Homework 1: AutoCalib

(Using 1 Late Day.)

Rohitkrishna Nambiar (115507944)
University of Maryland
College Park, Maryland 20740
rohit517@umd.edu

***Abstract*—In this homework we would be performing a single camera calibration following the work presented by Zhengyou Zhang [1].**

I. Introduction

Estimating the camera parameters such as focal length, principle point, distortion coefficients is called camera calibration. It is one of the most time consuming process part of any computer vision research involving 3D geometry. A robust and automatic method of camera calibration is presented in [1]. The camera calibration matrix **K** is given by

$$K = \begin{bmatrix} f_x & 0 & cx \\ 0 & f_y & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

and the radial distortion parameters is given by k_1 and k_2.

II. Data

The calibration method given in[1] relies on a calibration target. We use a checkerboard to estimate the camera intrinsic parameters. We have even number of squares on the X-axis and odd number on the Y-axis. We use a 9×6 checkerboard with each side equal to 21.5mm as we only consider the inner squares. 13 such images have been taken with different views from a Google Pixel XL phone with focus lock. A few of the images can be seen in Fig.1.

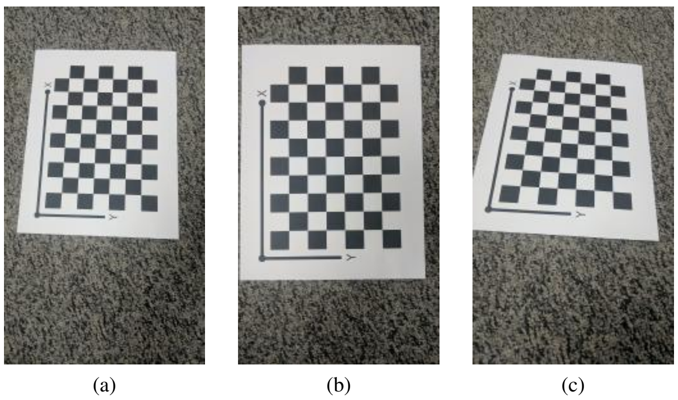
(a) (b) (c)

Fig. 1. Checkerboard Images

III. Initial Parameter Estimation

In this section we try to find out a good set of initial parameters that can then be fed into the non-linear optimizer. Before we proceed we define some of the common notations that will be used henceforth. **x** denoted the image points, **X** denotes the world points on the checkerboard, **k** denotes the radial distortion parameters, **K** the camera calibration matrix, **R** and **t** denote the rotation and translation of the camera wrt world coordinate frame.

Before we proceed further to estimate the parameters, we first establish a relation of Homography between the model plane and the image. We assume the world co-ordinates to be on the checkerboard and being a planar surface, the Z co-ordinate is taken to be 0. From the relationship of an image point and world point, using the above mentioned condition we get the following equations

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3}$$

The homography matrix **H** is given by a 3×3 matrix

$$\mathbf{H} = \mathbf{A} \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{4}$$

*A. Solving for approximate **K** or camera intrinsic matrix*

To find the corners of the checkerboard patter, we use the *cv2.findChessboardCorners* function in OpenCV. We input the pattern size which in our case is 9×6. We get the image co-ordinates of the corners of the checkerboard in sequence from top to bottom left to right. We similarly construct an array of world points given our knowledge of the dimension of each square.

To find out the homography matrix, we use DLT as we have knowledge of point correspondences. For every point, we obtain two rows given by 5.

$$\begin{bmatrix} X & Y & 1 & 0 & 0 & 0 & -xX & -xY & -x \\ 0 & 0 & 0 & X & Y & 1 & -yX & -yY & -y \end{bmatrix} \tag{5}$$

Similarly we stack 2n such equations and solve for the homography matrix. This is also given in section A of [1].

Once we obtain the homography matrix for each image, we follow the equations given in section 3.1 of [1] and obtain vector **b**. The extraction of the camera intrinsic parameters is done as given in section **B**. The initial estimate of the camera matrix K is given in Eq.11.

$$\mathbf{K} = \begin{bmatrix} 2054.26131 & -0.7333 & 761.6521 \\ 0 & 2037.3764 & 1350.7817 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

B. Estimate approximate *R* and *t* or camera extrinsics

Once we obtain the camera intrinsics, we obtain the extrinsics for each image using the following equations

$$\begin{aligned} r_1 &= \lambda A^{-1} h_1, \\ r_2 &= \lambda A^{-1} h_2, \\ r_3 &= r_1 \times r_2, \\ t &= \lambda A^{-1} h_3, \end{aligned} \tag{7}$$

where h_1, h_2 and h_3 are the columns of the homography matrix. Although a method to convert a normal matrix to a rotation matrix is given in Section C, it was found to be more erroneous and hence was neglected.

C. Approximate Distortion k

We assume the camera has minimal distortion with **k** given by

$$\mathbf{k} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \tag{8}$$

IV. NON-LINEAR GEOMETRIC ERROR MINIMIZATION

Once we have a good estimate of K, R, t and k we minimize the geometric error given by

$$\sum_{i=1}^{N} \sum_{j=1}^{M} ||x_{i,j} - \hat{x}_{i,j}(K, R_i, t_i, X_j, k)|| \tag{9}$$

where $\hat{x}_{i,j}(K, R_i, t_i, X_j, k)$ is the reprojection of the 3D point on the image plane with computed R, t, K and k. Scipy least squares optimizer is used to solve the non-linear least squares problem.

