CMSC733: Homework 1 - AutoCalib

Chayan Kumar Patodi
UID : 116327428
Email: ckp1804@terpmail.umd.edu

***Abstract*—One of the most important part of any computer vision research involves 3D geometry , which includes camera Calibration. Camera Calibration is important for 3D geometry and Camera Calibration is nothing but estimating parameters of the camera like the focal length, distortion coefficients and principle point.**

I. Introduction

The motive of this homework is to implement Camera Calibration using a more robust technique proposed by Zhang in his paper. We are provided with 13 images of Checkerboard pattern (9x6),which is used for the calibration. Using equations from the paper, intrinsic and extrinsic parameters are estimated. The parameters are optimized using a non-linear geometric error minimization. The results obtained from the above approach is analysed and re-projection error and final K matrix are presented.

II. Calibration Approach

A. Corner Detection

For each input image, I used the suggested function cv2.findChessboardCorners to find all chessboard corners in the image. The parameters are image and the dimensions of the chessboard (9*6), in our case.The output for one image can be seen in figure 1.

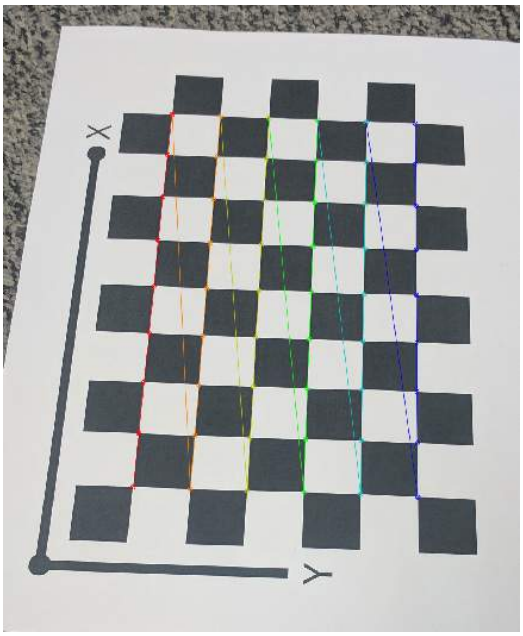

Fig. 1. Corners Detected in the Input Images.

The world coordinates for each corner are numbered from 1 to the number of corners on each row, and column.It's a meshgrid of 9*6, multiplied by the size of the square box (21.5mm).

B. Calculating Homography

The second step in the pipeline is to use these world coordinates and the corresponding detected points, we can estimate the homography between the model plane and each input image.This is achieved using the inbulit function, cv2.findHomography().The corners I selected for finding the homography were 1st,8th,46th and 54rth.Those can be seen in fig.2.

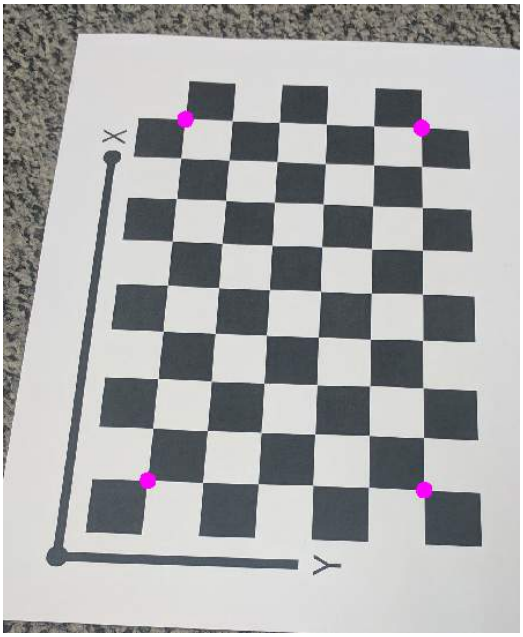

Fig. 2. Corners Selected for Homography in the Input Images.

C. Intrinsic Parameters Calculation

With the estimated homography matrices, I refer to section 3.1 in [1] and equations 99 to 105 from [2] to find the camera intrinsic parameters. We solve B =

$$A^T A$$

where A is the intrinsic camera matrix. Fig.3 shows the flow of calculating the intrinsic matrix, the equations the parameters in the matrix.

