Project : Finding Lane Lines on the Road

The goal / steps of this project are the following

- Make a pipeline that finds lane lines on the road

Overview

When we drive on the road, logically the driver see the road environment first. And then, drive can plan to move and control the car by following the lane center. Therefore, finding lane mark on the road is base and important thing for autonomous driving.

In this project, I wrote the code that find lane mark using Python and library modules. My software pipeline consist of Gaussian blur filter, ROI filter, Canny Edge and Hough Transform algorithms. And I inserted new function and modified function provided in this project(helper functions). I newly created line_estimation() function to estimate lines that is founded by Hough Transform function.

Included files list

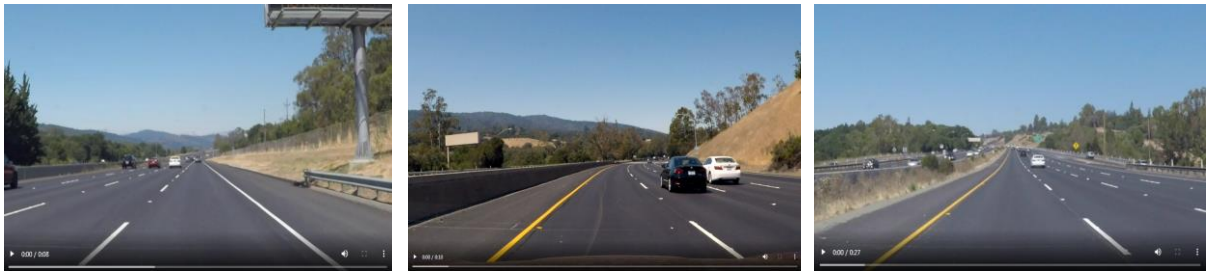
- *P1.lpython (code)*
- *test_image_output:*
  - 'solidWhiteCurve.jpg' / 'whiteCarLaneSwitch.jpg' / 'solidWhiteRight.jpg'
    'solidYellowCurve.jpg' / 'solidYellowCurve2.jpg' / 'solidYellowLeft.jpg'
- *test_video_output*
  - 'solidWhiteRight.mp4' / 'solidWhiteRight_no_LPF.mp4' 'solidYellowLeft.mp4'

Pipeline

1. Color Filtering
2. Canny edge Filtering
3. ROI(region of interest) Filtering
4. Hough Transformation
5. Lines estimation(averaging & extrapolation) and draw the lines

Input Image and Video

Image



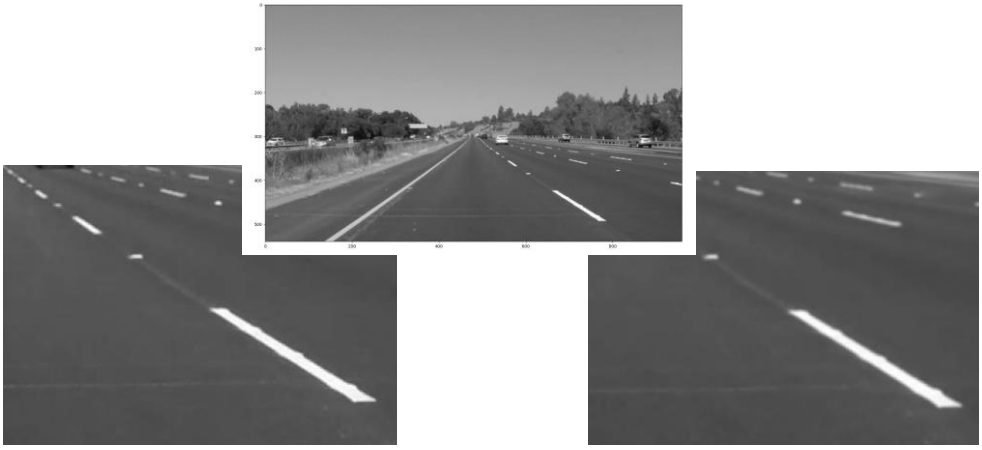
Video

Detail Pipeline

Color Filtering

✓ Transform the RGB image to gray scale image for canny edge processing.

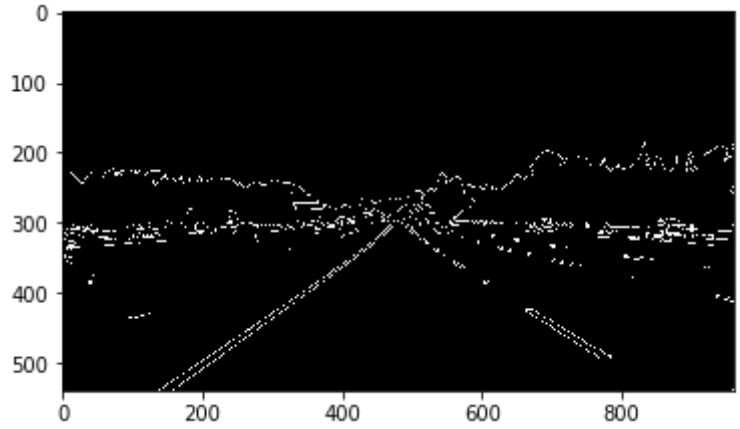
✓ In order to detect the edge, we need to filter image to smooth. Gaussian blur filter.



