1. Advanced Lane Finding Project

The goals / steps of this project are the following:

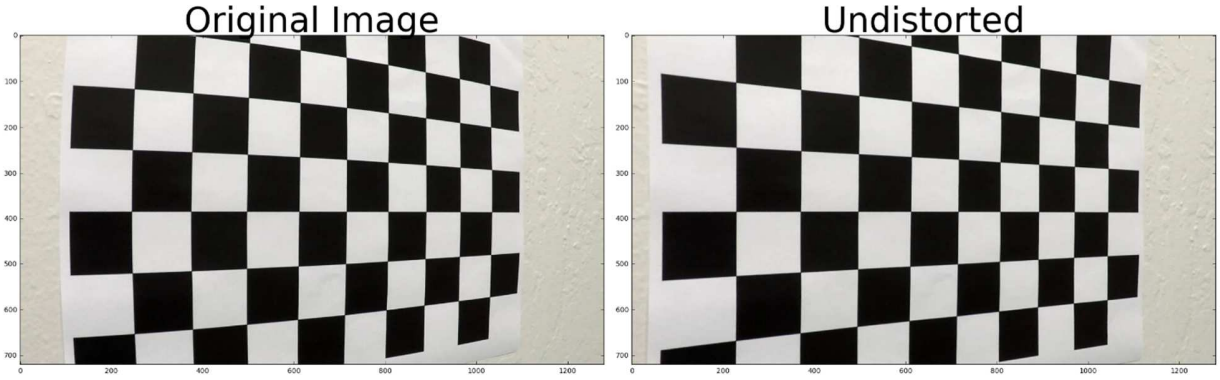
- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- 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.

2. Rubric Points

2.1. Camera Calibration

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

The code for this step is contained in the second code cell of the IPython notebook 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. 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:



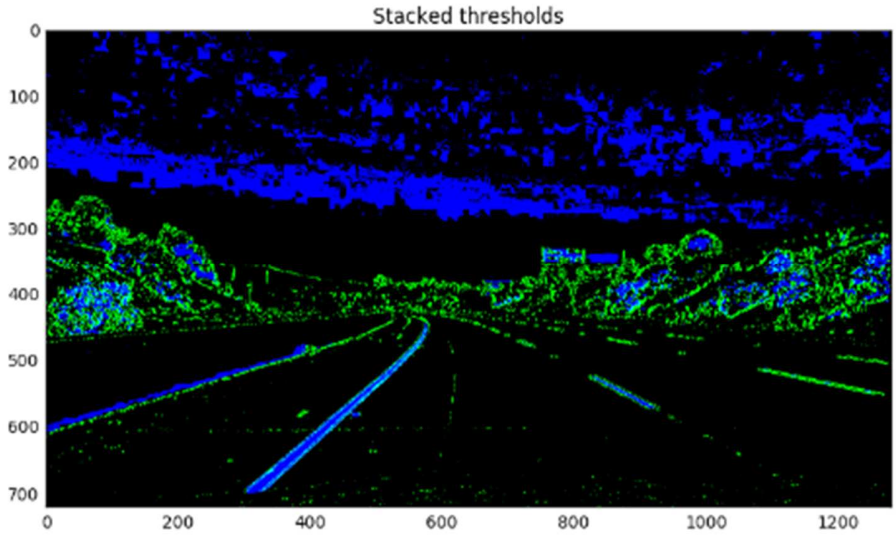
2.2. Pipeline (single images)

Has the distortion correction been correctly applied to each image?
Yes, the distortion correction applied to each image. I used the camera matrix and distortion coefficients obtained from the camera calibration routine

Has a binary image been created using color transforms, gradients or other methods?
Yes, I applied the following transformation

- **Gradient Threshold:** I converted the image from RGB to a gray image. Then, I applied an x direction sobel operator to extract the x-direction gradient of the gray image. Then I created a binary image by applying a min and max threshold on the gradient image.
- **Color Threshold:** I converted the RGB image to HLS image space. Then I applied saturation threshold to create a binary image.