$\boldsymbol{b} \leftarrow \text{Solve}(\mathbf{V} \cdot \boldsymbol{b} = \mathbf{0})$ ▷ $\boldsymbol{b} = (B_0, \ldots, B_5)$

$w \leftarrow B_0 B_2 B_5 - B_1^2 B_5 - B_0 B_4^2 + 2 B_1 B_3 B_4 - B_2 B_3^2$ ▷ Eqn. (104)

$d \leftarrow B_0 B_2 - B_1^2$ ▷ Eqn. (105)

$\alpha \leftarrow \sqrt{w/(d \cdot B_0)}$ ▷ Eqn. (99)

$\beta \leftarrow \sqrt{w/d^2 \cdot B_0}$ ▷ Eqn. (100)

$\gamma \leftarrow \sqrt{w/(d^2 \cdot B_0) \cdot B_1}$ ▷ Eqn. (101)

$u_c \leftarrow (B_1 B_4 - B_2 B_3)/d$ ▷ Eqn. (102)

$v_c \leftarrow (B_1 B_3 - B_0 B_4)/d$ ▷ Eqn. (103)

$$\mathbf{A} \leftarrow \begin{pmatrix} \alpha & \gamma & u_c \\ 0 & \beta & v_c \\ 0 & 0 & 1 \end{pmatrix}$$

Fig. 3. Intrinsic Parameter Calculation [2]

The Initial intrinsic Matrix calculated by the above method and used in this Homework is given by:

$$K = \begin{bmatrix} 2034.751 & 0.492 & 772.703 \\ 0 & 2017.904 & 1360.909 \\ 0 & 0 & 1 \end{bmatrix}$$

D. Extrinsic Matrix

After computing intrinsic camera parameters, we can estimate the extrinsic parameters i.e. Rotation and translation vectors.These are computed using the following equations from [1]:

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

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

$$r_3 = r_1 \times r_3$$

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

where,

$$\lambda = 1/\left\|A^{-1}h1\right\| = 1/\left\|A^{-1}h2\right\|$$

E. Distortion Coefficients

All the above steps from the pipeline are performed assuming there is no distortion in the images (the ideal case). Now to calculate the distortion coefficients we initialise K_c = [0,0] as an initial guess.

F. Non-linear Geometric Error Minimization

Once we have an initial guess for each parameter K,E ,Kc we perform Non-linear Geometric Error Minimization as suggest to improve them and obtain the updated values of the camera intrinsic parameters and the distortion coefficients. This was done using scipy.optimize.leastsquares as suggested.The updated camera intrinsic matrix is given by:

$$K = \begin{bmatrix} 2026.959 & 1.428 & 772.527 \\ 0 & 2009.836 & 1360.450 \\ 0 & 0 & 1 \end{bmatrix}$$

The updated distortion coefficients are given by:

$$K_c = \begin{bmatrix} 0.0901 & -0.4274 \end{bmatrix}$$

III. Discussion and Conclusion

Once the non-linear minimization is performed, we get the updated parameters. This optimization is based on minimizing the euclidean distance between the corner points in the image, and the corner points obtained after reprojecting the world coordinate point, using the updated matrices and coefficients. After optimization, we still don't get the perfect answer,as there is still some error left. To evaluate the accuracy of our output and as suggested in the pipeline, we compute the reprojection error , which is **1.714**. This is done by calculating the norm between what we got with our transformation and findChessboardCorners.We find the mean error, for all the input images.

As can be seen in the figure 4, we can see the difference in the detected corner points and projected corner points.

