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.
9. Created a lane class to keep track of lane parameters between frames. Ignored for images.
10. For video, use hist_search() and fast_search() (which tracks previous frame's lane location)

Using Ruberic Points

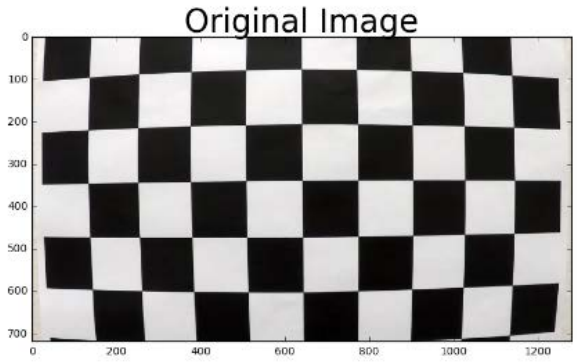
Camera Calibration

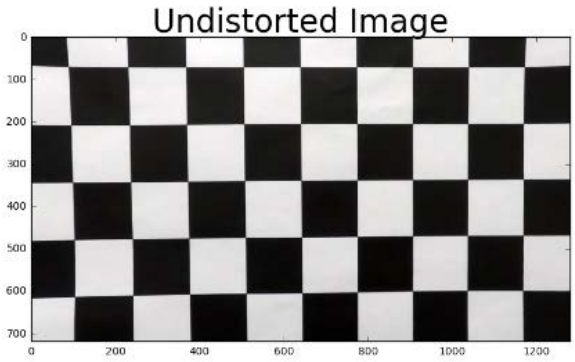
1. Have the camera matrix and distortion coefficients been computed correctly and checked on one of the calibration images as a test?

Yes and here's some description of how its implemented.
This will generate a list of object points and its corresponding image points and use it to calibrate the camera by generating the camera matrix and distortion coefficients and save them in a pickle file for later use.

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:

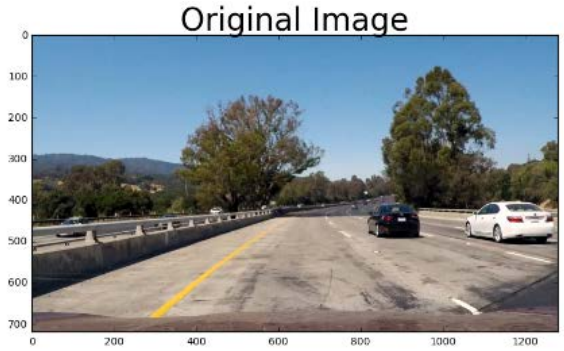




Pipeline (single images)

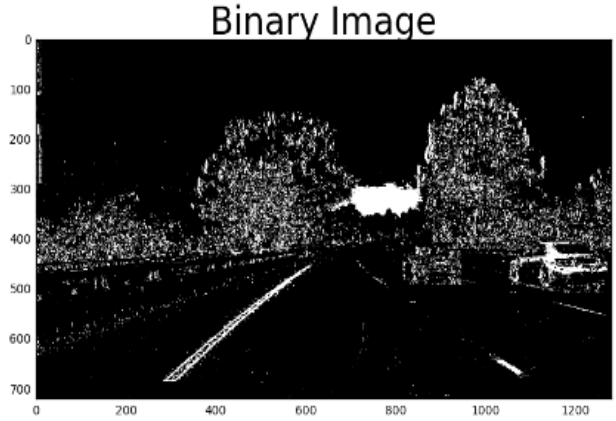
1. Has the distortion correction been correctly applied to each image?

Yes, see below. I have also create a lane class to keep track of important lane parameters between frames (to be used for video, but ignored for single images)



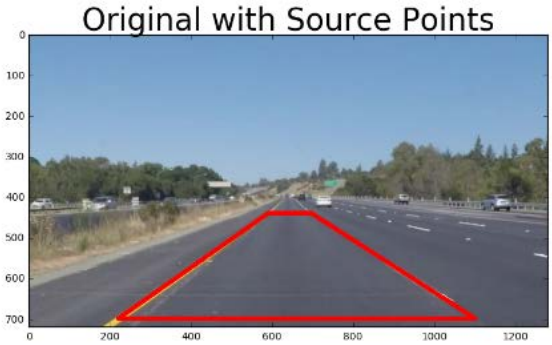
2. Has a binary image been created using color transforms, gradients or other methods?

