Finding Lane Lines on the Road

Overview

When we drive, we use our eyes to decide where to go. The lines on the road that show us where the lanes are act as our constant reference for where to steer the vehicle. Naturally, one of the first things we would like to do in developing a self-driving car is to automatically detect lane lines using an algorithm.

In this project we detect lane lines in images using Python and OpenCV.

To achieve the goal two modification of algorithm were done:

1. Identify all the lane dependent lines as they are presented in image/video stream
2. Extrapolated lanes' lines – left and right lane for each the image video.

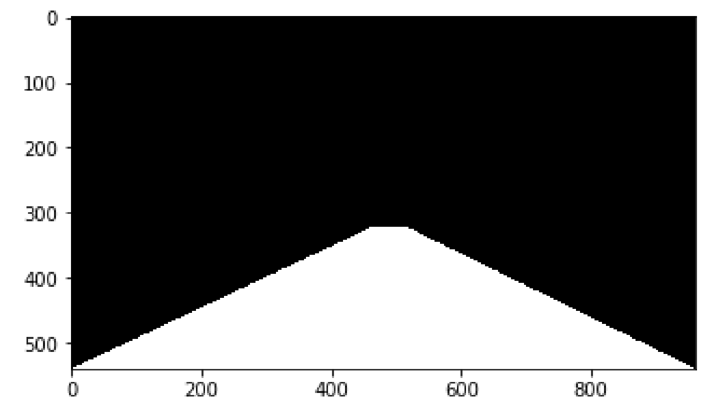
Main Pipeline

For lane identification such an algorithm is used for each input image:

1. Convert the image to gray scaled image using provided method ***grayscale***
2. Apply gaussian smoothing for the current image
3. Use **Canny** transformation with predefined parameters to receive the gradient image

   For the case current such values were chosen:

   low_threshold = 50
   high_threshold = 150
4. Provide the polygon to mask only interesting image area. Current polygon (mask) was defined for the each image:



5. Mask the input image with the current polygon using ***region_of_interest*** method
6. Use the **Hough** to identify lines of the lanes using **hough_lines** method. The Hough transform parameters were defined as:

   rho = 1 # distance resolution in pixels of the Hough grid

   theta = np.pi/180 # angular resolution in radians of the Hough grid

   threshold = 20   # minimum number of votes (intersections in Hough grid cell)

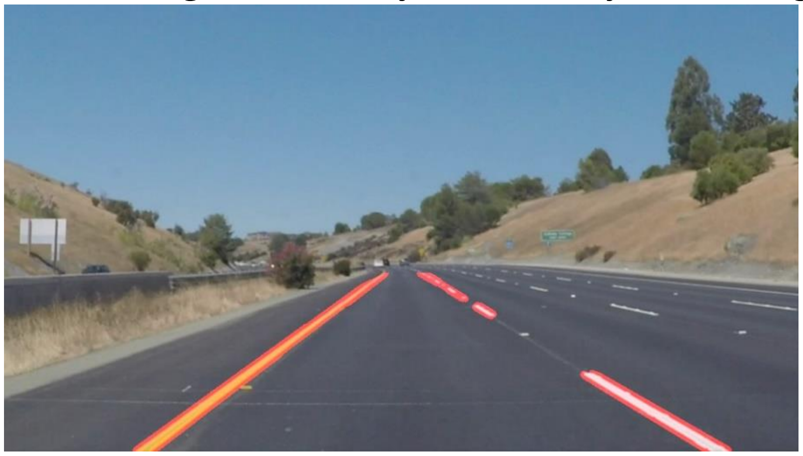
   min_line_length = 10 #minimum number of pixels making up a line

   max_line_gap = 10   # maximum gap in pixels between connectable line segments

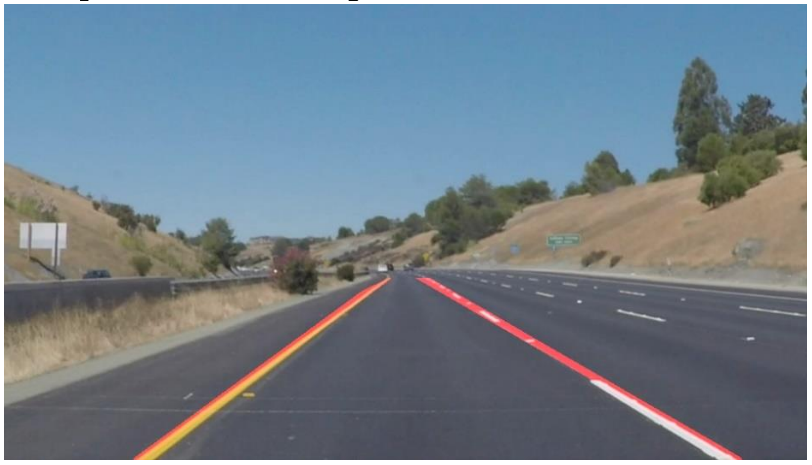
7. Add result lanes to the initial image using provided method ***weighted_img***

The **hough_lines** method is able to receive lines in two modes:

1 Detected edge lanes as they were initially on the image



2 Extrapolated left and right lanes



Line edges extrapolation algorithm

The result of edges extrapolation is two lines (if they were identified by Hough algorithm) which define left and right road lane.

Lane is identified as:

- **Left lane** if the tangent is negative
- **Right lane** if the tangent is positive

To avoid ***false positives results*** such lines are skipped:

- “horizontal lines” – the lines with angle less the 30 degrees
- Incorrect perspective lines – which means we skip the lanes with negative tangent from the right side of image and vise versa
- Strict vertical lines (where tangent is infinite value)

Each lane for simplicity is defined by only one line. The lane is defined using **best_chosen_line** strategy. Current strategy identifies the “best” line from the presented left/right lanes. Current chosen lane is continued through the all visible polygon and is returned as result.

Two possible strategies are defined in the code now:

3. ***The longest line strategy*** where the longest line from the identified left/right edges is chosen
4. ***The “most vertical” edge strategy*** where the most “vertical” edge is chosen. To achieve this The edge with the maximum absolute value of tangent is defined as the best edge.

By default **longest edge strategy** is used as it provides the better results on examples.

Shortcomings

- Not all the lanes identified correctly, especially for bad quality roads (from “challenge” video for example)
- As lanes extrapolated by one line only, incorrect lines or lines which are not correctly repeats the lanes geometry may appear

Possible improvements

- Better lane identification is needed to be provided. For such a case we may try to set more weights to points with possible lanes colors. For example, we can try to set some filter on the initial image to make yellow and white colors more contrast. And vice versa – smooth the “black” lines.
- Polyline for lane extrapolation might be better decision
- Algorithm to choose correct left/right lines may be improved
- Algorithm need to be verified on non-trivial examples with route turns (may lead non-acceptable results)