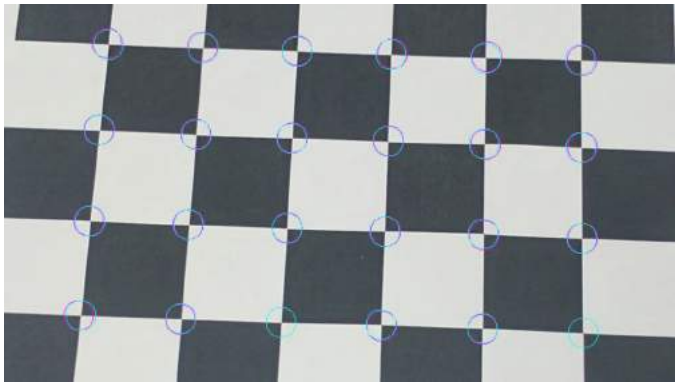

Fig. 4. Difference in corners.

IV. Results

The resultant parameters for the code file, would be in the following figure 4.

```
patodichayan@ChayanPatodi:/media/patodichayan/DATA/733/HW1$ python2.7 Wrapper.py

Initial Intrinsic Camera Matrix is:
[[2.03475109e+03 4.92810405e-01 7.72703920e+02]
 [0.00000000e+00 2.01790464e+03 1.36090928e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Updated Intrinsic Camera Matrix is:
[[2.02695977e+03 1.42856892e+00 7.72527797e+02]
 [0.00000000e+00 2.00983606e+03 1.36045054e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Distortion Coefficents:
[0.09010767231790394, -0.42745635536198673]

Mean Re-projection error is :
1.714585823618996
```

Fig. 5. Resultant Parameters.

The rectified images and the projected corners in those images can be found in the figures 6 to 18.

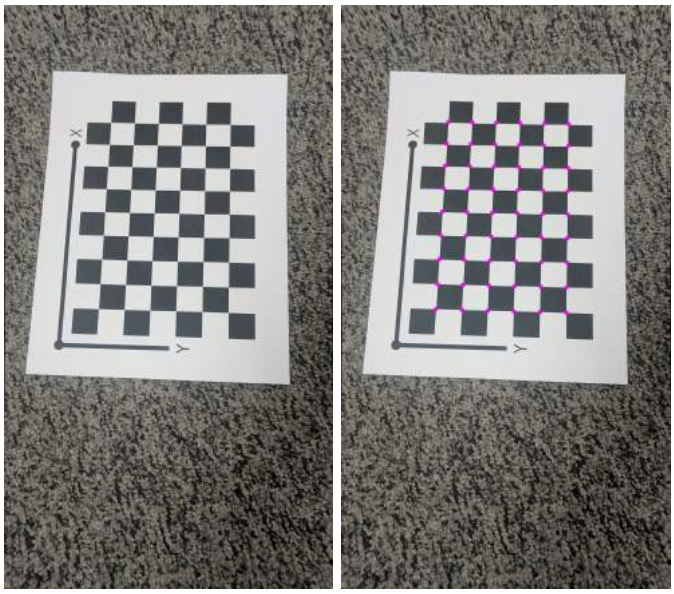

Fig. 6. Rectified Image and Re-projected Corners for Input Image 1.

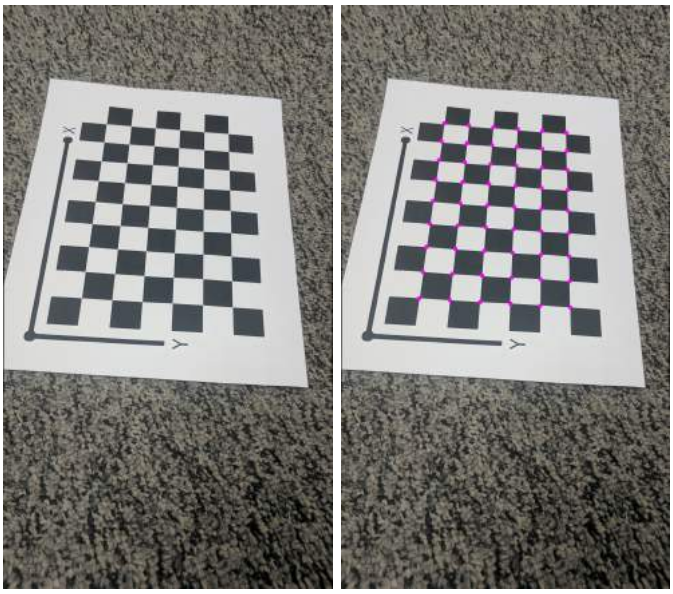

Fig. 8. Rectified Image and Re-projected Corners for Input Image 3.

