Homework 1: Camera Calibration

I-Sheng Fang
The Master's Degree Program in Robotics
National Yang Ming Chiao Tung University
isfang.gdr09g@nctu.edu.tw
Data collecting, Experiment, debug.

Hsiang-Chun Yang
Institute of Multimedia Engineering
National Yang Ming Chiao Tung University
yanghc.cs09g@nctu.edu.tw
Implementation.

Yu-Lin Yeh
Institute of Multimedia Engineering
National Yang Ming Chiao Tung University
s995503@gmail.com
Introduction, conclusion.

Abstract

In this assignment, we implement camera calibration from scratch and compare the result with the `cv2.calibrateCamera` in OpenCV. We also collect datasets with different lens and focusing mode as well as compare the results with baseline.

1 Introduction

Camera calibration is to find out the parameters of extrinsic matrix and intrinsic matrix in camera model. The extrinsic matrix transforms the object's position from the world coordinate system into the camera coordinate system. The intrinsic matrix transforms the camera coordinate system into pixel coordinate system. Therefore, we can use these two matrices to reconstruct the 3D position of the cameras from multiple 2D images.

There are many practical applications for camera calibration, such as the reproduction of 3D models of buildings, human organs, and natural landscapes or cultural monuments. It's convenient for the architects or doctors to their works. When people can't visualize the object in 3D, showing the reproduction of a 3D model is a good method. For instance,consumers can see the result of the finished product in advance[1]. In medical, 3D reconstruction can help doctor to accurate find out where is the tumor in the body.[2]

Nowadays, the multiple lens system in mobile phone becomes common. For example, the iPhone11 pro is equipped with ultra-wide-angle lens, wide-angle lens, and telephoto lens. In this assignment, We decide to use those camera lens to build the camera calibration data set and compare with auto and manually focusing.

2 Implementation Procedure

2.1 Homography matrix

We found that there is a `findHomography` function in the OpenCV package, but we still implement it by ourselves. `The findHomography` is only used to check whether our homography matrix is correct or not. In this part, we will explain how we compute the homography from scratch step-by-step. For the implementation of this section, please refer to `get_homography` function in `camera_calibration.py`.

The world coordinate can be projected to the pixel coordinate system by the following operations

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} f/s_x & 0 & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}$$

Since the 3D calibration rig is hard to compute, we can set the world coordinate system to the corner of the chessboard, so that all points on the chessboard lie in a plane that makes W to 0. Then simplify the transformation between two coordinate systems to

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} f/s_x & 0 & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} U \\ V \\ 1 \end{bmatrix}$$

By defining pixel coordinate p_i, world coordinate P_i and homography matrix H

$$p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}, P_i = \begin{bmatrix} U_i \\ V_i \\ 1 \end{bmatrix}, H = \begin{bmatrix} f/s_x & 0 & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

The equation becomes

$$p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim HP_i = \begin{bmatrix} h_1 P_i \\ h_2 P_i \\ h_3 P_i \end{bmatrix}$$

Note that h_i is the i-th row of H. After rewriting the equation, we can get

$$\begin{cases} u_i(h_3 P_i) - h_1 P_i = 0 \\ v_i(h_3 P_i) - h_2 P_i = 0 \end{cases}$$

Now we can form the following equation with n coordinates where n is the number of corners in a single chessboard photo.

$$\begin{bmatrix} -P_1^T & 0^T & u_1 P_1^T \\ 0^T & -P_1^T & v_1 P_1^T \\ \cdots & & \\ -P_n^T & 0^T & u_n P_n^T \\ 0^T & -P_n^T & v_n P_n^T \end{bmatrix} \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} = \mathrm{P}h = 0$$

To minimize $||\mathrm{P}h||^2$ over h subject to the constraint $||h||^2 = 1$, simply apply singular value decomposition to P.

$$\mathrm{P} = UDV^T$$

To get the homography matrix, we can set h equal to the last column of V and reshape it from 9×1 to 3×3.

$$h = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}, H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

By repeating the above operations N times where N equals the number of chessboard photos, we can get the homography matrix of each photo.

2.2 Intrinsic matrix

Now we can compute the intrinsic matrix after getting the homography matrix of each photo. For the implementation of this section, please refer to `get_intrinsic` function in `camera_calibration.py`. First, given the following equation where H and K is homography matrix and intrinsic matrix respectively.

$$H = [\mathrm{h}_1, \mathrm{h}_2, \mathrm{h}_3] = \begin{bmatrix} f/s_x & 0 & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} = \mathrm{K}\,[\mathrm{r}_1, \mathrm{r}_2, \mathrm{t}]$$

Notice that $\mathrm{h_i}$ represents the i-th column instead of row of matrix H. Then we can have the following equations

$$\begin{cases} \mathrm{r}_1 = \mathrm{K}^{-1}\mathrm{h}_1 \\ \mathrm{r}_2 = \mathrm{K}^{-1}\mathrm{h}_2 \end{cases}$$

Because r_1, r_2 and r_3 are orthogonal to each other, we can know that ${\mathrm{r}_1}^T\mathrm{r}_2 = 0$ and $||\mathrm{r}_1|| = ||\mathrm{r}_2|| = 1$. Then rewrite the above equations to

$$\begin{cases} {\mathrm{h}_1}^T\mathrm{K}^{-T}\mathrm{K}^{-1}\mathrm{h}_2 = 0 \\ {\mathrm{h}_1}^T\mathrm{K}^{-T}\mathrm{K}^{-1}\mathrm{h}_1 = {\mathrm{h}_2}^T\mathrm{K}^{-T}\mathrm{K}^{-1}\mathrm{h}_2 \end{cases}$$

Let $\mathrm{B} := \mathrm{K}^{-T}\mathrm{K}^{-1}$. It is obvious that B is symmetric and positive definite and K can be computed from B using Cholesky factorization.

