UDACITY

‹ Back to Self-Driving Car Engineer

Finding Lane Lines on the Road

REVIEW

HISTORY

Meets Specifications

I had a lot of joy reviewing your work :)

Hope to review some of your future works friend,

Farewell,

Sedar,
Twitter: @sedarolmez

Required Files

The project submission includes all required files:

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

All files were submitted correctly.

Lane Finding Pipeline

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

Really excited when reading your code, I quite enjoyed how you tested various parameters on the images, the reduced region of interest, and also the "Hough Lines" test! You've managed to successfully test various values for the parameters and chosen the correct values for optimum output, your code gives me the assumption that the value you use for the gaussian_blur function is `kernel_size = 5`, you have also realised that it being odd is inadmissible, this is because the value 3, 5, 7, 9... for this particular domain blurs the image to the right amount for the Gaussian Filter to be optimum. However, to reduce noise, I would slightly reduce the `kernel_size value to 3`.

Anyway, you may read more about the OpenCV API at:
http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=gaussianblur#gaussianblur

Not many students are able to ensure that the hough lines for the videos "white.mp4" or "yellow.mp4", but you have, which is quite intriguing :) even though a few minor changes for your parameter values like your threshold = 15, min_line_length = 40, max_line_gap = 15 has a inconsistency, i.e. how can you assume that the lane line will be a small threshold of 40? in length, a threshold would require you to test a set of values within a min and max threshold, so a more suitable value for these variables would be, say: `min_line_len = 100, max_line_gap = 160` and for threshold itself, which is the average length of the line, for this particular domain, you would need to increase the value to roughly `30` because it is more suited for this domain and would ensure both left and right hough_lines are the same in length, plus for the extra task your lines wouldn't be centered and overlapping because threshold is used when calculating each frame milli second.

Overall, a really good solution and you have met the requirements for this specification!

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

Congratulations, for your "white.mp4" and "yellow.mp4" you have conveyed a firm understanding of the OpenCV API, you have used various method presented in the API and lane lines are annotated, great attempt on the `challenge` video.

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

Not much to say about your `draw_lines_extrapolate` algorithm it works efficiently, however, I'd like to point you towards some conventions:

I would definitely recommend reading the python programming conventions:
https://www.python.org/dev/peps/pep-0008/ just to understand the foundation of python programming, especially when it comes to repeating code when you could just make functions with various parameters and just call them assigning them to various types of the same objects,

Regarding parameter values and in the future, try to test various values within thresholds for the domain you're coding for to never miss out a particular optimal output.

Not much to say about this requirement, you have passed it with flying colours and also I have elaborated a little more on the comments above :)

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.

I would like to focus on a particular point you've made: "My drawline_extrapolate function is not good when detect curvy line." and you're right, we could use colour maps from the Open CV api which detects a change in colour within the region_of_interest and performs depending on the colour, for animals it can be heat, for snow also: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/colormaps.html

Not much to say about this requirement, you have passed it with flying colours and also I have elaborated a little more on the comments above :)

DOWNLOAD PROJECT

RETURN TO PATH