CMSC733: Homework-1: AutoCalib

Saket Seshadri Gudimetla Hanumath
UID : 116332293
Email: saketsgh@terpmail.umd.edu

using 1 late day

***Abstract*—Accurate knowledge of the image projection parameters is an essential prerequisite for any kind of quantitative geometric measurement in computer vision. Auto-Calibration is a process that is used to obtain the said parameters(comprising of intrinsic camera matrix, lens distortion and position/orientation in space or the extrinsic parameters).**
In this homework the work of Zhengyou Zhang on Auto Calibration[1] has been implemented using OpenCV in python language.

I. Introduction

Estimating parameters of the camera like the focal length, distortion coefficients and principle point is called Camera Calibration. It is one of the most time consuming and important part of any computer vision research involving 3D geometry. An automatic way to perform efficient and robust camera calibration was presented by Zhengyou Zhang of Microsoft in this paper[1] and is regarded as one of the hallmark papers in camera calibration.

II. Data

The model under consideration is that of a Chess Board which can be seen in Figure 1. Thirteen images of the model taken from different pose have been used in this homework for the purpose of calibration. The Chess Board pattern is extremely helpful since the dimensions of each square is known and thus we can find perfect corners.

III. Procedure

The entire pipeline of this implementation can be summarised in the following steps -

- Images I_0, ...,I_{n-1} of the model are take under different views by either moving the model or the camera (or both).
- From each image m sensor points are extracted(observed), assumed to be in 1:1 correspondence with the points on the model plane.
- From the observed points, the associated homographies H_0, ..., H_{n-1} are estimated for each of the n views.
- From the view homographies H_i, the five intrinsic parameters $(\alpha, \beta, \gamma, u_c, v_c)$ of the camera are estimated using a closed-form (linear) solution. The lens distortion parameters are set to 0 initially.
- The extrinsic 3D parameters R_i, t_i are then calculated using the homographies and the intrinsic camera matrix computed earlier.

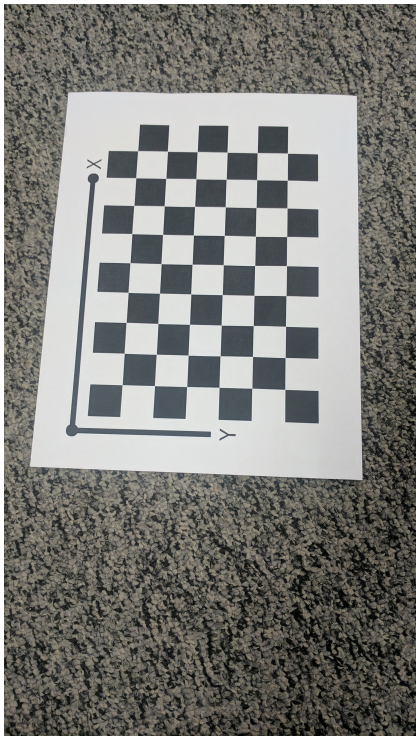

Fig. 1. Chess Board is used as Calibration target

- Using the initial estimates, all the parameter values are refined by non-linear optimization over all the views.

A. Acquiring Image and World Coordinates

The main purpose of this project is to ensure a seamless mapping from real world coordinates to image coordinates. And thus we want to find the most optimal values of the intrinsic camera matrix(K) and the extrinsic matrix(R and t). The relationship between the image and world coordinates is given by the transformation shown in the equation below. Here (u, v) are the image coordinates, s is a scaling parameter and (X, Y) are the world coordinates. R_i, t_i are the rotation and translation values of image i.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} K(R_i|t_i) \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{1}$$

where, K is the intrinsic camera matrix given by the equation -

$$K = \begin{bmatrix} \alpha & \gamma & u_c \\ 0 & \beta & v_c \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Here, without loss of generality the model plane is assumed to be on the XY plane.
The image coordinates are obtained using $cv2.findChessboardCorners()$ whereas the world coordinates can be defined with respect to the origin present on the chess board. The square dimensions of 21.5mm has been chosen as a unit of scaling.
The output of chess board corners can be seen in Figure 2.

