Lane_Finding_Procedure

January 29, 2017

0.1 Advanced Lane Finding

The processing steps of this project are the following:

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

0.1.1 Source code

The code was implemented in the following jupyter notebooks: 1) calibration.ipynb where camera calibration is performed. The distortion and camera matrix parameters are stored in pickle file named camera_cal.p 2) advanced_lane_finding.ipynb contains code where lane detection, curvature and car position estimation relative to the lanes are performed

0.1.2 Advanced Lane Finding Pipeline

1) load calibration matrices
2) load image/video frame as RGB image
3) apply calibration matrices to undistort image
4) apply color threshold
    - convert image into HLS color space
    - use threshold [100,255] on saturation (S) channel to extract white and yellow lines. However, shadow on road tend to fall within the same range
    - use threshold [0, 30] on hue (H) channel to filter out dark color e.g. shadow
    - do logical AND of binary images from S and H channels after thresholding to return a binary image
5) separately, transform the undistorted image to grayscale and perform sober edge detection that uses:

- gradient in x direction, threshold=(10, 255)
- magnitude threshold = (50, 255)
- direction threshold = (13.5 degree, 76.5 degree)
- these values are chosen among many possible combinations.They were selected after a trial-and-error procedure to detect the dashed lanes
- color thresholding was good at extracting the solid yellow left lane while sober edge detection was good at extracting the dashed line. Therefore, they were combined (logical OR) to give an optimal output
- In 'src_pts' array the 4 corners of the trapezoid ROI used to detect the lanes of interest
- Region masking with trapezoid is used to mask out the region which doesn't contain the information needed.

6) The binary image after color and gradient thresholding are warped using src_pts and dst_pts where the latter specifies the image positions that one wants the src_pts to warp into. The values of dst_pts shoud be chosen carefully so that the warped image capture the curve of lanes in longer distance.
7) We know road lanes are parallel to each other, so in warped image, we can assume they are parallel lines for the part of the image that is very close to the camera. We can exploit this information as follows:

    - apply another region masking in the bottom center region of warped image to remove textures/scratches in the middle of lane that would confuse the road finding
    - use bottom left and right corner of destination points for warping as starting x position for lane finding
    - Use windowing, and mean of histogram to find the next x center position towards the top of the image. Do this separately to find left and right lanes then combine them into a single binary image.
    - Then I am doing polynomial fitting on the left and right curve found
    - Correlation is performed on first two polynomial coefficients of left and right fit as these two coefficient determine the shape of the curve and the third is the y-offset of left and right lane which should be different and hence not used for correlation. Then the correlation is used to roughly determine if the lanes are curving to the same direction, if not, then perhaps lines were not found correctly and the previously found values are used instead.
    - Curvatures are then calculated
    - xm_per_pix is calculated by assuming lane width is 3.7, and divided by the distance of lane position (in pixel) in the warped image. From this, the car position offset relative to lane is calculated.
    - Finally, polygon is filled between the fitted polynomial lines in warped image. This is then unwarped and added to the undistorted color image.