Homework 1: AutoCalibration

Prateek Arora
Masters of Engineering in Robotics
University of Maryland, College Park
Email: pratique@umd.edu
Using 2 Late day

I. Introduction

Camera Calibration is one of the most time consuming but significant process for any computer vision research involving 3D geometry, of which robotics research is a prime example. Calibration of camera involved determining the distortion matrix and intrinsic and extrinsic parameters of camera. An automatic way to perform efficient and robust camera calibration was provided by Zhengyou Zhang of Microsoft[1]. In this report I have summarized my implementation of Zhang's technique for calibration.

II. The Calibration Process

A. Calibration Data

Zhang's implementation requires a calibration target to estimate the camera intrinsics parameter. For this project 13 images of a checkerboard were provided and are present in the data folder. Checkerboard pattern is extremely helpful since the dimensions of each square is known and we can find perfect corners.

B. Approach

To begin with we need to estimate initial values for intrinsic parameters, extrinsic parameters and the distortion coefficients. Following steps were performed to get initial estimate:

1) Find corner points in the checkerboard for all the given images. This was done using the function *cv2.findChessboardCorners*
2) Take a corner of the checkerboard to be the origin of world frame and find the world coordinates of every corner point relative to the origin.
3) Compute the homographies between the world coordinate and the image coordinates of corners found.
4) Compute the Intrinsic parameters for using homographies calculated in the previous steps. The initial estimate for intrinsic camera matrix computed is

$$\begin{bmatrix} 2055.6777, & 0.000, & 764.2554 \\ 0.0000, & 2038.4953, & 1349.205 \\ 0.0000, & 0.0000, & 1.0000 \end{bmatrix} \tag{1}$$

5) Extrinsic parameter is calculated from the intrinsic matrix which is $[R|t]$
6) The assumption in calculating the initial estimate is that there is no distortion present in the image at all. We thus assume the value of k_c to be [0,0]
7) Now we perform non linear optimization on all the values calculated above. This is done using *scipy.optimize.least_squares* function of the scipy library. The following values of the intrinsic camera matrix and distortion matrix were obtained after the optimization:

$$A = \begin{bmatrix} 2070.42774, & -0.196143403, & 758.823206 \\ 0.00000, & 2052.62501, & 1362.28088 \\ 0.00000, & 0.00000, & 1.00000 \end{bmatrix}$$

$$K_c = \begin{bmatrix} 0.06869167, & -0.37364624 \end{bmatrix}$$

III. Discussion and Conclusion

After estimating A, R, t and k_c,the non-linear minimization is performed the Intrinsic camera matrix to obtained refined values. The cost function used is the euclidean distance between the corner point in the image, and the corner point obtained after reprojecting the world coordinate point, using the intrinsic camera matrix A, extrinsic camera matrix R, t and the distortion coefficients K_c. The optimized values have some error and they highly depend on initialization values. To evaluate the accuracy of the calibration matrices obtained we compute the *reprojection error*.The values for the reprojection error is **1.834** . Reprojection error is calculated using the L2 norm between corner points the we get from our transformation and the corner points that we get from cv2.findChessboardCorners. Then the mean is taken across all the calibration images.

References

[1] A flexible new technique for camera calibration-zhengyou zhang. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf.

