Finding Lane Lines on the Road

Carolina Hoffmann-Becking [carolina-github] | Hong Kong | 26 April 2020

Project goals

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

Reflection

1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

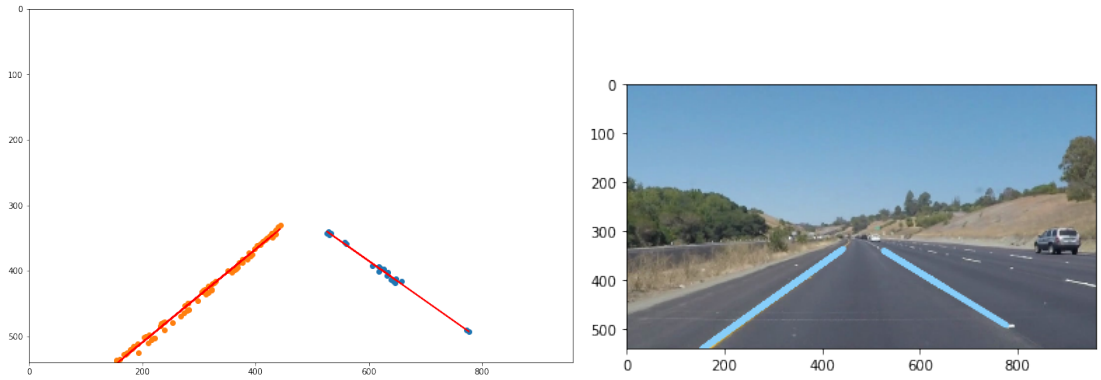
My pipeline to preprocess the series of images consisted of 6 steps:

1. Converted the series of images to **grayscale**.
2. Applied **advanced gaussian smoothing/blurring** with a kernel size of 9 to further improve the default 5x5 internal Gaussian of the Canny Function.
3. Detected the edges in the image with the **canny function** defining low and high threshold, which are recommended to have a 1:3 ratio.
4. Defined a four sided polygon mask / **region of interest** that captures the lane lines for the series of images.
5. Defined **hough transform** parameters to specify parameters to detect lines in the shape of lanelines only.
6. Applied mask on determined hough lines by filling pixels inside the polygon mask with **cv2.fillPoly** and returning the hough lines image only where mask pixels are non-zero using the **cv2.bitwise_and** function.

Initially, I would **draw lines** resulting from Step 6 on the image using **cv2.addWeighted**. However, for lane lines split in segments the full extent of the lane line could not be mapped out. In order to draw a single line on the left and right lanes, I modified the draw_lines() function by implementing a **linear regression algorithm**:

- Distinguished between left lane and right lane by
  - Slope (disregarding vertical and horizontal extremes)
  - X-values (left and right from image center)
  - Y-values (according to region of interest)
- Applied region of interest polygon on right and left lane arrays
- Separated x and y values inside region of interest to run linear regression
- Performed linear regression
- Draw lines on images using opencv cv2.line function

The workbook has included **checkpoints** with images and array prints over the course of the working process on developing a final solution. Below is an extract showing linear regression output on a plot and an image:



2. Identify potential shortcomings with your current pipeline and suggest possible improvements

Potential shortcomings are visible when running the project challenge. The manual definition of the polygon mask and the **impact of outliers on linear regression** due to the small dataset is tremendous indicated by lane lines “flicking and jumping” in the project challenge.

A possible improvement would be to improve the preprocessing of the images and resulting definition of lane lines. This improvement could decrease the number of outliers. In addition, training the regression algorithm on a series of images could improve the performance over time.

In addition, lane lines are **not extrapolated to the full extend of the image**, but rather until the “last” lane line detected. Start and Endpoints can be derived through the linear regression coefficient and intercept and added to the linear regression dataset of x and y value arrays.

Furthermore, a **GOI tool** can be designed to fully optimize canny function paramters.

Finally, **hough transform parameters** may be better understood to achieve optimal performance. Ideas for discussion are set around:

- Angular resolution in radians may be set with an angle of 180 or 210.
- Threshold is set at 2, however hough transform documentation shows threshold set at 200.
- Max_line_gap may not exceed 50% of min_line_length to ensure robustness and minimise outliers.

3. Additional comments

None