UDACITY

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

Advanced Lane Finding

REVIEW

CODE REVIEW

HISTORY

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Dear Student,

You have done some very impressive work in this project and you are almost finished. At this we just need to see a bit more improvement in the output video and hopefully my comments give you some ideas for how to make the necessary improvements so you can meet all specifications on your next attempt.

We look forward to seeing the progress you make with your next submission!😃

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.

It looks like you forgot to include your output images this time for the writeup. Please note that you will not necessarily be getting the same reviewer each time so you should be including the entire project on each submission

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).

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 distortion correction has been properly applied to images of highway driving, and I can see that this was implemented in the final 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.

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.

Good job warping the images to a birds eye view perspective!

Comment:

Ideally the lane lines will be parallel in the birds eye view images but it appears that your lines are converging towards the top of the image. If you adjust the coordinates of the perspective transform to make them closer to parallel it should help with some later stages of the pipeline.

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 using a histogram and sliding window search to find the locations of the lane lines, and using the previous detections to perform a targeted search in subsequent frames.

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.

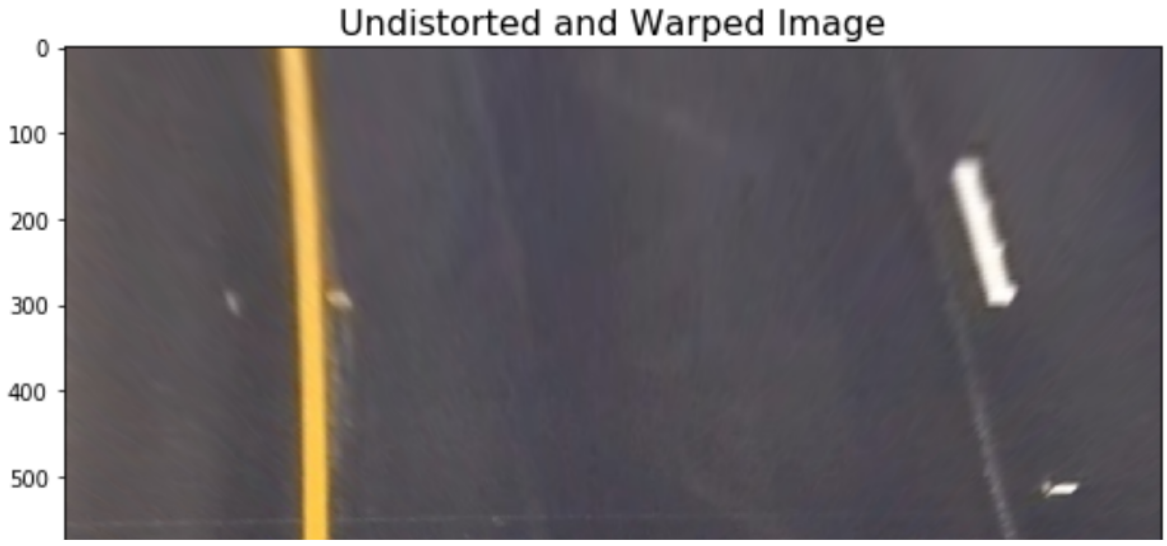
Well done here.👏 The values in the video look like reasonable estimates of the curvature of the road and the position of the vehicle in the lane.

Comment:

I still think you could be a bit more accurate with the pixel to meter conversions:

```
xm_per_pix=3.7/800, ym_per_pix = 25/720
```

If you look at this image:



The lane appears to be a bit wider than 800 pixels wide at the bottom, and I would guess that shorter than 25 meters of road are visible in front of the car.

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 output video looks great in almost every frame but there are still some pretty major errors in the part of the video with heavy shadows on the road.

Suggestions:

Because these errors are happening in the parts of the video with heavy shadows it could be that there is still some noise in the binary images so maybe you still need to tweak the thresholds a bit more. For the L channel maybe you can try putting the lower threshold up as high as 200 or even 225 and see if it helps.

It can also really help to implement some type of sanity checks which look for unreasonable detections and prevent erroneous lines from being displayed on the output video. Here are some questions you can ask about your detections:

- Are the two polynomials an appropriate distance apart based on the known width of a highway lane?
- Do the two polynomials have same or similar curvature?
- Have these detections deviated significantly from those in recent frames?

When a sanity check fails in a single frame, instead of displaying the lines that you know to have errors, you can discard the current detection and reuse the confident detections from prior frames.

You can also try averaging the detections across a series of multiple frames which helps to smooth the lines and minimizes the impact of a single bad detection.

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.

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH