Project: Advanced Lane Finding

Juan Cheng 2017.09.05

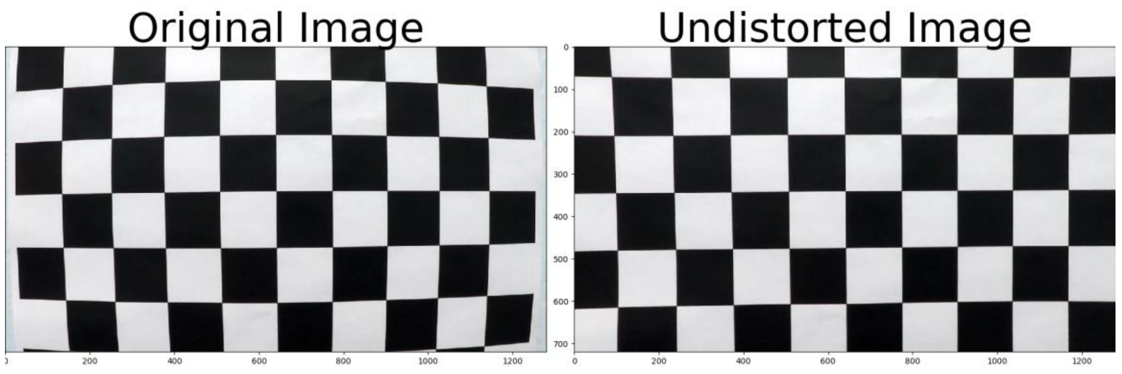
The goals / steps of this project are the following:

- 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.

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 this step is contained in lines 11 through 56 of the file called advanced_lane_pipeline.py.
I started 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 calibration image using the cv2.undistort() function and obtained this result:

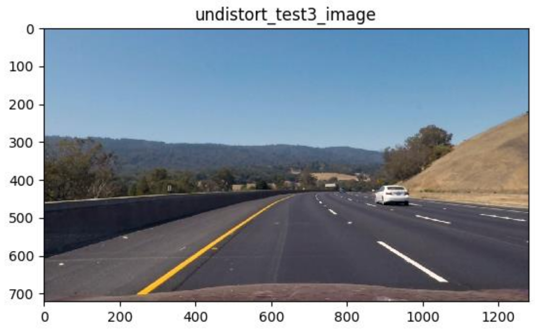


Pipeline (single images)

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

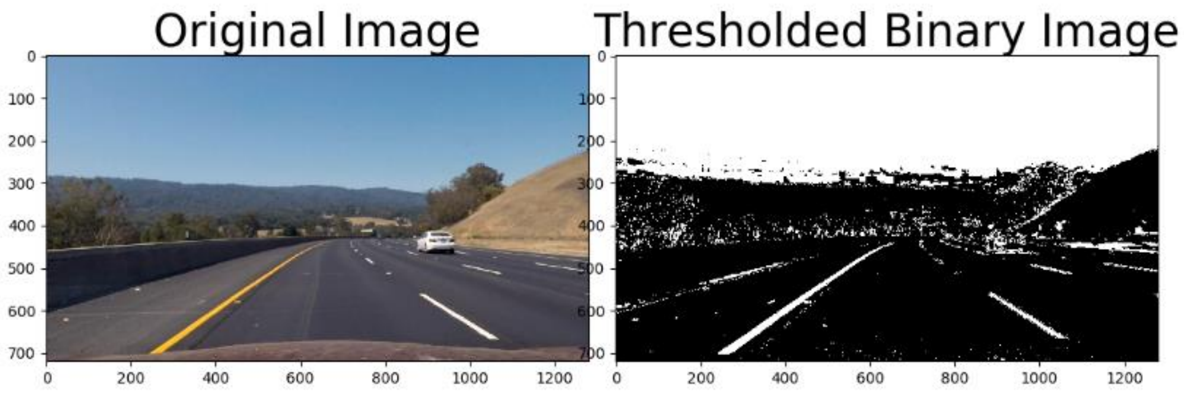
To demonstrate this step, I will describe how I apply the distortion correction to one of the test images like this one (original test3.jpg):

I used the same camera calibration and distortion coefficients, applying to the cv2.calibrateCamera() function and got this result:



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

Yes, I used sobelx gradient with kernel size 15 (choosed a larger odd number to smooth gradient measurements) and thresholds (20,100), combined with S-channel thresholds(90, 255) of HLS color space to obtain this result:



