Finding Lane Lines on the Road

Anishram Senathi

Contents:

Working Environment

Programme tested on Jupyter notebook using Python 3.8.5

Key packages: opencv, moviepy

Input: RGB video of driver's view

Output: Video with detected left and right lanes

Lane Detection pipeline

The idea is that lanes occur in a defined region of interest and at the edge of lane and road. For simplicity, lanes are classified into left and right based on slope and the respective best fit line is extrapolated to give the final lanes.

The pipeline consists of 5 steps

Gaussian Blur

A 3x3 kernel gaussian blur is applied to smoothen the image and remove noise.

Grayscale

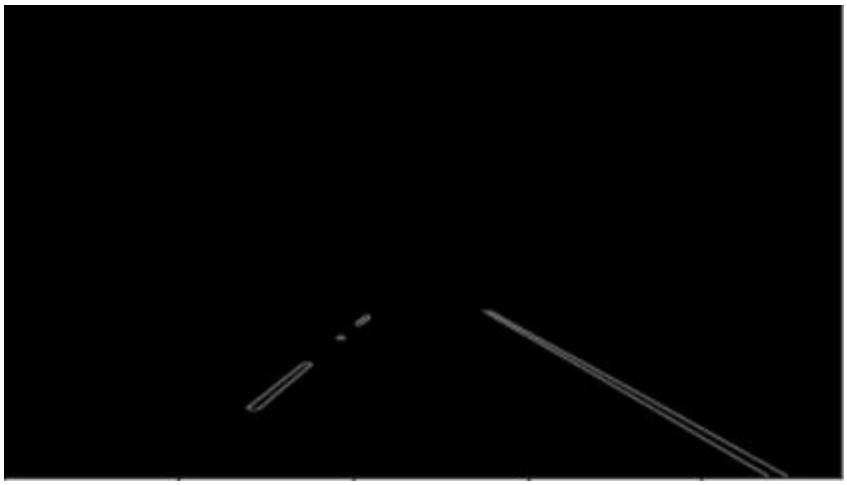
The image is converted to grayscale since further steps are based on pixel intensities and not color.

Canny Edge Detection

Next, Canny Edge detection algorithm is applied on the image to find the edge between lane and rest of the road.

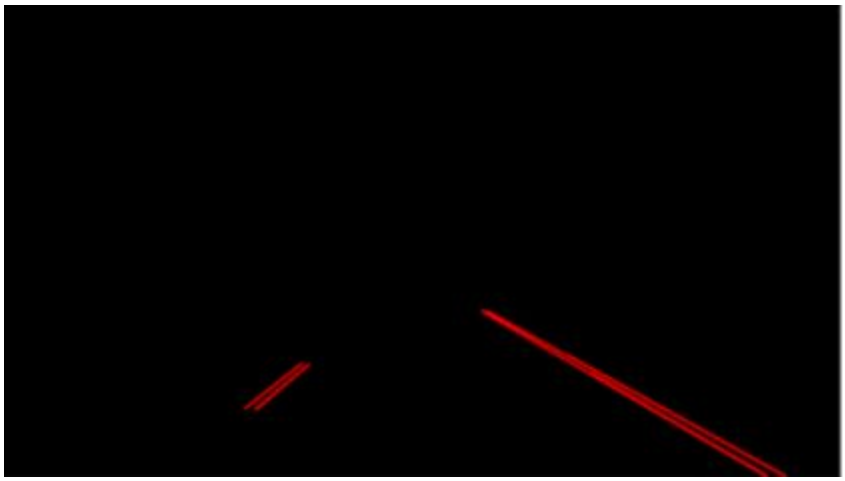
Region of Interest

Mask out areas other than our region of interest where the required lanes cannot be present. I have defined a trapezoidal region of interest taking some portion of road lying ahead of the car, omitting the horizon and leaving margin for the bonnet of the car.



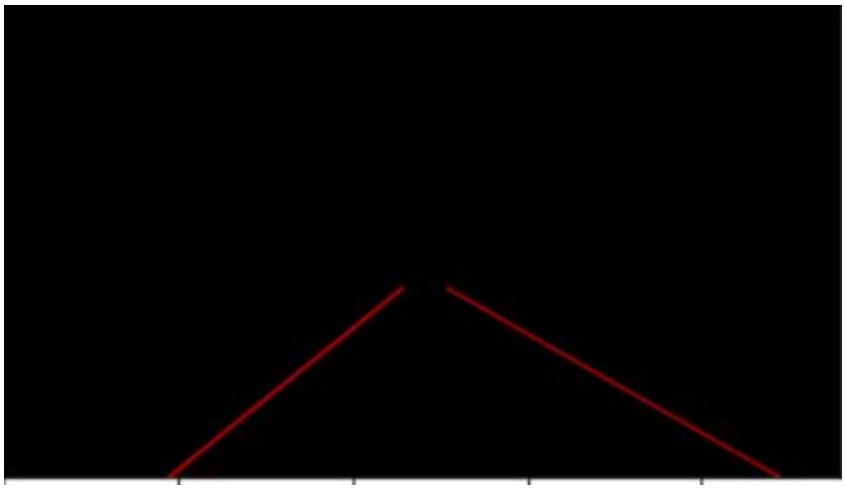
Hough Lines

Based on these edges, Hough transform is applied and many line segments are obtained.



Curve fitting and Interpolation

According to OpenCV matrices convention, lines with positive slope are classified as Right and those with negative slope are classified as Left. For each lane, the endpoints of these segments are taken and the best fit line is extrapolated and plotted to form the final lanes. If the slope of the best fit line comes out to be between -0.1 and +0.1 , the line is ignored.



Annotating Video

The detected lanes are annotated on the input video to get a sense of performance of the algorithm.

Reflection

This code involves finding lanes by classical image processing techniques using limited libraries. Hence it has its shortcomings, yet the parameters have been optimized to make the pipeline as robust as possible.

1) The region of interest has been set in terms of the frame size, instead of constant pixel values
2) Edge detection technique has been used instead of pixel value thresholding, to account for variations in lighting and color of the lanes and roads

Shortcomings:

1) It is assumed that the portion of lanes in front of the vehicle is a straight line, however that is not the case while turning
2) Lanes are not extrapolated beyond a point
3) Lanes are not being tracked, i.e. if the pipeline missed detecting lanes in a few frames, it doesn't make use of the fact that lanes were detected previously and they are static

4) Classical methods are not as robust as deep learning methods
5) Methods to remove wrongly detected lanes hasn't been implemented (for example lanes intersect)