Given matrix B

$$\mathrm{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix}$$

We can define the following equations for each homography matrix H

$$\begin{cases} \begin{bmatrix} h_{11} & h_{12} & h_{13} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \begin{bmatrix} h_{21} \\ h_{22} \\ h_{23} \end{bmatrix} = 0 \\ \begin{bmatrix} h_{11} & h_{12} & h_{13} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \end{bmatrix} - \begin{bmatrix} h_{21} & h_{22} & h_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \begin{bmatrix} h_{21} \\ h_{22} \\ h_{23} \end{bmatrix} = 0 \end{cases}$$

Then define $\mathrm{b} = [b_{11} \quad b_{12} \quad b_{13} \quad b_{22} \quad b_{23} \quad b_{33}]$ leads to the following system

$$\mathrm{Vb} = \begin{bmatrix} v_{11} \\ v_{12} \\ \ldots \\ v_{N1} \\ v_{N2} \end{bmatrix} = 0$$

Note that N is the number of chessboard photos, and v_{i1} and v_{i2} can be calculated by the equations below

$$\begin{cases} v_{i1} = \begin{bmatrix} h_{11}h_{21} & h_{12}h_{21} + h_{11}h_{22} & h_{13}h_{21} + h_{11}h_{23} & h_{12}h_{22} & h_{13}h_{22} + h_{12}h_{23} & h_{13}h_{23} \end{bmatrix} \\ v_{i2} = \begin{bmatrix} h_{11}^2 - h_{21}^2 & 2(h_{11}h_{12} - h_{21}h_{22}) & 2(h_{11}h_{13} - h_{21}h_{23}) & h_{12}^2 - h_{22}^2 & 2(h_{12}h_{13} - h_{22}h_{23}) & h_{13}^2 - h_{23}^2 \end{bmatrix} \end{cases}$$

Our goal is to minimize b over Vb. This can be done by apply singular value decomposition to V.

$$\mathrm{V} = UDV^T$$

By setting b to the last column of V, we can reconstruct matrix B by b. Then apply Cholesky factorization to B.

$$\mathrm{B} = LL^H$$

Notice that, since B is symmetric and positive definite, L^H in the above equation can be rewritten as L^T. The intrinsic matrix K now can be computed by

$$\mathrm{B} = LL^H = LL^T = \mathrm{K}^{-T}\mathrm{K}^{-1} \Rightarrow \mathrm{K} = L^{-T}$$

2.3 Extrinsic matrix

Once we have intrinsic matrix K, we can obtain the extrinsic matrix of each chessboard photo by the following equations

$$\begin{cases} \mathrm{r}_1 = \lambda \mathrm{K}^{-1}\mathrm{h}_1 \\ \mathrm{r}_2 = \lambda \mathrm{K}^{-1}\mathrm{h}_2 \\ \mathrm{r}_3 = \mathrm{r}_1 \times \mathrm{r}_2 \\ \mathrm{t} = \lambda \mathrm{K}^{-1}\mathrm{h}_3 \\ \lambda = 1/||\mathrm{K}^{-1}\mathrm{h}_1|| \end{cases}$$

$$\text{Extrinsic matrix} = [\mathrm{R}|\mathrm{t}] = [\mathrm{r}_1 \quad \mathrm{r}_2 \quad \mathrm{r}_3 \quad \mathrm{t}]$$

For the implementation of this section, please refer to `get_extrinsic` function in `camera_calibration.py`.

3 Experimental Result

In this section, we will explain our camera configurations, show the results of our method and compare them with the method of OpenCV.

3.1 Collecting data

We collected 6 sets of photos. The photo sets were respectively shot with the ultra-wide-angle lens, wide-angle lens and telephoto lens with iPhone 11 Pro. We also shot photos with autofocus mode by iOS camera app or manually focusing the lens by Adobe Lightroom app. Each photo set contained at least 9 pictures.