IV. Results

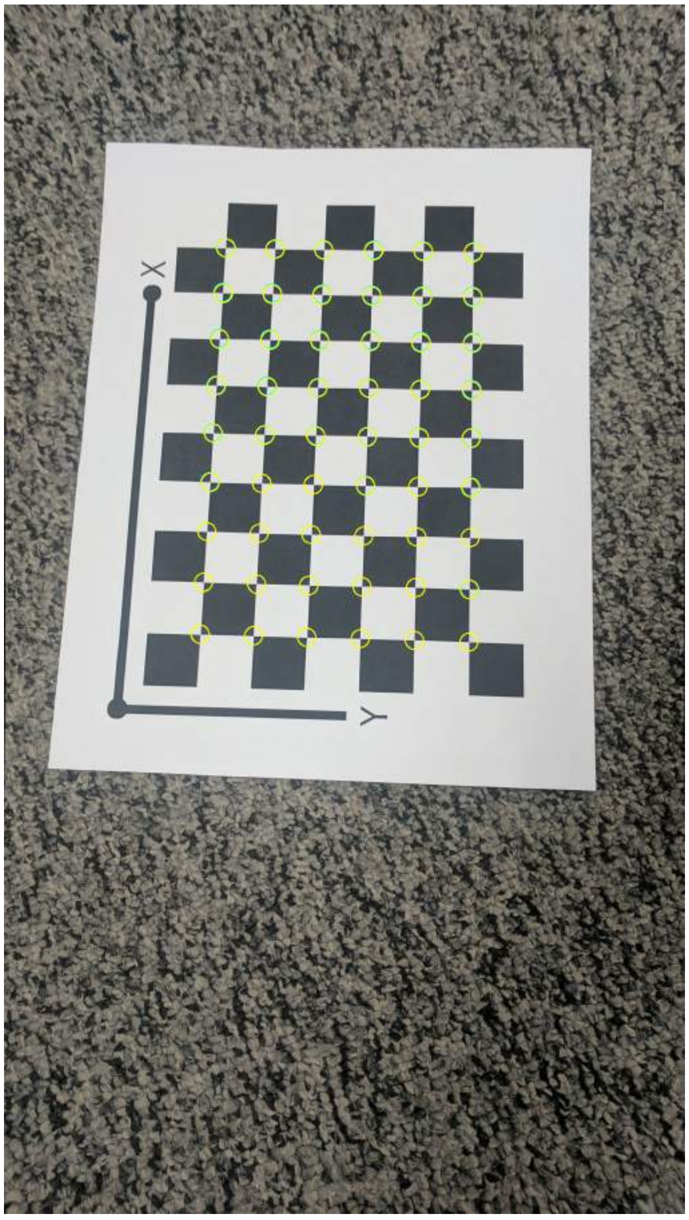

Figure 1: Rectified output of image 1

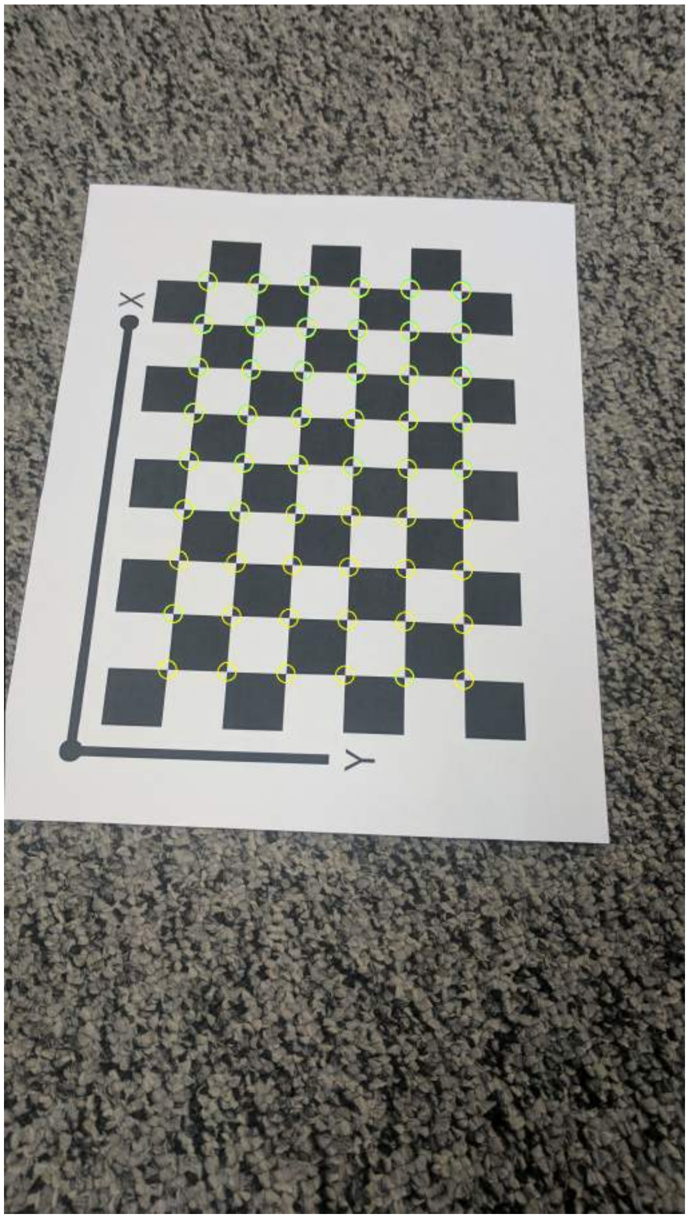