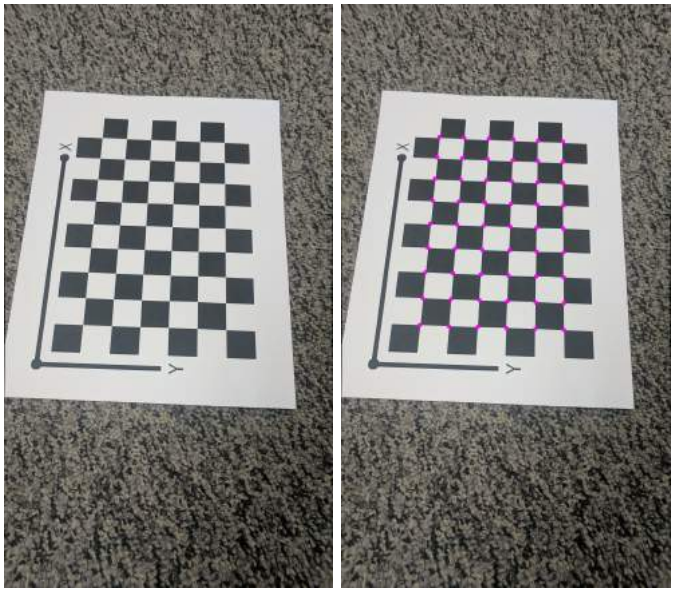

Fig. 7. Rectified Image and Re-projected Corners for Input Image 2.

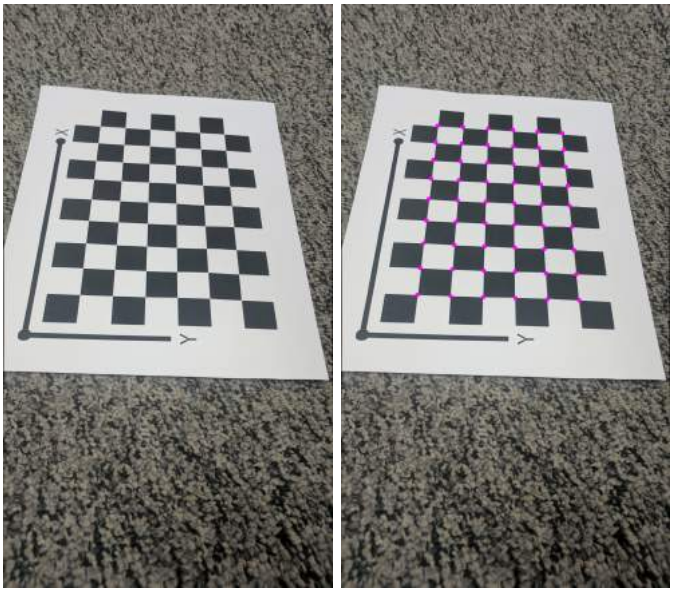

Fig. 9. Rectified Image and Re-projected Corners for Input Image 4.

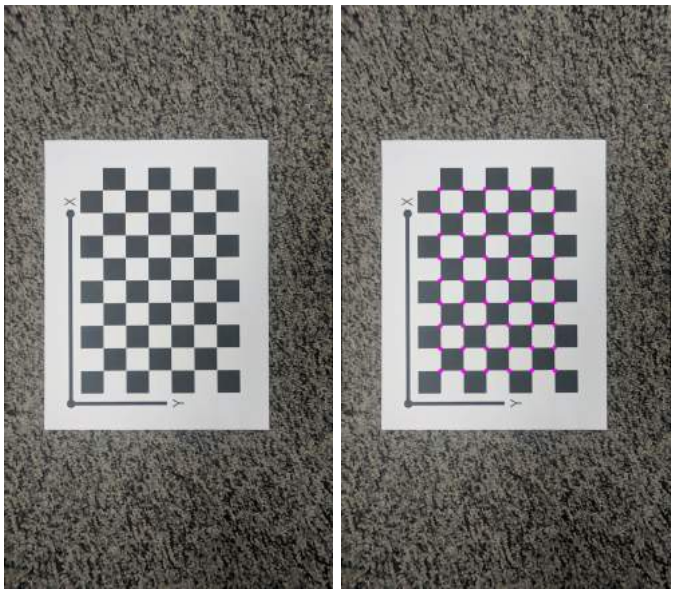

Fig. 10. Rectified Image and Re-projected Corners for Input Image 5.

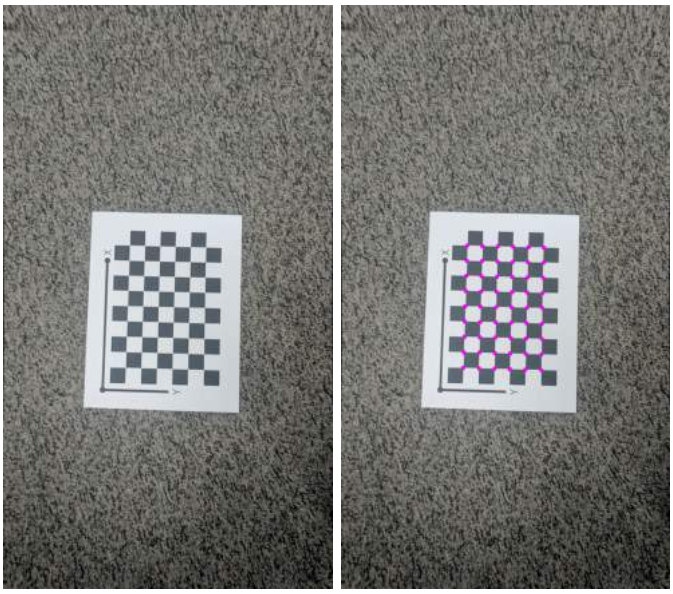

Fig. 12. Rectified Image and Re-projected Corners for Input Image 7.

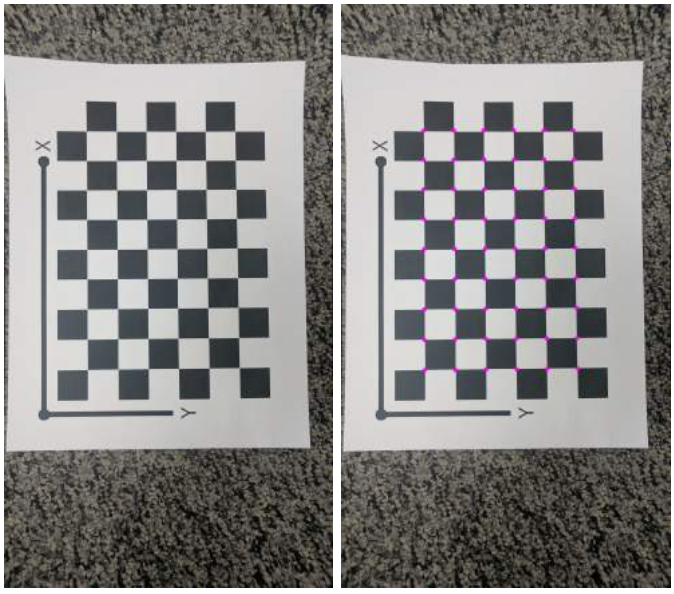

Fig. 11. Rectified Image and Re-projected Corners for Input Image 6.