We also compute the reprojection error after the optimization which was 0.96922. The final K matrix obtained after optimization is

$$\mathbf{K} = \begin{bmatrix} 2049.9616 & -1.2543 & 761.6437 \\ 0 & 2032.3582 & 1350.6240 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

The distortion parameters is obtained as

$$\mathbf{k} = \begin{bmatrix} 0.063711 & -0.360335 \end{bmatrix} \tag{11}$$

The images after reprojection compared with the detected corners are shown in following figures. The yellow box is the origin, green circles are the detected corners and the red cross are the reprojected points.

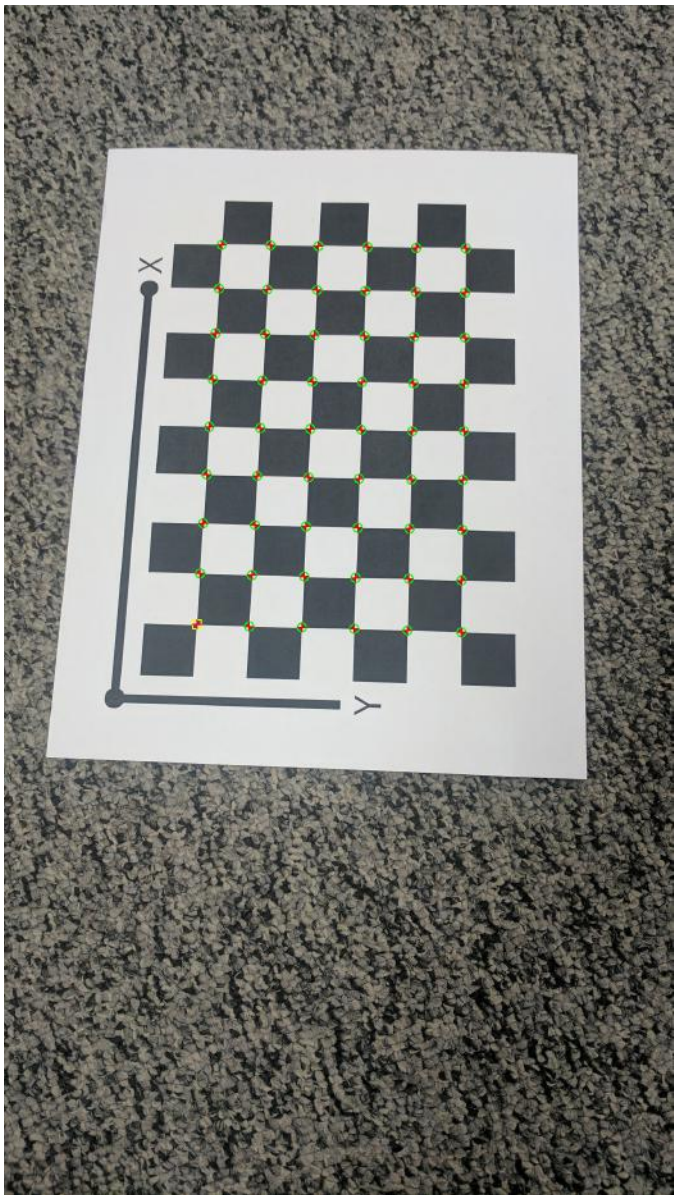

Fig. 2. Image 1 reprojection

Acknowledgment

I would like to thank Prof. Yiannis and the TA's Nitin and Chahat for the homework.

References

[1] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.