Initial image *Blur image*

Canny Edge Filtering

✓ With low and high threshold value, Canny edge detection algorithm execution.

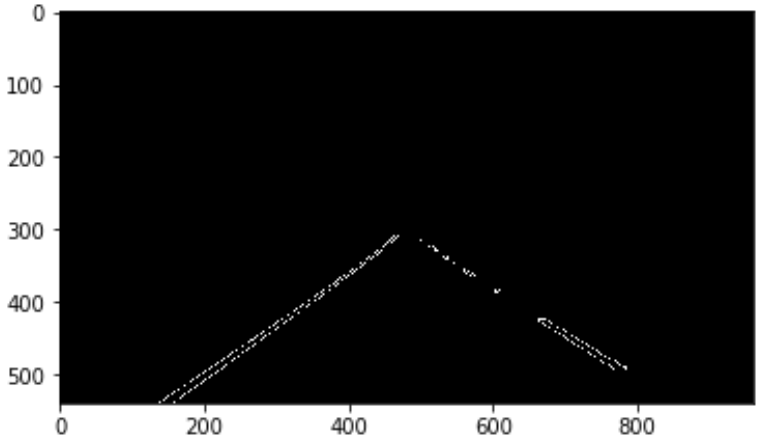


ROI Filtering

✓ Now, in the image, edge only exist. But, so many edges exist. Also we interest the specific region in the image.

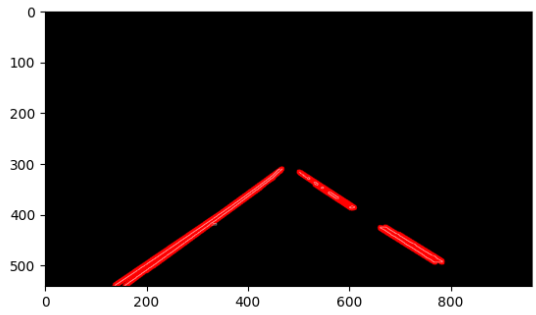
✓ So, we will extract only a specific area of the image and use it in the next algorithm.

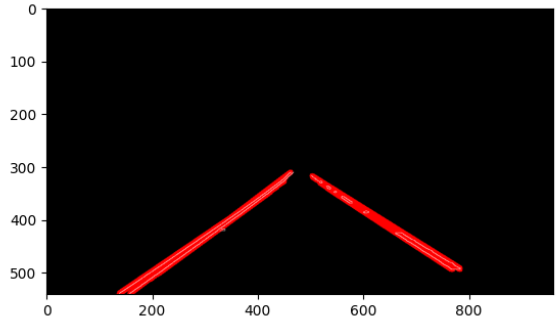
✓ We will define the ROI as a polygon shape and combine it with the currently filtered image above.



