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

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it!

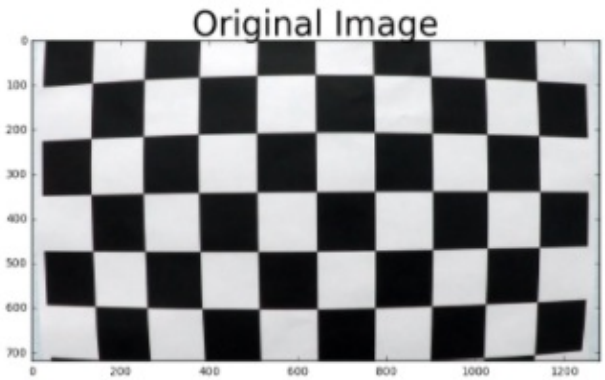
Camera Calibration

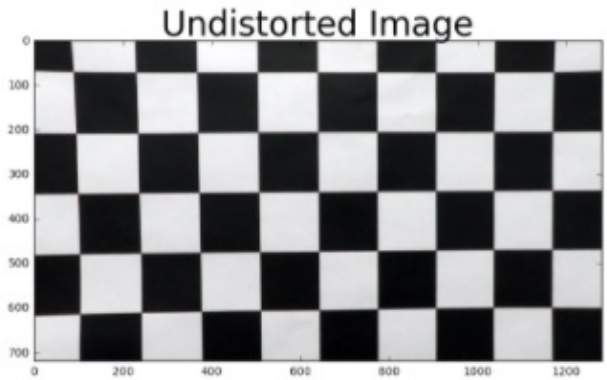
1. Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.

The code for this step is contained in the first section of the IPython notebook located in "./examples/example.ipynb".

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. Provide an example of a distortion-corrected image.

The code for this step is contained in the second section of the IPython

notebook located in "./examples/example.ipynb".

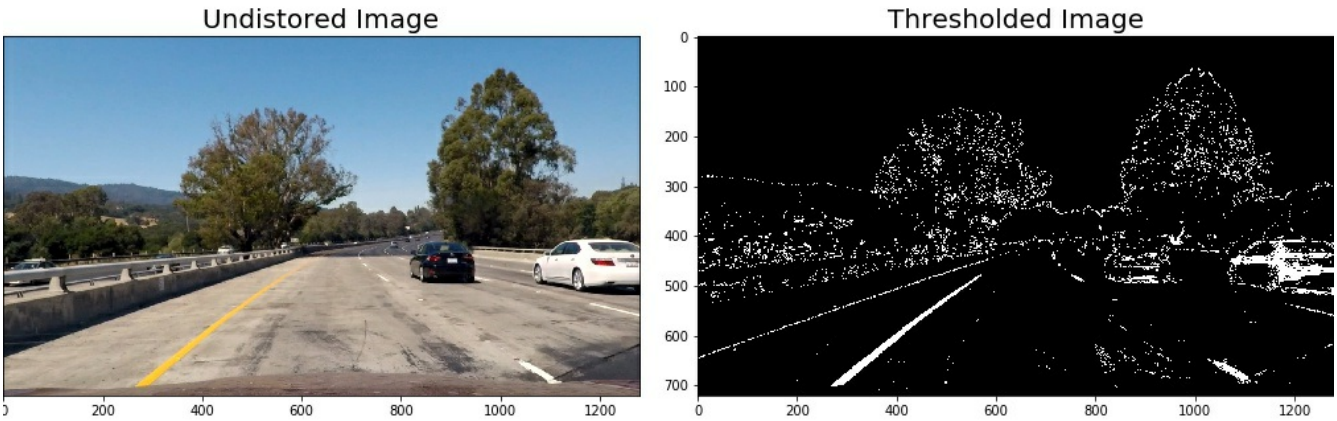
Given camera matrix and distortion params we can remove distortion caused by particular camera from images.

