

# [Project Writeup] Advanced Lane Lines [Template]

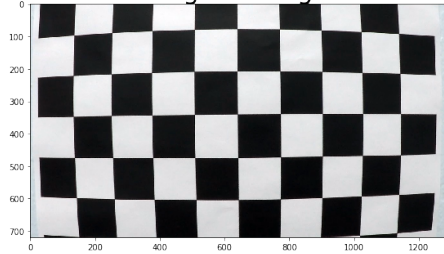
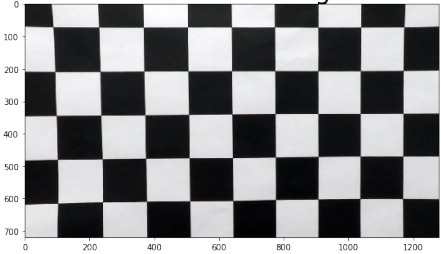
## PROJECT SPECIFICATION

### Advanced Lane Finding

#### Writeup / README

CRITERIA	MEETS SPECIFICATIONS
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. <a href="#">Here</a> is a template writeup for this project you can use as a guide and a starting point.	<p>The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled.</p> <p>I have added the images in the Distortion_Correction.html file.</p>

#### Camera Calibration

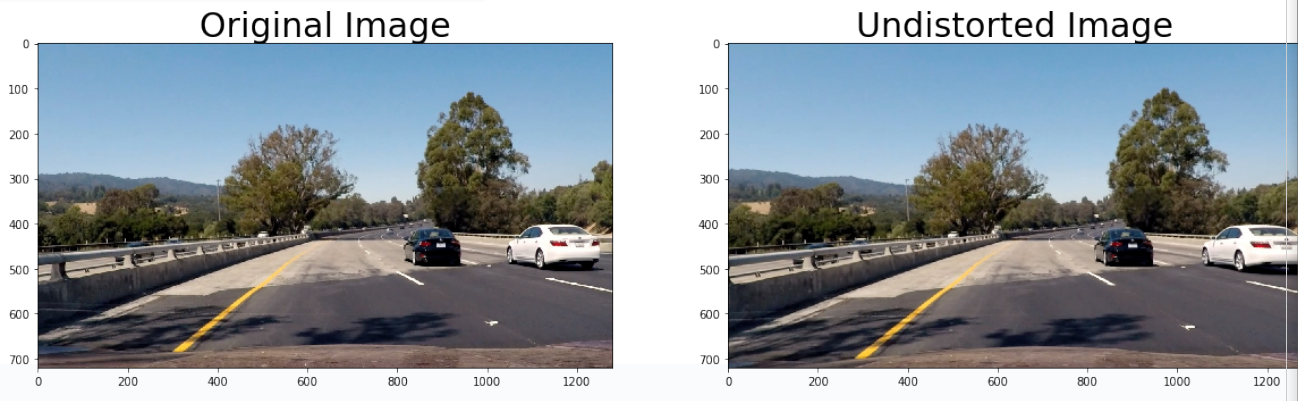
CRITERIA	MEETS SPECIFICATIONS
Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.	<p>OpenCV functions or other methods were used to calculate the correct camera matrix and distortion coefficients using the calibration chessboard images provided in the repository (note these are 9x6 chessboard images, unlike the 8x6 images used in the lesson). The distortion matrix should be used to undistort one of the calibration images provided as a demonstration that the calibration is correct. Example of undistorted calibration image is Included in the writeup (or saved to a folder).</p> <div><div><p>Original Image</p></div><div><p>Undistorted Image</p></div></div>

#### Pipeline (test images)

CRITERIA	MEETS SPECIFICATIONS
Provide an	Distortion correction that was calculated via camera calibration has been

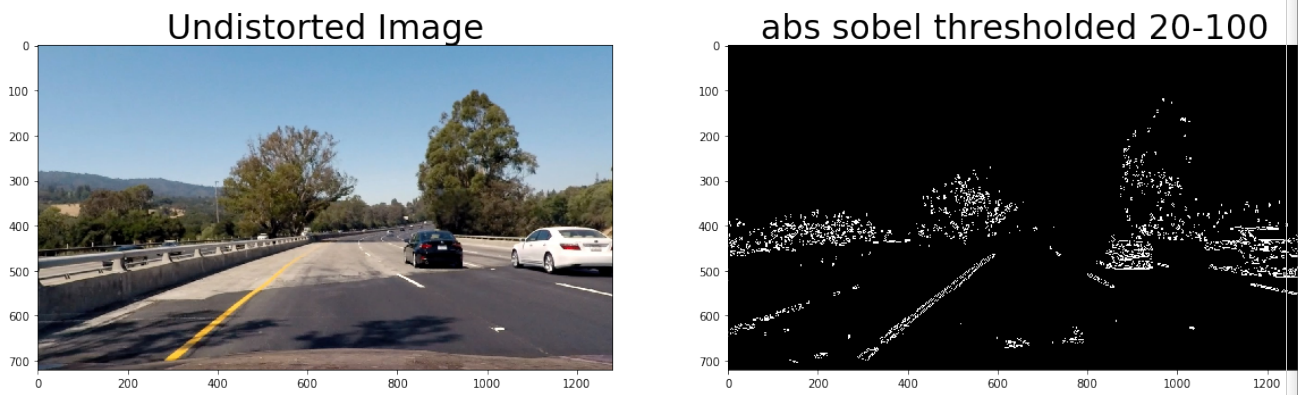
example of a distortion-corrected image.

correctly applied to each image. An example of a distortion corrected image should be included in the writeup (or saved to a folder) and submitted with the project.



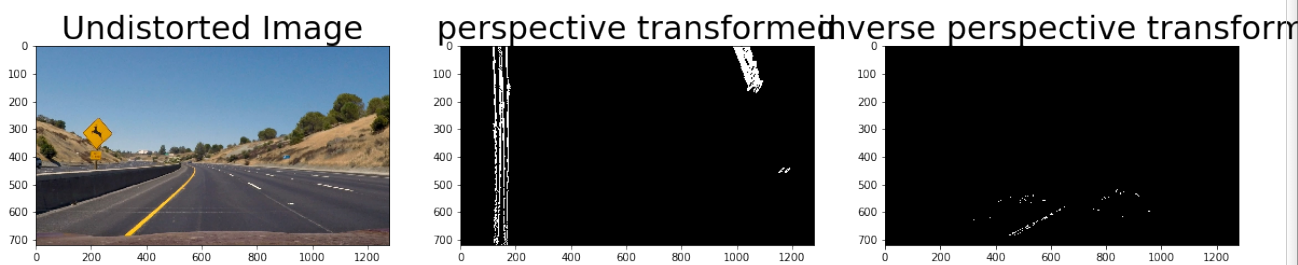
Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

A method or combination of methods (i.e., color transforms, gradients) has been used to create a binary image containing likely lane pixels. There is no "ground truth" here, just visual verification that the pixels identified as part of the lane lines are, in fact, part of the lines. Example binary images should be included in the writeup (or saved to a folder) and submitted with the project.



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

OpenCV function or other method has been used to correctly rectify each image to a "birds-eye view". Transformed images should be included in the writeup (or saved to a folder) and submitted with the project.



Describe how (and identify where in your code) you identified lane-line pixels and fit their positions

Methods have been used to identify lane line pixels in the rectified binary image. The left and right line have been identified and fit with a curved functional form (e.g., spline or polynomial). Example images with line pixels identified and a fit overplotted should be included in the writeup (or saved to a folder) and submitted with the project.

with a polynomial?



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.

Here the idea is to take the measurements of where the lane lines are and estimate how much the road is curving and where the vehicle is located with respect to the center of the lane. The radius of curvature may be given in meters assuming the curve of the road follows a circle. For the position of the vehicle, you may assume the camera is mounted at the center of the car and the deviation of the midpoint of the lane from the center of the image is the offset you're looking for. As with the polynomial fitting, convert from pixels to meters.

```
def compute_rad_curv(xvals, yvals):  
    ym_per_pix = 30/720 # meters per pixel in y dimension  
    xm_per_pix = 3.7/700  
    fit_cr = np.polyfit(yvals*ym_per_pix,  
xvals*xm_per_pix, 2)  
    y_eval = np.max(yvals)  
    curverad = ((1 + (2*fit_cr[0]*y_eval +  
fit_cr[1])**2)**1.5) \  
                /np.absolute(2*fit_cr[0])  
  
    return curverad
```

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



Pipeline (video)

CRITERIA	MEETS SPECIFICATIONS
Provide a link to your final video output. Your pipeline	The image processing pipeline that was established to find the lane lines in images successfully

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!)	processes the video. The output here should be a new video where the lanes are identified in every frame, and outputs are generated regarding the radius of curvature of the lane and vehicle position within the lane. The pipeline should correctly map out curved lines and not fail when shadows or pavement color changes are present. The output video should be linked to in the writeup and/or saved and submitted with the project.  <a href="https://youtu.be/2umBw6NIT5g">https://youtu.be/2umBw6NIT5g</a>
---	---

Discussion

CRITERIA	MEETS SPECIFICATIONS
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?	Discussion includes some consideration of problems/issues faced, what could be improved about their algorithm/pipeline, and what hypothetical cases would cause their pipeline to fail.  Manitude threshold and direction did not work very well for me.