Figure 2: Rectified output of image 2

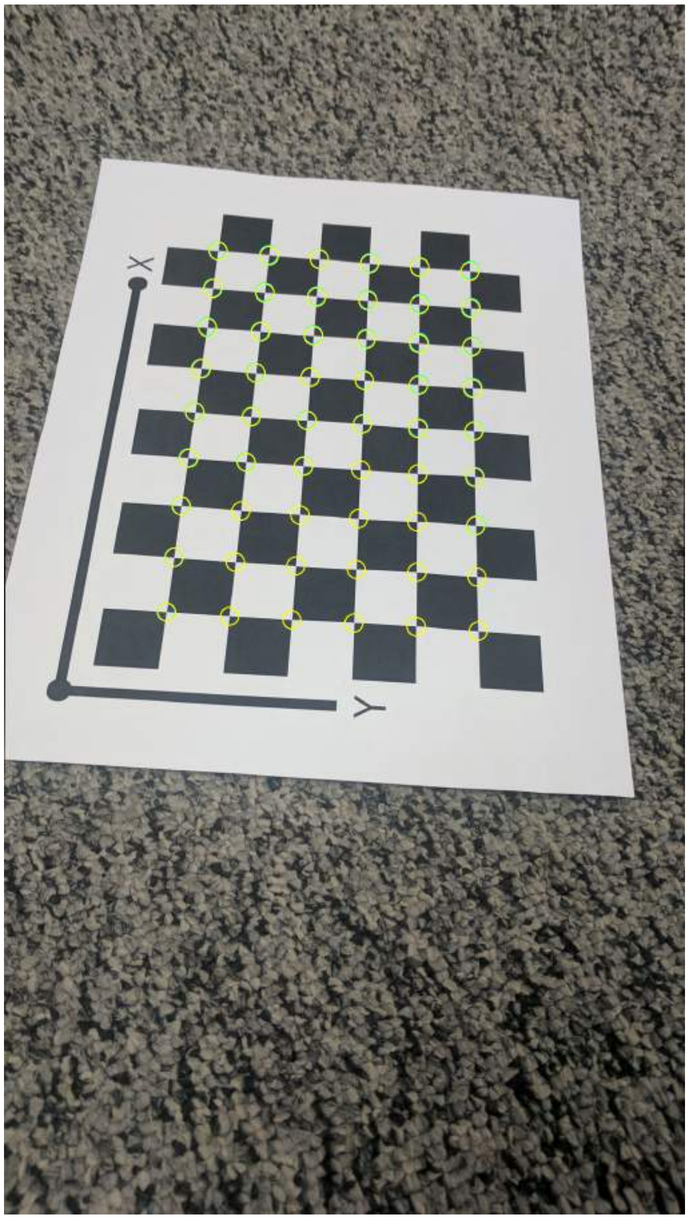

Figure 3: Rectified output of image 3

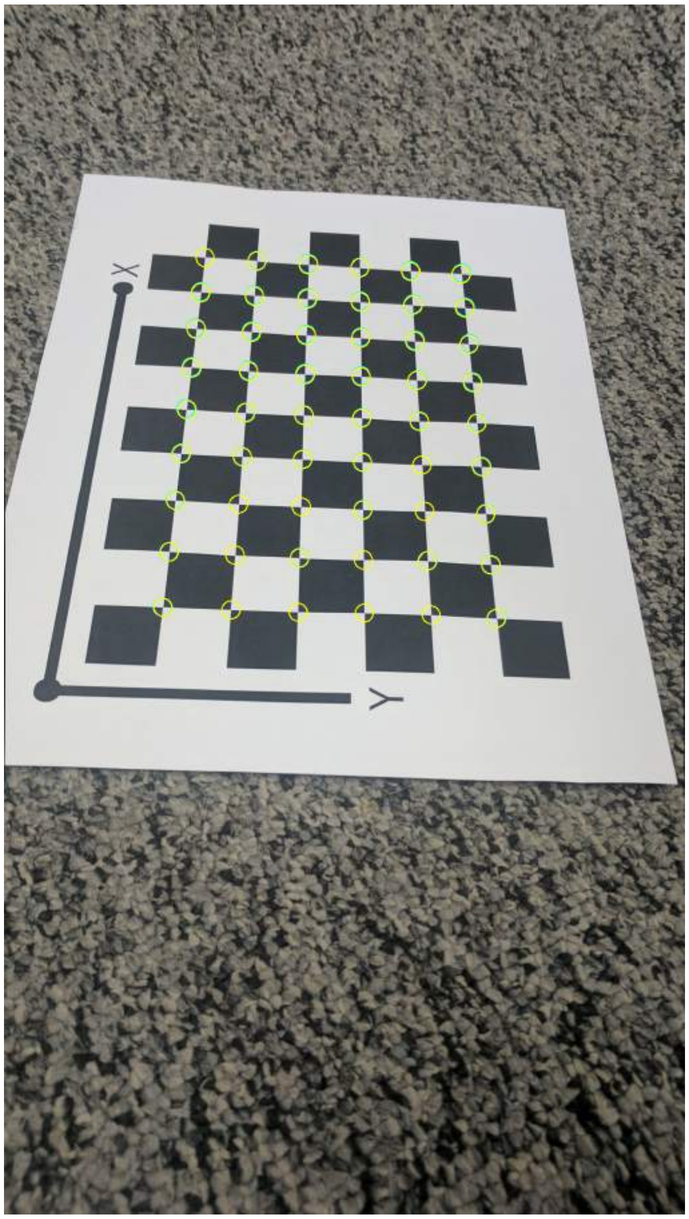

