Finding Lane Lines on the Road

—by Meng Wang

Finding Lane Lines on the Road

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

Before read on:
Most of the time I was working on the FLL-1.py file, only when I start to do the writeup, I noticed there is a jupyter notebook template, and the code structure there is very different from mine. Although I have migrate my code to fit the notebook template, it still look a bit messy, and I'm already tired with trouble shooting with jupyter notebook, it's not as convenient as my python IDE, so I give up on realizing the smooth filter in that code, thus the smooth filter feature is only available in the FLL-1.py.
I put a lot of effort on FLL-1.py, for the sake of practicing my coding skill, if you're willing to help me on improve coding style, please do review that file, run it in the project folder. It gives a cool feature that can output video at different stages, just need to play with the output_level parameter when initialize the LaneDetect object.

Reflection

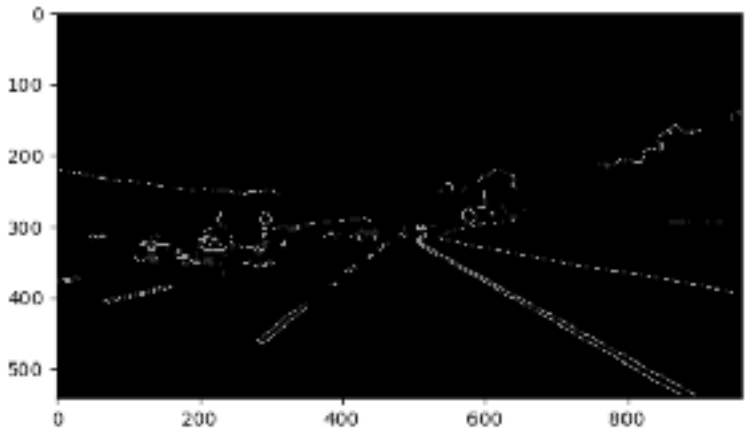
1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consisted of 7 steps:

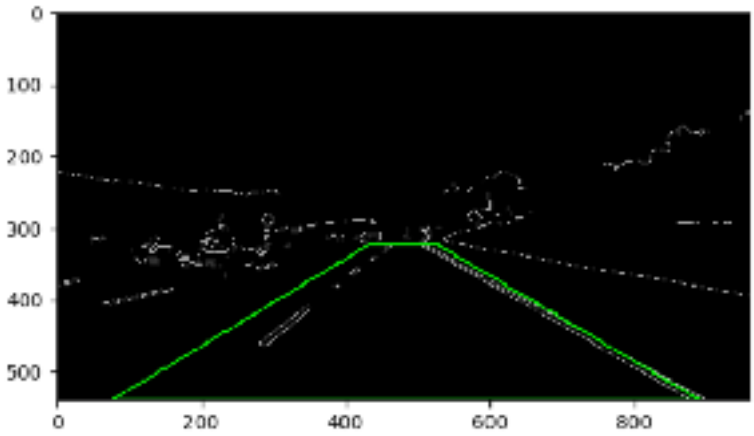
1. Converted the images to grayscale

2. Apply Gaussian blur

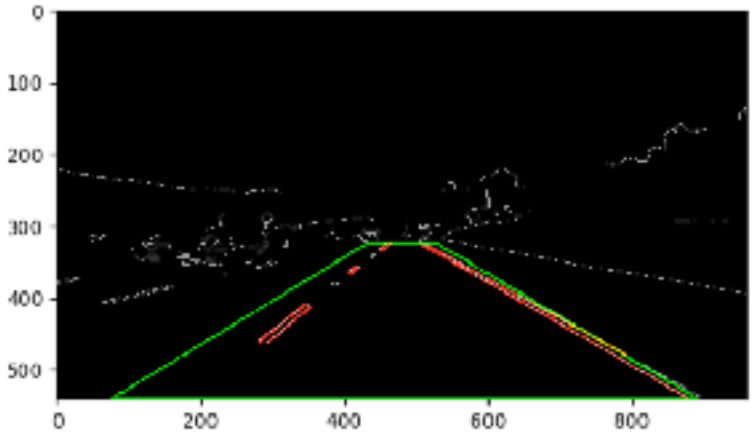
3. Using Canny edge detection to find edges



4. Apply a polygon to mask unwanted edges



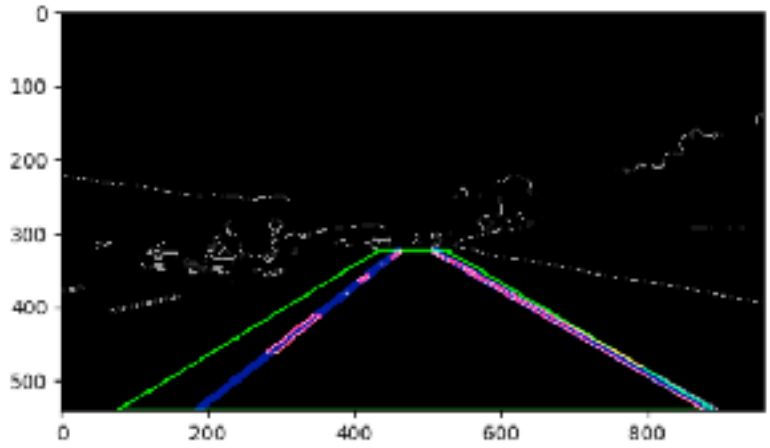
5. Find lane lines
   5.1. Find lines using Hough line detection method



   5.2. Find left and right lane lines using the vertices of the detected Hough lines.

6. *Smooth the result by average filter (this feature is only available in FLL-1.py)*

7. draw lanes on the original image



In order to draw a single line on the left and right lanes, I modified the hough_lines() function to produce the lane lines by modifying another function process_lines(). the function process_lines() would take the hough lines and return calculated lane lines using a new function find_lane(). The find_lane() function use liner regression of the hough line vertices to do the job.

2. Identify potential shortcomings with your current pipeline

Currently this algorithm rely heavily on Canny edge, if the contras is low, e.g. there's shadow on the ground like challenge.mp4, it may not be adapted to find the correct edges, although tuning parameters may make it better, but still not robust.

The filter I applied was only to average the result, not picking out obviously wrong result.

The polygon area is hand picked, may not work well in the turn.

The lane lines are approximated by straight lines, it could be fit with curves, another shortcoming is that when doing regression, it only take the Hough line's vertices, there're not many samples.

3. Suggest possible improvements to your pipeline

According to the shortcomings above, possible improvements could be:

1. Using color features to distinguish lane lines and other objects
2. Filter out the obviously wrong lane result, or restrict the result to a certain interval
3. Adapt the polygon region with the calculated lane result, assuming the lane won't give a sudden change among a few frames.
4. Approximate the lane lines to higher order curves.
5. the lane lines are actually an area space, take all the points in the area, and fit a curve cross all the areas, or at least all the points on the line boundary if for performance concern, not only some vertices.