Ptoject #4

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 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.

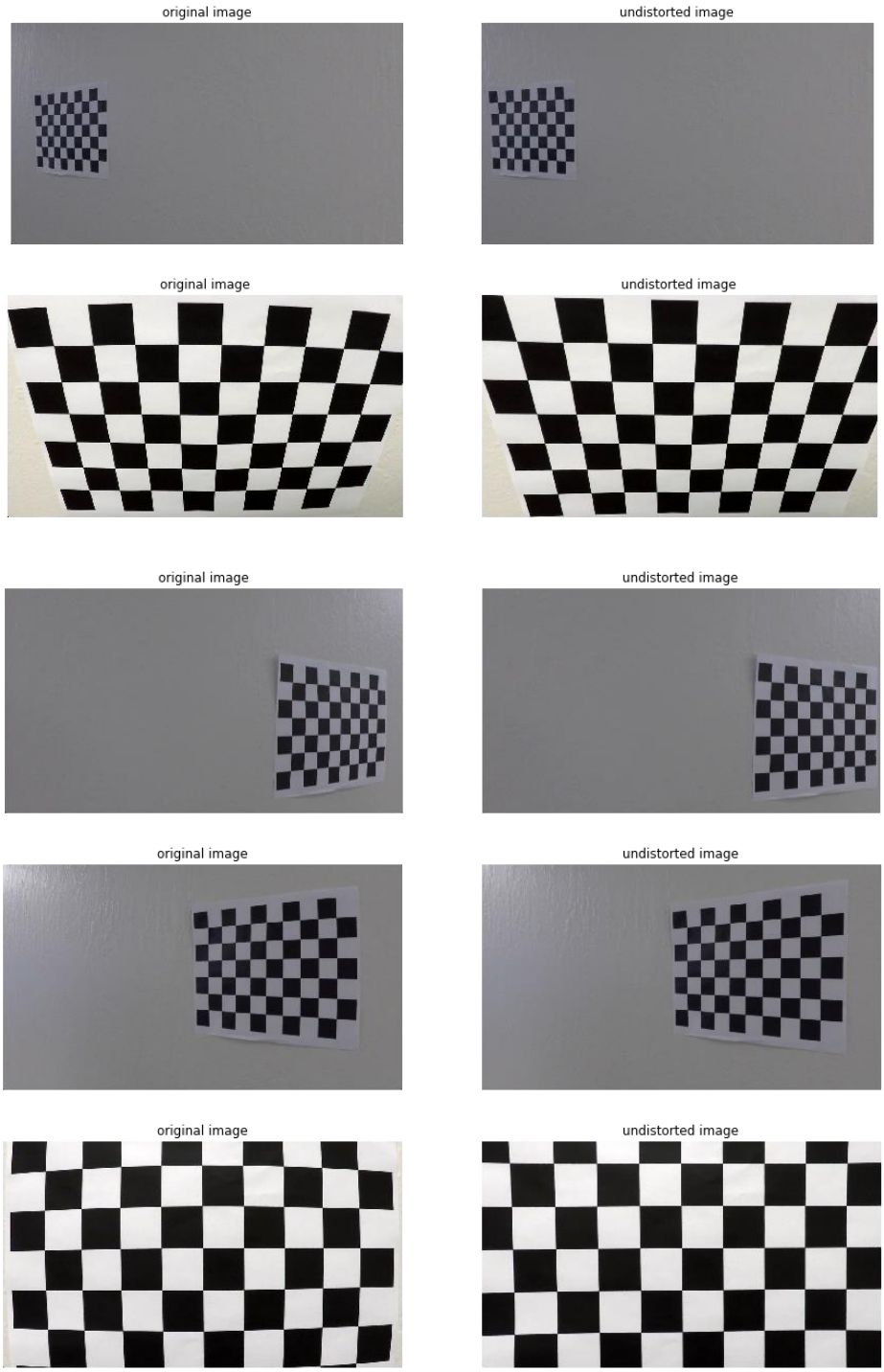
Rubric Points

Camera Calibration

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

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.