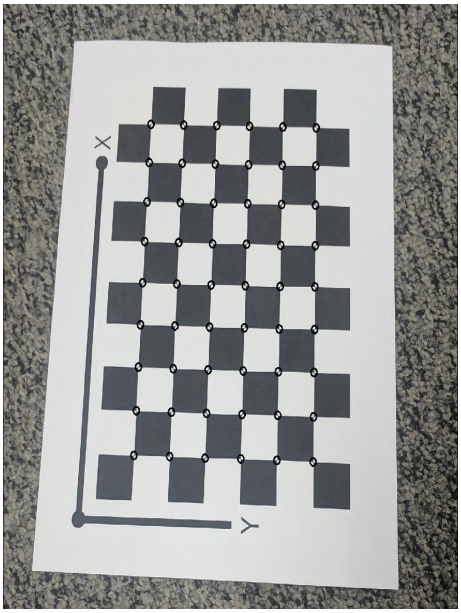

Fig. 2. Chess Board Corners (output image cropped for clarity)

B. Estimating Homographies

Homography between image coordinates(obtained from $cv2.findChessboardCorners()$) and their respective world coordinates is obtained through $cv2.getPerspectiveTransform()$ which requires four points for estimation. Four points chosen in the image must match with their respective world coordinates. The four chosen points can be seen as yellow colored points in Figure 3. The corresponding world are coordinates defined as $21.5 * [[1,1],[6,1],[1,6],[6,6]]$

C. Intrinsic Matrix Estimation

The initial parameter estimation involves solving a closed form solution defined by the equation(refer to section 3.1 in [1] for more details about the equations).

$$Vb = 0 \tag{3}$$

The solution is obtained by solving SVD for equation(3). The parameters of K can be found using b matrix.
The radial lens distortion is set to $k_c = [0,0]^T$.

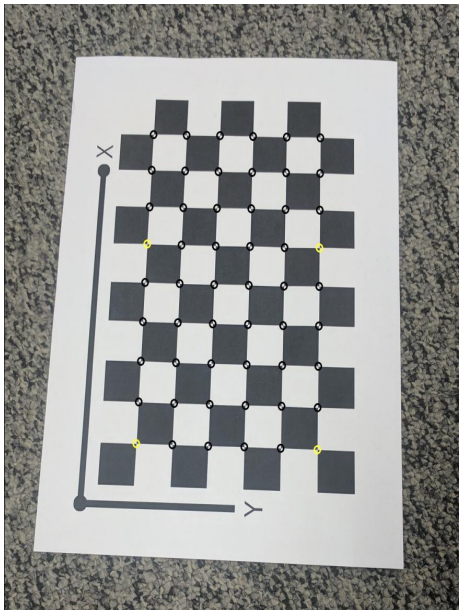

Fig. 3. Four points chosen for homography (output image cropped for clarity)

D. Extrinsic Parameter Estimation

The rotation and translation parameters can be estimated using the equations below -

$$r_1 = \lambda A^{-1} h_1 \tag{4}$$

$$r_2 = \lambda A^{-1} h_2 \tag{5}$$

$$r_3 = r_1 \times r_2 \tag{6}$$

$$t = \lambda A^{-1} h_3 \tag{7}$$

where A is the same as K matrix or the instrinsic camera matrix. h1, h2 and h3 are the column vectors of the homography matrix H

E. Non-Linear Optimisation

The optimisation of initial estimates are optimised using the function $scipy.optimise.least_squares()$. It is an implementation of the non-linear algorithm called **Levenberg-Marquardt Algorithm**. The algorithm tries to minimize the following cost -

$$\sum_{i=1}^{N} \sum_{j=1}^{M} ||x_{i,j} - \hat{x}_{i,j}\,(K, R_i, t_i, X_j, k_c)\,|| \tag{8}$$

here $x_{i,j}$ are the observed image coordinates(obtained from $cv2.findChessboardCorners()$) and $\hat{x}_{i,j}$ are the projected image coordinates.

F. Image Rectification and Re-Projection of Points

The images are rectified using the optimal estimates of the intrinsic parameters and the lens distortion coefficients and are given to $cv2.undistort()$ function to rectify the images. These images are then read and homographies are calculated as before. Extrinsic parameters are obtained as before but using

the final K matrix. The world points are then re-projected and re-projection error is calculated. The rectified image and the re-projected points can be seen in Figure 4, 5 and 6 respectively. In the figure 5 and 6 the black circles are obtained from $cv2.findChessboardCorners()$ and red circles are the projected points. As these overlap pretty well(indicating very minute projection error) the black circles are rarely visible.

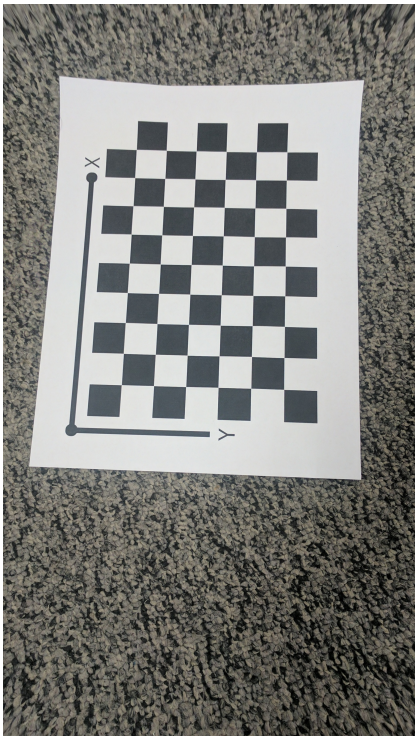

Fig. 4. rectified image 0

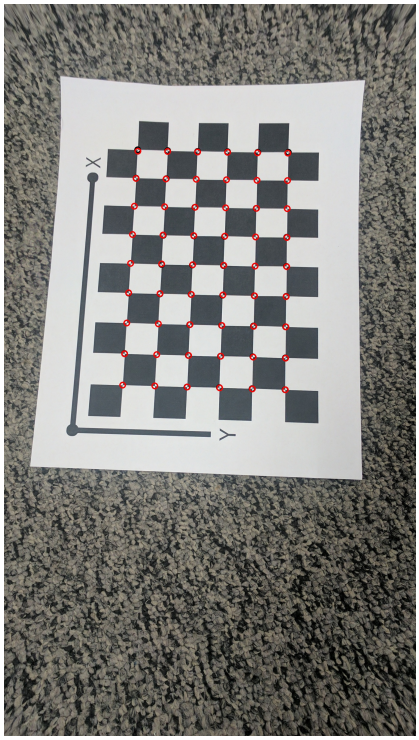

Fig. 5. re-projected points in image 1

IV. Results

A. Initial intrinsic camera matrix K

$$K = \begin{bmatrix} 2.03730563e+03 & 4.84486257e+00 & 7.57382370e+02 \\ 0.00000000e+00 & 2.02388697e+03 & 1.37763686e+03 \\ 0.00000000e+00 & 0.00000000e+00 & 1 \end{bmatrix} \tag{9}$$

B. Final intrinsic camera matrix K

$$K = \begin{bmatrix} 2.03371233e+03 & -2.21824766e-01 & 7.57693564e+02 \\ 0.00000000e+00 & 2.02204892e+03 & 1.37725480e+03 \\ 0.00000000e+00 & 0.00000000e+00 & 1 \end{bmatrix} \tag{10}$$

C. Final lens distortion matrix k_c

$$k_c = \begin{bmatrix} 0.08771432 \\ -0.6170109 \end{bmatrix} \tag{11}$$

D. Re-projection error

The mean re-projection error obtained was **2.011877**.

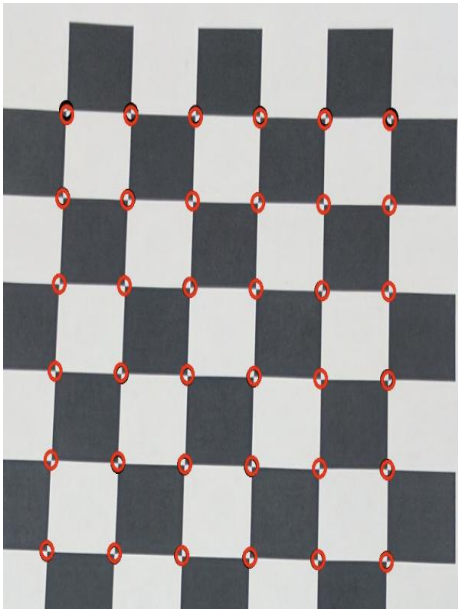

Fig. 6. re-projected points(cropped fig 5 for clarity)

V. APPENDIX

The re-projected points on the rectified images can be seen in the following images from Figure 7-18.