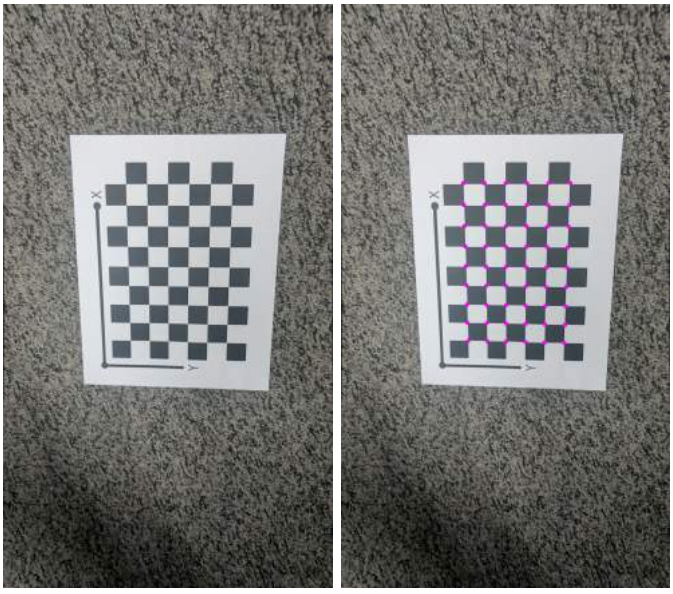

Fig. 13. Rectified Image and Re-projected Corners for Input Image 8.

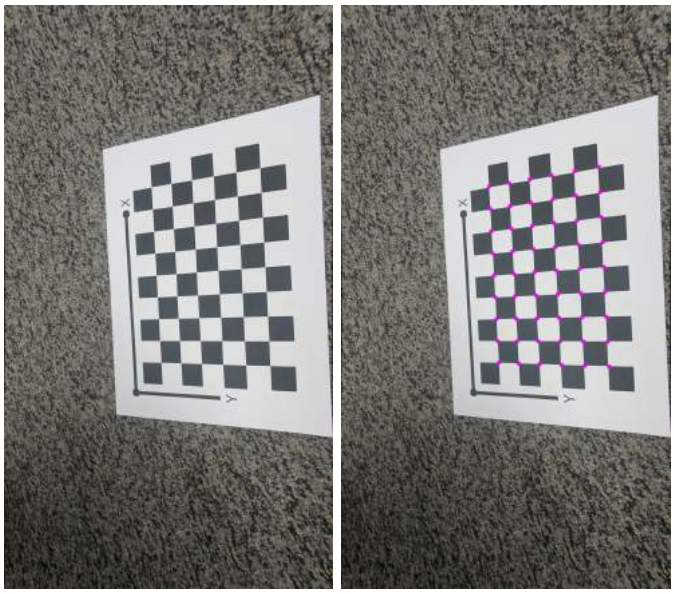

Fig. 14. Rectified Image and Re-projected Corners for Input Image 9.

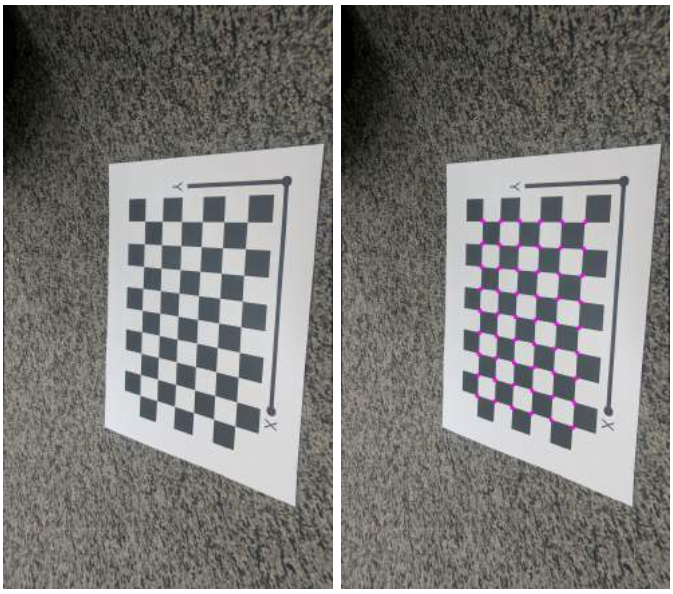

Fig. 16. Rectified Image and Re-projected Corners for Input Image 11.