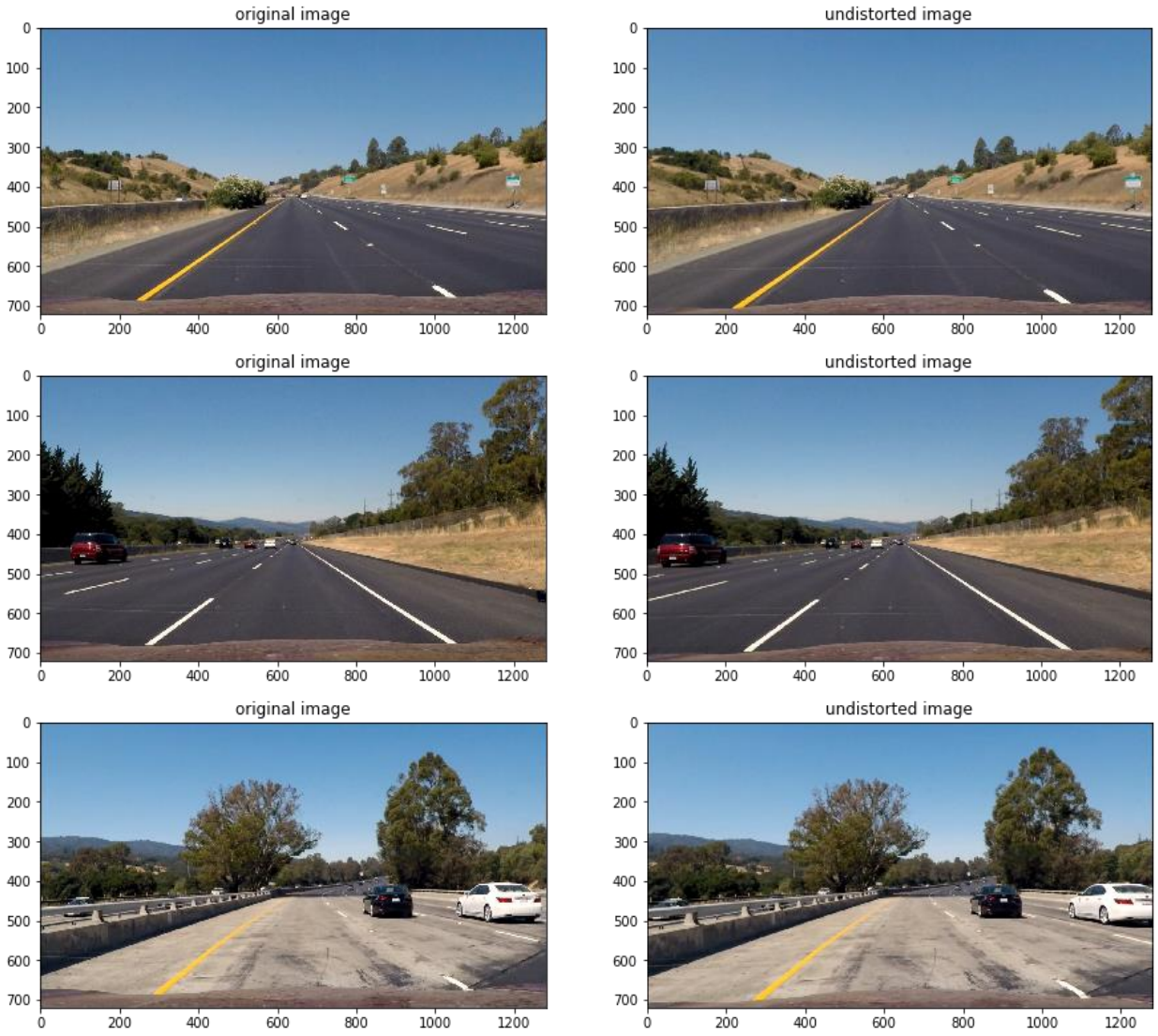
I then used the output `objpoints` and `imgpoints` to compute the camera calibration and distortion coefficients using the `cv2.calibrateCamera()` function. I applied this distortion correction to the test image using the `cv2.undistort()` function and obtained this results: (some sample results)



Pipeline (single images)

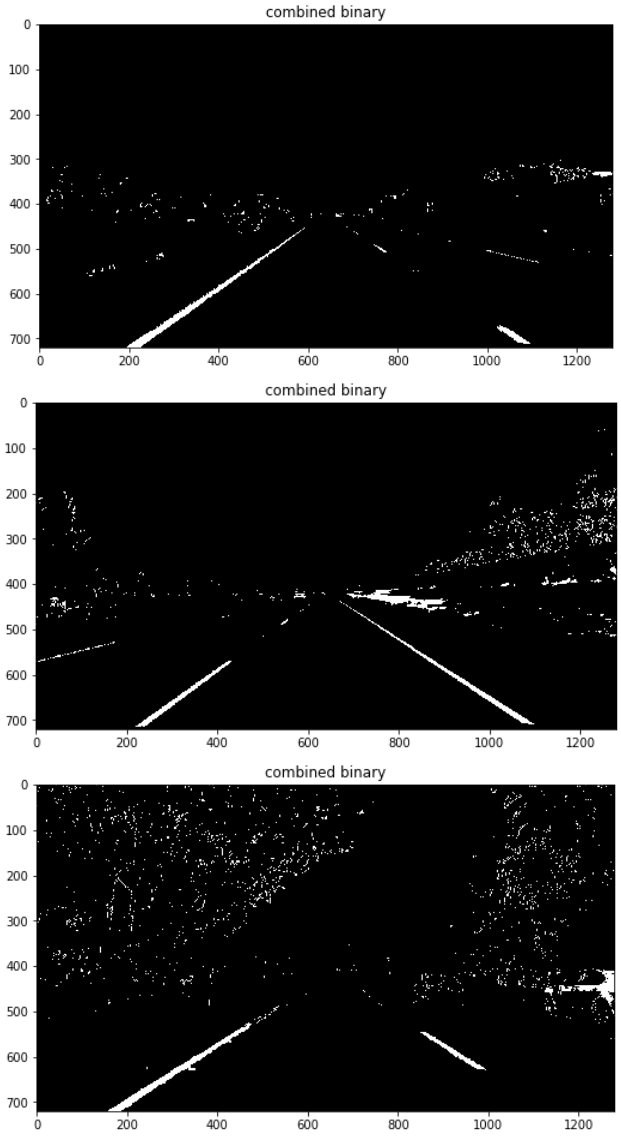
1. **Has the distortion correction been correctly applied to each image?**

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?

