Finding Lane Lines Project

Samuel Navarro
Udacity Self Driving Car ND

May 6, 2019

1 Description

I wanted to make the code in both C++ and Python. I'm a beginner at programming with C++ and because OpenCV has a very nice API both in Python and C++ this was a great oportunity for me.

However, this made me to waste a lot of time building the pipeline in C++ and because I didn't wanted to be more delayed, you can see that the Python code is very basic (and pretty ugly actually). But I just wanted to keep up and planned to make the Advanced Lane Lines with better code both in C++ and Python.

2 Pipeline

The current pipeline is as follows:

- Get edges with Canny edge detection after we convert the image to gray scale.
- Mask the image on the desired region.
- Get both lines (right and left) of the image based on the slope.
  - The criteria was hardcoded and there is room for improvements here.
- Get x_1, y_1, x_2 and y_2 from the right and the left line.
- Fit the right and left lines with from the points that we obtained previously with 'cv2.fitLine'.
- Compute the lines to be drawed based on the average of all the previous lines.

3 Reflection

One of the problems I encounter it was that there where frames when you find that the fited line is not on the lane. Because of that I knew that I needed some kind of average. I think that a simple average would do part of the job but that wouldn't be enough because you could not deal with fast changes in the lines in the future. For that reason I believe that drawing the lines based on Exponential Moving Average would be betters because you can give more weight to the recent observations. For now, I proceeded with a simple average of the fitted lines both in the left and the righ line.

Another problem I saw was the fact that the frames could be from different sizes. At the beginning I hardcoded the variable y_fin so it can be roughly at the middle of the image. But when I started to work on the challenge video, I notice that the frames where different and in a real life scenario they will vary between videos. So I divided the frame.

3.0.1 Improvements

Some of the improvements are:

- Better criteria to divide the lines between right line or left image. Right now is hardcoded.
- Get the lines with EWMA; not just a simple average.
- Because I'm always curious about the differences between run you code with GPU or CPU, a GPU implementation would be desirable.
- Benchmark code between CPU, GPU and C++ vs Python implementation.