Figure 4: Rectified output of image 4

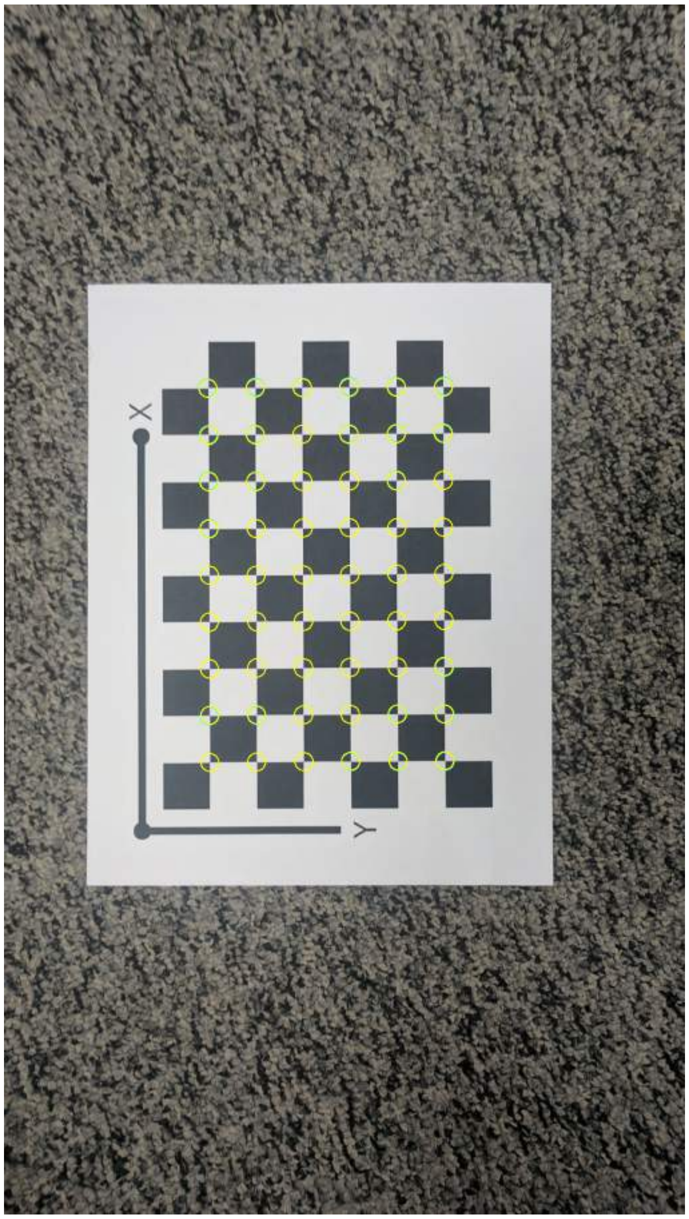

Figure 5: Rectified output of image 5

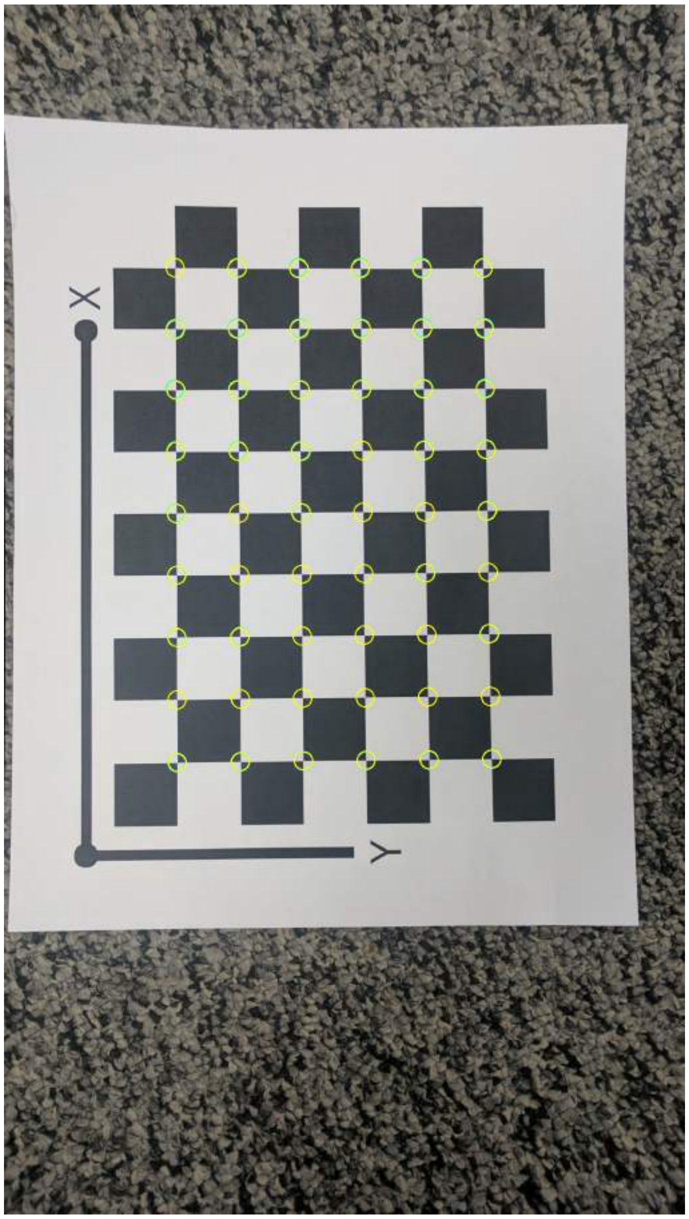