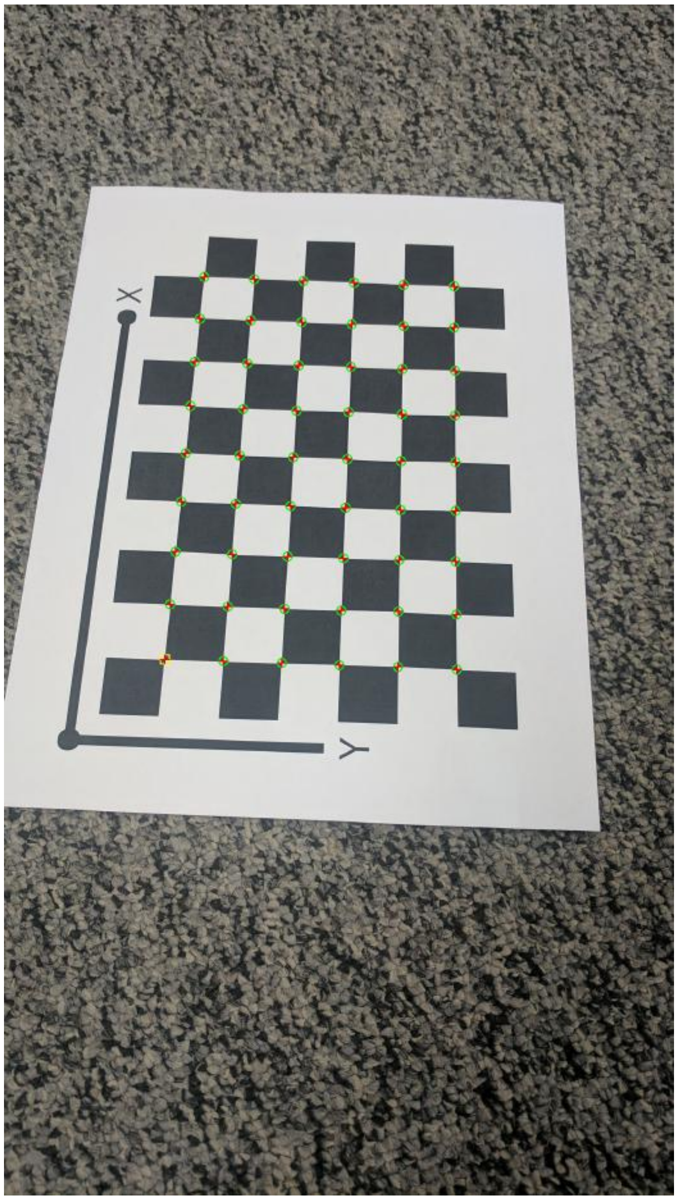

Fig. 3. Image 2 reprojection

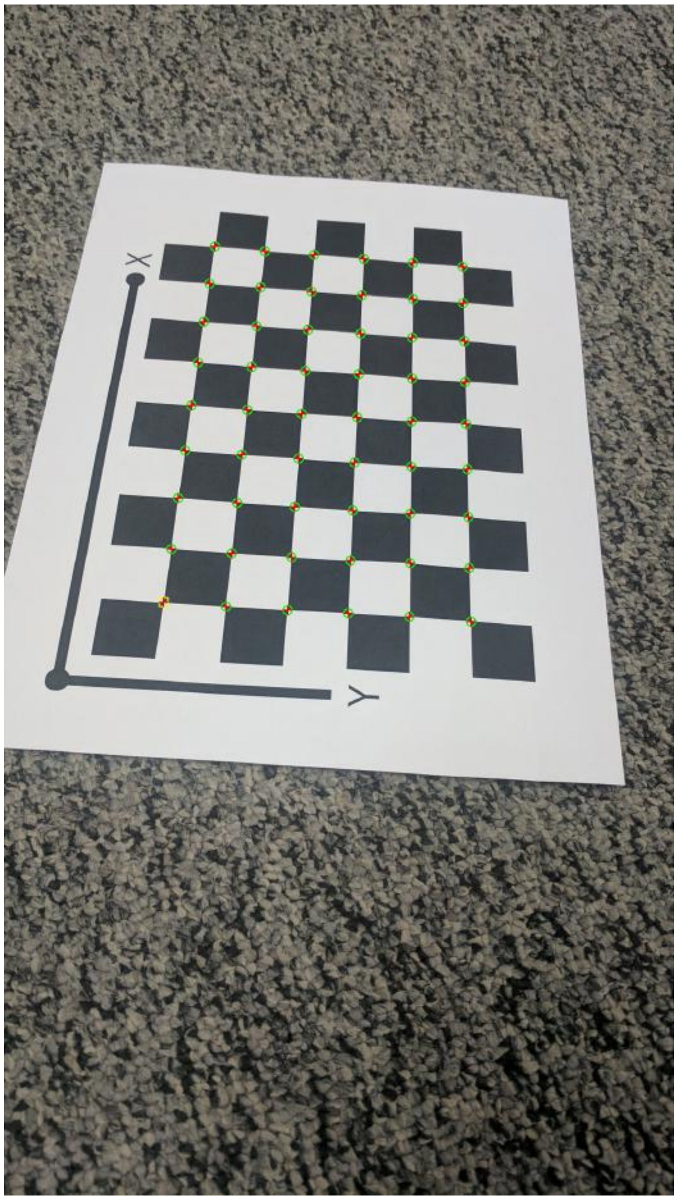

Fig. 4. Image 3 reprojection

