**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.

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

### Writeup / README

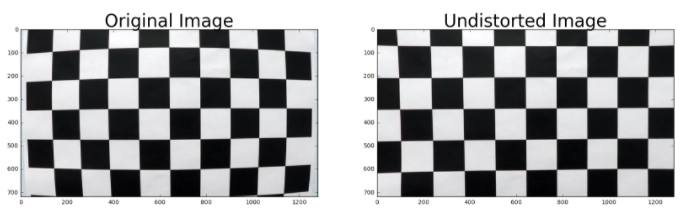
#### 1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. [Here](https://github.com/udacity/CarND-Advanced-Lane-Lines/blob/master/writeup_template.md) is a template writeup for this project you can use as a guide 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 to calibrate the camera is contained in the CalibrateCamera function. This functions looks for all images called “calibration\*.jpg” in the ‘camera\_cal’ folder. This uses a 9X6 chessboard to calibrate the camera. I saved the camera matrix and distortion info to a pickle file. I found calibrating the camera each time I ran the project was very time consuming. I check to see if the pickle file exists, and if it does I load the values from it. If it doesn’t exist, then I go ahead and calibrate the camera.

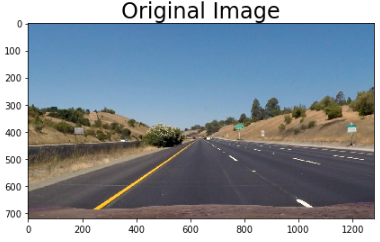
[](https://github.com/udacity/CarND-Advanced-Lane-Lines/blob/master/examples/undistort_output.png)

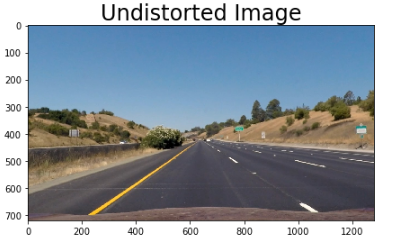
### Pipeline (single images)

#### 1. Provide an example of a distortion-corrected image.

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

To undistort the image, I used the camera calibration obtained in the previous step and used cv2.undistort() to undistort the image.



