SDC Project 4

Advanced Lane Finding Project

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.

Rubric Points

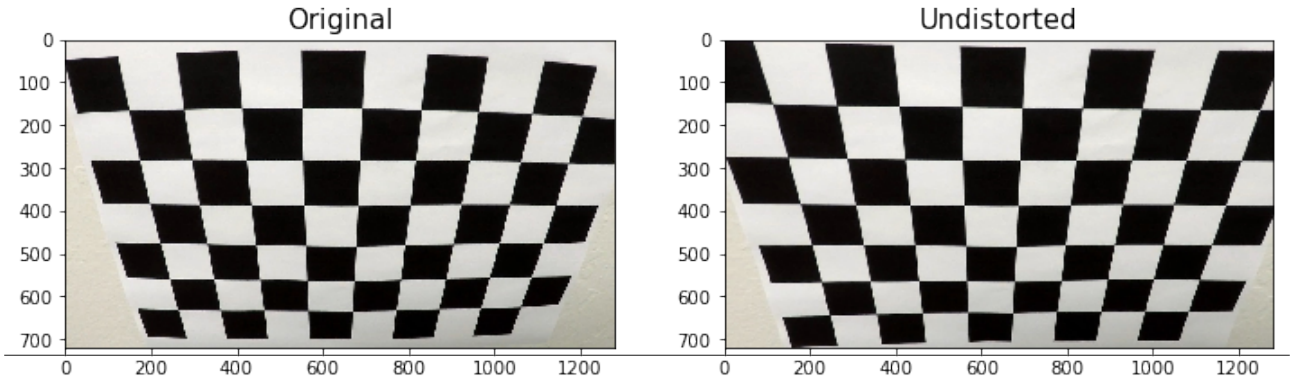
Camera Calibration

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

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.

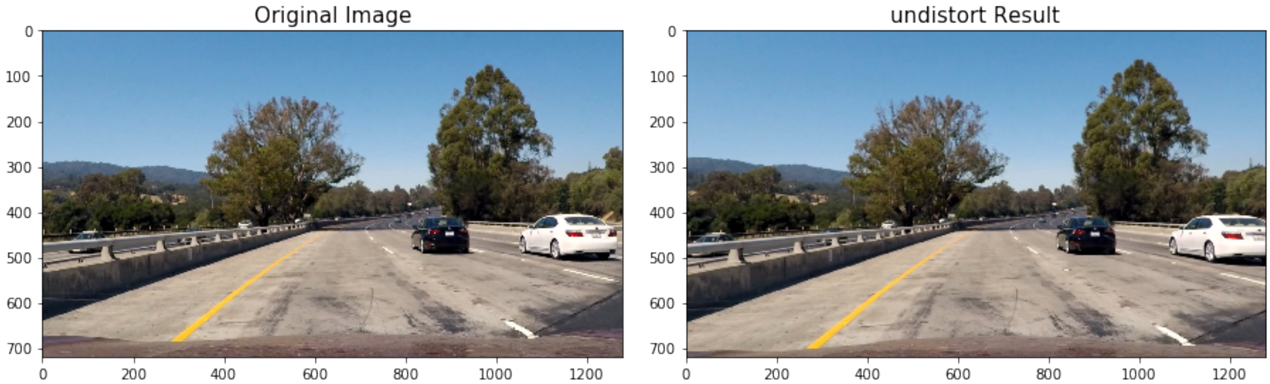
I then used the output objpoints and imgpoints to compute the camera calibration and distortion coefficients using the cv2.calibrateCamera() function. I applied this distortion correction to the test image using the cv2.undistort() function and obtained this result:

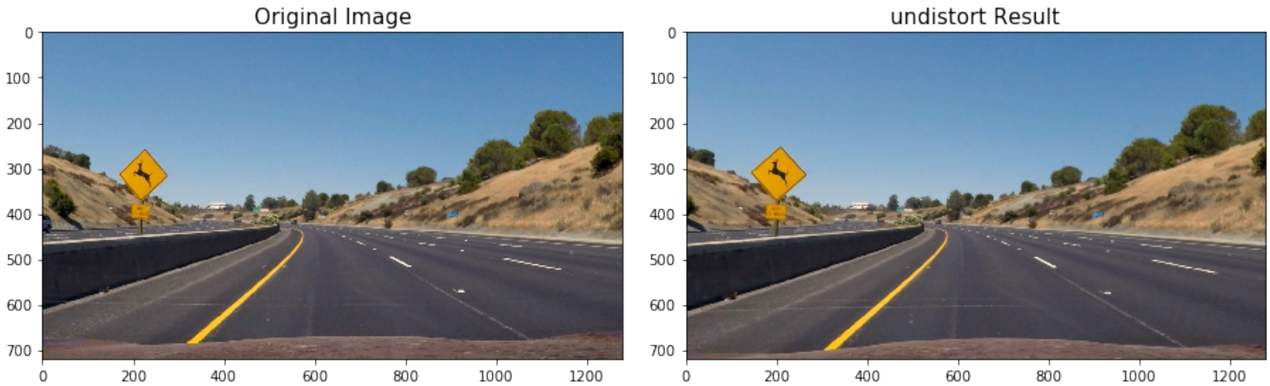


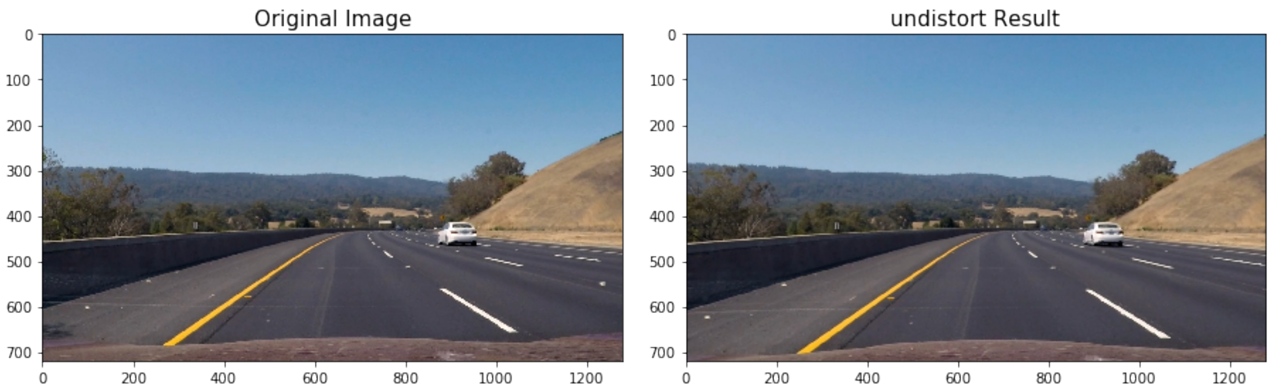
Pipeline (single images)

