Finding lane lines on the Road

SDC ND Project 1 – Writeup

The goals / steps of this project are the following:
* Make a pipeline that finds lane lines on the road
* Reflect on your work in a written report

Reflection:

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

There are two parts to this problem: Information extraction and Processing it to draw the needed lines

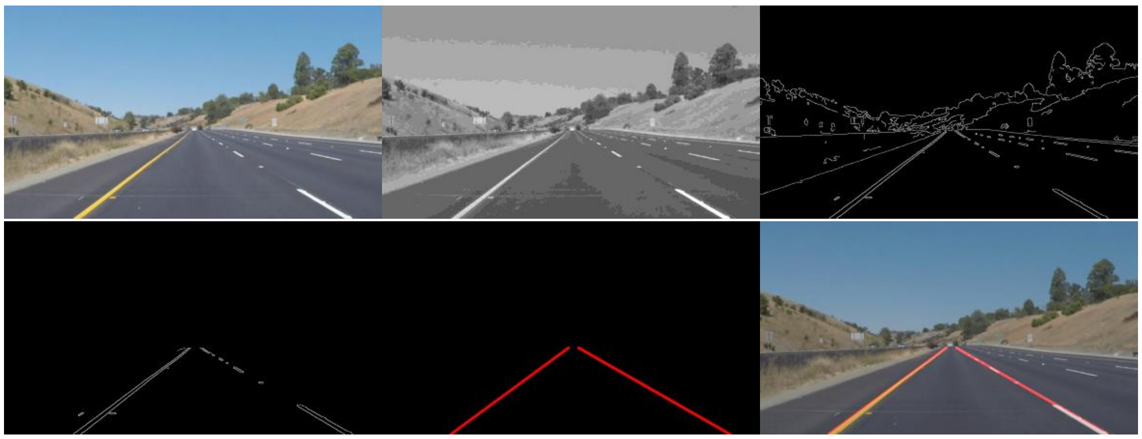
A. Information Extraction:

1. This part of the pipeline converts the image to grayscale, so that it can be passed to the Canny edge detection function, to extract only the edges from the image, after it is blurred to reduce noise.
2. The output of the above step is fed to a region selection function, that removes all pixels except the ones in the narrowing region of interest where we expect to find the lane lines ahead of the car.
3. Once the edges are masked to only the region of interest, Hough transform technique is used to detect lines from the edge pixels, which is fed to the second part of the pipeline - data processing

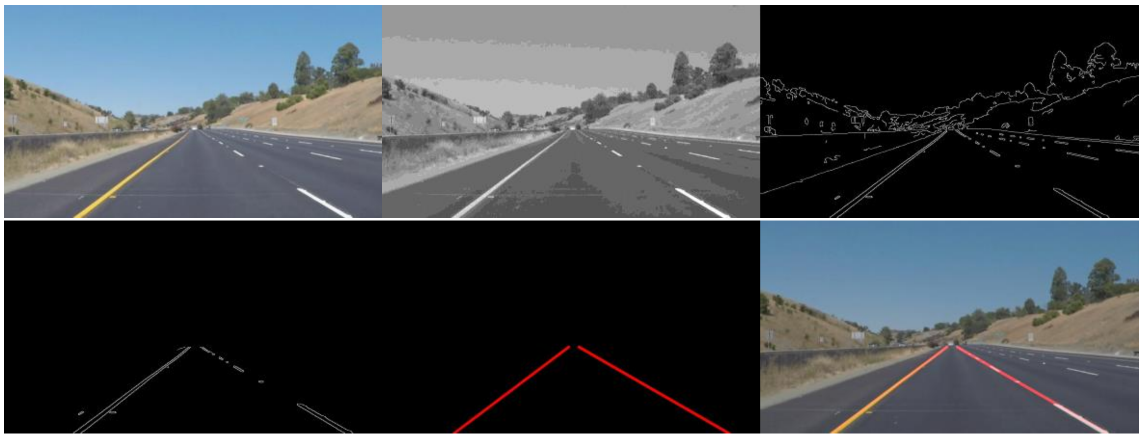
B. Processing:

1. The lines (point pairs) returned by the HoughPLines function need to be separated between the ones belonging to the left lane and the right lane. Several different approaches were tried:
   1.1. **Slopes**: Left lane lines would have a positive slope and the right ones, negative. Problem with this approach is when the camera is aligned too close to a line, the slope becomes infinite and does not help with the separation.
   1.2. **Polyfit**: this approach takes all the points and tries to fit a line between all points. This didn't work well due to the nature of how polyfit works.
   1.3. **Left-to-left**: Any points that lie on the left half of the image are assigned to the left lane and likewise for the right.
2. Once the points are separated in the two groups, they could be extrapolated using either of the two approaches:
   2.1. **Farthest points**: Here the distance between all possible combinations between points was calculated to determine the farthest points. Once known, these points were then used to

calculate slope and intercept of the average lane lines, followed by the drawing it out on the image with the height of the image as the starting y value, until the last point found earlier.



2.2. **Polyfit**: np.polyfit was carried out to determine the best fit slope and intercept of left and right point groups, followed by drawing the lines starting at the bottom of the image to the center. For images that did not have sufficient points for a lane, the previous polyfit coefficients were used.



Identify potential shortcomings with your current pipeline

1. One shortcoming of this approach is of course, curved paths. The pipeline expects and draws straight lane lines.
2. Another issue here is that, changes in the color of the pavement and also the shadows cast by nearby objects can interfere with the detection & processing of line data for lanes.
3. Parameters need to be adjusted for every input source, depending on the road, camera and its setup.

Suggest possible improvements to your pipeline

1. Some additional image processing may help remove noise related to shadows
2. Curve fitting may be used to make the pipeline more robust and make it work on turns