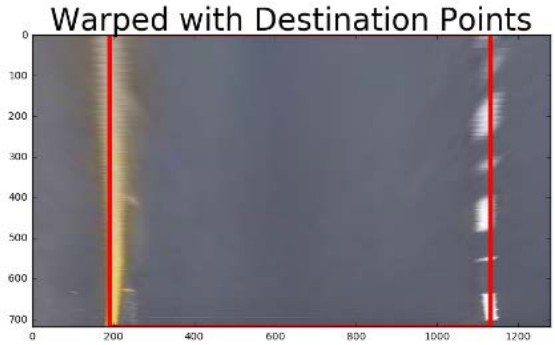
Yes, see below. I used color (using HLS color space) and gradient thresholding. For gradient thresholding, I used gradient directions below 0.6 radians (i.e. closer to x-direction) as lanes have stronger gradient in the x-direction. But the reason I used a range of 0 to 0.6 radians instead of near 0 is to better detected non-straight line lanes (i.e. with curves).



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

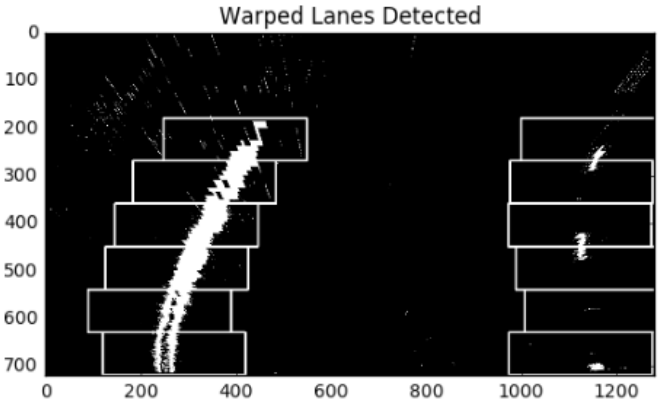
Yes, see below. The code for my perspective transform includes a function called warp(). The warp() function takes as inputs an image (img) as well as a transformation matrix generated using source (src) and destination (dst) points. I chose to hardcode the source and destination points.

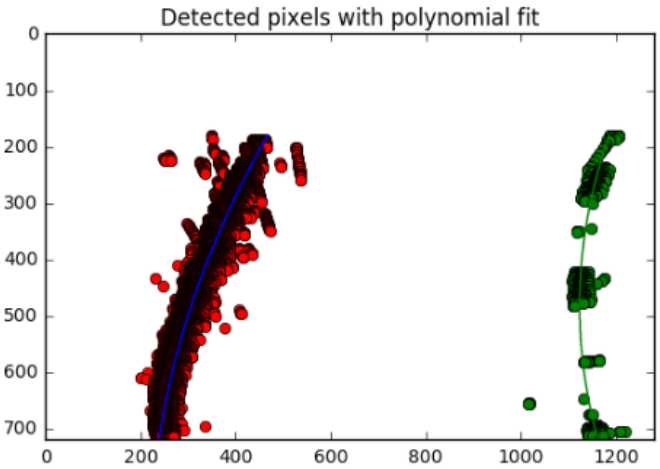




4. Have lane lines pixels been identified in the rectified image and fit with a polynomial?

Yes, see below. To identify the lane pixels, I use hist_search function. And once I have the detected pixels, I curve fit it using a second order polynomial.





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 the center of the lane?

Yes, see below. More images are stored in the output_images directory.

Pipeline (Video)

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

Yes, checkout 'project_video_output.mp4.'

2. Has some kind of search method been implemented to search and keep track of lane lines?

Yes, I make use of the lane class (mentioned above) to keep track of important lanes parameters between frames. Moreover, to speed the lane detection process, once I build strong confidence in 25 detected frames, I use a faster way (fast_search()) to detect lanes by searching in a window centered around the polynomial fit detected in the previous frame. And if the detection confidence decreases for 4 consecutive frames, then I revert back to the histogram based approach (hist_search()) until I build strong confidence again to go back to fast_search.

Future Improvement

Currently the pipeline doesn't work on the challenge videos. The problem is that with the existing color and gradient masks, the lanes are not detected in the challenge videos. If I get more time in the future, I'll address it.