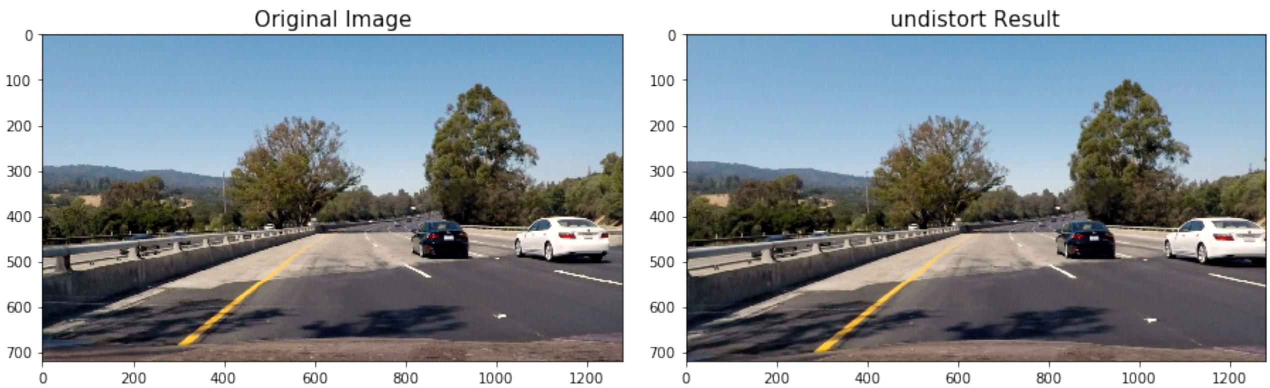
1. Has the distortion correction been correctly applied to each image?

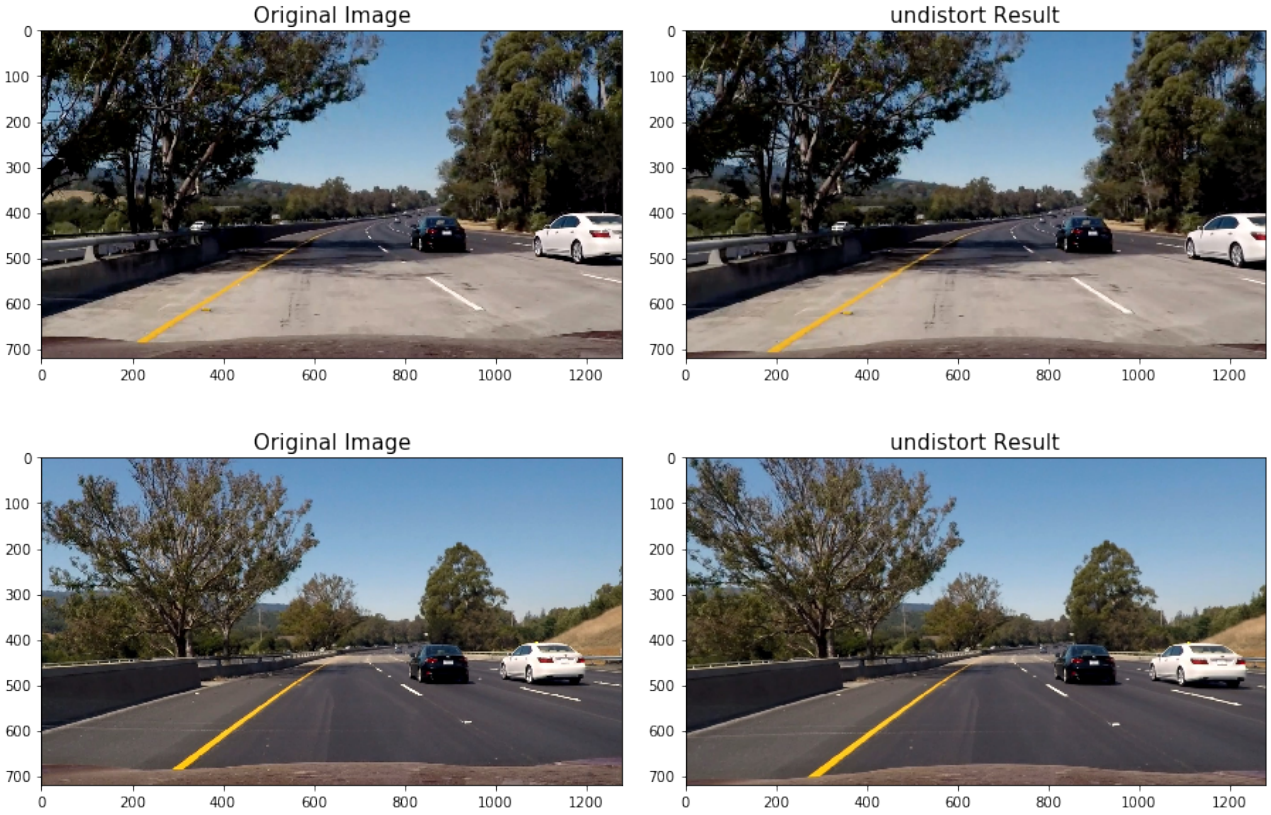
I use the algorithm describe before applied to each 6 images, and the results show as below:







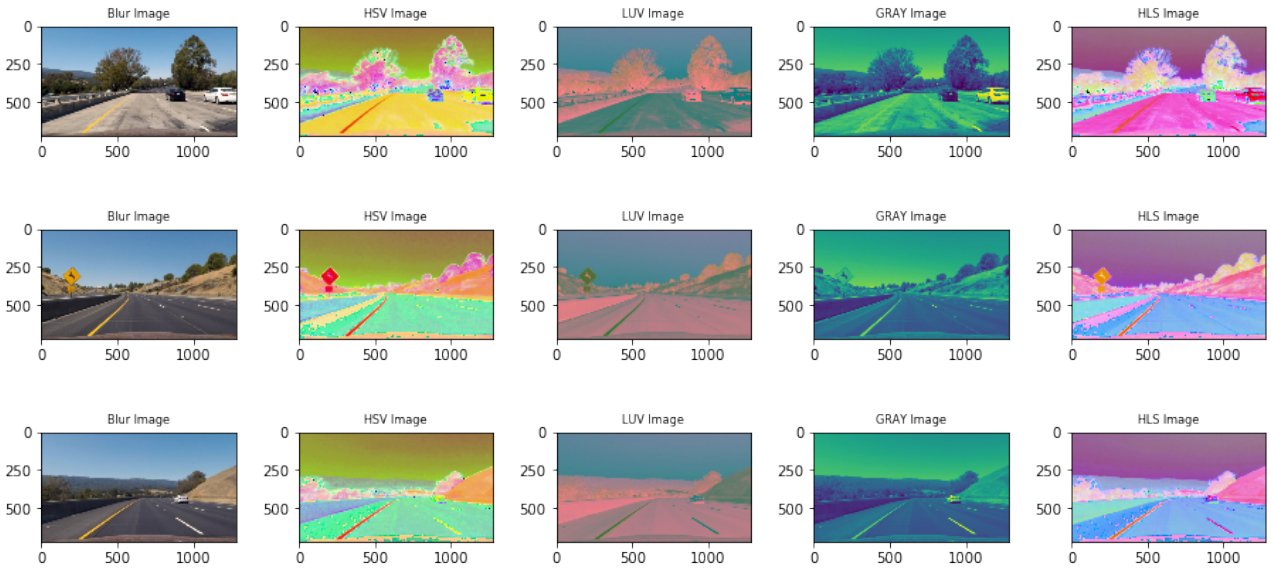


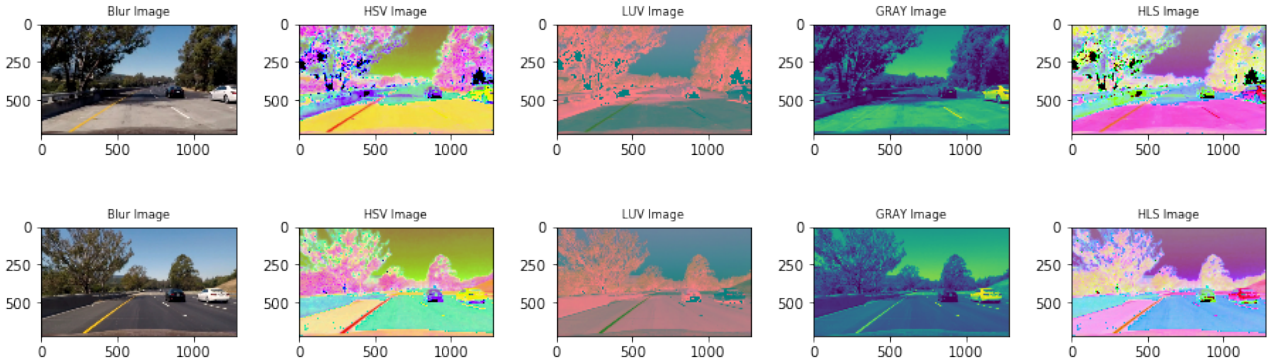


The left image is original input images, and the right images is distort results, the results is works correctly and works well in each images.

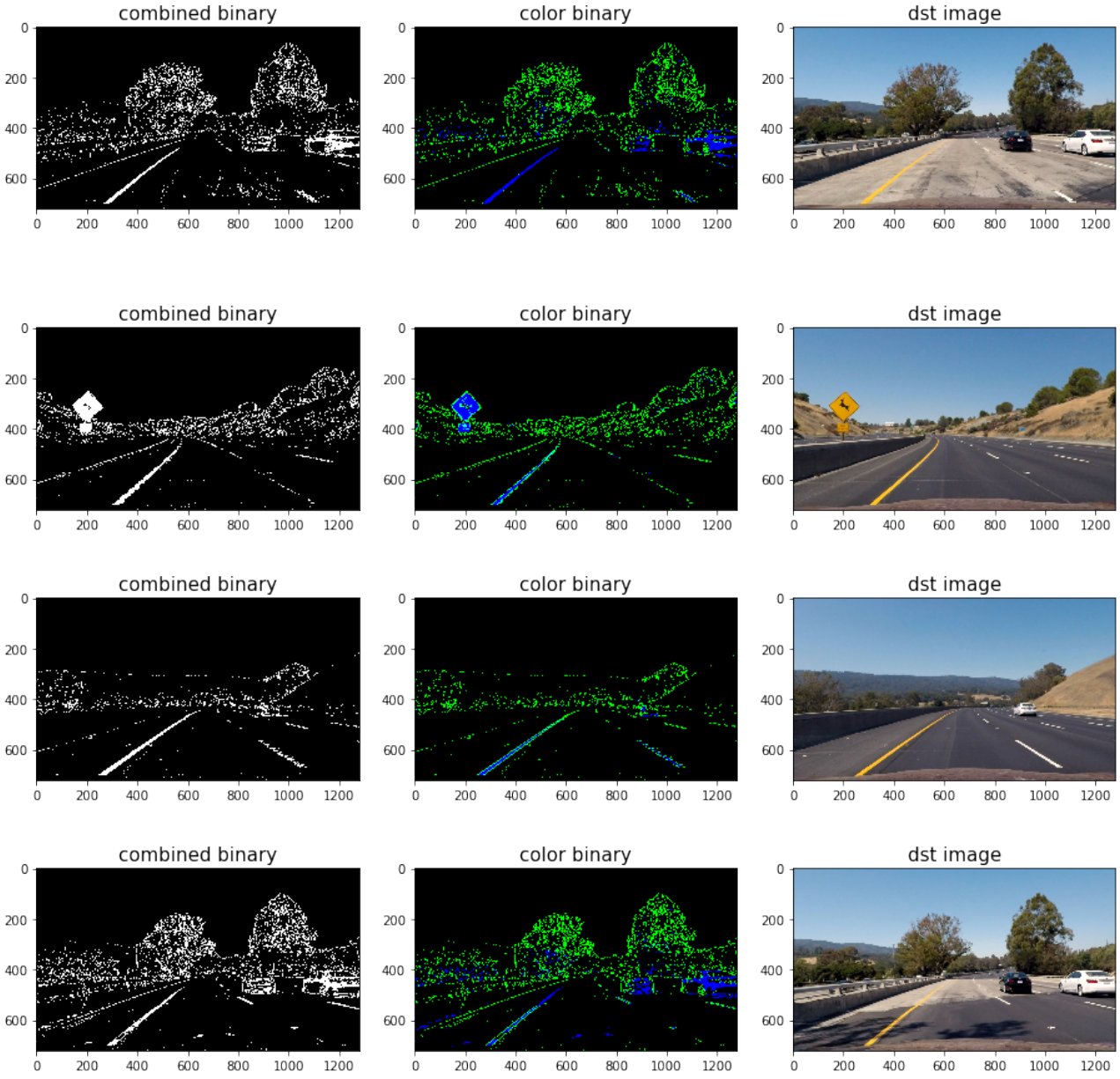
2. Has a binary image been created using color transforms, gradients or other methods?

