UDACITY

‹ Return to Classroom

Finding Lane Lines on the Road

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

You asked:

> I would be interested how my coding style/level can be compared to a regular software engineer. In general the question would be: How much is missing to start a career as software engineer?

Your code style is pretty good. It is well commented, contains exception and handles corner cases - well done! As a suggestion, I would add a detailed description of each function in the `Image` class as you did it of the `Line` class. Overall great job with coding style and project itself! Congratulations and good luck on your next project/term!

Required Files

The project submission includes all required files:

- **Ipython notebook with code**
- **A writeup report (either pdf or markdown)**

All files are provided!

Lane Finding Pipeline

The output video is an annotated version of the input video.

Good job with annotated video!

In a rough sense, the left and right lane lines are accurately annotated throughout almost all of the video. Annotations can be segmented or solid lines

Visually, the left and right lane lines are accurately annotated by solid lines throughout most of the video.

Lines are a little bit jittery but each is a single solid line generally centered on the actual lane lines. It looks almost as in `P1_example.mp4`!

Reflection

Reflection describes the current pipeline, identifies its potential shortcomings and suggests possible improvements. There is no minimum length. Writing in English is preferred but you may use any language.

You are right that algorithm can be improved a little bit. It is not so good in the following conditions:

- curved lane
- shadows on the road
- reduced visibility due to inclement weather conditions

You can see it if you apply it to the last optional video without any changes.

These problems can be resolved by applying some of the following items (actually, some of them will be used in the advanced lane finding project):

1. Play more with region of interest or use two separate regions of interest around lane positions instead of one.
2. Reject all lines with slopes out of an acceptable range.
3. Filter hough lines and discard those lines that are not crossing bottom and top lines of selected region.
4. Play more with color processing to reduce shadows effect or different lane colors. For example here you can look for colors and brightness and dropping all obviously incorrect dark colors.
5. Also spline interpolation may be good for curved roads.
6. A low-pass or high-pass filters can be added to reduce the noise

7. Use second-degree polynomial for curved lanes.

I encourage you to tune your pipeline to implement a better algorithm. You can for example try to implement algorithm provided in this question:
http://stackoverflow.com/questions/36598897/python-and-opencv-improving-my-lane-detection-algorithm

Here is also an excellent paper about detecting curves and faded lines:
http://airccj.org/CSCP/vol5/csit53211.pdf

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START