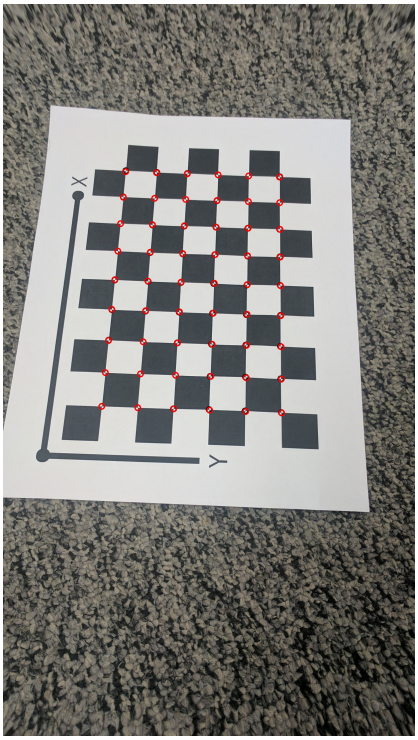

Fig. 7. re-projected points in image 2

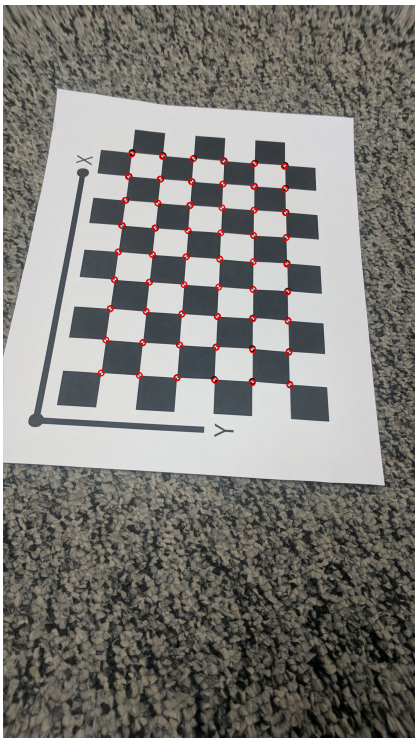

Fig. 8. re-projected points in image 3

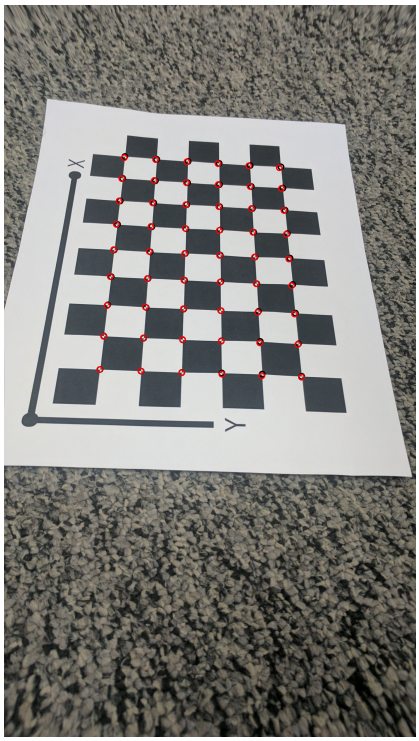

Fig. 9. re-projected points in image 4

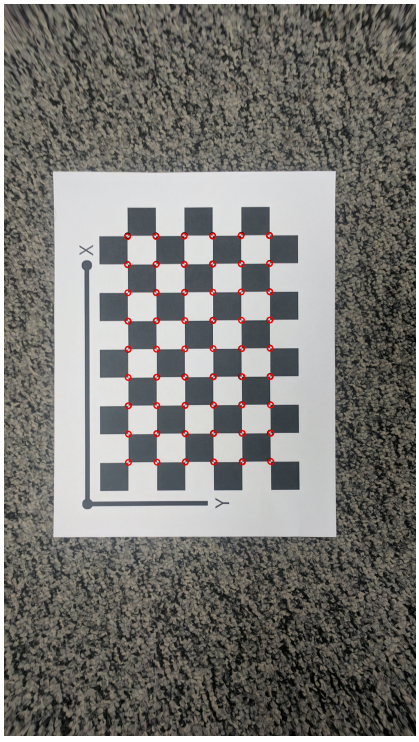

Fig. 10. re-projected points in image 5

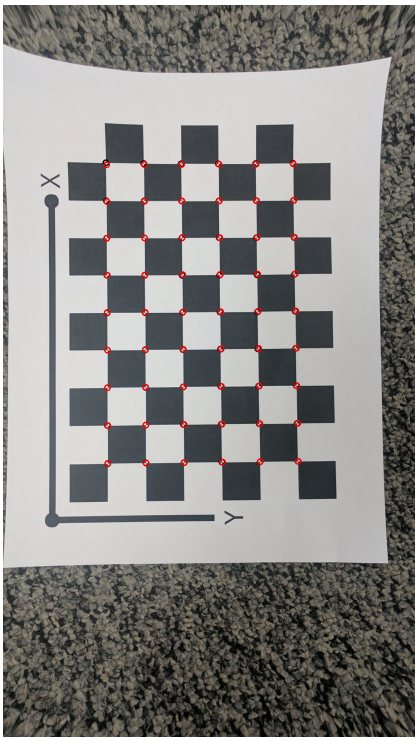

Fig. 11. re-projected points in image 6

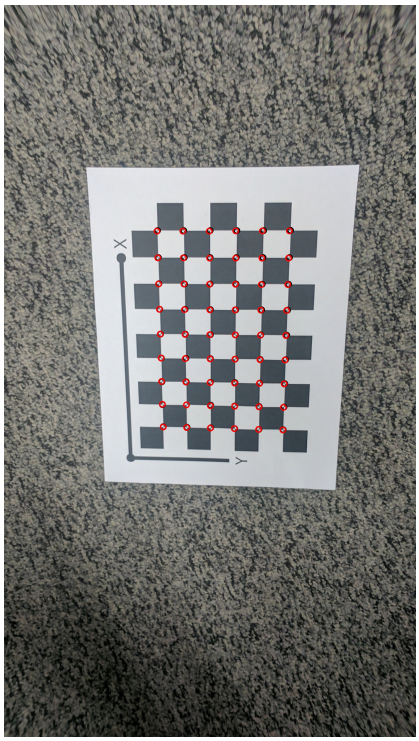

