# 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. | 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 un-distort 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). |

I created a function called “FindCorners” to find the corners on the calibration images provided with the repository.

Within the function, I am looping over all the images, read the files one by one, grayscale them and call **cv2.findChessboardCorners**. The number of corners: 9 horizontally, 6 vertically which I store in variables.

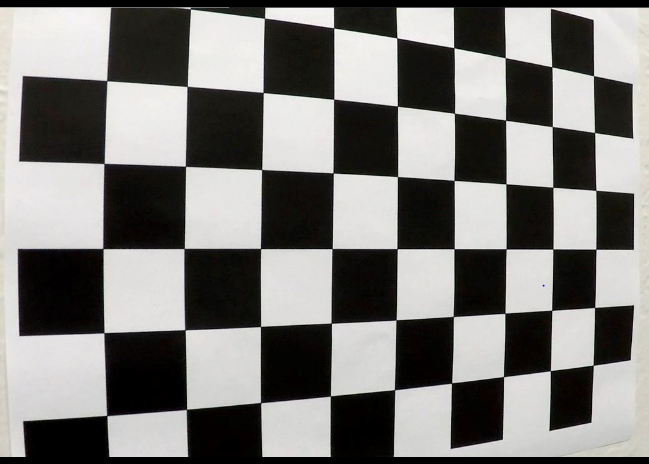
the result of **cv2.findChessboardCorners** is the detected coordinates of the corners on the chessboard image (calibration image corners) which we link to 3D real chessboard image corners (undistorted image corners). The format of the coordinates of the 3D image corners is (x,y,z), but z=0 always! Numpy.mgrid generates the coordinates for the 3D image.

I use the detected calibration image and undistorted image corners to calibrate the camera with function, **calibrateCamera of cv2**. The function return 5 values, but for calibration we need only the distortion coefficients and the camera matrix we need to transform 3D object point to 2D image points.

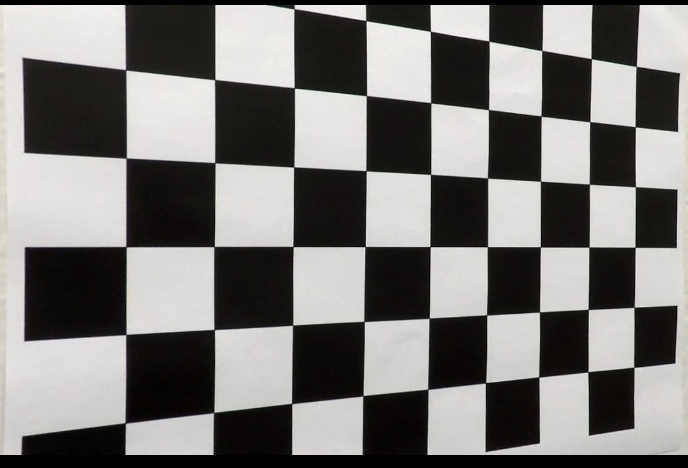
To undistort an image, I used **cv2** function **undistort**, which takes the calibration image, distortion coefficients and camera matrix as input and return the undistorted image.

This is how the distorted and undistorted images look like:

DISTORTED



DISTORTED



# Pipeline (test images)

|  |  |
| --- | --- |
| Provide an example of a distortion-corrected image. | Distortion correction that was calculated via camera calibration has been 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. |

To correct a distorted image, I applied the above mentioned cv2.undistort function to all the test images. The previously determined distortion coefficients and camera matrix were used as parameter.

This is a distorted image:



Undistorted image (the space between the back of the white car and the edge of the image is less than on the original image):



|  |  |
| --- | --- |
| 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 with a polynomial? | 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., spine 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. |

|  |  |
| --- | --- |
| 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. |

|  |  |
| --- | --- |
| Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly. | The fit from the rectified image has been warped back onto the original image and plotted to identify the lane boundaries. This should demonstrate that the lane boundaries were correctly identified. An example image with lanes, curvature, and position from center should be included in the writeup (or saved to a folder) and submitted with the project. |

# Pipeline (video)

| **Criteria** | **Meets Specifications** |
| --- | --- |
| 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!) | The image processing pipeline that was established to find the lane lines in images successfully 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. |

# 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. |