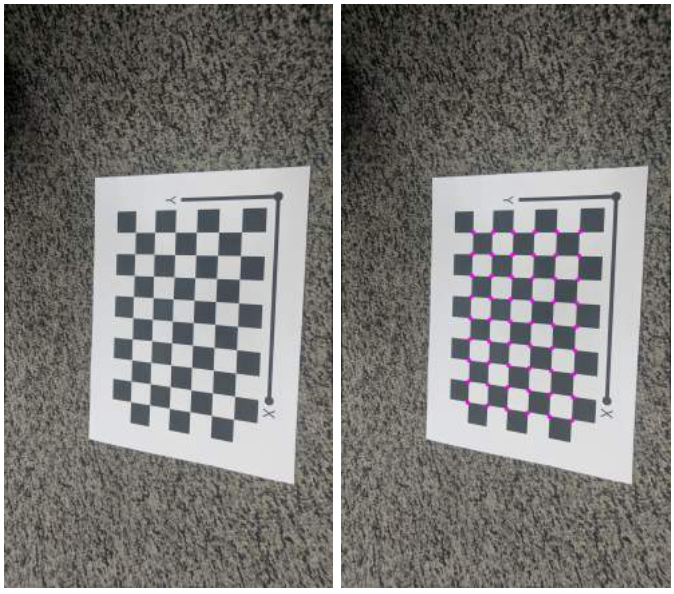

Fig. 15. Rectified Image and Re-projected Corners for Input Image 10.

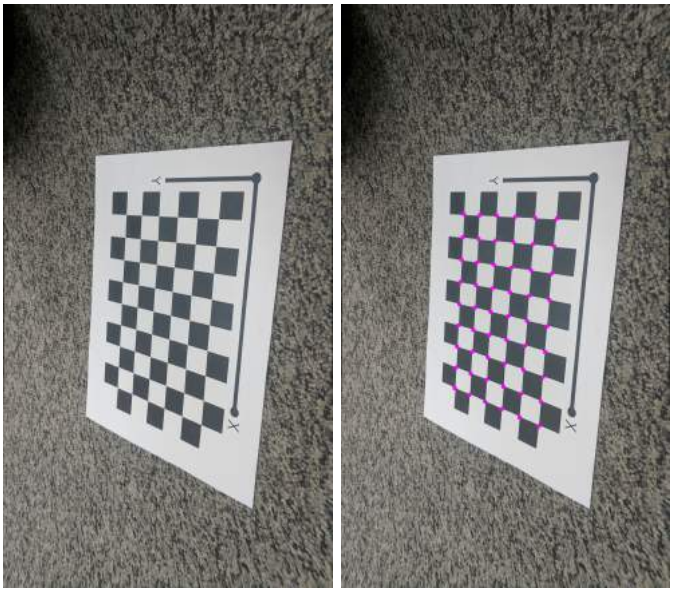

Fig. 17. Rectified Image and Re-projected Corners for Input Image 12.

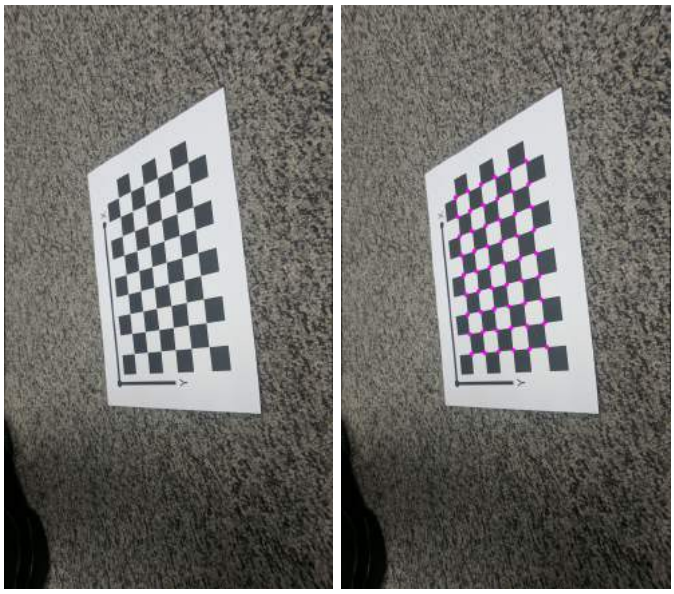

Fig. 18. Rectified Image and Re-projected Corners for Input Image 13.

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/nonlinmin