UDACITY

PROJECT

Advanced Lane Finding

A part of the Self Driving Car Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Dear Student,

I loved reviewing your project you have successfully completed one of the toughest projects in Term 1.
You must have learned a lot from this project including OpenCV.
I wish you all the success for your last project which is quite easy. 😃

Advanced lane Lines with Aaron Brown

Few Articles

Robust lane finding using advanced computer vision techniques
Advanced Lane Finding Using OpenCV by Paul heraty
Experiment Using Deep Learning to find Road Lane Lines by Paul Heraty

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.

Your writeup includes statements and supporting 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).

You implemented what you have learned from the lessons. Nice work.

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.

Well done here 👏

✓

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.

You have used gradient based and color based methods to produce a binary image containing likely lane pixels.

✓

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.

Your transformation looks good. Well done.
Images have been properly rectified to a birds eye view perspective which shows the lane lines as parallel to each other.

✓

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.

The left and right line have been identified and fit with a curved functional polynomial form.

✓

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 is consistent with the strength of the curve. Good job with estimating the lane curvature and vehicle's lane placement position.

✓

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.

You warped back the detected lane onto the original image. Nice work 👏

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 image processing pipeline that was established to find the lane lines in images successfully processes the video.
Your pipeline performs well on most of the frames.

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.

You have written about the hypothetical cases which would cause your pipeline to fail.
Also you have provided good strategies to overcome it.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

☆☆☆☆☆

Student FAQ