To demonstrate this step, I will describe how I apply the distortion correction to one of the test images like this one:

2. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

The code for this step is contained in the third section of the IPython notebook located in "./examples/example.ipynb".

I used a combination of color and gradient thresholds to generate a binary image. Here's an example of my output for this step.



3. Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

The code for my perspective transform includes a function called warp(), which appears in the fourth section of the IPython notebook. The warp() function takes as inputs an image (img), as well as source (src) and destination (dst) points. I chose the hardcode the source and destination points in the following manner:

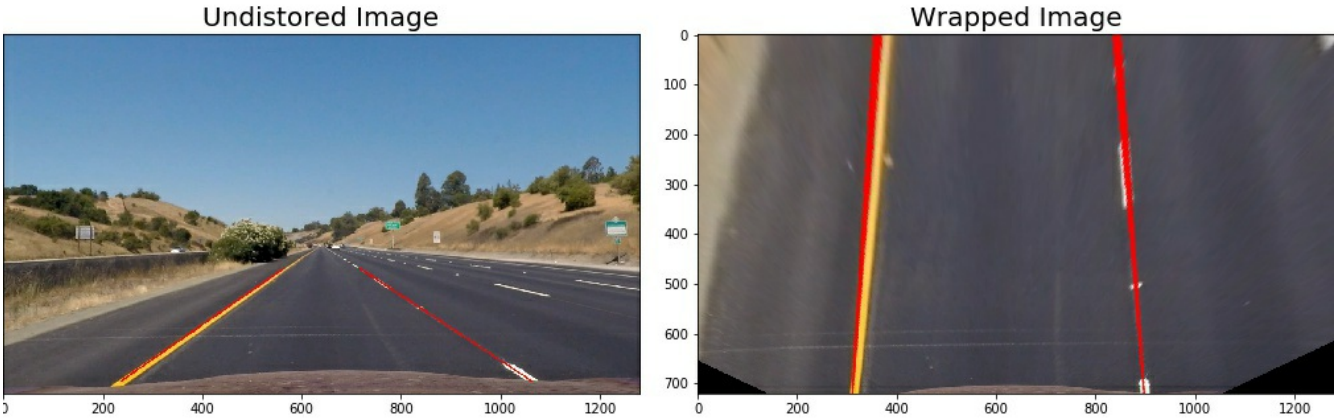
```
#select source points
    srcpoint1 = (img.shape[1] * 0.429,img.shape[0] *
0.652 ) #top left
    srcpoint2 = (img.shape[1] * 0.566,img.shape[0] *
0.652) #top right
    srcpoint3 = (img.shape[1] *
0.820,img.shape[0]-25) #bottom right
    srcpoint4 = (img.shape[1] * 0.195,
img.shape[0]-25) #bottom left

#select destination points
    dstpoint1 = (220,0) #top left
    dstpoint2 = (1050,0) #top right
    dstpoint3 = (1050, img.shape[0]) #bottom left
    dstpoint4 = (220,img.shape[0]) #bottom right
```

This resulted in the following source and destination points:

Source	Destination
580, 460	320, 0
755, 720	960, 0
1150, 695	960, 720

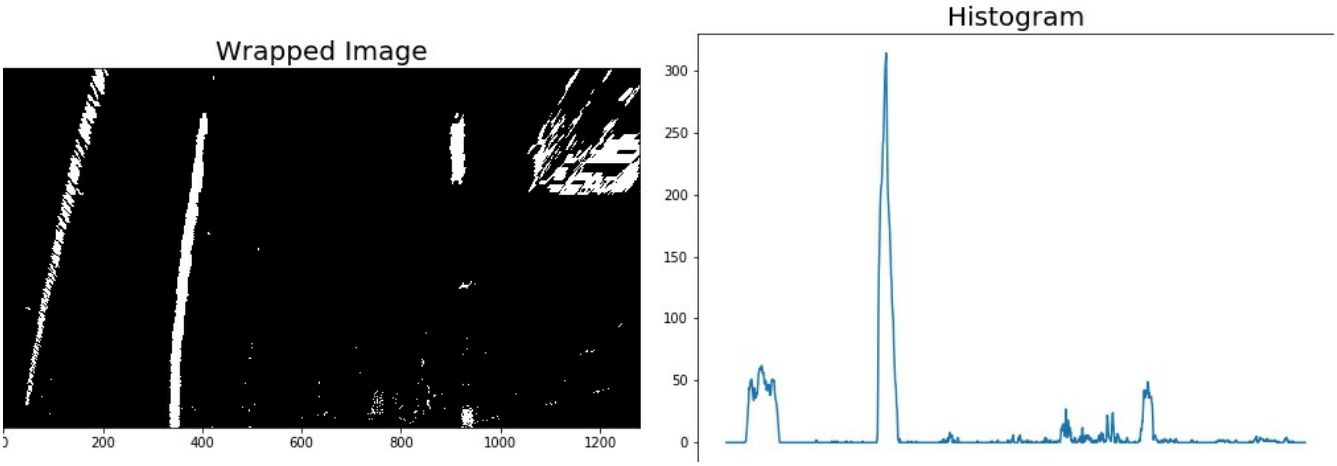
195, 695 320, 720

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.

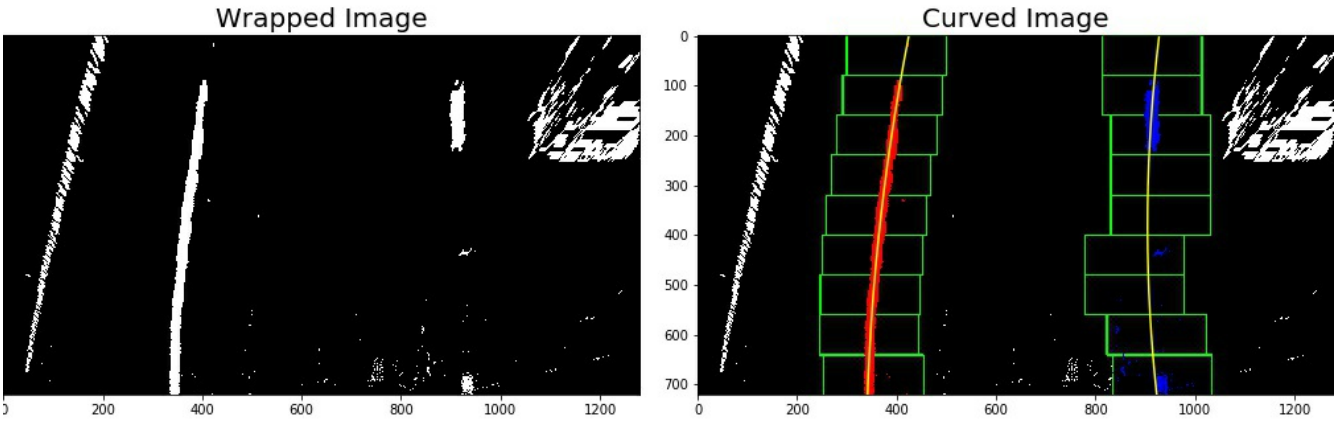


4. Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?

In the code for my lane indentification is used histogram approach for finding my starting points for left and right lanes, which appears in the fifth section of the IPython notebook.



And fit my lane lines with a 2nd order polynomial for left and right lanes like this:



5. Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

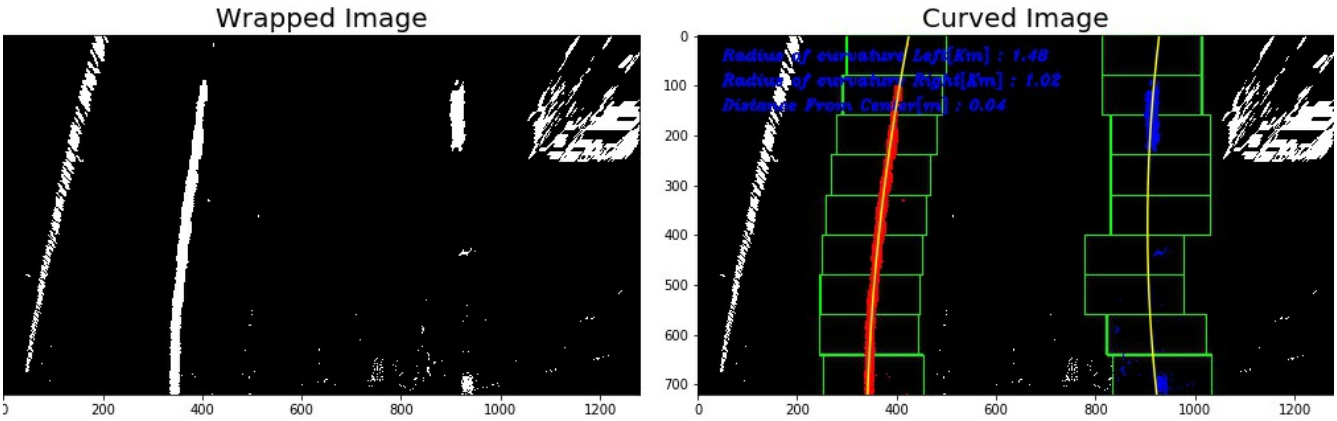
I did this in sixth section of my Ipyhton notebook. Differentiated the 2nd degree polynomial and found the curvature value for the bottom of the image i.e near my car.

And for finding lane vehicle center i did this in three steps.

1. Found the center of the image.

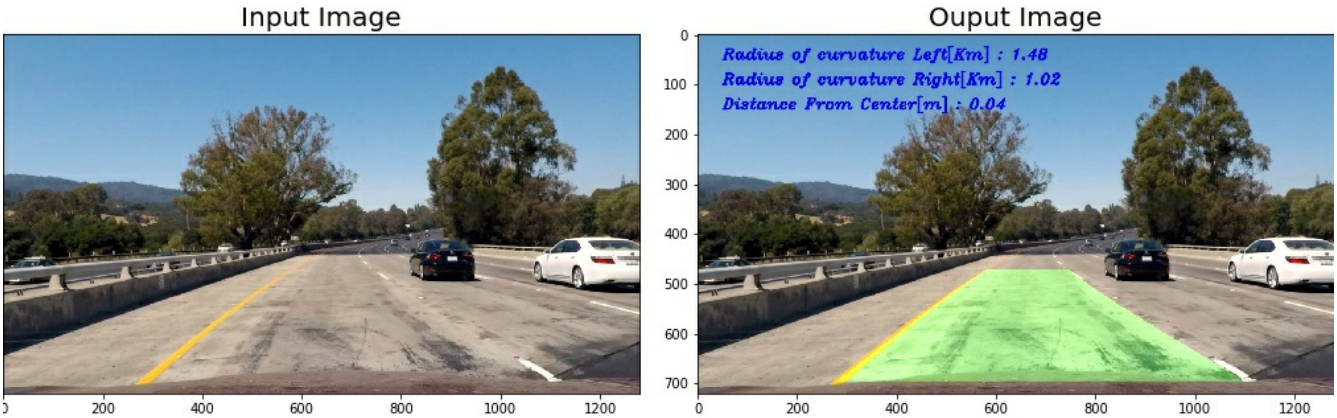
2. Found the difference between left lane starting point and right lane starting point of the image and caluclated their center.

3. Finally, subtracted the above two to get position of the vehicle with respect to center of image.



6. Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly.

I implemented this in seventh section in my code in Ipython notebook in the function final_result(). Here is an example of my result on a test image:



Pipeline (video)

1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (wobbly lines are ok but no catastrophic failures that would cause the car to drive off the road!).

Here's a link to my video result

Discussion

1. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

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.