Figure 6: Rectified output of image 6

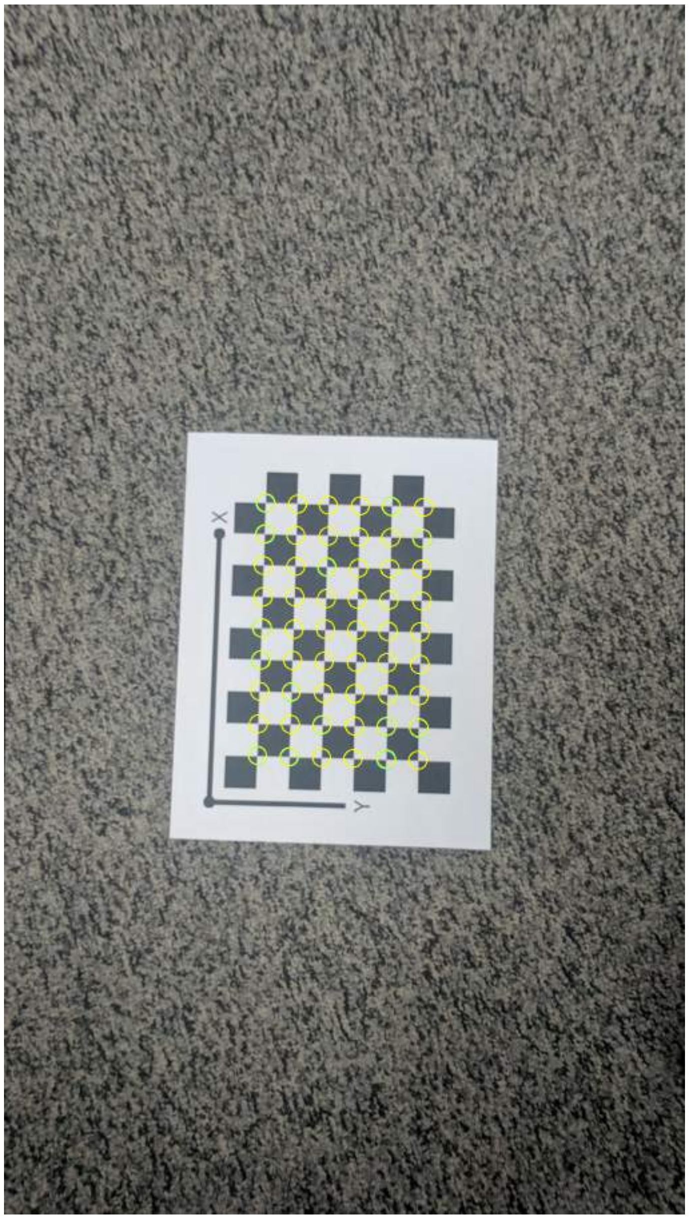

Figure 7: Rectified output of image 7

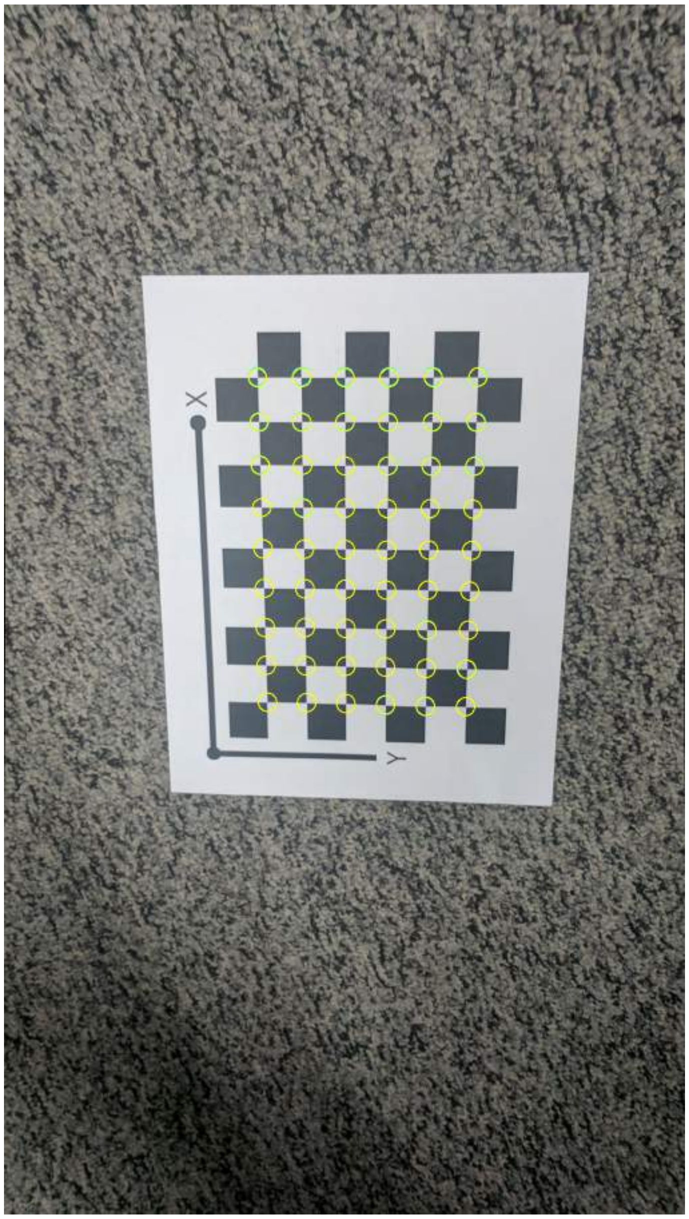

Figure 8: Rectified output of image 8

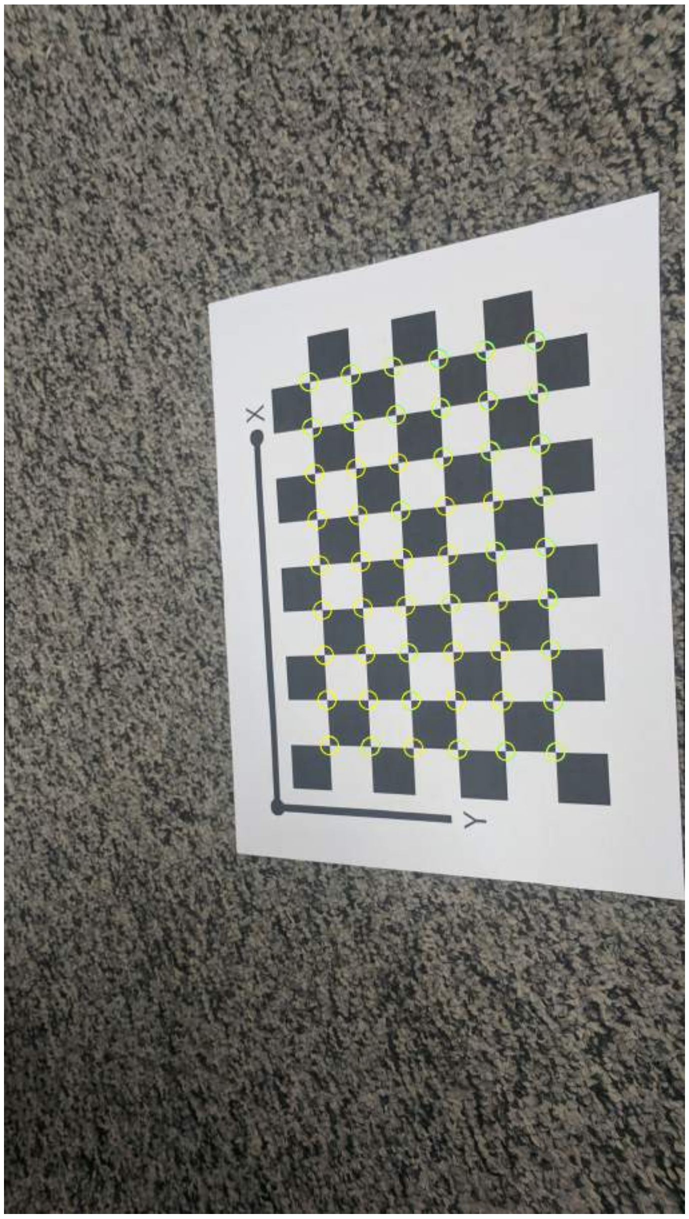

Figure 9: Rectified output of image 9

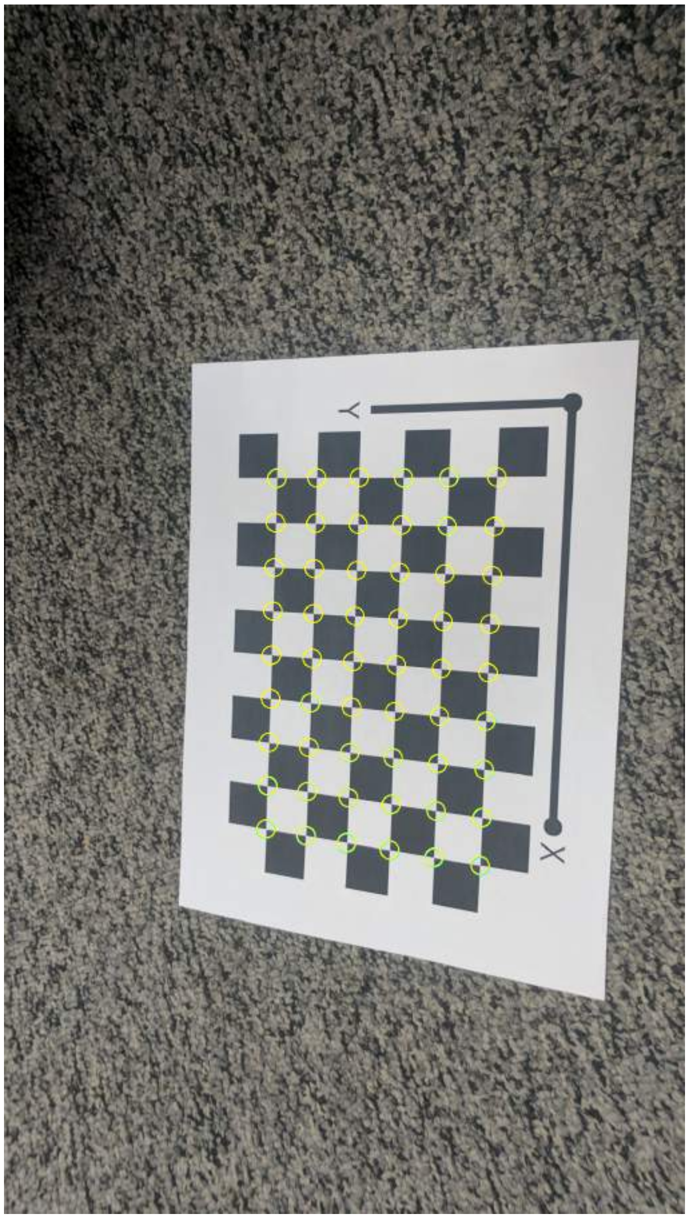