Fig. 13. re-projected points in image 8

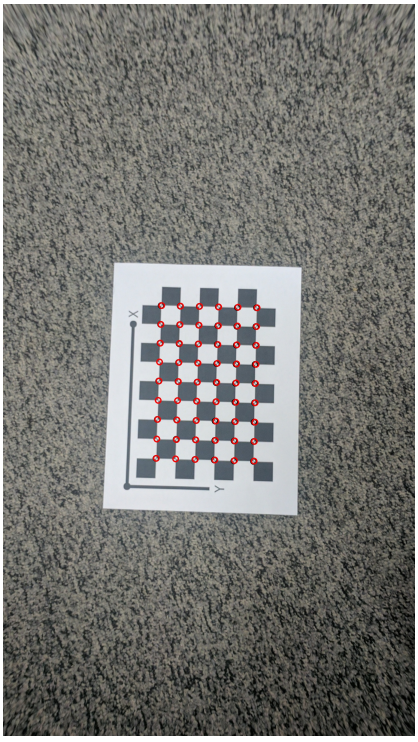

Fig. 12. re-projected points in image 7

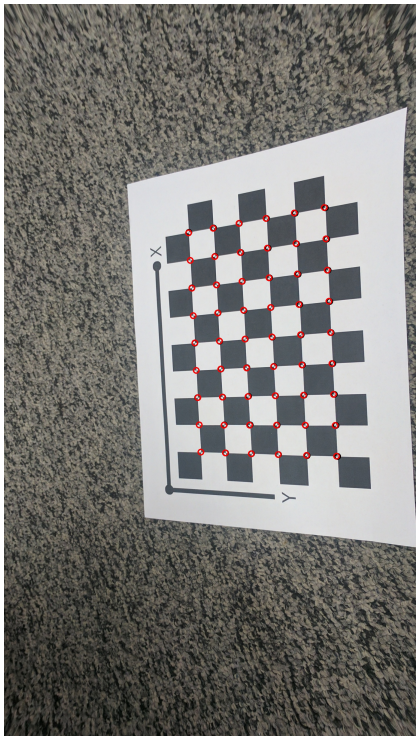

Fig. 14. re-projected points in image 9

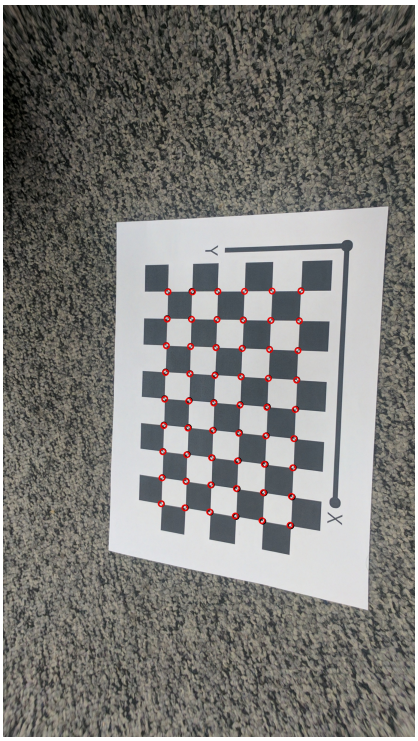

Fig. 15. re-projected points in image 10

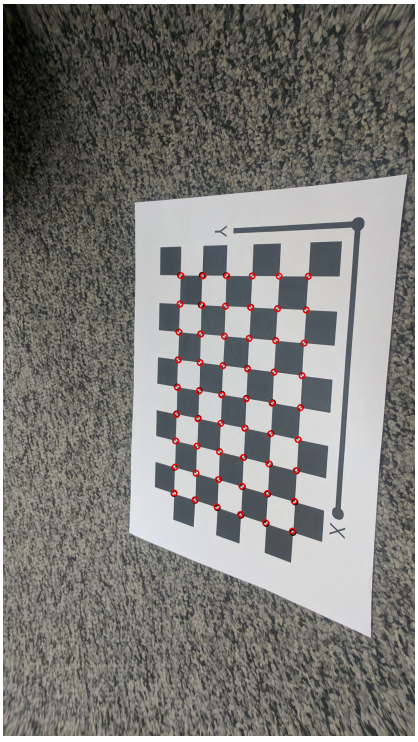

Fig. 16. re-projected points in image 11

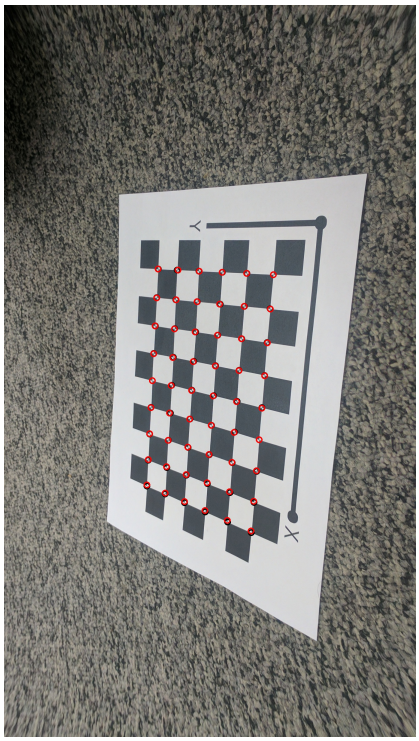

Fig. 17. re-projected points in image 12