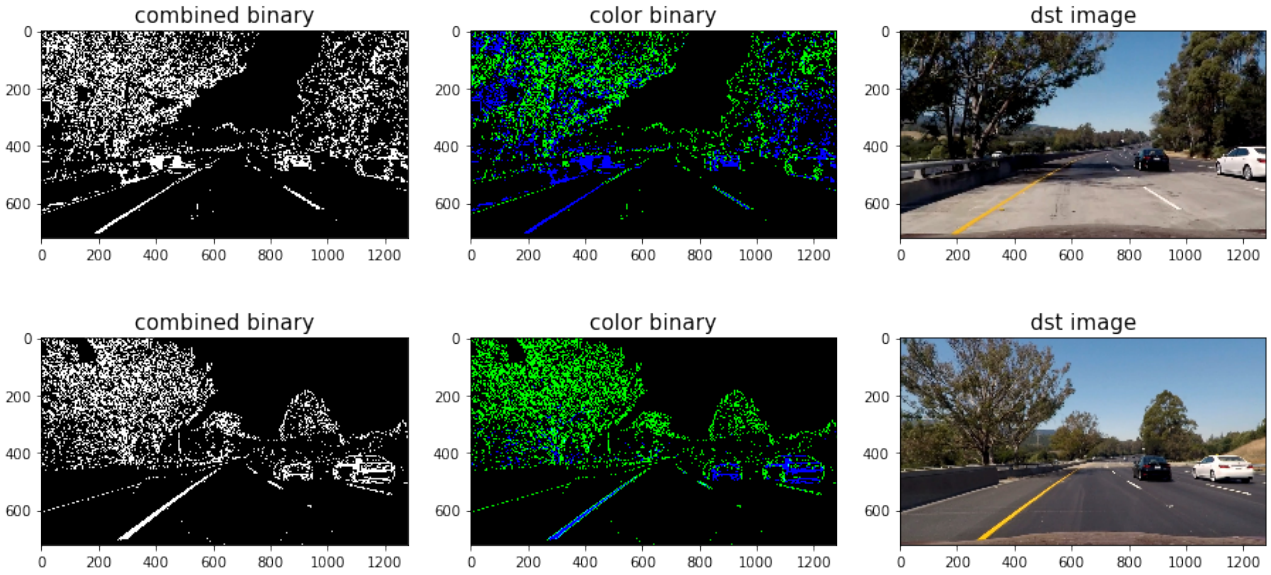
I test each images under different color channels, I test HSV, LUV, GRAY and HLS channels, and for each channels, the results show as below:



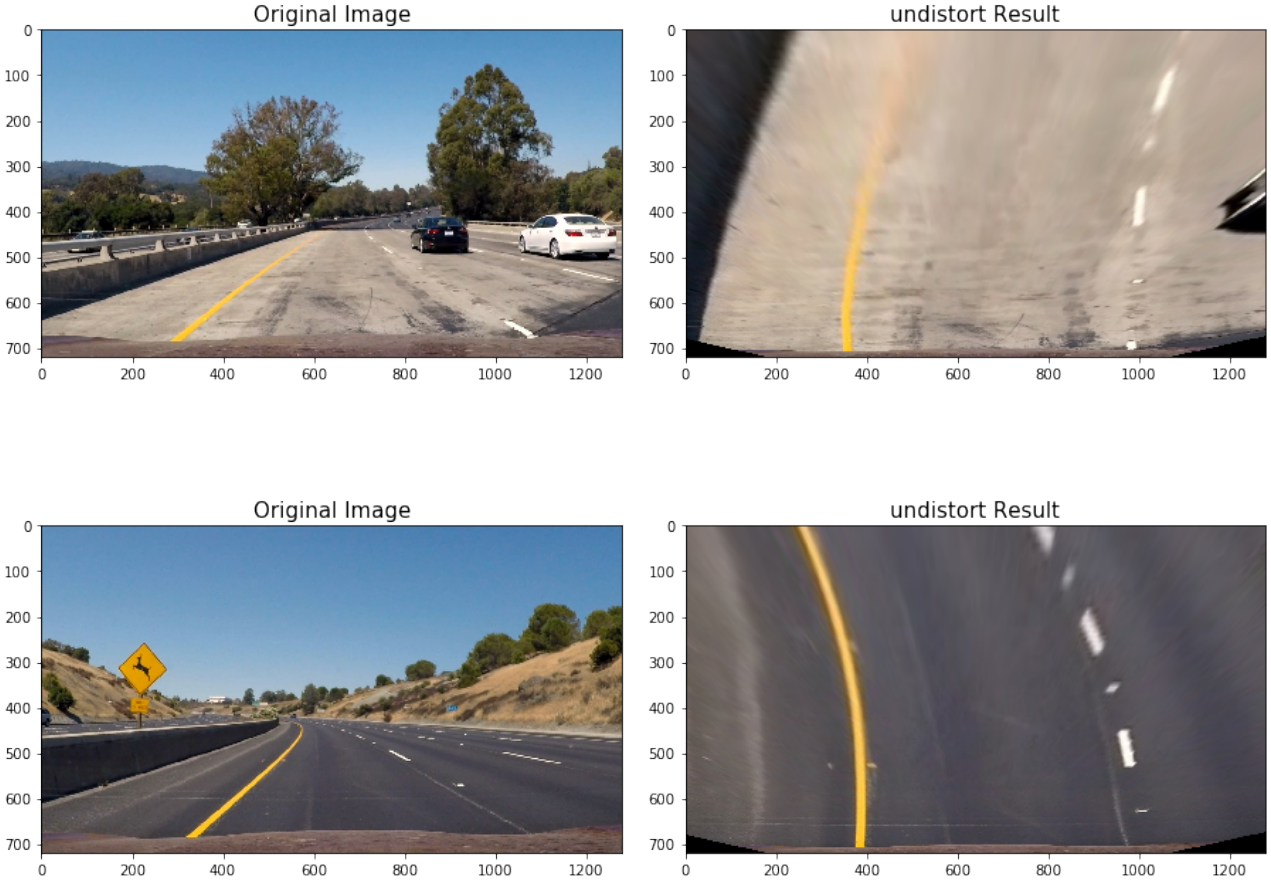


and I find the S channel in HLS channels is better than others, so I transform the RGB image to HLS and GRAY channels, then I applied sobel and gradient methods to GRAY level image, and threshold the results, also I combined the results with threshold S channel. The results show as below:

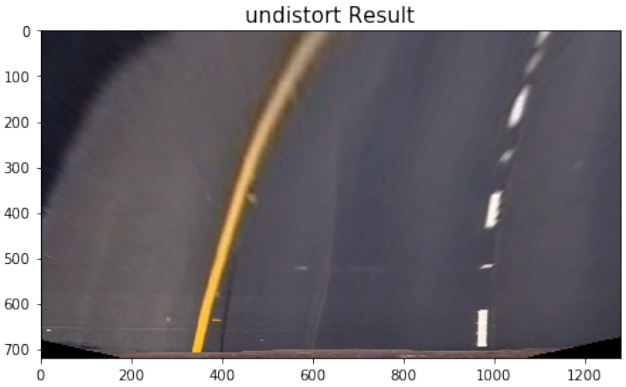




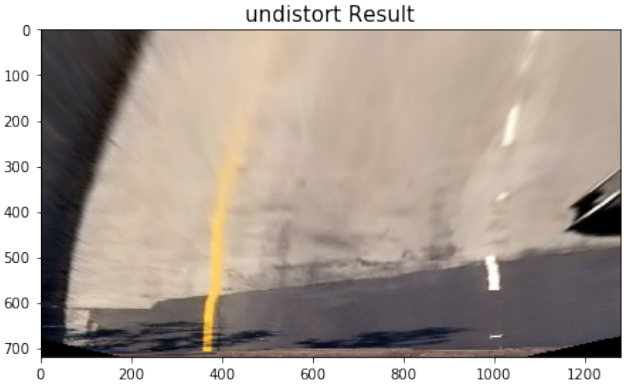
3. Has a perspective transform been applied to rectify the image?
After I got the binary results, I used the perspective transform to the input image, I set the src and dst matrix as below:
dst = [[320, 0][320,720][960,720][960,0]] src=[[2585,450][200,720][1130,720][695,450]]
the results displayed below:



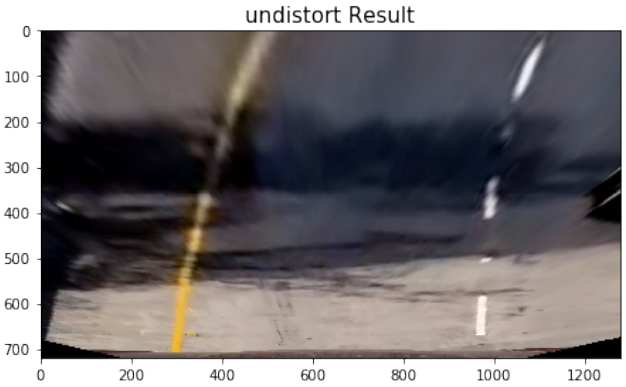

Original Image


undistort Result

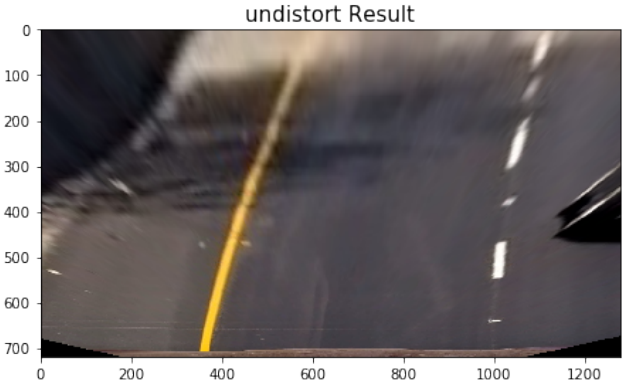

Original Image


undistort Result

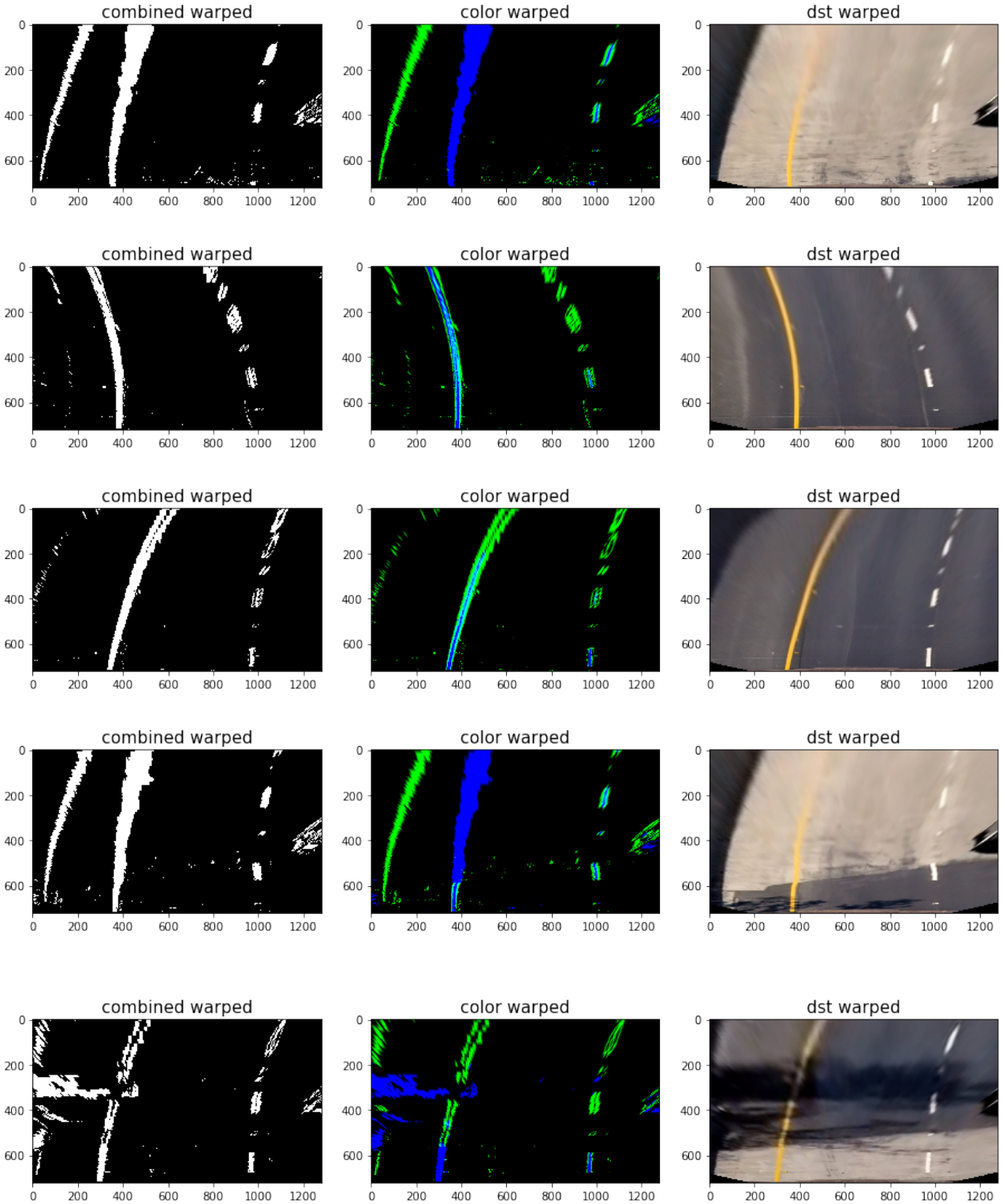

Original Image

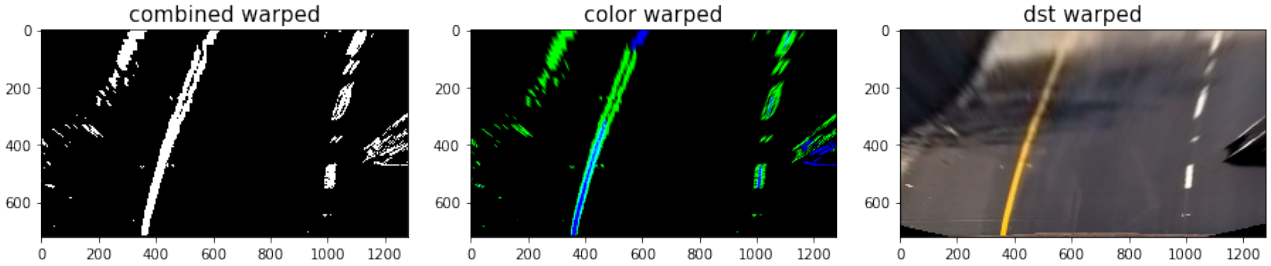

undistort Result


Original Image


undistort Result

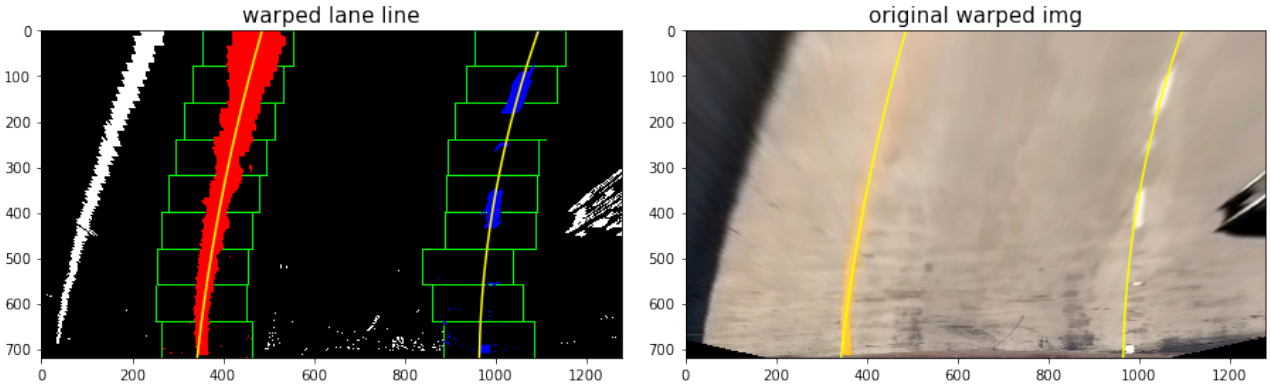
lists as below is combined warped results and color warped images:

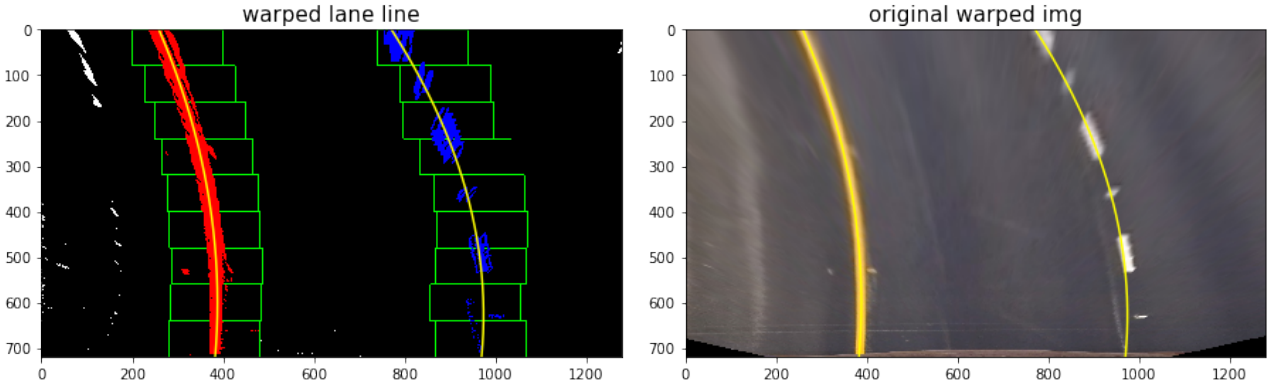


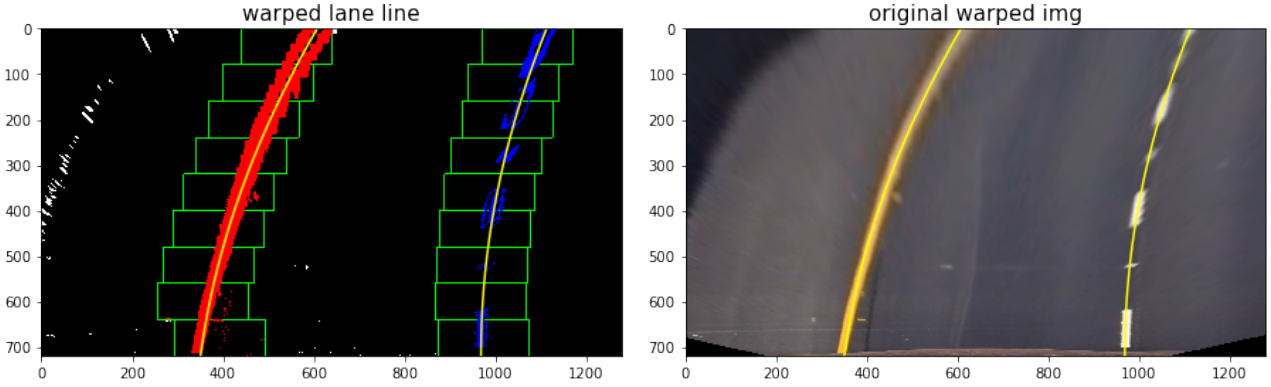


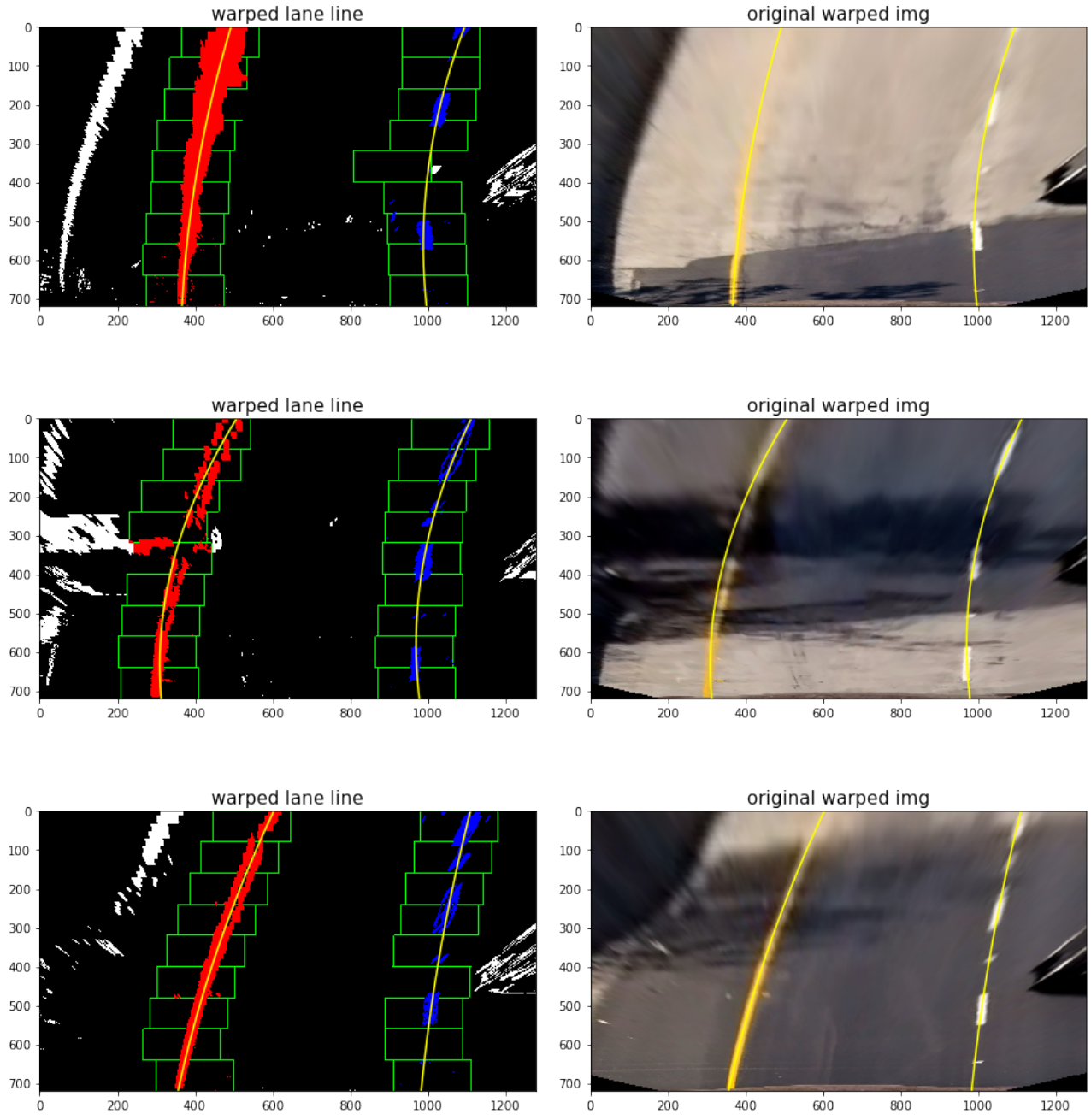
4. Have lane line pixels been identified in the rectified image and fit with a polynomial?

This step, I follow the materials provide in the classroom to write the lane line detection algorithm, I used the second order polynomial to each lines, and got the final results below:









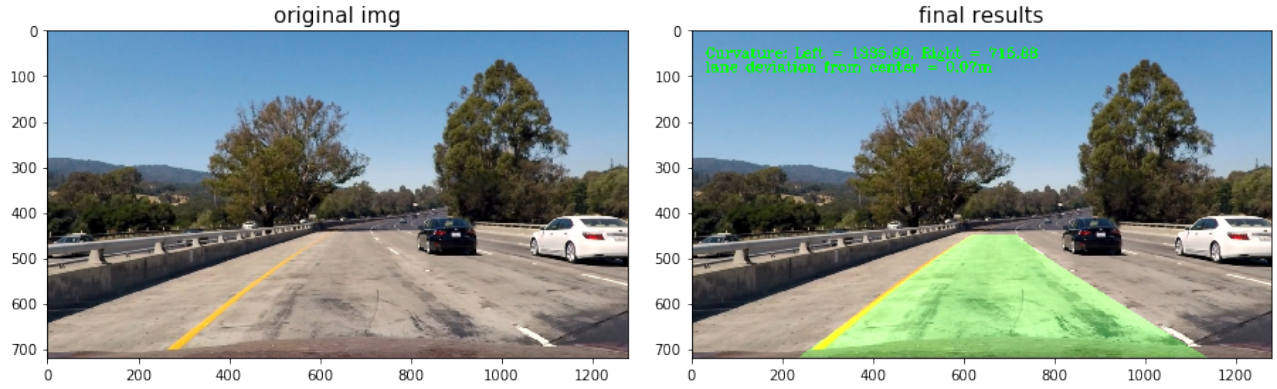
5. Having identified the lane lines, has the radius of curvature of the road been estimated? And the position of the vehicle with respect to center in the lane?

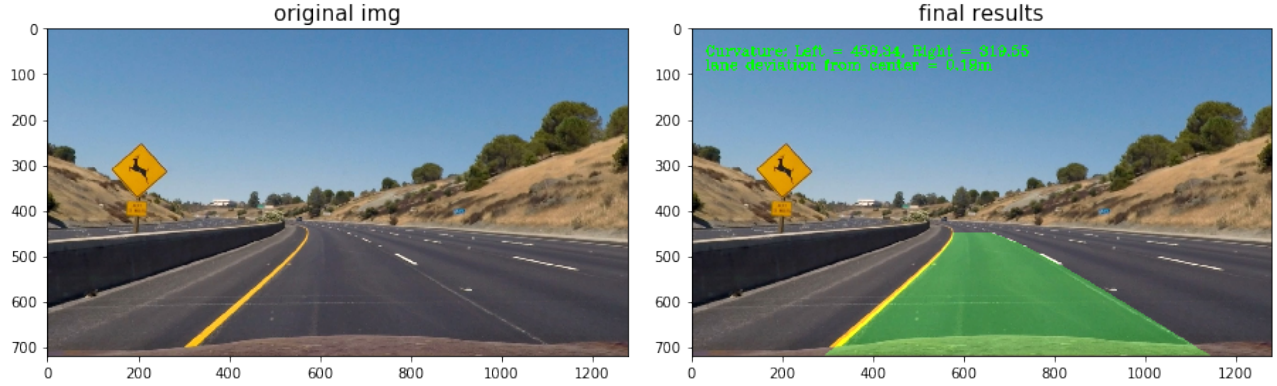
I also follow the classroom materials to write the curvature algorithm, and I Define conversions in x and y from pixels space to meters, then fit new polynomials to x,y in world space, and calculate the new radii of curvature, the final step is calculate Lane Deviation from center of lane. The code of this step please see the Jupiter Notebook.

