UDACITY

Back to Self-Driving Car Engineer

Advanced Lane Finding

REVIEW

CODE REVIEW

HISTORY

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Great work with the project! You are almost there. Just further tweak your thresholding and lane finding algorithm, and you will have achieved better results 😄

```
Except that, can you please direct me to some resource for Automatic Perspective Transform poi
nt selection ?
Also, some additional resources would be really helpful for reading.
```

Once you have passed the project, we will try to provide you with some resources for these! Please make sure you include this note in your next submission as well :)

I hope the review helped you. If you feel there's something more you would have preferred from this review please leave a comment. That would immensely help me to improve feedback for any future reviews I conduct including for further projects. Would appreciate your input too. Thanks!

Hopefully, you enjoyed this project! Good luck with the next review! 😄

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.

You have a very well written writeup. Mentioning what code to focus on, relevant images, and brief explanations of what you have worked on.

I wanted to bring up a general point about writeups for such projects, which, through personal experience and talking to other students, is something I feel is important to know -

These reports/writeups are not just for the purposes of reviewing the project. It serves a more important function for you too. People often face some issue in talking about their work/projects in front of others or during interviews with sufficient details. They can come off as unprepared or not having put enough effort in their own project or worse - plagiarized them. In such cases, these writeups/reports are very useful tools in revising before an interview. You are quickly able to understand what you worked on, talk about some of the finer details, and even answer questions related to the project and how you could improve upon it. So, always focus on such writeups and how you can detail them out without overdoing it. Good practice for any potential interviews or even for updating your resume!

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 work on correctly calculating the calibration matrix and distortion coefficients!

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 was correctly applied to the images. Good job on that!

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.

Excellent work here! You combined color transforms and gradients (sobel) to achieve pretty good results.

Especially for using colorspaces like LAB and LUV. You are quite right, as per your writeup. Sometimes the S channel in HLS/HSV is less robust in shadows and colorspaces to experiment like LAB or LUV help counter that. You might be able to get some really good results without utilizing sobel at all, in fact.

To quickly try and iterate through tuning for thresholding (which personally, I found to be quite a painful task!) you can create a sort of interactive GUI using OpenCV. Check out OpenCV's `createTrackbar()` function. Here's a resource to look into this - https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_trackbar/py_trackbar.html

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 selected good source and destination coordinates to warp the images to a birds-eye view, and the lines appear to be parallel.

I would encourage you to think about what alternatives can you use here to dynamically calculate the source and destination points. Can you perhaps utilize the hough transform to identify an initial line segment for each lane line, and use the end points for that lane as coordinates to the perspective transform? Try it out (after passing the project) :)

A question for you to think about - Do you think applying the transform before thresholding would be more or less helpful than applying the transform after thresholding? Which one might be a better approach if you were to implement this on actual hardware, assuming language of choice?

Good work! 😄

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.

Excellent work by implementing the histogram peaks and a sliding window method to identify lane pixels in the warped images.

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.

Nicely done!

You correctly calculated the radius of curvature and vehicle position.

However, your printed/displayed values can be a bit on the higher side at times. You can try to fine-tune your conversion coefficients (pixel to meters) a bit if you wish to be more accurate.

Something for you to think about - Since the results update with every frame, do you think you could apply a low-pass filter (weighted average) here to smooth out these results as well? Do you think that's of any value to an actual SDC, if you smooth out these values instead of the absolute/exact value at each timestep?

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!

