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

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.

The implementation process is presented in the README file and example images for each stage of the implemented pipeline are provided.

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).

The process of camera calibration has been successfully applied to un-distort the test image.

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.

The distortion correction has been correctly applied to real-world images.

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 input image was converted into the HLS colour space, derivatives were computed in X, and binary thresholds were applied on each output. The combined binary image contains clearly visible parts of the lane lines.

In addition to that, you may want to try extracting the R channel from the RGB space due to its high value for yellow and white 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.

You did a good job finishing this part of 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.

The data points identified by computing a histogram were fit with a polynomial and both lane lines have been correctly identified.

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 required parameters were correctly computed providing users with information on the estimated radius of curvature and the position of the vehicle with respect to the centre of the road.

These measurements are printed on the video frame.

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 image processing pipeline does a good job processing video frames. You were able to detect curving lines and generally your algorithm does not fail when shadows or color changes are present.

There are some scenarios where the mapped area jitters. Apart from taking snapshots of frames from these particular episodes, tuning hyperparameters, and testing the modified pipeline on these snapshots; you could refer to the resources below to find some methods that you could incorporate:

http://programmingcomputervision.com/downloads/ProgrammingComputerVision_CCdraft.pdf

http://diml.yonsei.ac.kr/papers/Real-time%20Illumination%20Invariant%20Lane%20Detection%20%20for%20Lane%20Departure%20Warning%20System.pdf

https://medium.com/@vivek.yadav/robust-lane-finding-using-advanced-computer-vision-techniques-mid-project-update-540387e95ed3#.n1ph5k1og

https://medium.com/@heratypaul/udacity-sdcnd-advanced-lane-finding-45012da5ca7d#.c0ujnakj9

https://medium.com/@heratypaul/experiment-using-deep-learning-to-find-lane-lines-c668a6e42070#.5flnqnf0a

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 provided a thorough reflection on the implemented algorithm, identified its limitations, and suggested possible improvements. Well done.

DOWNLOAD PROJECT