Udacity CarND Project 4

Advanced Lane Detection

🡪Steps followed were:-

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 colour transforms, gradients, etc., to create a threshold 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.

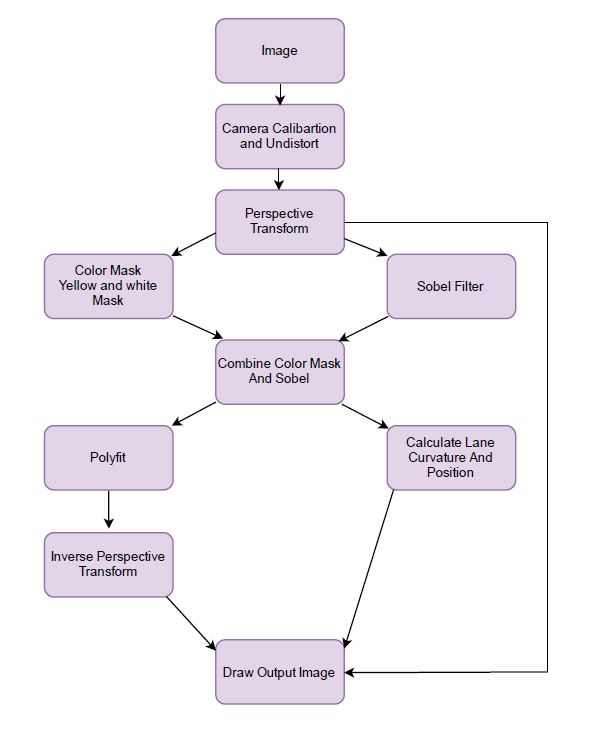
6. Determine the curvature of the lane and vehicle position with respect to

centre.

7. Warp the detected lane boundaries back onto the original image.

8. Output visual display of the lane boundaries and numerical estimation of

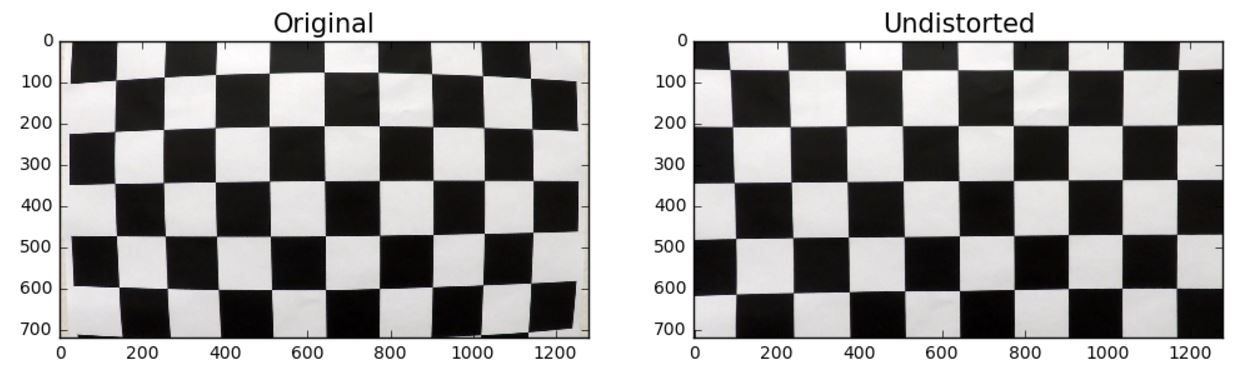
lane curvature and vehicle position.

🡪Flowchart:-

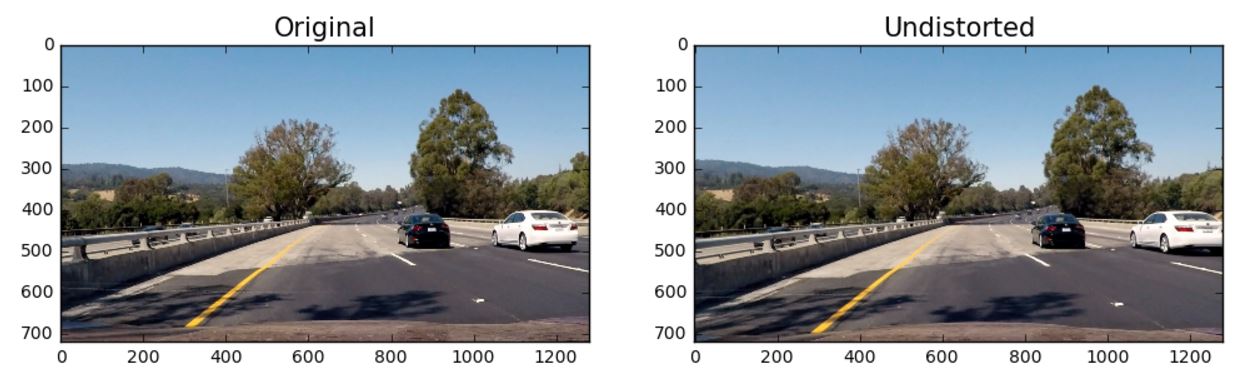
🡪Camera Calibration:-

In order to compute the camera matrix and distortion coefficients, CameraCalibration uses OpenCV’s **cv2.findChessboardCorners** function on grayscale version of each input image in turn, to generate a set of image points. I then use numpy.mgrid in order to create a matrix of (x, y, z) object points.

Once I have the image points and the matching object points for the entire calibration dataset, I then use **cv2.calibrateCamera** to create the camera matrix and distortion coefficients. Then **cv2.undistort**, returns an undistorted version of the user’s input image.

Code is implemented in **camera\_Calibraton** and **undistort** functions.

🡪 Example of a distortion-corrected image:-

 After calculating mtx and dist, we use them to undistort our testimages using **undistort** function.