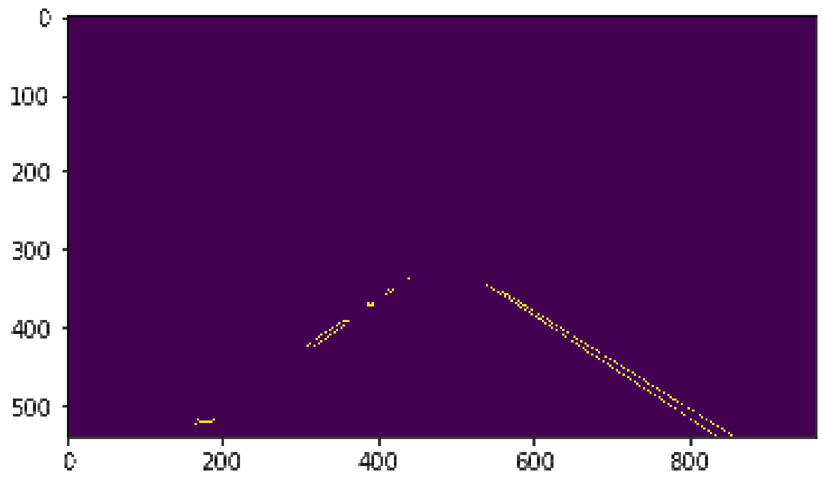
Noise reduction: In this stage, Gaussian-Blur is applied to reduce noise from Grayed image in first step. This helps in better performance of Canny Edge Detection algorithm.



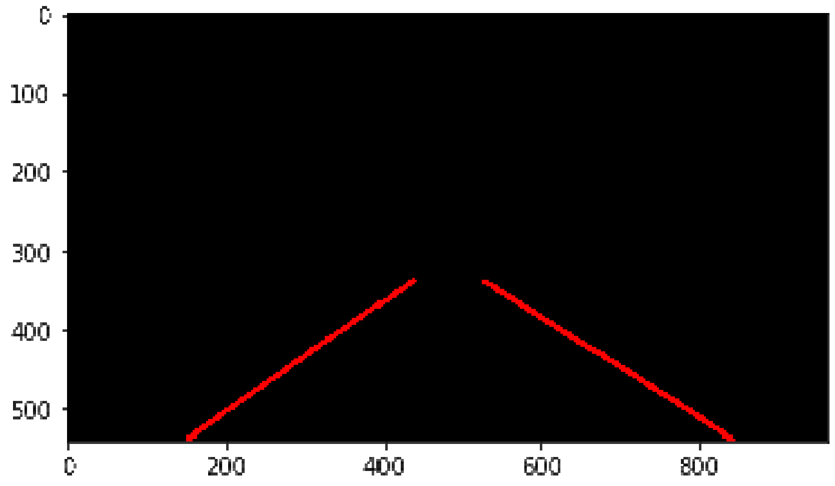
Finding Edges using Canny: Canny edge Detection is performed to find sharp edges in the Grayed image after noise reduction. Canny algorithm detects sharp edges in image, any of which can be potential candidate for lanes.



Extracting ROI: Edges detected by Canny can be scattered across whole image. But here we are trying to find lanes, which makes our search region confined near to horizon to bottom of image. A region of interest(ROI) is defined for it.



Hough Transform: Hough Transform takes the edges with ROI as input and find set of lines. These lines are then fitted together for generating solid lines which represent right and left lane. The output of this process is a blank image with lines for the lanes(one for each). These final lines are then drawn on original image and displayed.



Fine tuning:

The major challenge in the activity was to display solid lines for lane markers. Towards this goal, I segregated the lines obtained from Hough transform based on the slope of each line:

Slope = Change in Y / Change in X

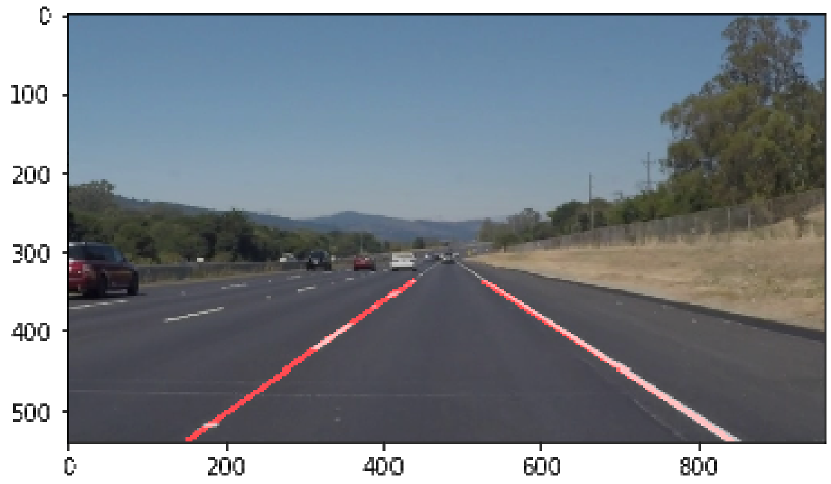
As origin in opencv is from top-left, '-'ve slope indicates left lane lines and '+'ve slope indicates right lane.

The set of each lane lines are then least squared(1st degree) for finding slope and intercept of a line which is most fitting for the scattered line points. This is performed using polyfit() function.

Based on this information, a equation of 1st degree is formed for y= mx + b(equation of a line). This is performed using poly1d().

Finally, using the equation obtained, the corresponding X co-ordinates are calculated for each min and max value of Y(from the lines of hough transform). This gives us extreme points of lines which are then drawn to find a solid line for each lane.

Final Output(merged with input image):



Shortcomings:

Following are some of the shortcomings of the algorithm in my opinion:

- Curves in lane lines are not detected. Currently straight lane marking only detected.
- Camera based image processing does not work efficiently in foggy and low light condition. This algorithm also will be affected by the environment condition.

Possible Improvements:

The algorithm will perform better if the following conditions are taken care:

- This algorithm can be upgraded to display curves in lane lines.
- Object Classifiers can be used for accurate results.