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

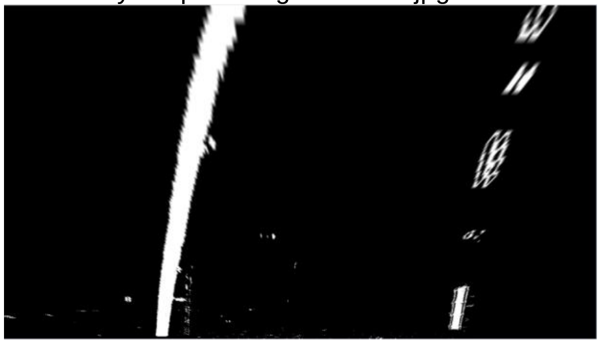
Yes. The code for my perspective transform includes a function called warper() , which appears in lines 117 through 130 in the file advanced_lane_pipeline.py. The warper() function takes as inputs an image (img), as well as source (src) and destination (dst) points. It outputs a warped image and Minv(inversed perspective transform, for later usage). I chose the hardcode the source and destination points in the following manner:

```
src = np.float32(
    [[(img_size[0] / 2) - 55, img_size[1] / 2 + 100],
    [((img_size[0] / 6) - 10), img_size[1]],
    [(img_size[0] * 5 / 6) + 60, img_size[1]],
    [(img_size[0] / 2 + 55), img_size[1] / 2 + 100]])
dst = np.float32(
    [[(img_size[0] / 4), 0],
    [(img_size[0] / 4), img_size[1]],
    [(img_size[0] * 3 / 4), img_size[1]],
    [(img_size[0] * 3 / 4), 0]])
```

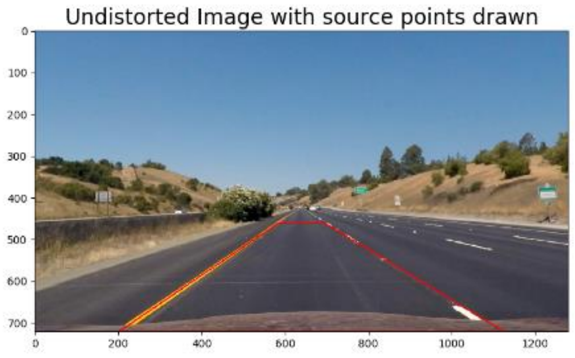
This resulted in the following source and destination points:

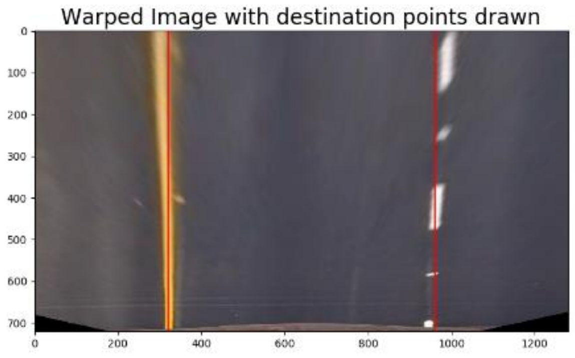
Source points	Destination points
585, 460	320, 0
203, 720	320, 720
1126, 720	960, 720
695, 460	960, 0

The binary warped image on test3.jpg is as follows:



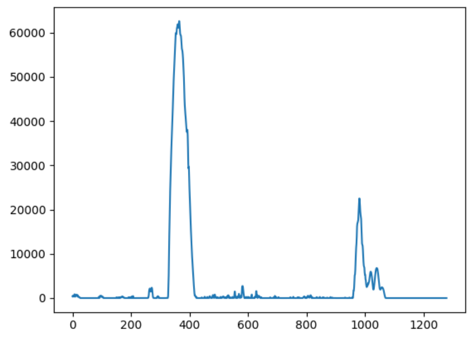
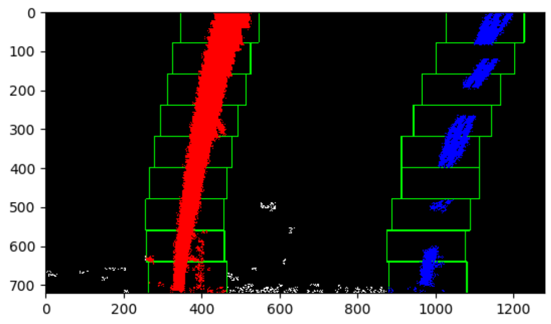
I also verified that my perspective transform was working as expected by drawing the src and dst points onto a test image (straight_lines1.jpg) and its warped counterpart to verify that the lines appear parallel in the warped image.