Figure 10: Rectified output of image 10

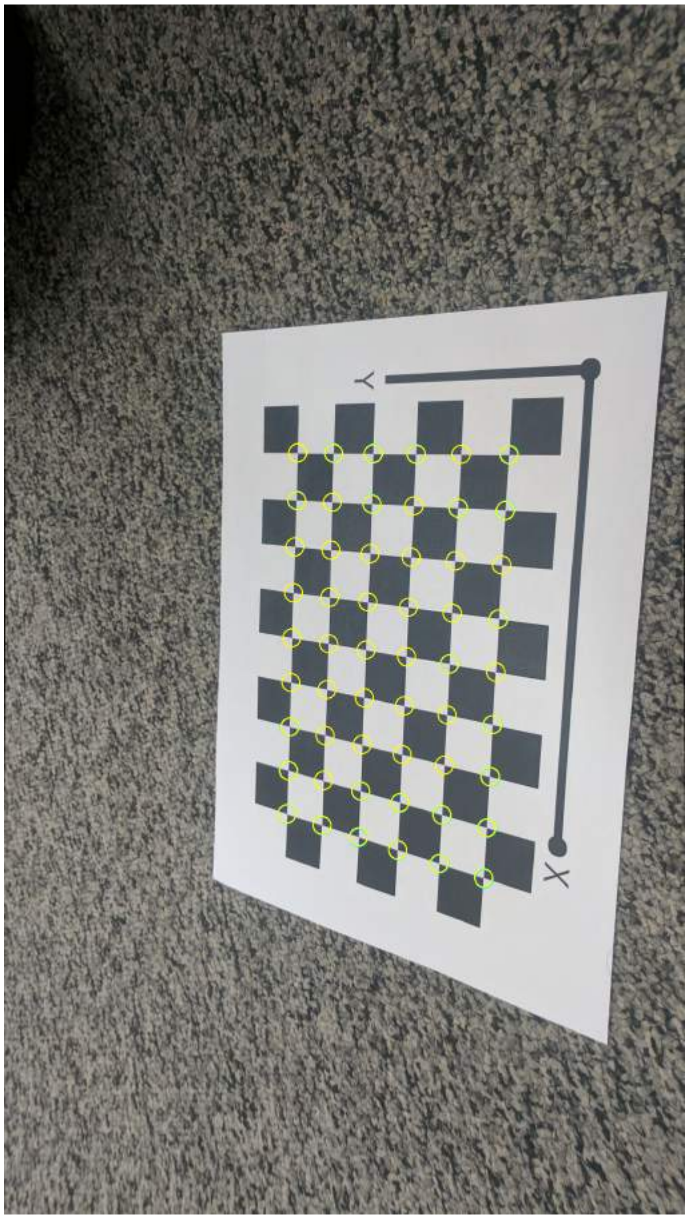

Figure 11: Rectified output of image 11

