Finding Lane Lines on the Road

Report

Author: Aneeq Mahmood

Email: aneeq.sdc@gmail.com

Finding Lane Lines on the Road

The following goals have been achieved in this project

- Make a pipeline that finds lane lines on the road: For this purpose, a file Project.ipynb has been written. In this file, some of the functions have already been provided (and discussed during the lecture and quiz phase in the Udacity classroom). I wrote a pipeline called "process_image", which can take images as input and detects lane in those images. The images are to be saved after processing in a separate directory. The same process is done then to 2 video files which have been provided in the project repository.
-
- Reflect on your work in a written report

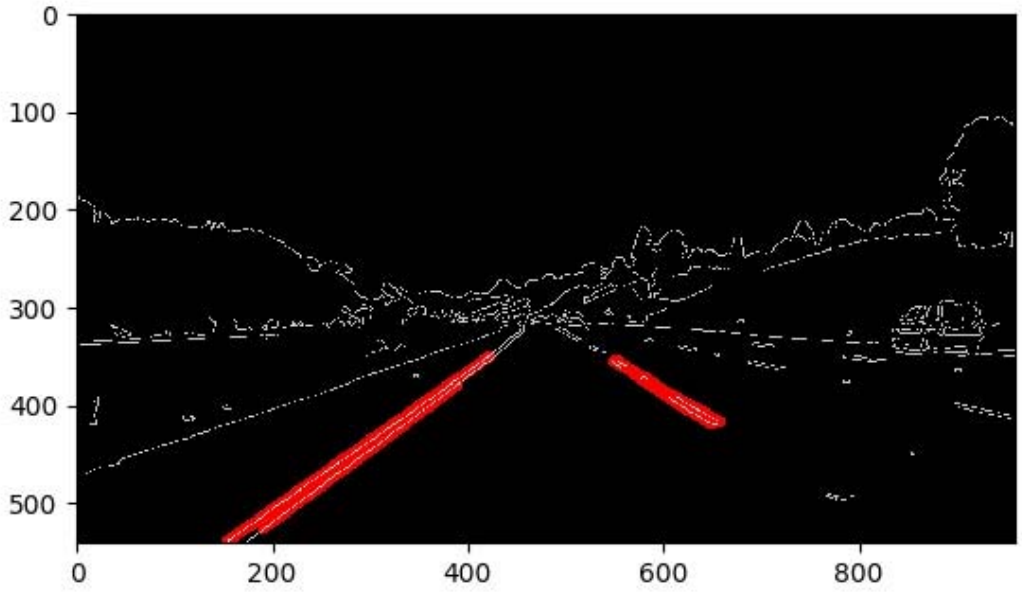
Reflection

1. Description of the pipeline.

My pipeline consisted of a few steps, most of which have already been shown and discussed in the lectures inside the Udacity classroom. The steps include:

a) Obtain the image from a director and convert it to greyscale for smoother edge detection, and then blurred afterwards to further soften some of the edges
b) The area of interest (AOI) is identified and masked inside those greyscale images.

c) Hough lines are then detected inside the masked AOI, and they are shown below



d) The output from Hough line processing are lines (x and y coordinates) detected in the AOI.
e) Once the lines are obtained, their slopes are calculated to check which lines are on the left or right. Each set of positive or negative slope-lines' coordinates are then put through a line-fitting routine using python's built in np.linalg.lstsq() function. The function return slope of the line and bias. Using this slope and bias, points are chosen at the bottom of the image and in the middle to draw lines, as indicated below

f) Afterwards the same process is done on individual frames within video clips

2. Identify potential shortcomings with your current pipeline

Most of the settings for the Hough line and polyfill () calculations are based on parameters which are fixed for one set of videos, and would need to be changed. This quite clearly happened while processing 'challenge.mp4' video. In that case, older settings from previous video clips were not suitable.

Another shortcoming is that if line detections settings are too strict, then cv2.HoughLinesP() might not return any line. In this case, there will be no lane to be seen in real time.

3. Suggest possible improvements to your pipeline

First improvement will be to make process_image() function accept more arguments and not just one argument of image. In this way, all the videos can be passed a set of values related to edge detection and Hough lines calculations.

There is no check if the case arises that no lines are being detected in an image after Hough-related calculations. Right now I have chosen values suitably so that this situation will not arise. However a proper check against this situation is needed. For example, the lines form previous image(s) can be used in this scenario.

Most of the video challenge.mp4 is handled correctly. However, tyre skid marks are often taken as lines in certain segments of that clip. I thought about filtering using of current and previous lines to detected those outliers (for example 3 sigma outlier detector). Due to time shortage I have not carried it out, and hence *did not submit* an incomplete solution.