UDACITY

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

DISCUSS ON STUDENT HUB

Advanced Lane Finding

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Greetings Student,
Congratulations, you made it!👏👏
It is obvious that you have put a lot of thought and hard work into this project, and the results are very impressive! The pipeline does an excellent job of identifying the lane lines in the project video, and the write-up is very thorough, making your project a pleasure to review! Keep up with this hard work and surely, you will make a great Self Driving Car Engineer.

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! The solution briefly introduced the steps that were taken to achieve the result and explored the difficulties and possible improvement in the whole project. Nice work!

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).

Good Job here, you successfully calculated the correct camera matrix and distortion coefficients by using all the calibration chessboard images.

Suggestion & Comment

For more information on camera calibration, the following links can be very important:

- Calibrate fisheye lens using OpenCV—part 1.
- Calibrate fisheye lens using OpenCV—part 2.
- Camera Calibration.

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.

You successfully applied the distortion correction to un-distort the real-world image. Great 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.

Good job using a combination of color and gradient thresholds to generate a binary image. The image saved does contain the lane lines. Very good work here!

Suggestion & Comment

- Has any consideration been made on using channels from other color spaces like HSV, YUV, Ycbcr?
- Here is an article from Utah State University which has a proposed architecture for lane line detection
- This article from a previous student could also aid in understanding how various combinations can play out.
- This journal gives a review of lane detection techniques.
- This article proposes a method of lane detection based on contrast analysis.

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.

It clearly shows you understand how to rectify each image to a `birds-eye view`. Nice!

Extra Material

Please, you may want to check more information on perspective transform:

- Perspective Transform.
- Perspective Transformation.
- Efficient Implementation of Image Warping on a Multimedia Processor.

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.

A nice job was done in identifying the lane lines in the rectified binary image as depicted in the solution. Keep it up!

Extra Material

Here are some resources about polynomial fitting:

- Polynomial Curve Fitting
- Discrete Mathematics: Polynomial Fitting
- Curve Fitting with Linear and Nonlinear Regression
- Fitting of a Polynomial using Least Squares Method

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, you successfully measured the Radius of Curvature and the vehicle position and shown it in the exported graph.

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.

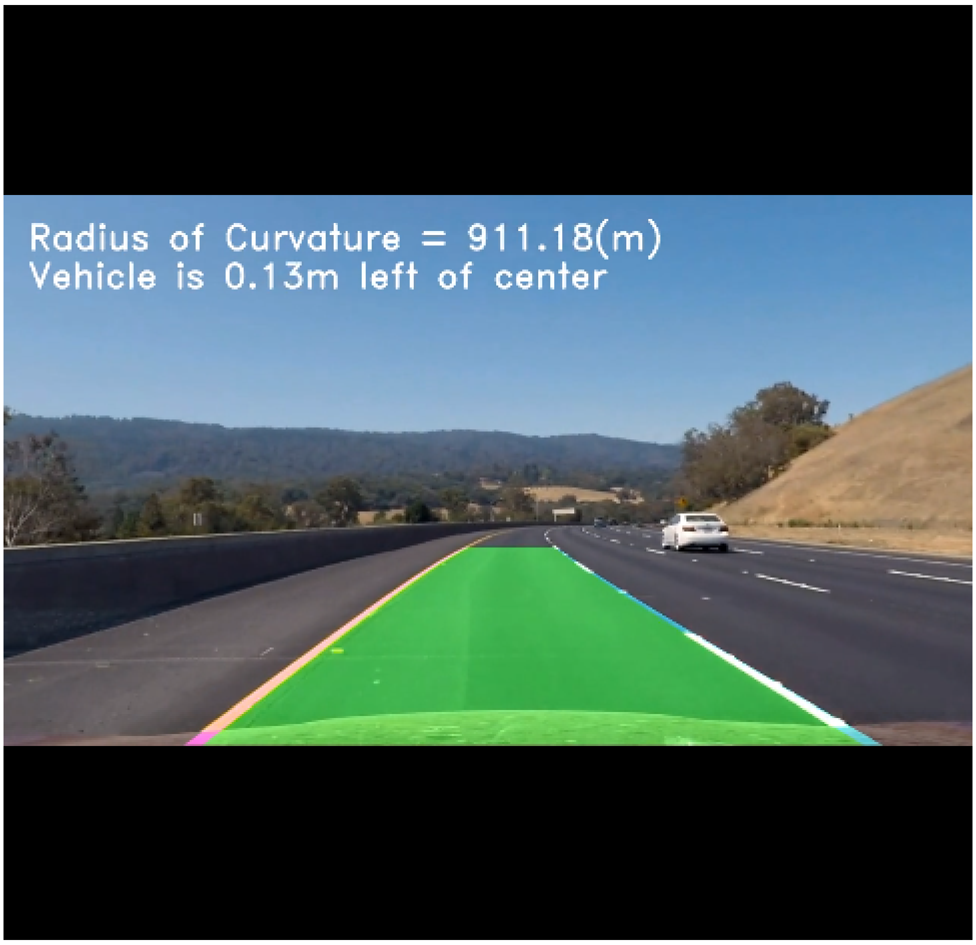
saved to a folder) and submitted with the project.

It is absolutely wonderful that you could accurately capture the lane boundaries on warping back the rectified onto the original. Well done! :+1:

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 exported accurately captured the lane lines with the instant measurements of Radius of Curvature and Vehicle Position. I have to admit that a very outstanding job was done in this part.


Radius of Curvature = 911.18(m)
Vehicle is 0.13m left of center

Suggestion & Comment.

Below I present some suggestions and links to help accomplish a robust pipeline:

- I will recommend taking snapshots of frames from this particular episode, tuning parameters, and testing the modified pipeline on these snapshots. Please, make use of the resources below to find some methods that you could incorporate:
- Programming Computer Vision with Python
- Robust lane finding using advanced computer vision techniques

- Robust lane finding using advanced computer vision techniques
- Udacity SDCND : Advanced Lane Finding Using OpenCV
- Experiment Using Deep Learning to find Road Lane Lines
- Advanced Lane Finding - Udacity SDCND

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.

It is nice to know you have planned some potential strategies to make the model more robust. Hopefully, you could implement it soon to explore the result. :)

Suggestion & Comment

Deep learning techniques can also be used in the current implementation, check out the following links to know more:

- Nvidia's end to end deep learning for Self driving cars
- Experiment using Deep Learning to find lane lines
- Deep learning technique from ford's engineers

DOWNLOAD PROJECT

RETURN TO PATH