Hough Transform

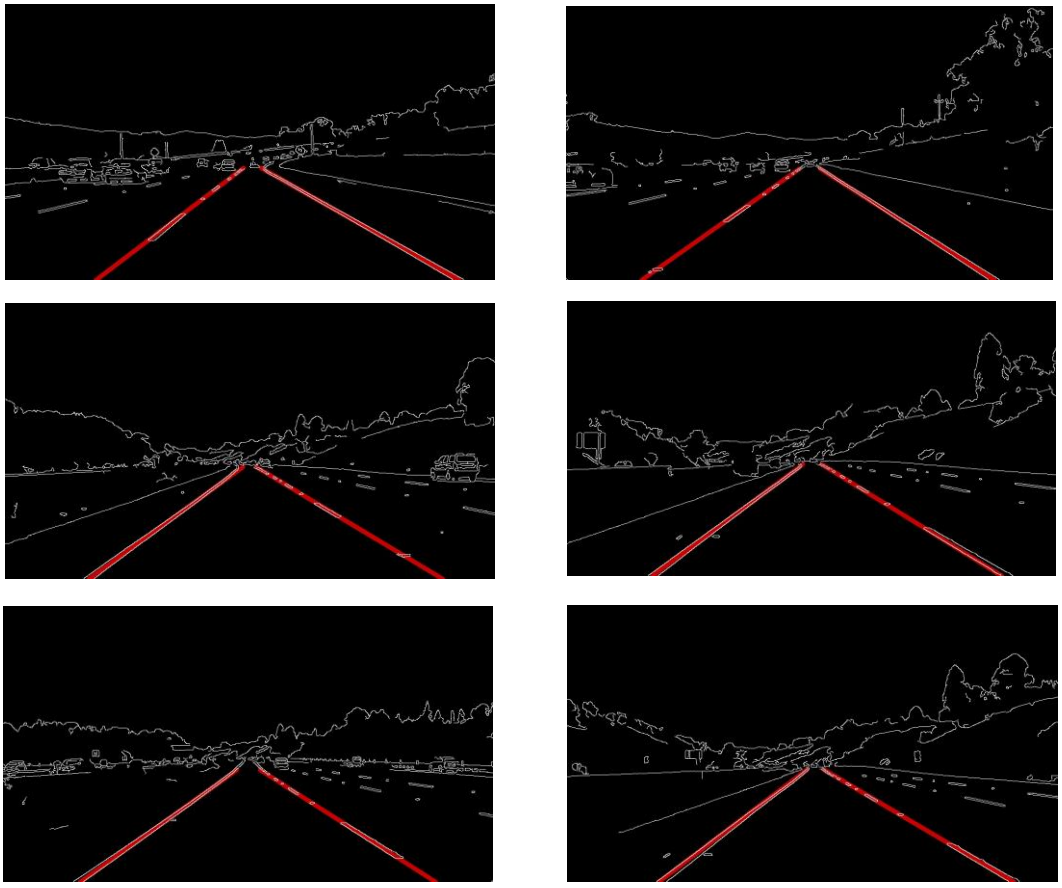
- ✓ In the image obtained through ROI filtering, we now need to find a straight line.
- ✓ Analysis from pixel space to Hough space is performed through the Hough transform. (rho, theta)
- ✓ The converted image is as follows picture. It may appear as three lines, but in data is output as multiple lines.
- ✓ In addition, as shown in the figure below, in the case of a dashed-line, it will appear uncontinuous.
- ✓ However, if you do parameter tuning, it may appear as shown in the picture on the right. Since this is a special case, a separate line estimation is required.





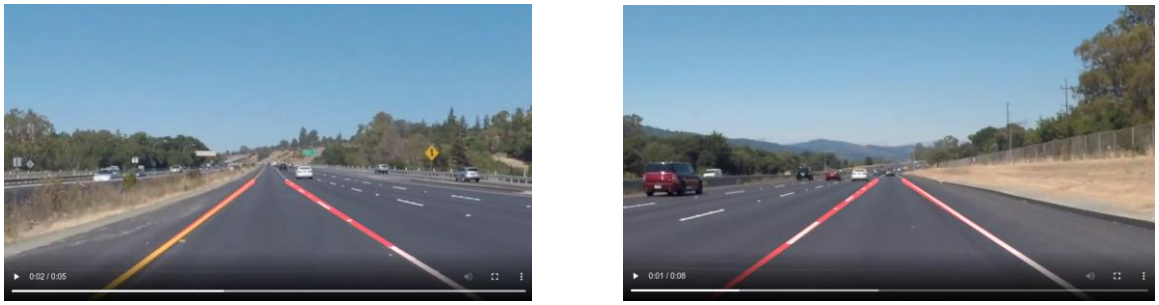
Line estimation and draw the line

- ✓ The line estimation and line drawing functions are called by the hough transform function. But it's an important part, so I'll explain it at the end.
- ✓ The line drawing function draws two end points of each straight line extracted from HOUGH.
- ✓ Here, through the slope and intercept values of each straight line, you can exclude lines that are unnecessary or noise.
- ✓ Also, the left lane and the right lane can be identified through the slope.
- ✓ The line estimation function estimates one extrapolated and continuous line by calculating the average of the slope and intercept values of each line.
- ✓ However, as you can see from the test, this simple estimated straight line causes very chattering in the video input.
- ✓ Therefore, the influence of chattering can be reduced by using a low pass filter.



Test on Videos

For testing in the video, the pipeline code is executed every frame and the result is output. The results output through the line estimation algorithm described above show more robust lane detection.



The video can be viewed through the link below. (If the link doesn't work, copy the address and check it in your browser) The video consists of a total of three.

https://youtu.be/-JxcuLNGgn0

https://youtu.be/y5mijSuAC9c (none Low Pass Filter)

https://youtu.be/5XHwSMI5q7U

Potential Shortcoming

Decreased recognition performance due to lighting

Depending on the brightness, it may be difficult to distinguish between lanes in the image.

Decreased perception of faint lanes

If lanes are cleared and appear blurred, they may not be recognized

Inability to recognize curved lanes

Since it is a straight lane recognition algorithm, it may not be able to recognize curved lanes.

Possible Improvements

Improved recognition performance for shadows

It will be possible to improve recognition performance through parameter tuning in an environment such as challenge.mp4 video.

Added recognition function for curved lanes

A detection technique using a third-order polynomial for a curved lane in an environment such as a challenge.mp4 image will be applicable.