Oooh, binary image... you mean these?



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

The code for my perspective transform is includes two functions called `perspective_transform()` and `bird_eye()`, which appears in helpe functions part. `bird_eye()` function gets the original image, calibration matrix and undistortion coefficients, src and dst points and finally thresholds of s_channel in hls color space, sobel gradient on l_channel of hls color space and R_channel of RGB color space as follow:

```
bird_eye(image, mtx, dist, src, dst, s_thresh=(125,255),
sx_thresh=(10,100), R_thresh = (200, 255))
```

I tested different ways and channels and color spaces and finally selected R_channel from RGB and s_channel from hls color space and combined these binary color metrices with sobel x gradient on l_channel of hls color space. It's performance is good.

Src and dst points selected as follow using the following code:

```
src = np.float32(
[[(img_size[1] / 2) - 55, img_size[0] / 2 + 100],
[((img_size[1] / 6) - 10), img_size[0]],
[(img_size[1] * 5 / 6) + 60, img_size[0]],
[(img_size[1] / 2 + 55), img_size[0] / 2 + 100]])
```

```
dst = np.float32(
[[(img_size[1] / 4), 0],
[(img_size[1] / 4), img_size[0]],
[(img_size[1] * 3 / 4), img_size[0]],
[(img_size[1] * 3 / 4), 0]])
```

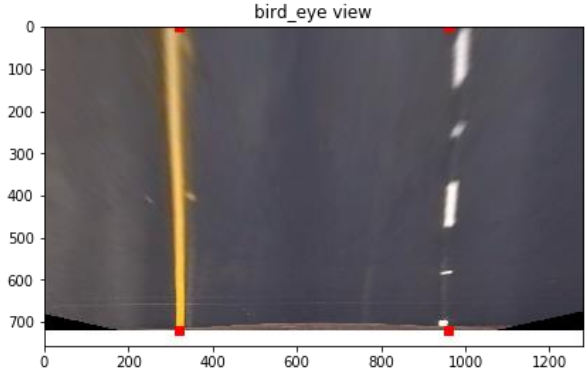
src:

```
array([[  585.        ,   460.        ],
       [  203.33332825,   720.        ],
       [ 1126.66662598,   720.        ],
       [  695.        ,   460.        ]], dtype=float32)
```

dst:

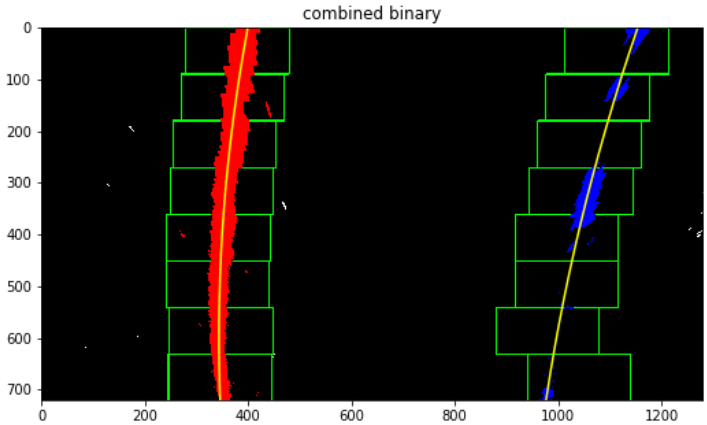
```
array([[ 320.,    0.],
       [ 320.,  720.],
       [ 960.,  720.],
       [ 960.,    0.]], dtype=float32)
```

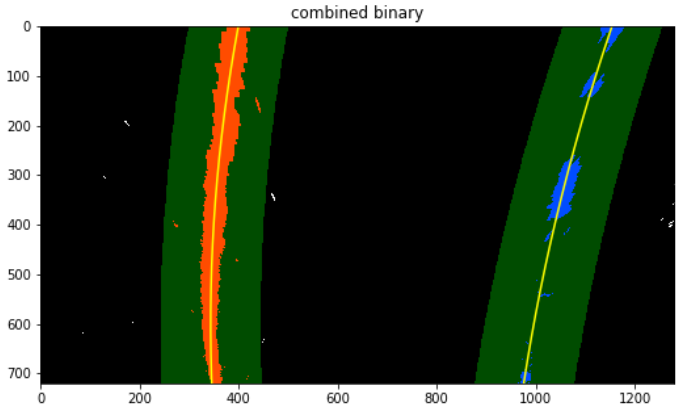
I verified that my perspective transform was working as expected by drawing the `src` and `dst` points onto a test image and its warped counterpart to verify that the lines appear parallel in the warped image:



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?**

Yes. You can see results in video. Here is a link to my video result.

Pipeline (video)

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

Yes. Here is a link to my video result.