UDACITY

‹ Return to "Self-Driving Car Engineer" in the classroom

DISCUSS ON STUDENT HUB

Advanced Lane Finding

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations! You passed this Advanced Lane Finding project!

Your result is great, the video works smoothly. Your code is organized in a very good way, very clean code, I really like it.
You use the Jupyter Notebook tool to tune the parameter, that's a quite creative way to tune the parameters, it would be useful in the following projects as well.
For your pipeline, even though it works perfectly in the video, there maybe some further challenges when we apply it in real cases. The detection in each new frame may not work perfectly, you may want do some sanity check to the new result,
if (sanity_check_pass()):
use result from new frame
else:
use previous result

You can find the common used sanity check here:
https://classroom.udacity.com/nanodegrees/nd013/parts/168c60f1-cc92-450a-a91b-e427c326e6a7/modules/5d1efbaa-27d0-4ad5-a67a-48729ccebd9c/lessons/7cb63828-36aa-4cea-9239-700b5ea41f0b/concepts/7ee45090-7366-424b-885b-e5d38210958f

Sanity Check
Ok, so your algorithm found some lines. Before moving on, you should check that the detection makes sense. To

confirm that your detected lane lines are real, you might consider:

Checking that they have similar curvature
Checking that they are separated by approximately the right distance horizontally
Checking that they are roughly parallel

Overall, you did great work, I believe you spent a lot of effort to tune the video to work better. I hope you learned a lot in this project.

Good luck to you!

Writeup / README

The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled.

Camera Calibration

OpenCV functions or other methods were used to calculate the correct camera matrix and distortion coefficients using the calibration chessboard images provided in the repository (note these are 9x6 chessboard images, unlike the 8x6 images used in the lesson). The distortion matrix should be used to un-distort one of the calibration images provided as a demonstration that the calibration is correct. Example of undistorted calibration image is Included in the writeup (or saved to a folder).

Pipeline (test images)

Distortion correction that was calculated via camera calibration has been correctly applied to each image. An example of a distortion corrected image should be included in the writeup (or saved to a folder) and submitted with the project.

A method or combination of methods (i.e., color transforms, gradients) has been used to create a binary image containing likely lane pixels. There is no "ground truth" here, just visual verification that the pixels identified as part of the lane lines are, in fact, part of the lines. Example binary images should be included in the writeup (or saved to a folder) and submitted with the project.

The tool is super awesome! It save a lot of effort to rerun to code for comparison. Thank you!

OpenCV function or other method has been used to correctly rectify each image to a "birds-eye view". Transformed images should be included in the writeup (or saved to a folder) and submitted with the project.

Methods have been used to identify lane line pixels in the rectified binary image. The left and right line have been identified and fit with a curved functional form (e.g., spine or polynomial). Example images with line pixels identified and a fit overplotted should be included in the writeup (or saved to a folder) and submitted with the project.

Here the idea is to take the measurements of where the lane lines are and estimate how much the road is curving and where the vehicle is located with respect to the center of the lane. The radius of curvature may be given in meters assuming the curve of the road follows a circle. For the position of the vehicle, you may assume the camera is mounted at the center of the car and the deviation of the midpoint of the lane from the center of the image is the offset you're looking for. As with the polynomial fitting, convert from pixels to meters.

The radius of curvature values looks good now. Great work!

The fit from the rectified image has been warped back onto the original image and plotted to identify the lane boundaries. This should demonstrate that the lane boundaries were correctly identified. An example image with lanes, curvature, and position from center should be included in the writeup (or saved to a folder) and submitted with the project.

Pipeline (video)

The image processing pipeline that was established to find the lane lines in images successfully processes the video. The output here should be a new video where the lanes are identified in every frame, and outputs are generated regarding the radius of curvature of the lane and vehicle position within the lane. The pipeline should correctly map out curved lines and not fail when shadows or pavement color changes are present. The output video should be linked to in the writeup and/or saved and submitted with the project.

The video looks almost perfect now.

Discussion

Discussion includes some consideration of problems/issues faced, what could be improved about their algorithm/pipeline, and what hypothetical cases would cause their pipeline to fail.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review