Advanced Lane Finding

审阅

代码审阅

HISTORY

Requires Changes

还需满足 2 个要求 变化

Great job putting together an impressive first submission of this project!👏 This is a very difficult project so please do not become frustrated by not passing on the first try. There are just a couple of changes that need to be made, 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.

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).

Perfect, the chessboard images have been used to calculate the camera calibration matrix and distortion coefficients, and

these were used to remove distortion from one of the calibration images, demonstrating the effect.

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.

Awesome job applying a combination of color and gradient thresholds to create a binary image with clearly visible lane lines.

Suggestion:

If you want to continue to explore additional color channels, I have seen that the L channel from LUV with lower and upper thresholds around 225 & 255 respectively works very well to pick out the white lines, even in the parts of the video with heavy shadows and brighter pavement. You can also try out the b channel from Lab which does a great job with the yellow lines (you can play around with thresholds around 155 & 200).

✓

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.

Nice job with the perspective transform, the lane lines appear very close to parallel in the birds eye view images.👍

✓

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 lane line pixels in the warped binary images, and fitting polynomials to your detections for each line.

Suggestion:

After finding a confident detection in one frame, in the next frame you can skip the histogram search altogether and go straight to searching in a tight proximity around the polynomials from the previous frames.

I recommend implementing something like this:

```
if line was detected in previous frame:
```

```
      skip sliding windows
else:
      sliding window search
```

This should hopefully help prevent the lines from deviating too drastically from one frame to the next.

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.

You are very close here but there is one step that was missed in the process of converting the radius of curvature measurement from pixels to meters, and as a result the numbers showing in the video are a bit overestimated.

The problem is here:

```
left_curverad = (1+(2*left_fit_cr[0]*y_eval + left_fit_cr[1])**2)**(1.5) /np.absolute(2*lef
t_fit_cr[0])
right_curverad = (1+(2*right_fit_cr[0]*y_eval + right_fit_cr[1])**2)**(1.5) / np.absolute(2
*right_fit_cr[0])
```

here `y_eval` is in units of pixels and needs to be converted to meters, for example: `y_eval*ym_per_pix`.

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 video looks really good and is very close to meeting specifications. The positions of the lane lines have been accurately identified in almost every frame of the video, but unfortunately there are a couple of significant errors where the detected lane lines have deviated from the actual locations of the lane lines, and we will need to see these improved.

Here are some examples of the errors that need to be fixed.

Suggestions:

Because these errors are happening in the parts of the video with heavy shadows and/or brighter pavement, it will probably be helpful to continue to improve the methods used for binary thresholding to create a cleaner binary image. I hope the color channels and thresholds that I mentioned earlier can help with this.

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.

minimizes the impact of a single bad detection.

Discussion

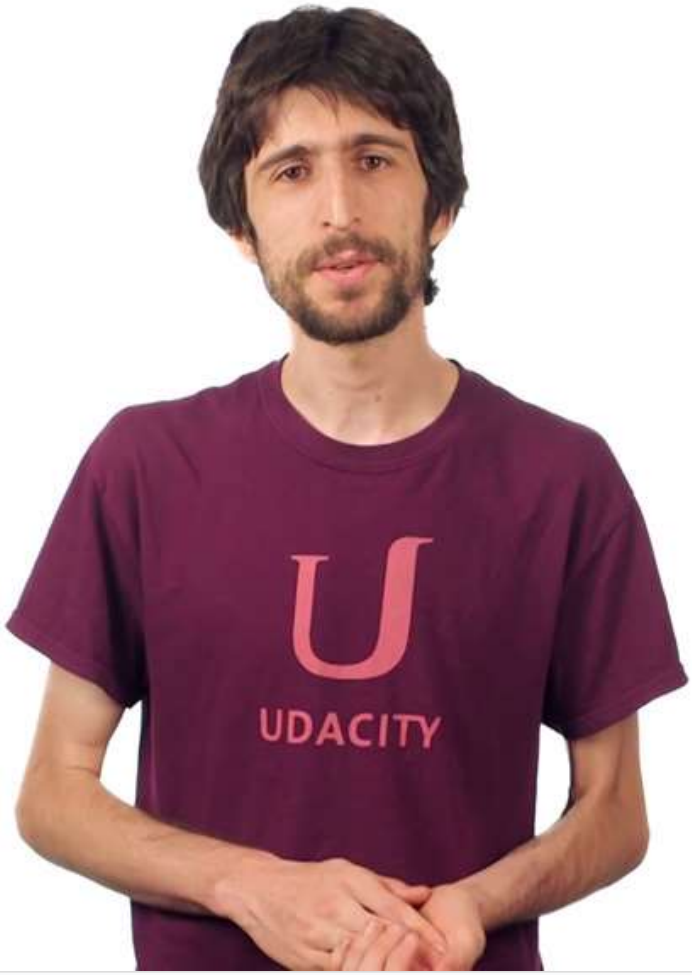
✓

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.

Good job reflecting on the project and pointing out some areas where the current implementation could still be improved.

重新提交

下载项目



重新提交项目的最佳做法

Ben 与你分享修改和重新提交的 5 个有益的小贴士。

观看视频 (3:01)

返回 PATH

给这次审阅打分