Advanced Lane Finding Project
Anishram Senathi

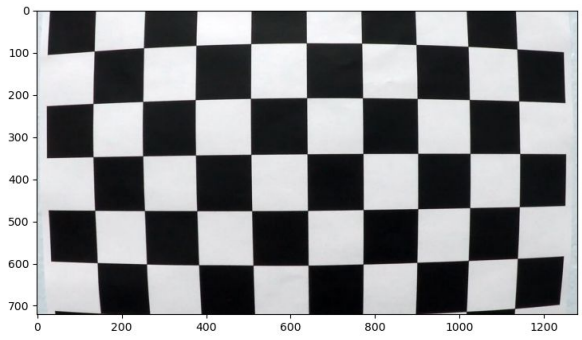
The goals / steps of this project are the following:

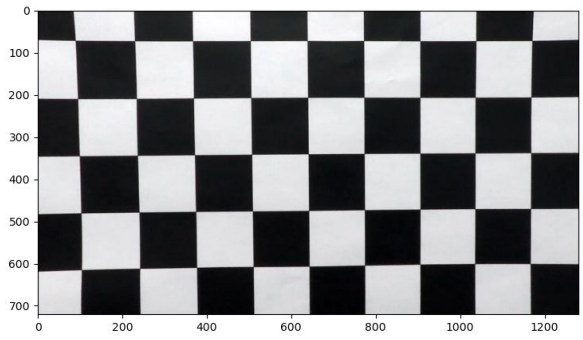
1. Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
2. Apply a distortion correction to raw images.
3. Use color transforms, gradients, etc., to create a thresholded binary image.
4. Apply a perspective transform to rectify binary image ("birds-eye view").
5. Detect lane pixels and fit to find the lane boundary.
6. Determine the 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.

Camera Calibration

1. Have the camera matrix and distortion coefficients been computed correctly and checked on one of the calibration images as a test?

The code for this step is contained in the first code cell of the IPython notebook.
I start by preparing "*object points*", which will be the (x, y, z) coordinates of the chessboard corners in the world. Here I am assuming the chessboard is fixed on the (x, y) plane at z=0, such that the object points are the same for each calibration image. Thus, *objp* is just a replicated array of coordinates, and *objpoints* will be appended with a copy of it every time I successfully detect all chessboard corners in a test image. *imgpoints* will be appended with the (x, y) pixel position of each of the corners in the image plane with each successful chessboard detection.
I then used the output *objpoints* and *imgpoints* to compute the camera calibration and distortion coefficients using the cv2.calibrateCamera() function. I applied this distortion correction to the test image using the cv2.undistort() function and obtained this result:





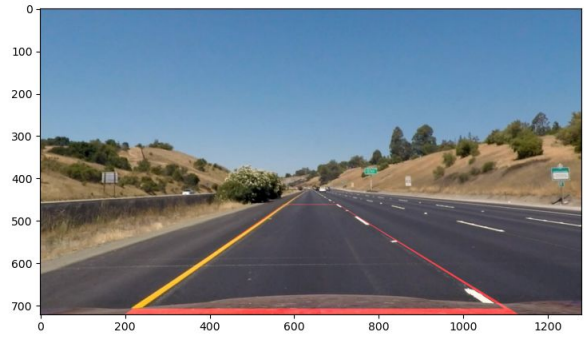
Pipeline (single images)

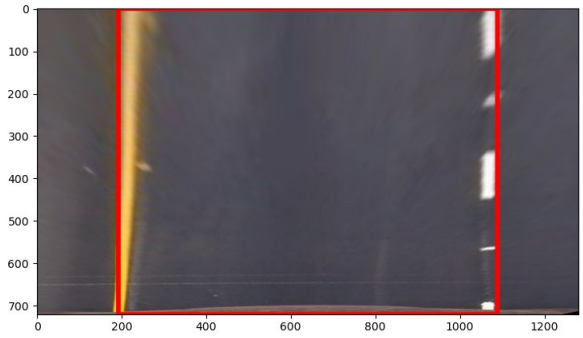
1. Has the distortion correction been correctly applied to each image?

To demonstrate this step, I will describe how I apply the distortion correction to one of the test images like this one:

2. Has a perspective transform been applied to rectify the image?

In the function *calculateTransformMatrix,* I have hardcoded the source and destination points in terms of the image size. After transforming a sample image containing straight parallel lanes, the result yields an image where lanes appear parallel and hence the transformation seems good.





For the project video, the transformation matrix looked like:

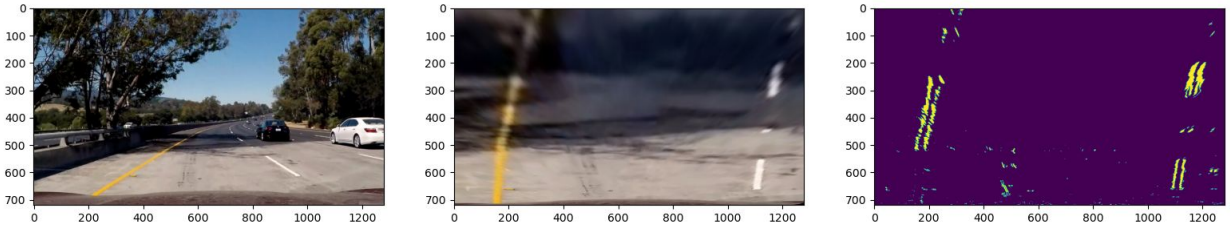
src	**dst**
[576. 460.8]	[192. 0.]
[204.8 720.]	[192. 1280.]
[1126.4 720.]	[1088. 1280.]

[704. , 460.8]	[1088. 0.]

