Finding Lane Lines on the Road

Finding Lane Lines on the Road

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

Solution implemented in Python 3.6. The Jupyter notebook is available at…

https://github.com/JesseR2/CarND-LaneLines-P1/blob/master/P1.ipynb

Sample output videos at…

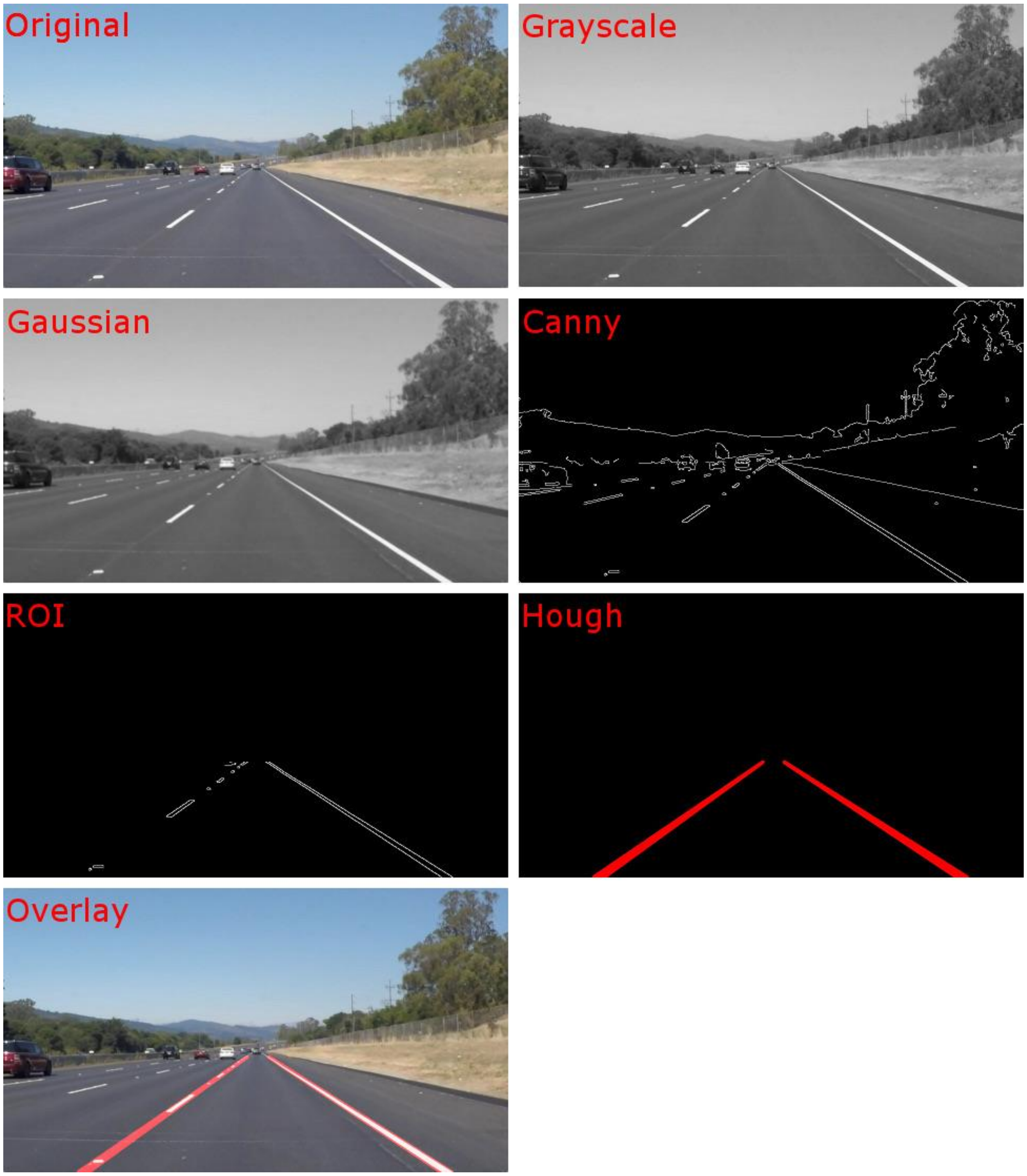
https://youtu.be/w-Opy34AP7A

https://youtu.be/QbBABMVB2Ww

1. Pipeline Description

- Convert to grayscale – 'cv2.COLOR_BGR2GRAY'
- Gaussian blur – 'cv2.GaussianBlur' with a 5x5 kernel
- Canny edge detection – 'cv2.Canny' – parameters 50, 100
- Region Of Interest – array of vertices used to mask trapezoid around road
- Probabilistic Hough – 'cv2. HoughLinesP' used to find lane lines
- Draw average of lines – 'draw_lines()' – modified version detailed below
- Overlay Lines – 'cv2.addWeighted'

Samples images at each stage of the pipeline are shown below.



2. Modifications to the draw_lines() function.

The original draw_line() function drew all Hough lines on the image, many of which were not valid lane lines. I rewrote the function to filter lines by slope removing any that were outside the expected range. Then I grouped the lines into left and right side lines and extrapolated them up and down to the edges of the ROI. I averaged all left side and all right side vertices and drew the resulting lines. To improve the appearance of the lines I added two additional lines to each side. These had the same upper vertex but new lower vertices to each side of the actual lower point resulting in lines that had more apparent depth.

3. Testing

The pipeline was tested in several sample images as seen in the notebook (github link above) and on two short video clips. Lines on the video were implemented frame by frame and did not use any averaging over time.

4. Identify potential shortcomings with your current pipeline

This pipeline was a good introduction to some of the OpenCV tools and basic image and video processing but the pipeline is not very robust in its ability to process images that are not similar to those provided for the project. Some examples where it might fail include changes in camera setup, scene lighting, scene weather, road markings, curvature of the lane, etc.

5. Suggest possible improvements to your pipeline

- Separate processing of left and right lines
- HSV or HSL processing
- Active road color values identification with color mask
- Using data from previous frames to smooth the video
- Detect line position at various distances and used splines to draw curved lanes
- Add a neural network based lane detector