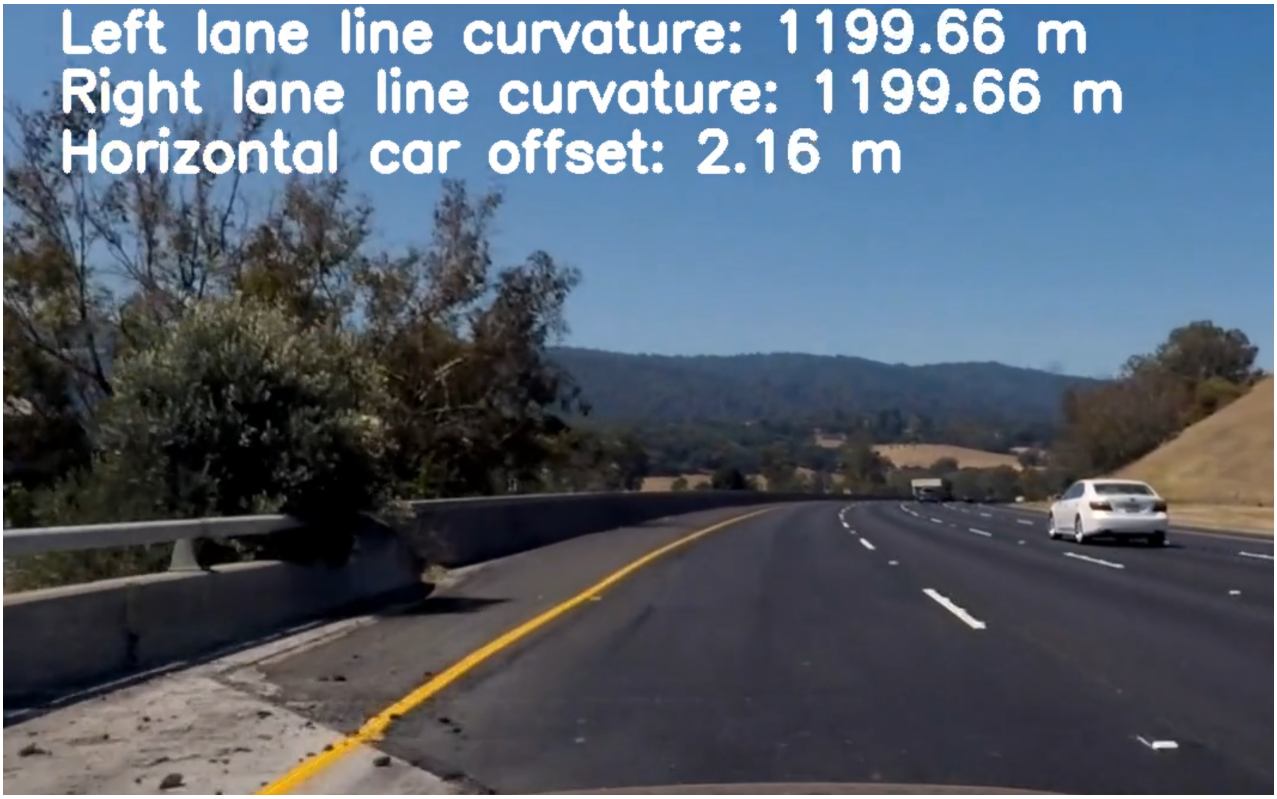
Your correct results are visible from your attached video as well.

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.

Overall, you have done some really good work here.

But, at one instance in your video, your lane detection fails completely for a few seconds (0:24 to 0:28) -



There could be several reasons for this. You can try the following out to fix this -

- Fine-tune your thresholding parameters to make sure your get a good thresholded binary image for this region. Make sure you can detect the lane lines for this part of the road.
- Recheck your sanity checks and lane deteection , especially the one you point out in your writeup - `To improve stability, after a pre-defined number of frames , blind search is done again. Currently after 75 frames.` Is this causing it to skip a few frames because the blind search, after a failed lane pixel detection, yields nothing?
- You can consider averaging over previous N frames to improve your current detection. There is less of chance of a "missing detection" with such an approach.

If you follow along your code's logic, you will be able to pinpoint this. The above should help narrow it down, but first step should be to make sure you can identify the lane pixels for those frames/images (that part of the road).

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.

Great work! The points you discuss for improvement are quite good and shows your understanding of your project.

Such details can help you further on when you wish to either expand on the project or revise it for interviews for example. And is a good exercise on how you could theoretically improve your work too!

Additionally, I would also like you to think about couple of questions (not important/required for resubmission) -

- Do you think having 3D road data/information could help produce a more robust pipeline for lane detection in actual autonomous vehicles? Or are 2D images, like for this project, reasonable for such a task in almost all cases/scenarios?

- How could or would you go about implementing the pipeline using a Deep Neural Network/Model approach instead of using classical Computer Vision? See, if you could try this out as an additional project to up your skills whenever you feel you have sufficient knowledge with ML/DL, if you'd like to!

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review