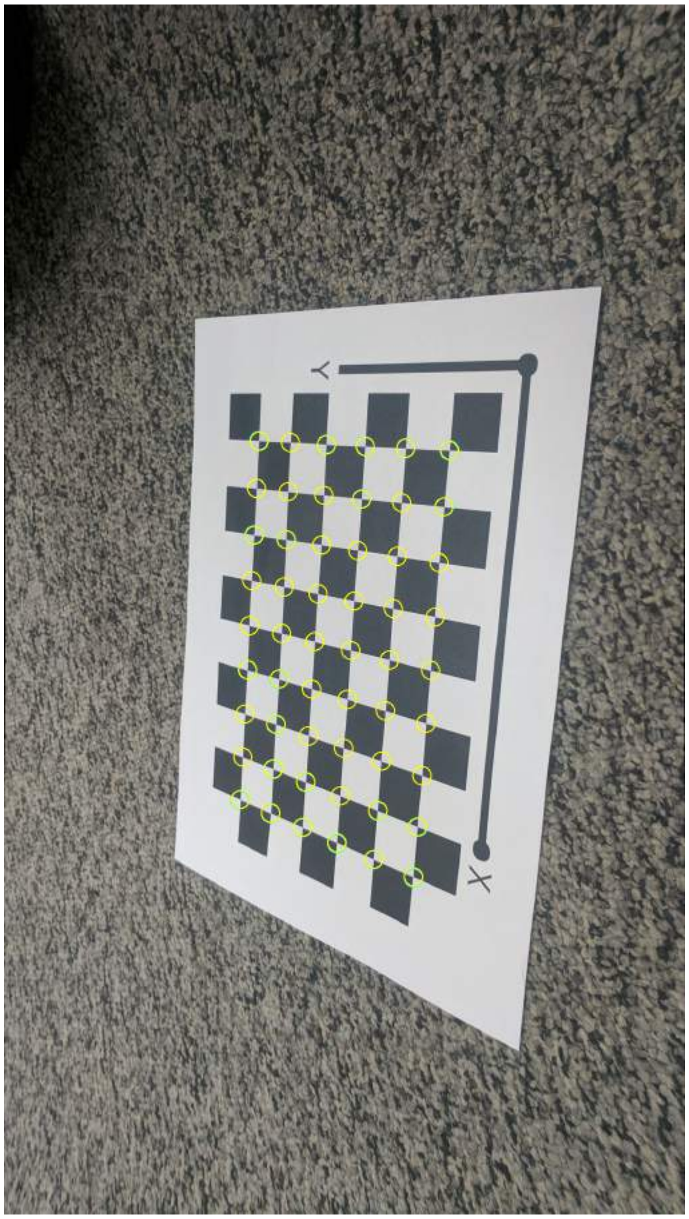

Figure 12: Rectified output of image 12

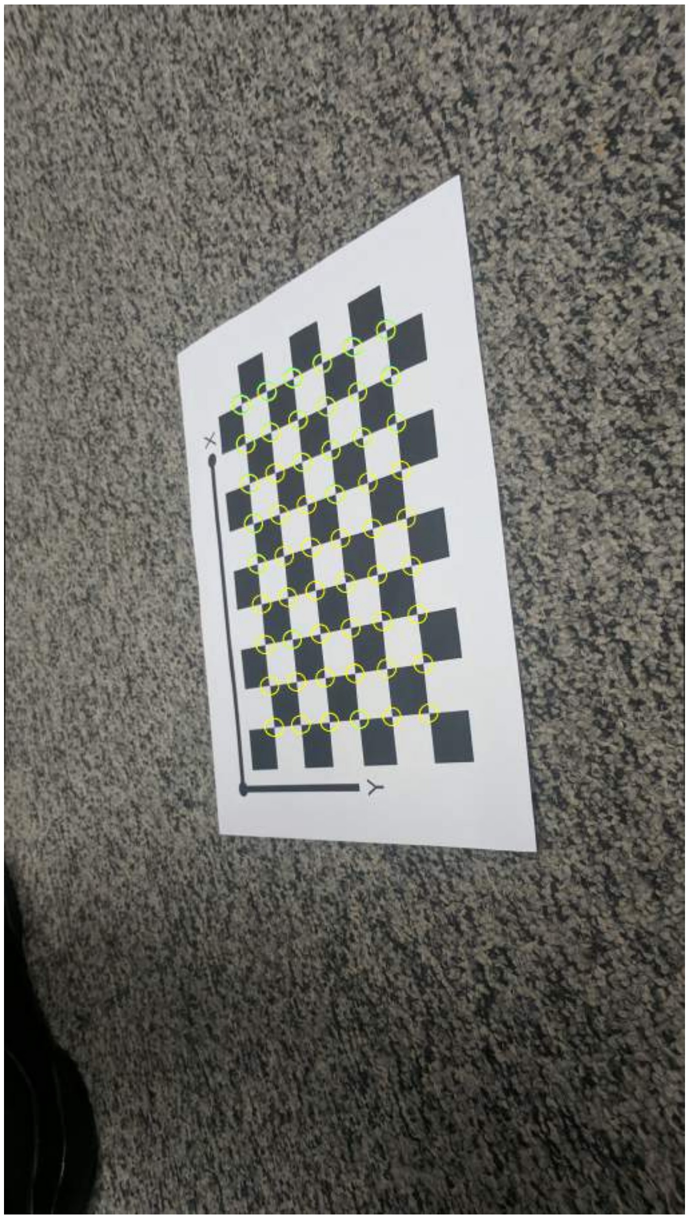

Figure 13: Rectified output of image 13