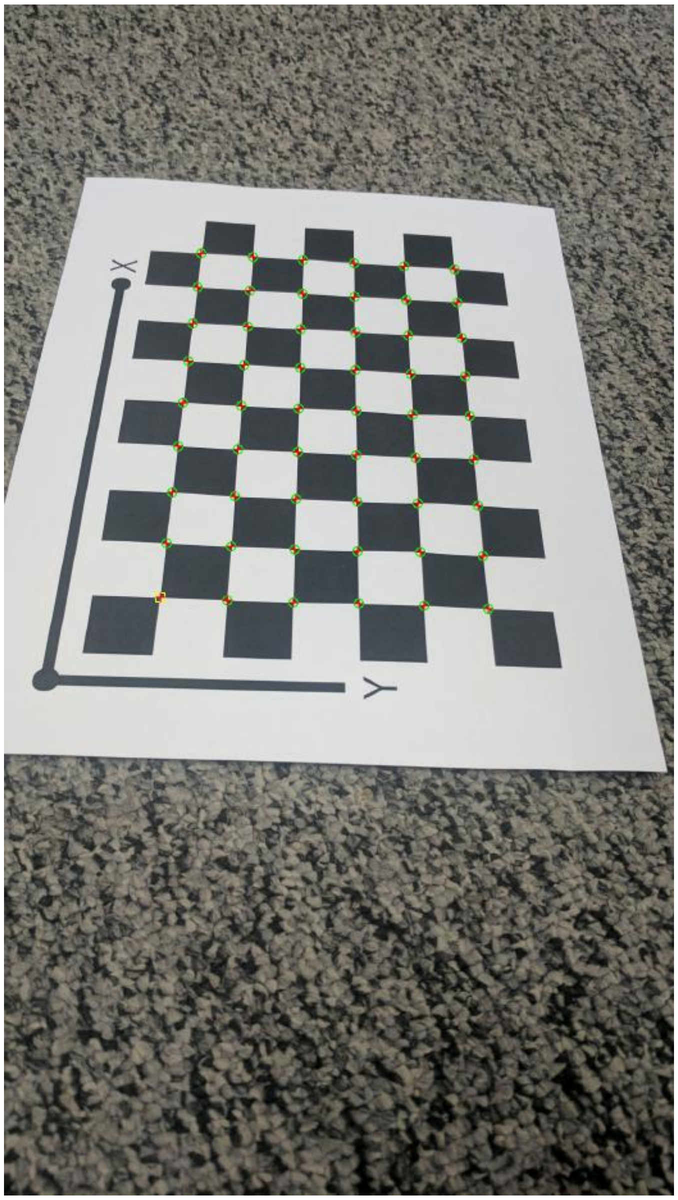

Fig. 5. Image 4 reprojection

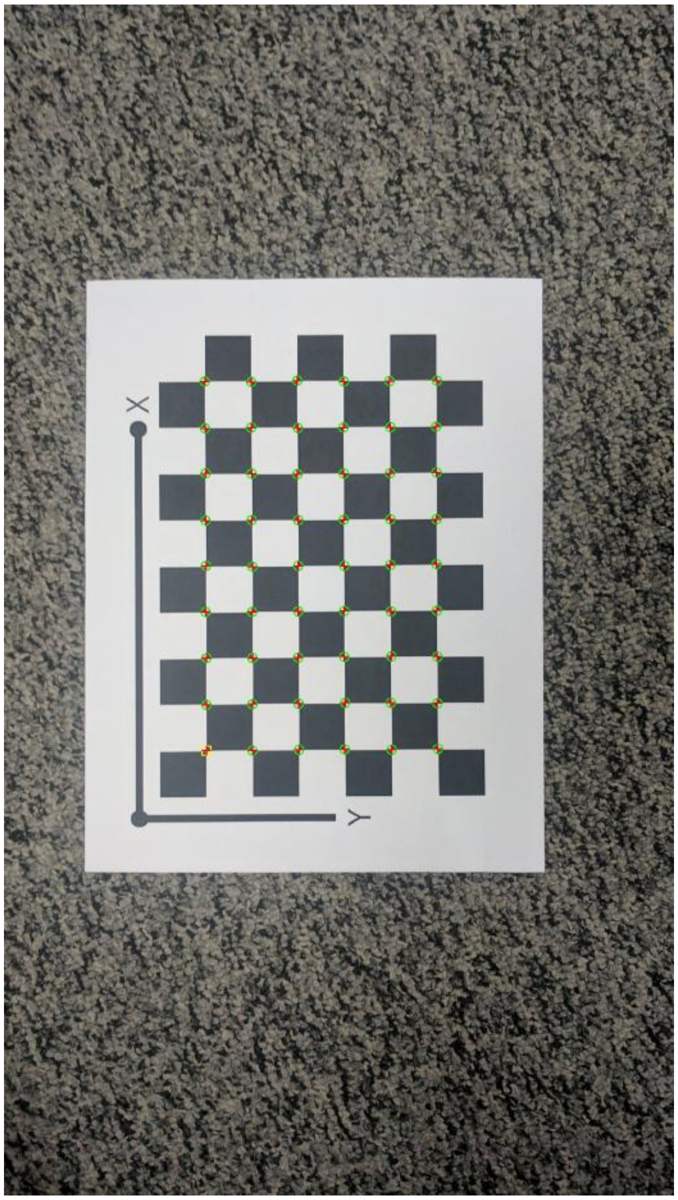

Fig. 6. Image 5 reprojection

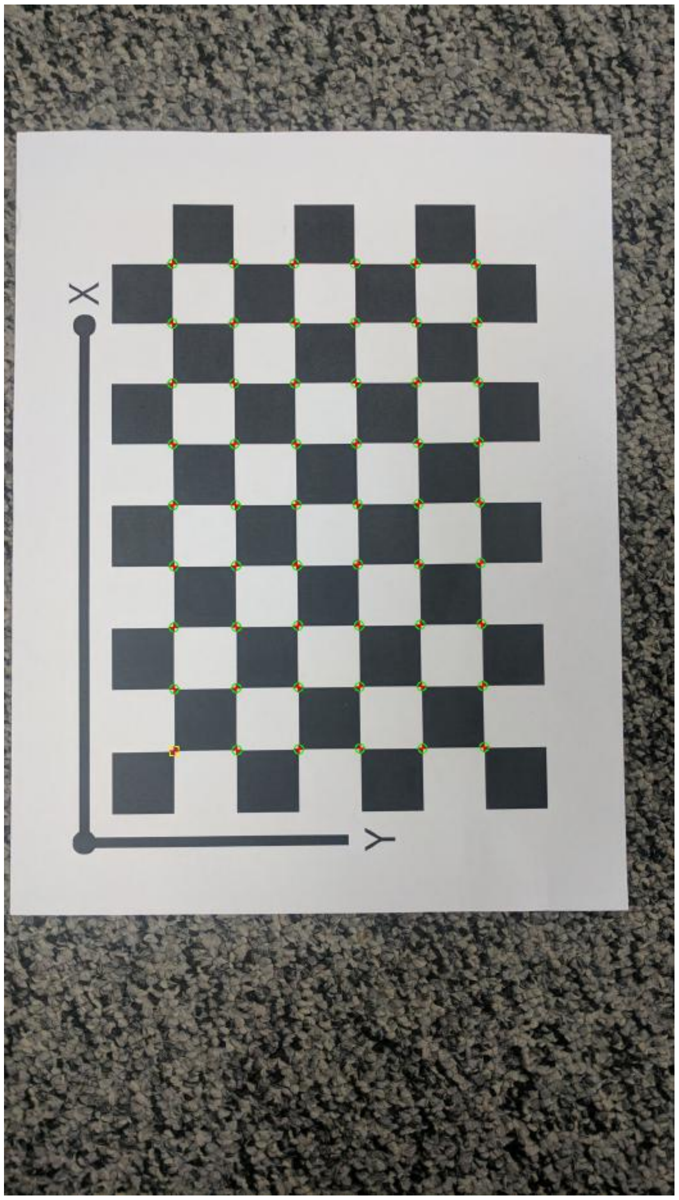

Fig. 7. Image 6 reprojection

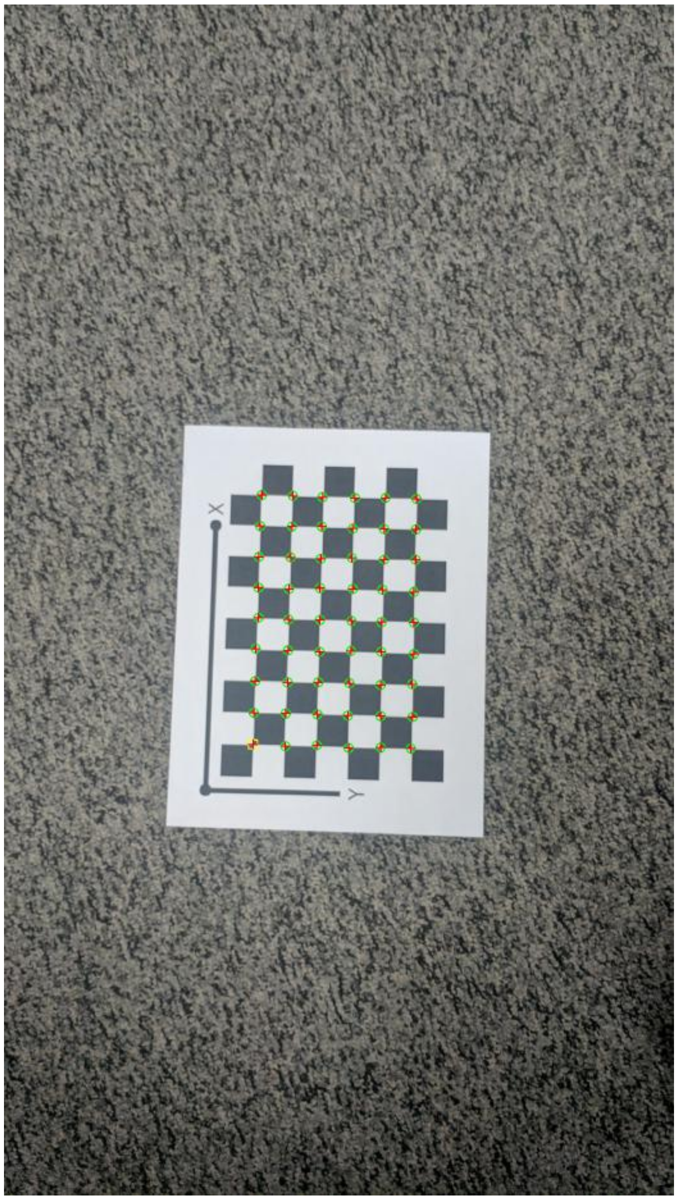

Fig. 8. Image 7 reprojection

