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

Great job! Your write-up was comprehensive and easy to read. It was clear that you had a firm understanding of this module.
Can't wait to see what you build next. Keep it up! 🎉😸💭

### 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.**

Great job! 😀

Your README is well-structured and easy to follow. Thanks for providing images as examples of your project's outputs - it helps in understanding your process and pipeline!

### 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).**

Nice! ✅

### 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 undistorted test image shows that the distortion correction has been properly applied to the road images, and I can see that this was applied in the pipeline as well. Nice work!

**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.**

Great job using a mixture of features from the RGB color space and the Sobel operator.

SUGGESTION

Check out the Q&A video for a walkthrough of this project, including debugging tips for color and gradient thresholding.

**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.**

Appropriate source and destination coordinates have been selected to warp the images to a birds eye view with parallel lane lines.

**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.**

Great job!
Another way to fit a line to the points would be to use spline interpolation: https://docs.scipy.org/doc/scipy/reference/tutorial/interpolate.html#spline-interpolation

**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.**

Great work!
Check out this link for more information on the curvature of radius formula:
http://www.intmath.com/applications-differentiation/8-radius-curvature.php

Be sure to check with the U.S. Department of Transportation's website to see if your curvature seems reasonable for highway speeds:
https://safety.fhwa.dot.gov/geometric/pubs/mitigationstrategies/chapter3/3_lanewidth.cfm

**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.**

Great job! You've correctly overlaid the lanes and the curvature/offset information on the rectified image.

## 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.**

Awesome - your video shows that your pipeline stays accurate for most images, and is robust to the presence of shadows or other occlusions.

Here is a tip that can help you get faster feedback when refining your pipeline (this can be useful in the next project as well).

If you want to test the video processing pipeline on a short segment of the video without waiting for it to process the entire video, you can use the `.subclip()` method from `VideoFileClip`. For example, if you want to test the part of the video where the car goes over the second bridge, you could use: `VideoFileClip("project_video.mp4").subclip(38,43)`, and it will process the 5 second clip between 38 and 43 seconds.

## 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.**

Thanks for your thoughtful discussion!

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ Reviewer Agreement