Advanced Lane Finding

In this project, a robust lane detection algorithm which can also provide curvature of the road and the lateral position of the vehicle with respect to the center of the lane has been developed. In the following section, all of the criterias will be addressed. Brief explanation of how the code is working will also be provided.

The Project Rubric

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.

This document is provided for this purpose

Camera Calibration

Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.

The provided chessboard pictures first converted to grayscale. Then with the opencv function each corner of the chessboard is found. Since the distance of the each corner of a chessboard should be equal, the image is adjusted with this information via opencv function.

Pipeline (test images)

Provide an example of a distortion-corrected image.

The following image shows clearly the effect of distortion correction.

Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

The common points of sobelx and sobely is combined since the lanes can appear on both of them clearly. With taking only common points, some of the distortion is removed. The same idea is used for magnitude and direction of the gradients. Since the lanes are expected in a certain angle, this information is utilized. However this information alone is too noisy. To remove it, the magnitude of the each gradient is also thresholded and only common points are used. On some sections of the road, color is really helpful since it doesn't affect from shadows etc. Because of this, HLS and Lab colorspaces is used to get the lines. The three combined images are also combined to add up all information from each combination. Few of the examples are given below. The results can be seen under output_images folder or in the LaneFinder.ipynb

Describe how (and identify where in your code) you performed a perspective transform and provide an

example of a transformed image.

On one of the corrected images with straight lanes, four points on the lanes are chosen. They are ([574,466],[710,466],[1069,701],[232,701]). Then, another four points are chosen to create a shape of a rectangle. These settings can be found in the function called as warpSettings. The result of this function can be seen below, applied to a combined binary image and the original image.

Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?

First, bottom half of the image is taken, and a histogram of number of active pixels on the y direction is calculated. On the histogram graph, maxima points are found and those are assumed as the mean of the lanes. Then the total image is divided in 9 regions. For each region, a small margin of pixels above the mean value is activated as the lane pixels. For the next region, mean of the previous region is used, so if there is a curvature, the windows also slide with the lanes on the image. Then, on the activated pixels, a second degree polynomial fit is used to find the equation of the lane. An example image of the process can be seen below.

Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

Two separate functions are created for this purpose. To calculate the radius of curvature, the formula used can be seen here. Since it is a second degree polynomial, calculating it's first and second degree derivatives are easy. For the vehicle's position it is assumed that the camera is attached in the center of the vehicle which means that if the center of the image aligns with the center of the lane, the distance is zero. The number of pixels are calculated between the center of the image and the left and right lanes. Since the width of the lane is already known approximately, this information has been used to check the validty of the road. If the number of pixels are too high from the expected value, it means the lanes are not correct. The only problem is, the values are in pixel, and to make sense we need meters. For conversion, known width of the lane, and length of a one lane mark is used. The pixel to meter rates are calculated by counting the pixels and using the real values in meters.

Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly.

Pipeline (video)

Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (wobbly lines are ok but no catastrophic failures that would cause the car to drive off the road!)

The final video is here.

Discussion

Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

The patched parts of the road was problematic in the first trials, which can be seen near the end of my jupyter notebook. The pipeline will probably fail if the shadows are too dominant on the road, because those type of differences are also shown in gradients of the image. Something depending more on color information would help the function, however it is harder to detect the colors especially under changing lighting conditions. Hard curves or different lane widths would also cause problem, because for these problems some of the parameters should changed (for hard curves, inverse perspective and window margin parameters, for the lane widths, the valid distance parameter). To make this function robust, instead of hard parameters, adaptive parameters can be implemented. And also, if there is no lane found, instead of skipping the frame, different alternatives can be tried on the same frame until there is a valid hypothesis for a lane.