Advanced Lane Finding Project

The goals/steps of this project are the following:

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to the center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

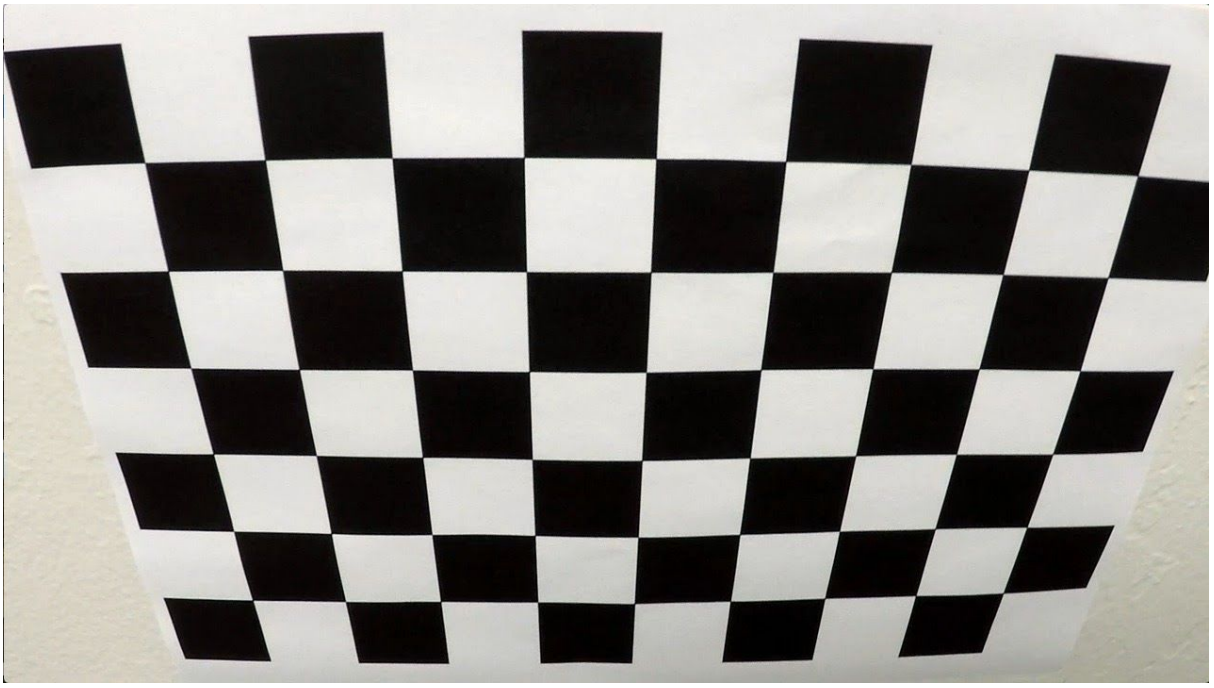
Rubric Points:

Camera Calibration

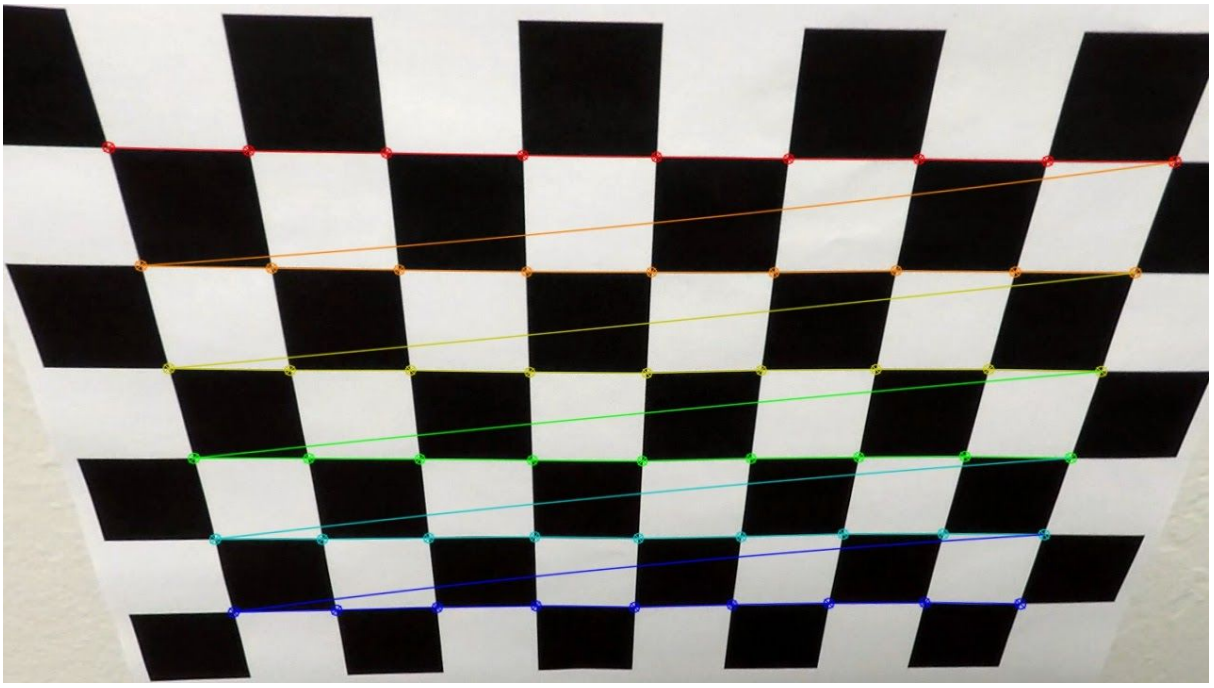
1. **Have the camera matrix and distortion coefficients been computed correctly and checked on one of the calibration images as a test?**

The code for camera calibration is under "examples/P2.ipynb". I start by preparing "object points", which will be (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:



The above figure represents a distorted image.

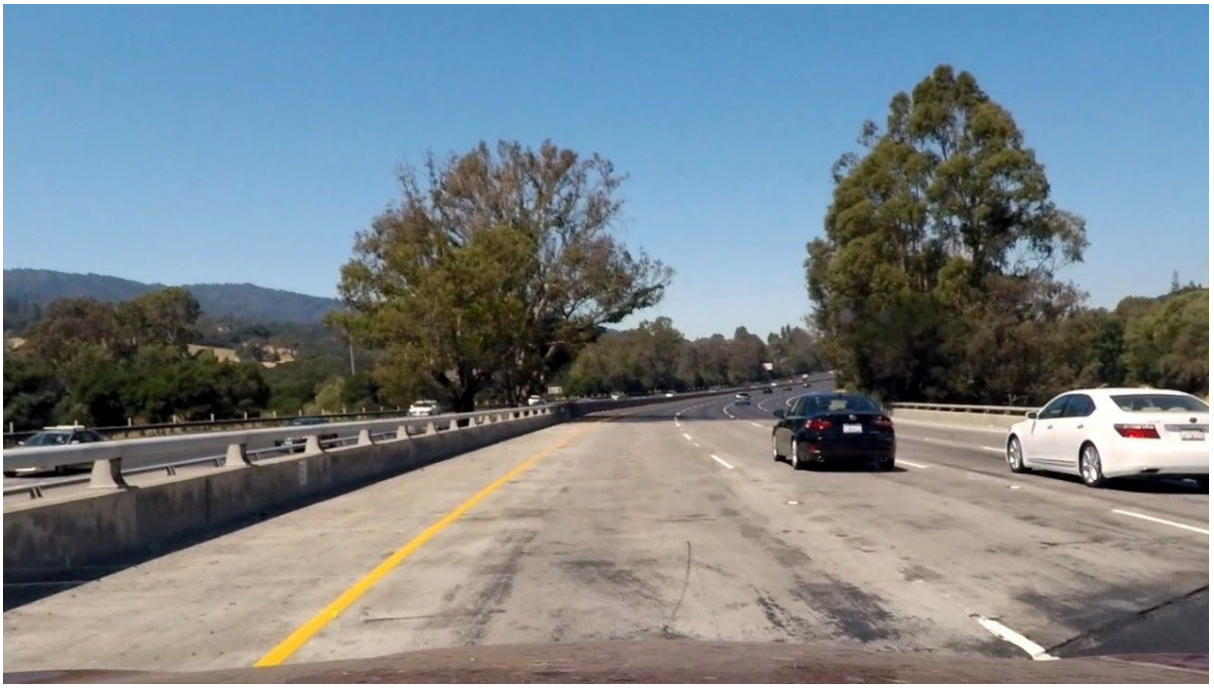


The above figure represents an undistorted image.