Transformation Matrix :
[[-6.98447970e-01 -1.45848743e+00 1.05521959e+03]
[-4.03717455e-16 -1.99556558e+00 9.19556595e+02]
[-0.00000000e+00 -2.38667173e-03 1.00000000e+00]]

3. Has a binary image been created using color transforms, gradients or other methods?

A combination of color transform and gradients were used to generate the binary image. Primarily, Sobel gradients along x direction were used to filter the lane pixels. It was found to contain a lot of noisy pixels due to shadows or other patterns on the road and hence a small threshold was applied on the saturation of the pixels. This method fairly succeeded in detecting the lane pixels for most conditions.



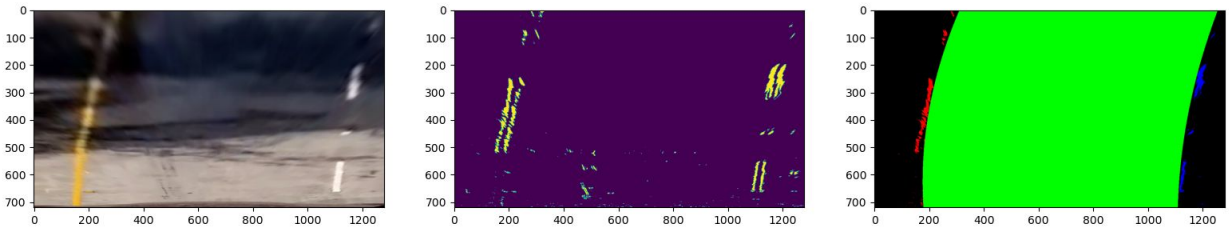
4. Have lane line pixels been identified in the rectified image and fit with a polynomial?

Lane line pixels have been identified in a 2-stage process.

1) Histogram based method for finding new lanes
2) Lane Tracking method if the past lane was known

<u>Histogram based method</u>
Taking the thresholded image, pixels are accumulated along the y-axis and a histogram is plotted along the x-axis. Peaks are detected in the left and right halves of this histogram and these peaks roughly denote the starting x-center of the left and right lanes respectively. Then as we move from bottom to top in 9 steps (windows), we mark all thresholded pixels within a margin of 100 pixels along the detected x-center as lane line pixels. The mean of x-values of these pixels form the x-center of the next window respectively. As this method proceeds, we get all lane line pixels of left and right lanes respectively.

A quadratic line is fit for left and right lanes respectively and the best fit line is plotted on the image as the lanes. The region between the lanes is plotted as the lane region in green as shown below.



The equation of lanes were found out to be:
Left lane: x = 2.05185063e-04 * y2 -2.87341916e-01 * y + 4.03710917e+02
Right lane: x = 1.68404398e-04 * y2 -2.28357764e-01 * y + 1.06957346e+03

Lane Tracking method

If the lanes were detected in the previous image and were good enough (that is, the coefficients for square term and linear term for both lanes differ by less than 0.1), this information can be used in the current frame to detect lanes faster and more accurately. Now, instead of forming histogram windows, the current lanes are assumed to lie within a certain margin of the quadratic lines previously formed. All thresholded pixels lying in this quadratic region are marked as lane line pixels and the best fit lines and region are plotted just like explained above.

5. Having identified the lane lines, has the radius of curvature of the road been estimated? And the position of the vehicle with respect to center in the lane?

The radius of curvature was calculated using the formula:
((1 + (2*A*y_eval*ym_per_pix + B)**2)**1.5) / np.absolute(2*A)

Where A and B are obtained from the equation of lane: Ay^2 + By + C
The radius is calculated for both right and left lanes and the minimum of the two is chosen as the radius of curvature.

The position of vehicle with respect to center of lane is calculated by considering the midpoint of bottom edge as the position of the vehicle.
Both these values are converted into real world coordinates by multiplying them with the conversion factor in x and y axes.

Pipeline (video)

1. Does the pipeline established with the test images work to process the video?

To run the pipeline on video, lanes are first tried to detect using the lane tracking algorithm, and whenever it fails sanity checks, new lines are detected using the histogram algorithm. If even the new lines fail sanity checks, then the lanes detected in the previous frames are used. The first frame of the video is of course, passed onto the histogram method. The sanity checks used are:

- Checking if the lane coefficients are close enough
- Checking if the radius of both lanes are close enough
- Checking if the distance between the lanes is close to 3.7 metres

Redundant values like perspective transform matrices are calculated only once in the beginning and the values are stored. The other procedures are carried out for every frame just like explained above for single images.

The video result can be found at “project_video_output.mp4” in the project folder.

README

1. Has a README file been included that describes in detail the steps taken to construct the pipeline, techniques used, areas where improvements could be made?

Discussion

Approach

1. Since the pipeline involves considering a region of interest and lane tracking and sanity checks, which mitigate type 1 error, the role of thresholding stage was to keep type 2 error to a minimum and could afford to have some type 1 error

2. One critical step is to determine when to switch from lane tracking state to new lane finding state since the errors could follow to the next frames. So I introduced some sanity checks to decide between these two methods and if previous lane values need to be reused.

Where the pipeline would fail:

1. When an uneven road or a speed breaker is encountered, the true perspective transform matrix would change and hence lanes might be wrongly detected
2. If the are other lines on the road, like dividers or sections of recently cemented roads, etc, they might be wrongly detected as lanes
3. Vehicles and pedestrians in the region of interest would cause trouble while detecting lanes. Hence the pipeline would fail incase of traffic
4. If the lanes are not parallel, for example at a T-turn, the pipeline would fail
5. Thresholding parameters are fixed and hence it it would fail for soft lanes and night time

Scope of improvement:

1. Instead of gradient based thresholding, more robust methods can be implemented to detect new lanes
2. The perspective matrix can be experimentally determined instead of hardcoding it