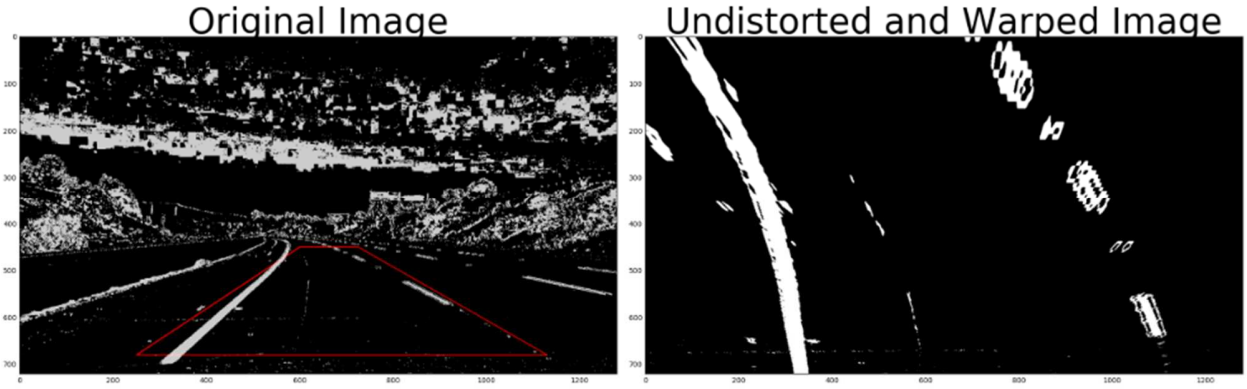
I created a final binary image by applying OR on the threshold binary images. The final color binary image looks like following
Blue color represents the contribution of saturation threshold and green color represents gradient threshold



Has a perspective transform been applied to rectify the image?
The code for my perspective transform is in 2nd cell of ipython notebook. I chose the hardcode the source and destination points in the following manner:

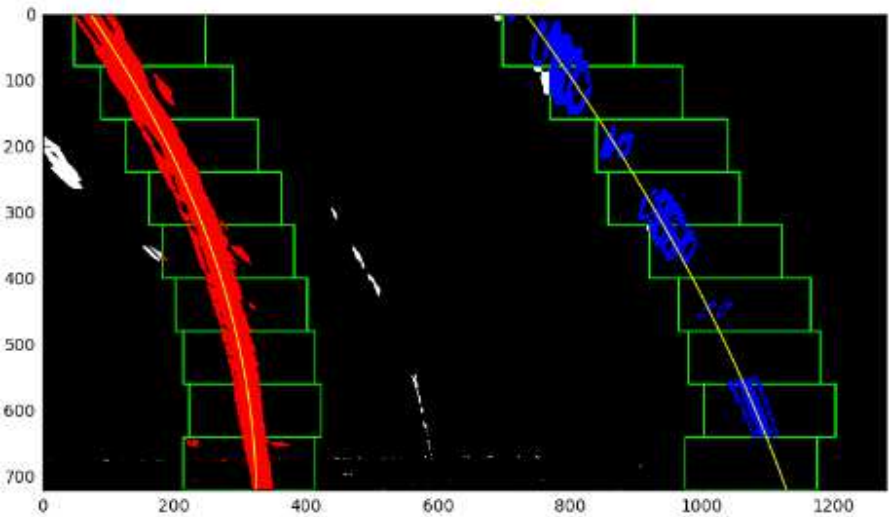
```
# For source points I'm grabbing the outer four detected corners
corners=np.zeros((4,2),np.float32)
corners=[[600,450],[725,450],[1130,680],[250,680]]
src = np.float32([corners[0], corners[1], corners[2], corners[3]])

# For destination points, I'm arbitrarily choosing some points to be
# a nice fit for displaying our warped result
# again, not exact, but close enough for our purposes
dst = np.float32([[250,0], [1130, 0], [1130,720], [250,720]])
```

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



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

Yes, I am able to detect the lane lines in the warped image and able to fit with a polynomial



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, I estimated the radius of curvature

2.3. Pipeline (video)

Does the pipeline established with the test images work to process the video?
Yes, here is the link

Discussion

Here I'll talk about the approach I took, what techniques I used, what worked and why, where the pipeline might fail and how I might improve it if I were going to pursue this project further

I used the following technique for robust detection of lane lines in a video:

- Smoothing: As soon as lanes are detected, I average the curvature coefficient of the last 10 frames and used this value for the current lane line. This approach has reduced the flickering of lane line detections

- When I am not able to detect a lane line confidently, then I use the previously smoothened curvature coefficients to draw the lane lines

- I would like to use better thresholding techiques (i.e. usage of other color spaces) to filter out only the lane lines more robustly.