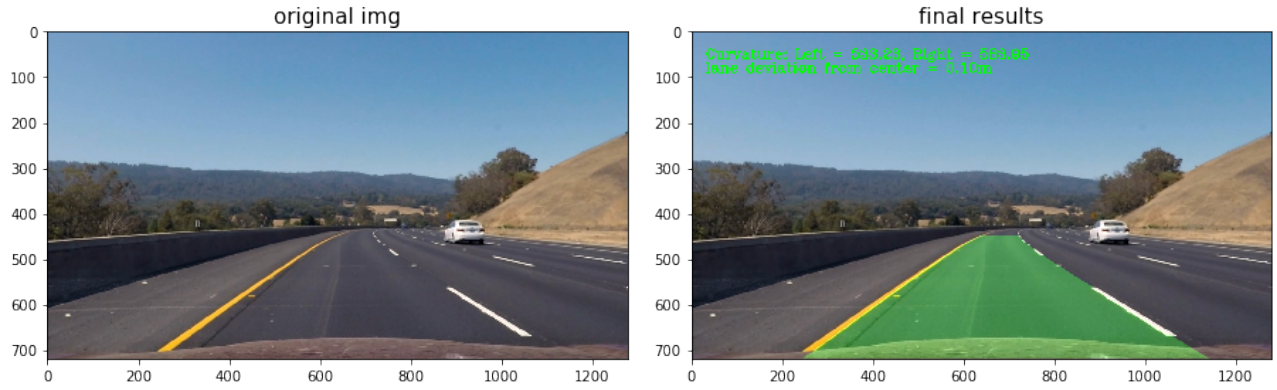
Pipeline (video)

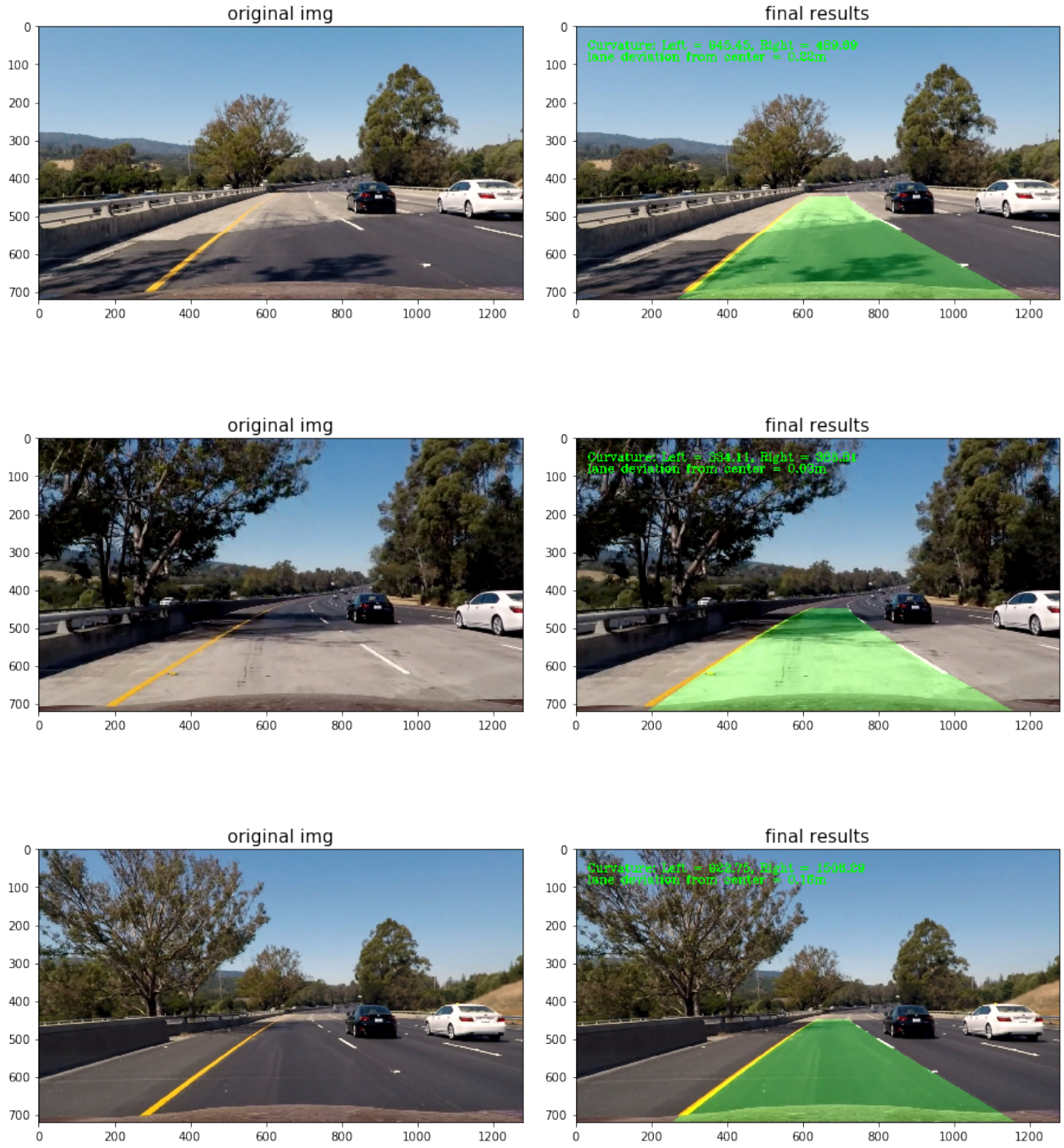
1. Does the pipeline established with the test images work to process the video?
The pipeline of my algorithm list as below:
1. distortion correction applied to the input images

2. applied the binary image algorithm to the distorted results
3. doing warp for the binary results
4. applied lane line detection methods to the warped results
5. then applied the curvature methods to detected results
6. doing sanity_check with previous fitting lines, and return True or False, if True ref = current, if False current fitting line = previous fitting line;
7. draw the lane results to the original image;
The results of test image as below.









The Video Output please see the output_images(named project_video_output.mp4)

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?

This report can seen as README file, and the improvement of this methods is the dst and src set, also the binary image methods is not good, also we can use deep learning algorithm may can get better results.

Discussion

The pipeline of lane line detection may not well, and the technical used here is pure computer vision methods, we do not use any machine learning based algorithm, so the results may not too stable, and the algorithm can not learned from our input dataset. But this methods is easy to implemented and easy to understand. The polynomial and radius curvature algorithm used here make the results seems better than our first project. I think we may can use some machine learning based algorithm to find the lane line or using segment based eep learning methods, it may can get better results than this one.