PROJECT

Advanced Lane Finding

A part of the Self Driving Car Engineer Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Nice job with this submission. You did an amazing job and your project meets all specifications defined by the project rubric for this project. Congratulations on passing the fourth project in the SDC Nanodegree and best of lucks on the ones ahead.
Cheers and Keep up the good work !!!

Here are some useful links that might help you further grasp the subject of this project
https://www.youtube.com/watch?v=hnXkCiM2RSg&feature=youtu.be
https://www.python.org/dev/peps/pep-0008/
http://stackoverflow.com/questions/36598897/python-and-opencv-improving-my-lane-detection-algorithm
https://zulko.github.io/moviepy/ref/VideoClip/VideoClip.html#moviepy.video.VideoClip.VideoClip.subclip
https://www.youtube.com/watch?v=WaxAbRy258A
https://docs.scipy.org/doc/numpy/reference/generated/numpy.hstack.html
https://docs.scipy.org/doc/numpy/reference/generated/numpy.vstack.html

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).

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.

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 "birdseye view". Transformed images should be included in the writeup (or saved to a folder) and submitted with the project.

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.

Rate this review

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.

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.

Good job, your pipeline is pretty robust, detecting the lanes even when going through shadows.

Suggestion

My intuition on the deviation you reported is that the shadow on the road is polluting the warped binary which in turn confuses the search for lane lines.

Further improving the pipeline can be made easier by creating a small subclip where the pipeline has difficulties as processing that would be much faster. For instance, to extract a 3 second video from 41 to 43 you can use:

```
subclip = VideoFileClip("project_video.mp4").subclip(41, 43)
```

The subclip of the affected areas can be used to evaluate each step of the pipeline to find out where it might be failing (thresholding, warping, lane fitting etc).

Another idea would be implementing a side-by-side visualization of each step of the pipeline such as this so you can visualize for each mistake of the pipeline which step is the root cause. This can be achieved by concatenating the image arrays horizontally and vertically using numpy.hstack and numpy.vstack.

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.

DOWNLOAD PROJECT