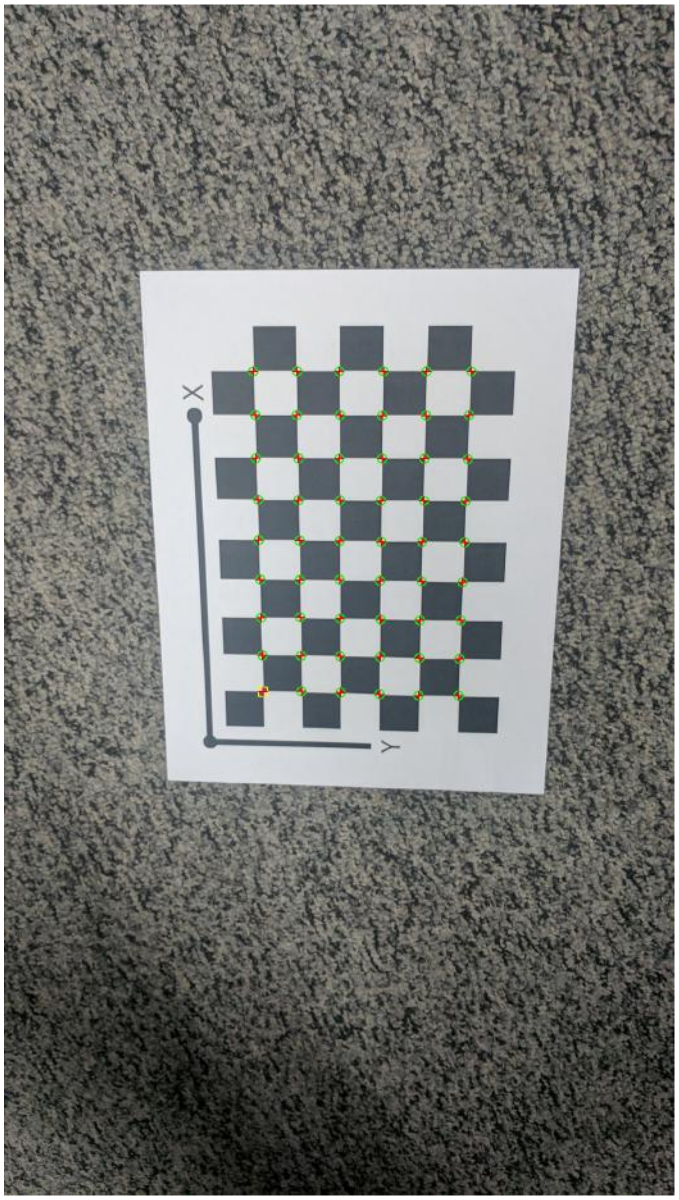

Fig. 9. Image 8 reprojection

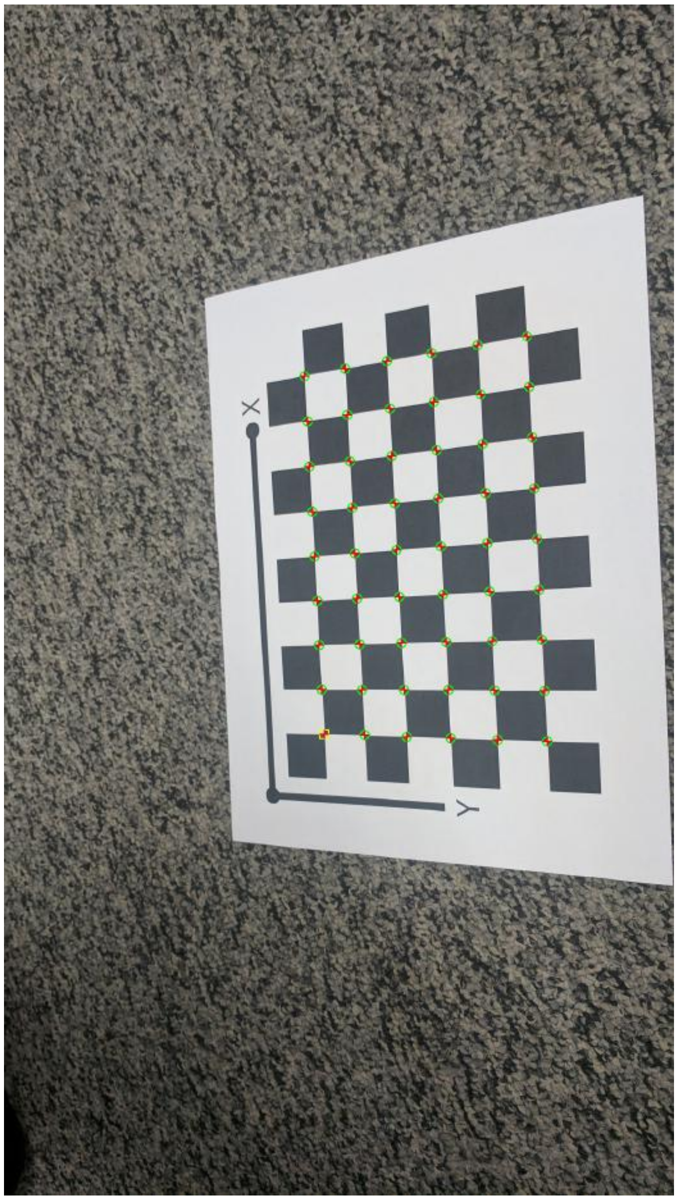

Fig. 10. Image 9 reprojection