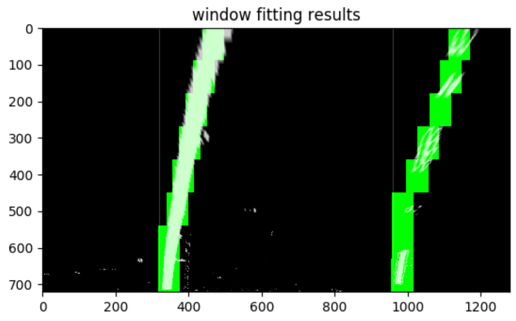


4. Have lane line pixels been identified in the rectified image and fit with a polynomial?

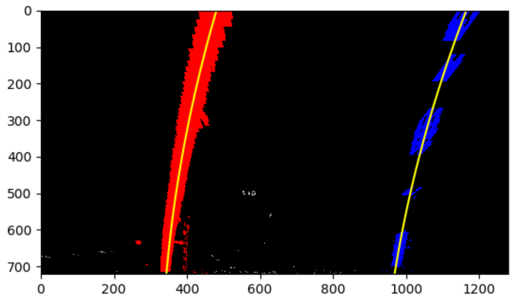
Yes, I tried two methods as directed in the course. One is using peaks in a histogram and sliding window searching as follows:

The other is using convolution to find the fitting lane lines:



And then I fitted my lane lines with a 2nd order polynomial kinda like this:



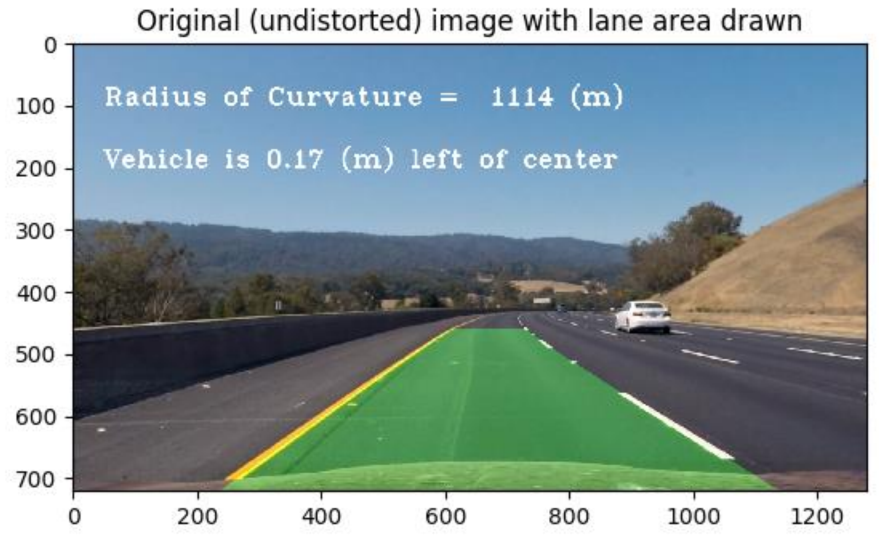
5. Having identified the lane lines, has the radius of curvature of the road been estimated? And the position of the vehicle with respect to center in the lane?

Yes, I did calculated the radius of curvature and the offset of vehicle to the lane center, as implemented in lines 261 through 303 in my code.
The curvature is calculated part of polyfit(), as it's known as a function of polynomial coefficients.
Assume the camera is mounted at the center of the car, such that the lane center is the midpoint at the bottom of the image between the two lines. The offset of the lane center from the center of the image (converted from pixels to meters) is the vehicle position from the center of the lane.

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 lines 284 through 318 in my code. I plotted white text to show vehicle offset when it's less than 0.35m, yellow text to show vehicle offset when it's greater than 0.35m and less than 0.5m, red text to show vehicle offset when it's greater than 0.50m as warnings.

Here is an example of my result on a test image:


Original (undistorted) image with lane area drawn
Radius of Curvature = 1114 (m)
Vehicle is 0.17 (m) left of center
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

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!).

I've included in the submission my video result (project_video_output.mp4) from project_video.mp4
Also tried challenge videos with 5 seconds clip outputs:
challenge_video_output.mp4, harder_challenge_video_output.mp4

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?

I see there are a few lane fit glitches from the project video output, which implies the pipeline doesn't work very well on those sharp turn corner cases. When I ran pipeline on the video stream, I didn't keep track of all the interesting parameters that I measure from frame to frame. So each frame basically did histogram and sliding window search from scratch. Future improvement can be made on keep tracking of recent detections, smoothing measurements and performing more sanity check. For example, in my current

implementation, I report the radius of curvature using the mean value from both left and right curvature (line 301). However, when the left and right lane curvatures are quite different from each other, some filter algorithm should be applied to rule out the insane one.
Also, the preprocessing of the images could be more robust and better. I've only used sobelx gradient and s channel thresholds to obtain the binary image. For challenge videos, it doesn't work perfectly. Other methods like gradient magnitude and gradient direction thresholds can be combined as well to tune for better performance on corner cases.