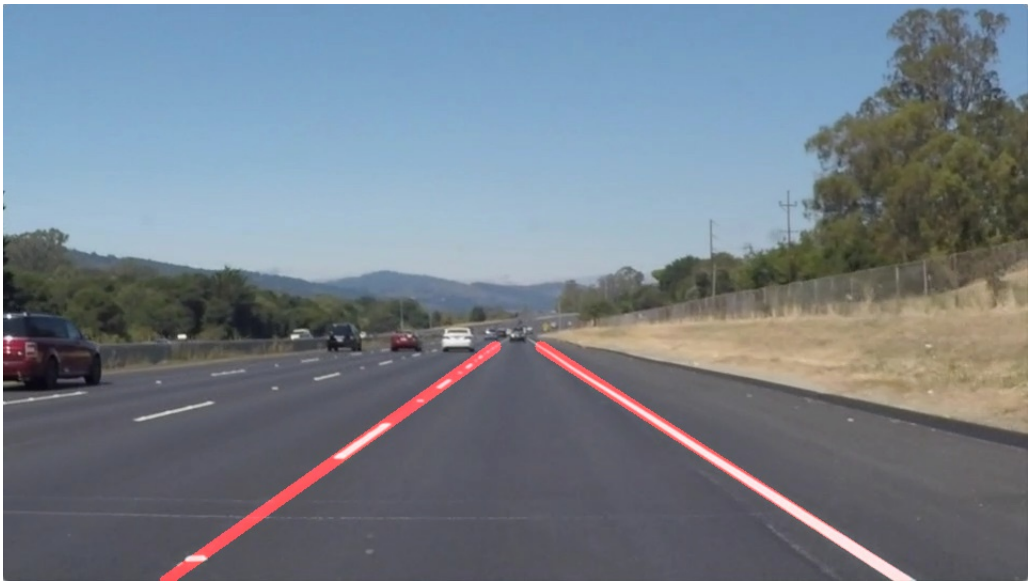
Finding Lane Lines on the Road

Udacity CarND



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.

This project will detect lane lines in images using Python and OpenCV. In addition, some test still images and videos are given.

Reflection

1. Algorithm Pipeline

Lane line detection algorithm pipeline consisted of 5 steps:

- convert the input RGB image to grayscale
- apply Gaussian smoothing the grayscale image
- apply Canny edge detection algorithm, and get edge image
- define a Region of Interested (ROI) area, and generate a mask based on ROI and edge image
- use Hough Transform to detect lines on mask image, and generate the result

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by following algorithm:

- iterate all lines detected from Hough Transform
- for each line, calculate the slope and intercept based on $slope = (y_2 - y_1)/(x_2 - x_1)$ and $intercept = y_1 - slope * x_1$
    - classify the line into right line if $slope >= 0.5 \; and \; slope <= 0.9$
    - classify the line into left line if $slope >= -0.9 \; and \; slope <= -0.5$
- calculate the average of slope and the average of intercept in left line set, and draw one left lane line
- calculate the average of slope and the average of intercept in right line set, and draw one right lane line

2. Identify potential shortcomings with current pipeline

Some shortcoming could be:

- low generalization capability
    - Current pipeline is only using basic digital image processing method. There are too many handcraft feature and empirical threshold values.
- the system won't work if the quality of image is not that good
    - foggy or raining
    - night
- the result on challenge video is not good enough

3. Suggest possible improvements to your pipeline

Some improvement could be:

- calculate the final line based on regression instead of averaging slope and interception
- include memory, which means prediction on current frame is also based on previous frames
- use a machine learning framework instead of traditional image processing method
- use high level feature to make system more robust
    - deep convolutional neural network