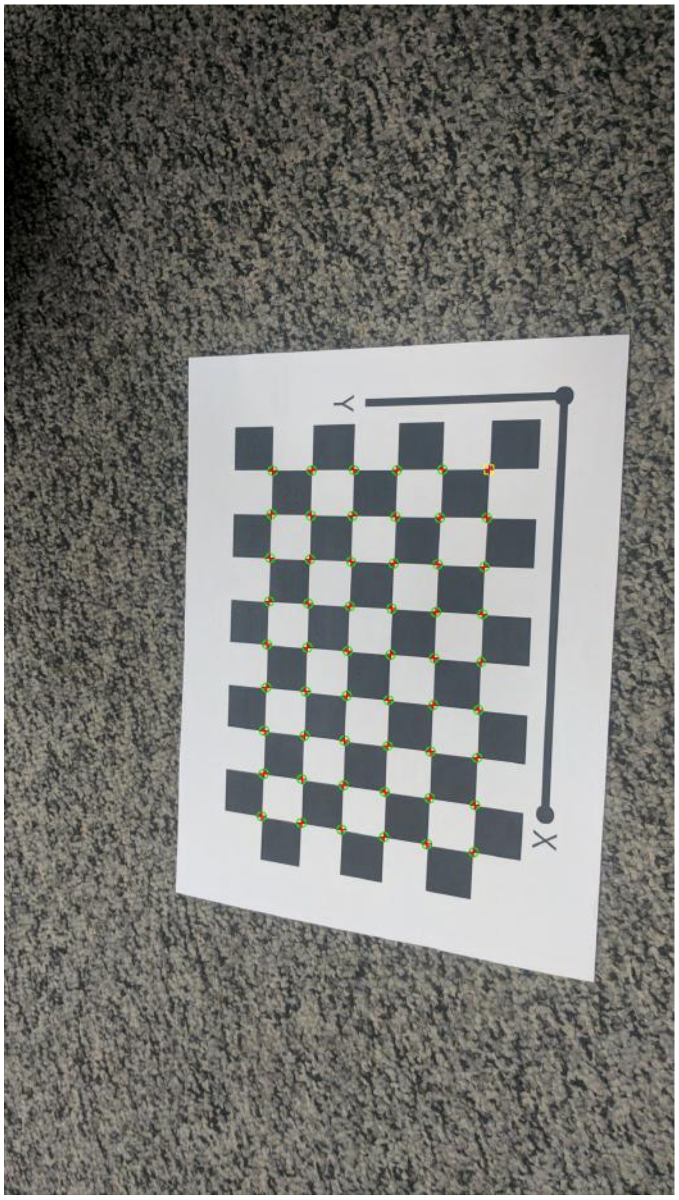

Fig. 11. Image 10 reprojection

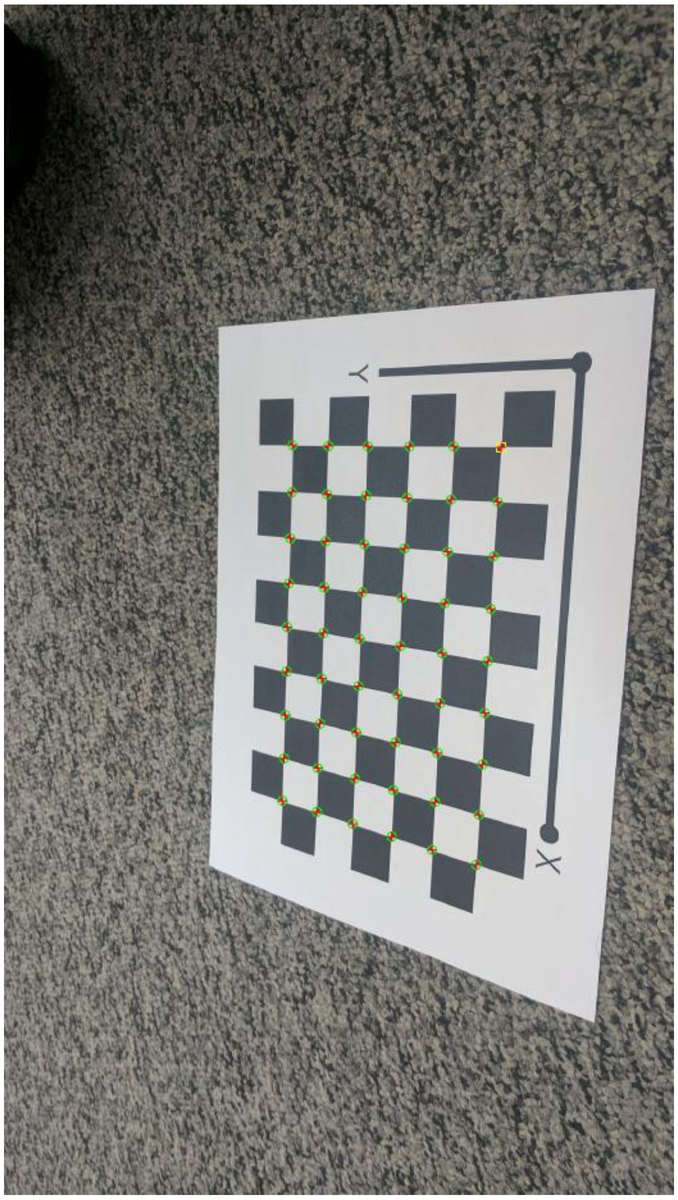

Fig. 12. Image 11 reprojection