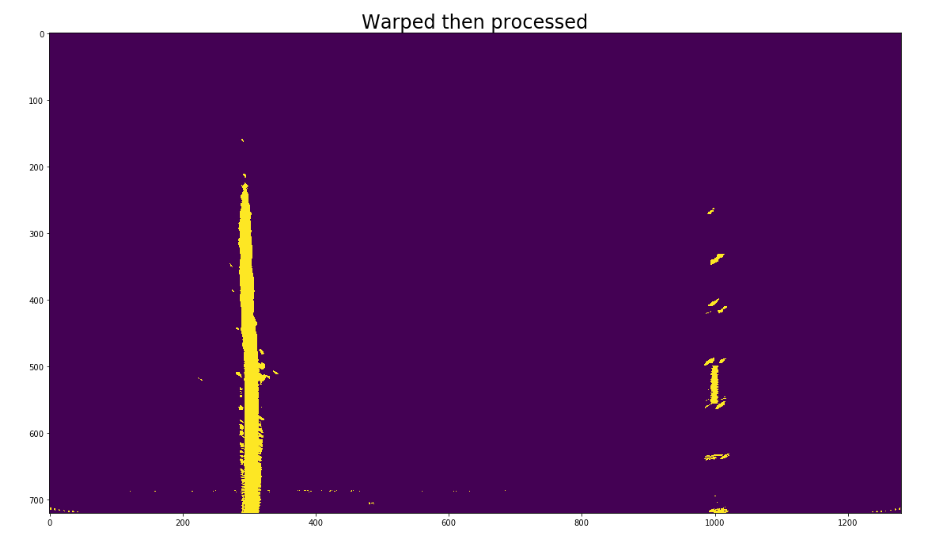


#### 2. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

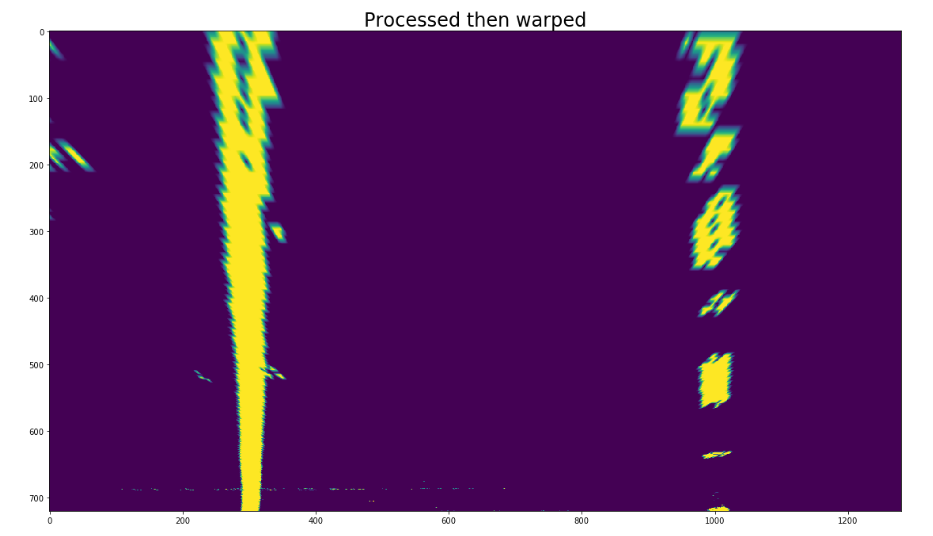
For this step, I used a number of different methods before finally settling on one which worked pretty well. Originally, I used the methods shown in class, where we got the gradient of x and y, as well as the magnitude and direction of the gradient. I received very mixed results with this. I then watched the P$ Q&A video and decided to go with the gradients of X and Y, and do a color threshold using the S channel of a HLS image, and the V channel of a HSV image. This produced very clean and crisp results.

Once I did this, I ran in to another problem. When the images were undistorted the lines could get blurry. This caused them to not show up when I did the gradient and color work on a warped image. I originally thought it would be more efficient to do the threshold and gradient work on the warped image, being it had less ‘data’ in it. I found I was wrong, and it worked a lot better when I did the gradient and threshold work on the full image, and *then* perform the image warp. Warping of the image is described in the next section

Here is an image which was warped prior to processing



Finally, the image when it was processed and then warped.



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

The perspective transform (warp) was performed using dynamic points which can change with the size of the image. I used an offset of 15% of the image size. I also used points similar to what I used for the mask in the first lane finding project:

botWidth = .76 – this is the width of the bottom of the trapezoid

midWidth = .08 – this is the width of the top of the trapezoid

heightPct = .62 – this is the height of the trapeziod

bottomTrim = .935 – this trims the hood of the car off

src = np.float32([[img\_size[0]\*(.5-midWidth/2), img\_size[1]\*heightPct],

[img\_size[0]\*(.5+midWidth/2), img\_size[1]\*heightPct],

[img\_size[0]\*(.5-botWidth/2), img\_size[1]\*bottomTrim],

[img\_size[0]\*(.5+botWidth/2), img\_size[1]\*bottomTrim]])

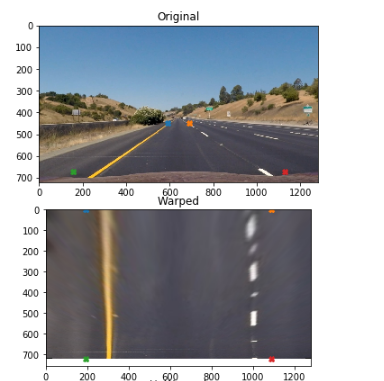
dst = np.float32([[offset, 0],

[img\_size[0]-offset,0],

[offset, img\_size[1]],

[img\_size[0]-offset, img\_size[1]]])

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?

#### 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 lines # through # in my code in my\_other\_file.py

#### 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 step in lines # through # in my code in yet\_another\_file.py in the function map\_lane(). Here is an example of my result on a test image:

[](https://github.com/udacity/CarND-Advanced-Lane-Lines/blob/master/examples/example_output.jpg)

### 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](https://github.com/udacity/CarND-Advanced-Lane-Lines/blob/master/project_video.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?

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.