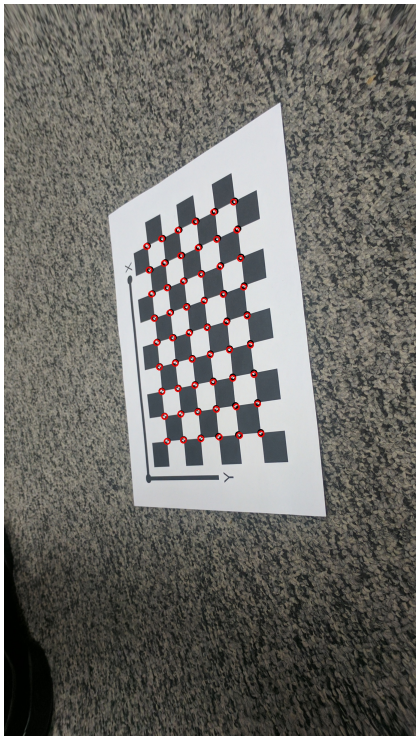

Fig. 18. re-projected points in image 13

VI. Conclusion

Auto Calibration for the given images as given in Zhang's paper has been successfully performed with a mean re-projection error of **2.011877**

References

[1] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf.

[2] http://staff.fh-hagenberg.at/burger/publications/reports/2016Calibration/Burger-CameraCalibration-20160516.pdf

[3] https://cmsc733.github.io/2019/hw/hw1/