FINDING LANE LINES ON THE ROAD

• ***The goals of this project is to make a pipeline that finds lane lines on the road***

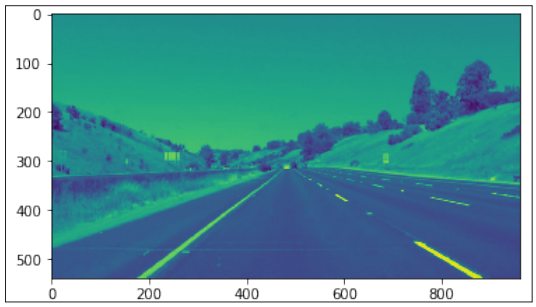
PROJECT OVERVIEW

• The first Project in the Udacity Self-Driving Car Nanodegree is about implementing a pipeline that detects lane lines in images. While the pipeline is created for a single image, it can be applied to video footage by breaking the video down into frames, passing the frames through the pipeline, and then reconstructing the video.
Identifying lanes during driving is a very common task to humans in order to perform a safe trip from point A to B. An important task to keep the car within the constraints of the lane. This is also a very critical task for an autonomous car to perform. This writeup describes a pipeline that can be used for simple lane detection using Python and OpenCV.

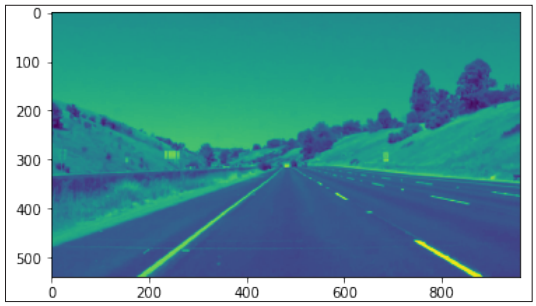
1. LANE DETECTION PIPELINE

The original image

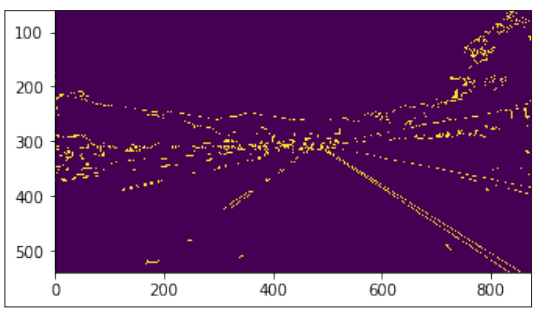
Step.1 : Apply the grayscale function to the original image in order to reduce the amount of data and complexity we are dealing with.



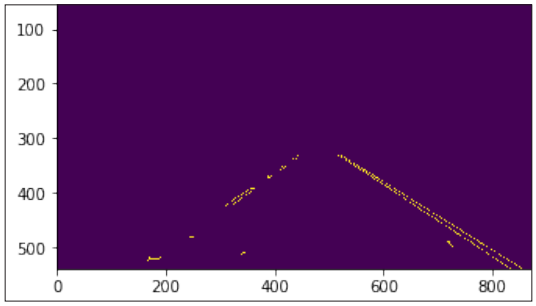
Step.2 : Define a kernel size and apply a Gaussian smoothing, Gaussian blur is pre-processing step used to reduce the noise from the image. the purpose of this step is to remove many detected edges and only keep the most prominent edges from the image.



Step.3 : Define the parameters for CannyEdgeDetection & apply, Canny edge detection is an algorithm that measures the change in connected pixel values (the gradients). We use it to turn our image into pure black and white where white represents the largest gradients



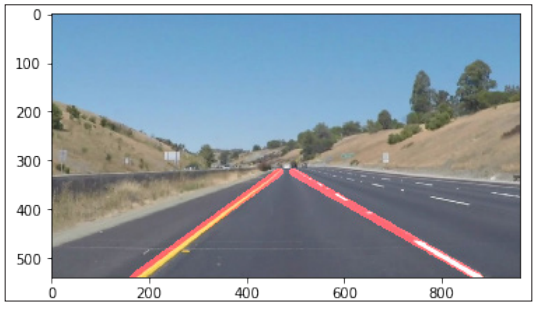
Step.4 : Define the region of interest, This helps in removing the unwanted edges detected by the Canny, because the output contains many detected edges that are not lanes. Region of Interest is a polygon that defines an area in the image, from where edges we are interested.



Step.5 : Apply the Hough Transform, Hough Transform is the technique to find out lines by identifying all points on the line. This is done by representing a line as a point. And points are represented as lines/sinusoidal

Step.6 : Consolidate and extrapolate the Hough lines and draw them on the original image. We need to trace complete lane markings. For this the first thing we need to distinguish between the left lane and right lane, and we can do this by evaluating the sign of the slope of a line (+ for left lines and - for right lines, we ignore vertical lines).
After identifying left/right lane Hough lines. We will average those lines if there are many lines detected for Lane, and extrapolate them if there are partially detected.

In the last step, we simply draw the final lines onto the original image, I build a similar function to the classic draw lines (extrapolate_draw) and give it as an argument an array of the extrapolated Left / right average lines, I also edited the hough_lines function to return the solid lanes lines on the final image regarding this modification (voir Notebook).



2. SHORTCOMINGS

Hough Lines based on straight lines do not work well for curved roads/lanes. The Region of Interest assumes that the camera stays at the same location and lanes are flat which makes the Region of interest the weak element in the actual pipeline. So a big effort is needed to identify the right polygon vertices for curved roads and find out how to deal with other lines that we could find in the middle of the road hence detected in the region(not a smart algorithm). We can also find roads without lane markings where this won't work.

2. POSSIBLE PIPELINE IMPROVEMENTS

The curve lanes detection in autonomous vehicles play vital rule in field of self-driving cars. I think using complex video/ computer vision algorithms with deep learning techniques is the key to develop the lane/curve path detection and improve the definition of the region of interests and making it dynamic & smart.