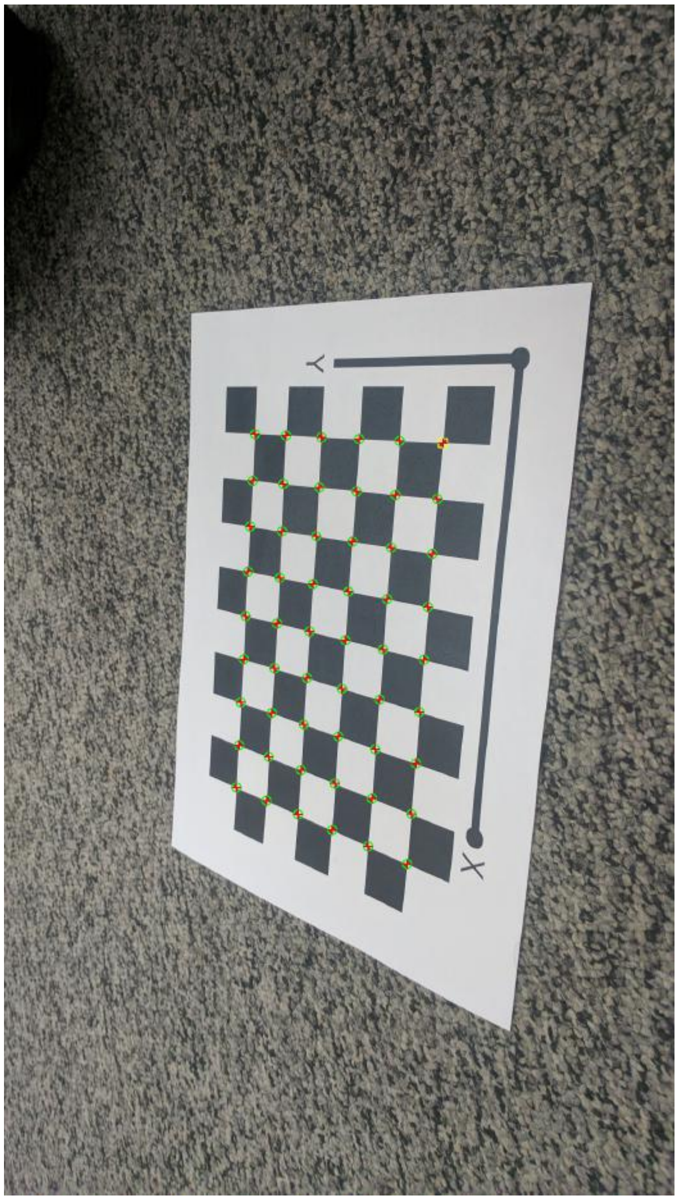

Fig. 13. Image 12 reprojection

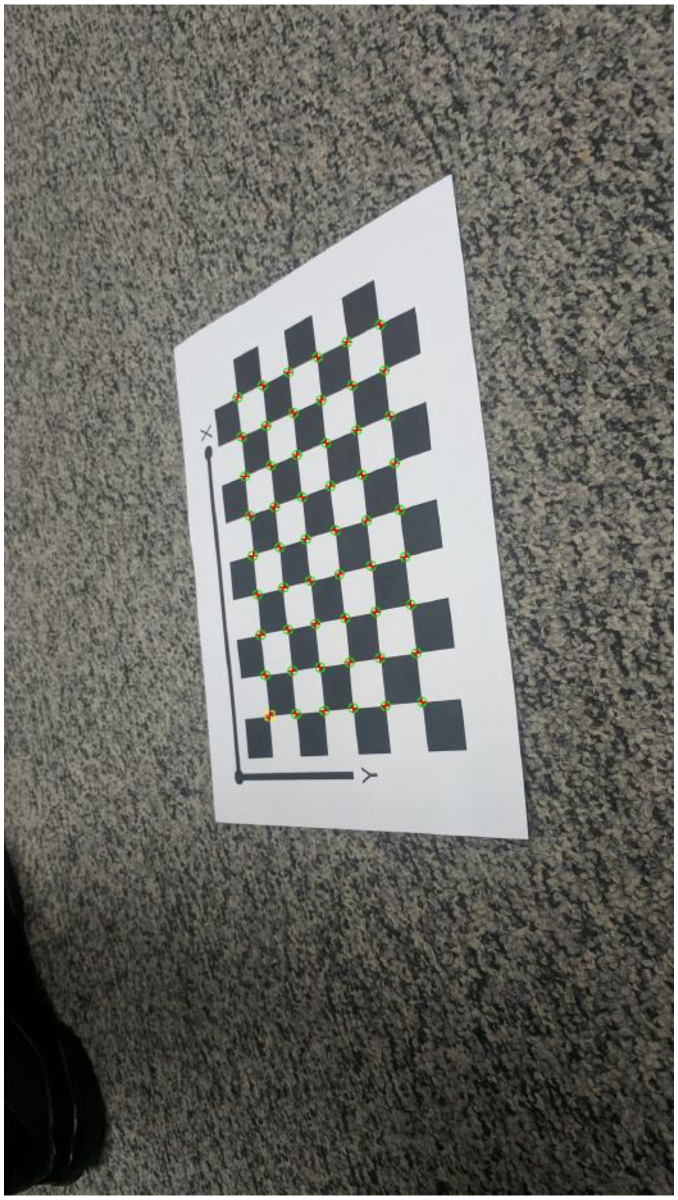

Fig. 14. Image 13 reprojection