UNIVERSITY, SCHOOL OR DEPARTMENT NAME

Project Additional Writeup

Lei Yang

January 11, 2018

1 Advanced Lane Findings

This project is focus on using computer vision to find the lane lines. I used the suggested pipelines to finish this project: The goals / steps of this project are the following:

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

All the codes are in one ipynb file.

2 Addition writeup

2.1 Add a chessboard undistort image

The camera matrix and distortion coefficients are calculated based on the lecture. More specifically, I used 2 opencv functions: cv2.findChessboardCorners and cv2.calibrateCamera to find the camera matrix and distortion coefficients from a number of chessboard images. The camera matrix and distortion coefficients (mtx, dist) are used in later analysis. One example of undistorted chessboard image can be shown as Fig 2.1.

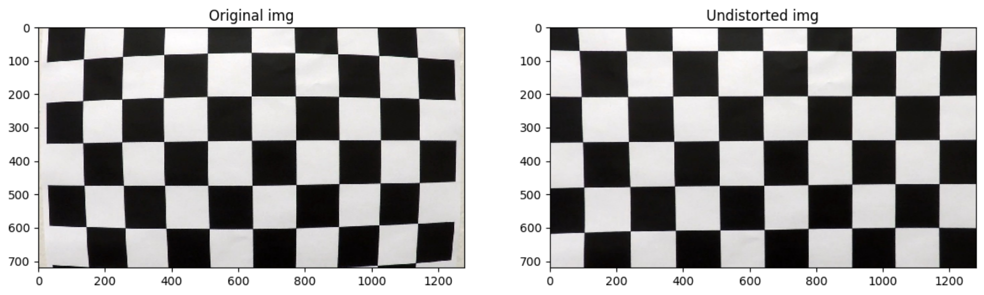

Figure 2.1: Example of undistort on chessboard. Left is original and right is undistorted image

Based on the reviewer's suggestion, I also made it clear about detected lanes. The detected lane pixels can be illustrated as Fig 2.2.
An example of image with lanes, curvature, and position from center can be illustrated as Fig 2.3.

3 Answers of the reviewer's questions

3.0.1 Almost there, all it's missing here is the use of the coefficients in meter space instead of pixel space to calculate the curvature and deviation.

- I added conversion from pixel space to meter space in [46] findLanes function. left_fit_cr = np.polyfit(lefty*ym_per_pix, leftx*xm_per_pix, 2) and right_fit_cr = np.polyfit(righty*ym_per_pix, rightx*xm_per_pix, 2)

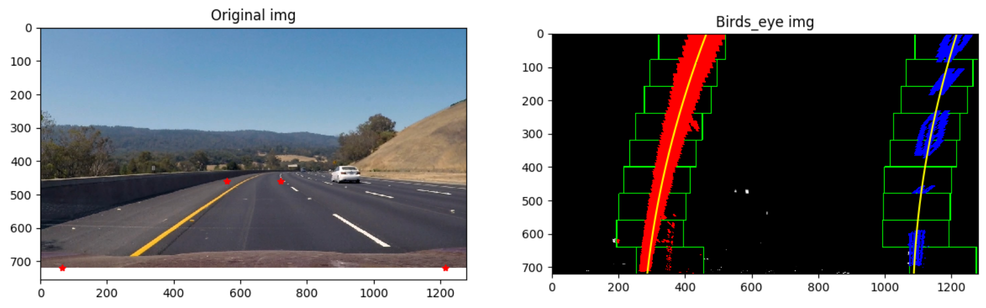

Figure 2.2: yellow lines denotes detected lane pixels

3.0.2 Please add to the write-up an image showing the annotated lanes with curvature and deviation from center.

- Annotated lanes with curvature and deviation from center is shown in Fig 2.3

3.0.3 Unfortunately, the pipeline fails to find the correct lanes on some frames. A small deviation is OK, but cases like this could cause an accident..

- The new generated video file achieved much more reasonable results. The video also shows the whole content rather than a clip.

3.1 Discussion

- This pipeline is highly limited by color thresholding performance. For example, yellow color can be best detected in saturation color space. Which means only specific color can be detected very well.
- ROI regions and other hyper-paramters have to be pre-defined. The algorithm have to be fine-tuned properly with a new camera.
- Conventional computer vision method still suffer's more with different environments. For example, low light conditions in the evening. Generally, this method is not reliable under severe condition. The machine learning or deep learning methods which could use large number of training data to handle the variety of weather and conditions.

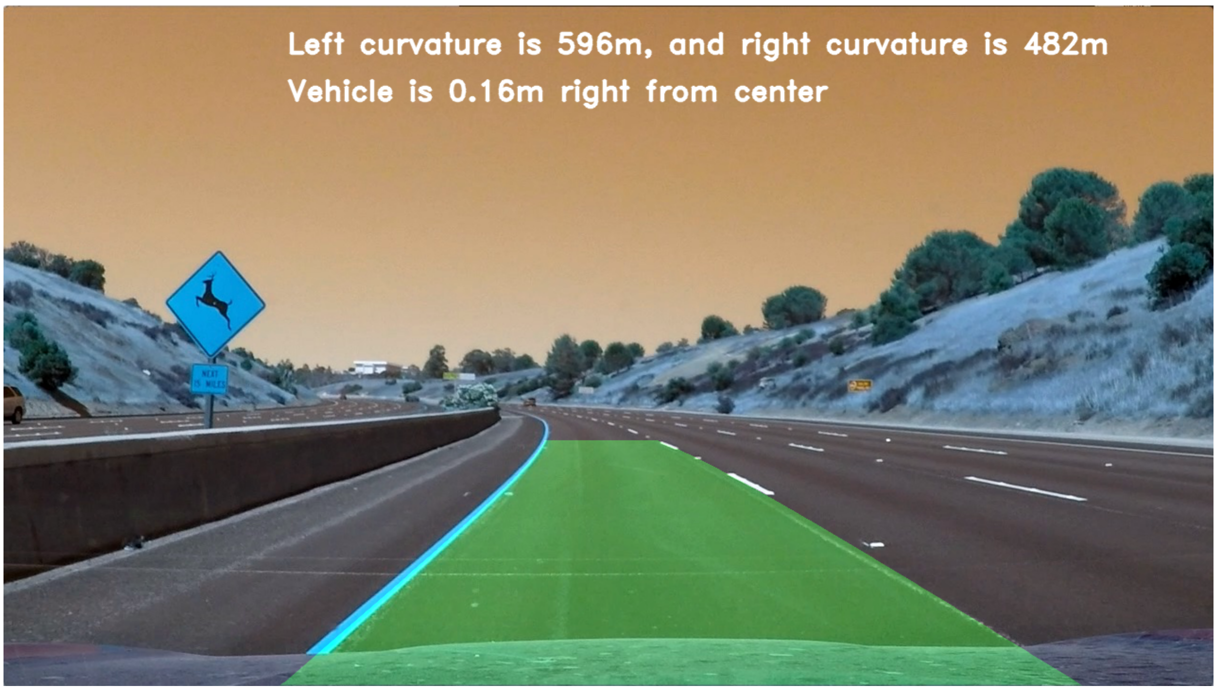
Left curvature is 596m, and right curvature is 482m
Vehicle is 0.16m right from center

Figure 2.3: an example of pipeline applied on test image