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 binary image been created using color transforms, gradients, or other methods?**

Compute thresholded binary image using a combination of color threshold and gradients for S channel in HLS color space and Sobel X gradient derivative.

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

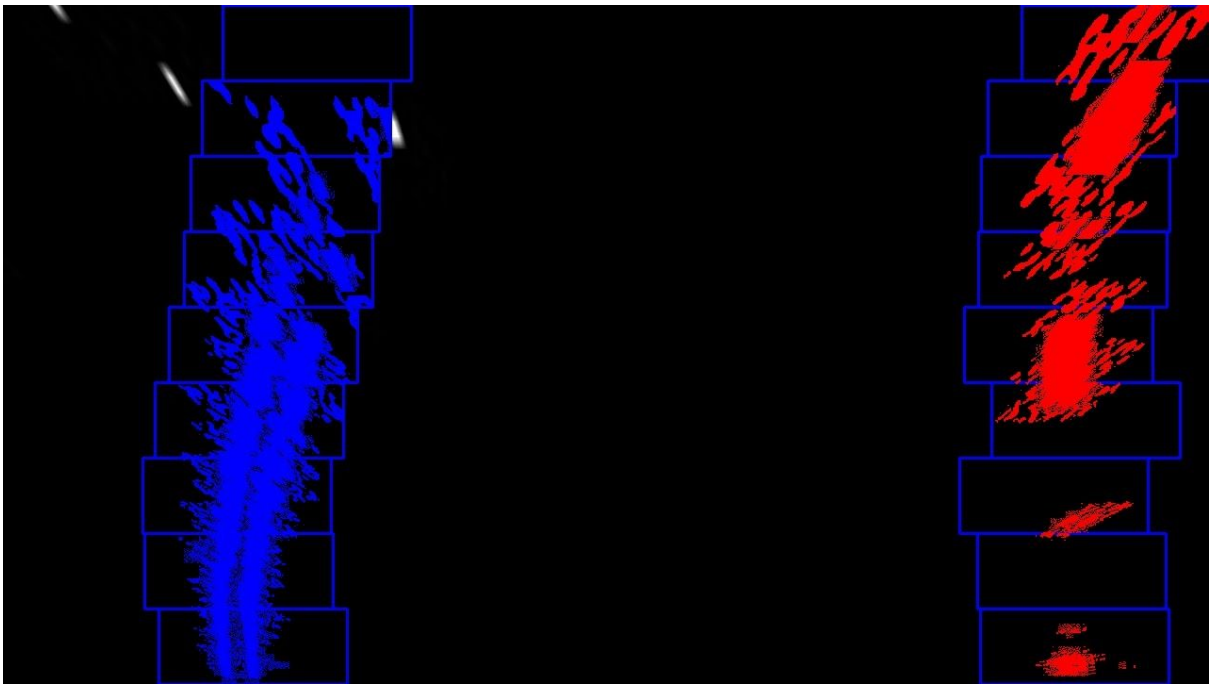
The code which I used to perform perspective transform is given below-

```
bottomY = 720
topY = 450

left1_x, left1_y = (200, bottomY)
left2_x, left2_y = (585, topY)
right1_x, right1_y = (700, topY)
right2_x, right2_y = (1100, bottomY)
src = np.float32([[left2_x, left2_y], [right1_x, right1_y], [right2_x, right2_y], [left1_x, left1_y]])
    dst = np.float32([[offset, 0], [img_size[0]-offset, 0], [img_size[0]-offset, img_size[1]], [offset, img_size[1]]])

M = cv2.getPerspectiveTransform(src, dst)
MInverse = cv2.getPerspectiveTransform(dst, src)
```

M represents the transformation matrix from src to dst and MInverse represents the transformation matrix from dst to src.

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

Then I did some other stuff and fit my lane lines with a 2nd order polynomial kinda like this:



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?**

   Yep, sure did!

Pipeline (video)

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

   It sure does! I have uploaded the video called "project_video_output.mp4".

2. **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?**

   Yes, I have included the README.md is included.

Discussion

The proposed approach works for project_video.mp4 but does not work for challenge video because of the shadows on-road and the steep lane curvature.