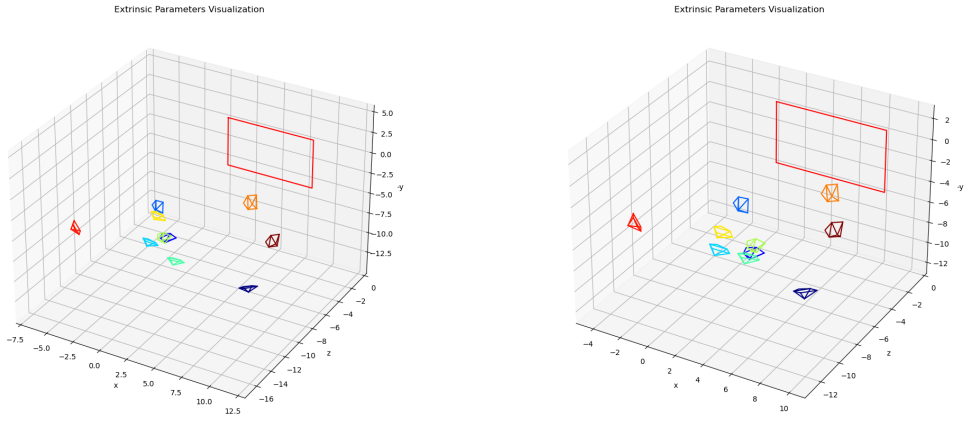
3.2 Qualitative results

We compare our method to the baseline, `cv2.calibrateCamera` in OpenCV. Two sets of results with different lens and focusing mode are shown in Figure 1, 3 and Figure 4.

4 Discussion

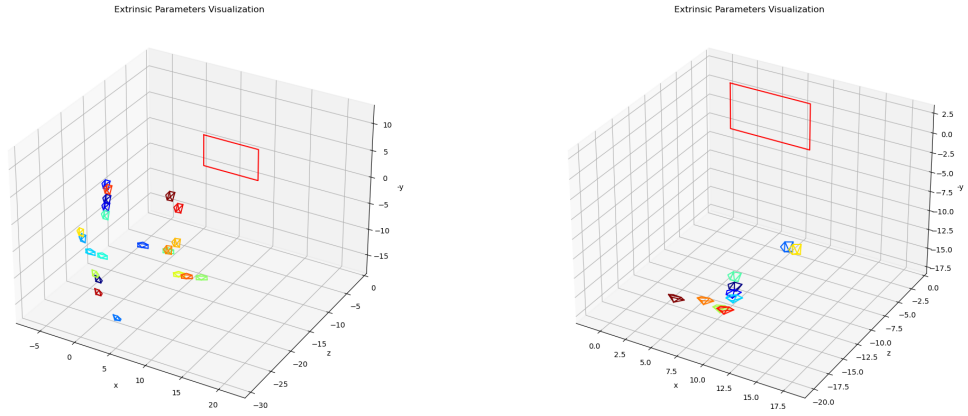
As Figure 1, 3 and 4 shown, our implementation has the close result with `cv2.calibrateCamera` in OpenCV. We discover that the larger focal length has more different results. Our results are shown the auto focus doesn't effect the camera calibration.

During the collecting data, we discover that the background color (Figure 5) effects our camera calibration results as Figure 2 shown. The method of OpenCV is more robustness. We traced the documents of OpenCV and found that OpenCV uses global Levenberg-Marquardt optimization algorithm to minimize the re-projection error between image points and projected object points. `cv2.calibrateCamera` also consider with the distortion. We guess these are the main reasons causing the different results between `cv2.calibrateCamera` and our method.



Ours OpenCV

Figure 1: The camera calibration results with data provided by TA



Ours OpenCV

Figure 2: The camera calibration results with black background

5 Conclusion

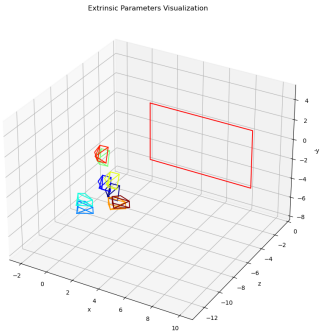
Through the analysis of experiment results, we realize that camera calibration will be affected by the external environment and the camera configuration. Our experiment shows that autofocus mode does not affect camera calibration which is different from the assignment instruction. We also discover that the background of the chessboard affects the performance of camera calibration. Despite the different results between our implementation and OpenCV, tracing the source code of `cv2.calibrateCamera` inspires us to learn other camera calibration approaches. To sum up, this experience will enable us to do more advanced camera model research in the future.

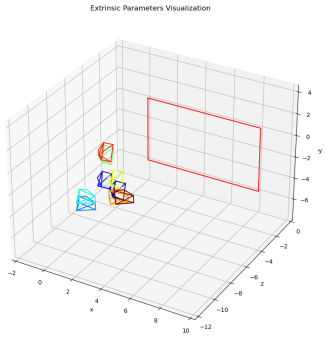
References

[1] G. D. Liu & W.C. ZHAO (2011) The Application of Computer Vision Techniques for Image-based 3D Reconstruction on iOS Platform,*institutional Repository of NCTU*

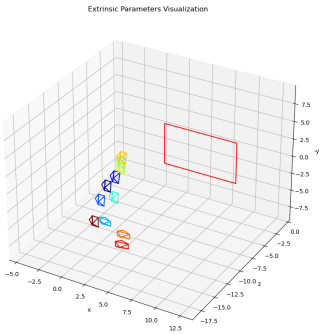
[2] D. Y. Jian (2002) Auto-Calibration, Reconstruction and Assessment of Clinical Lesions from Endoscopic Image Sequence*Airiti Library*

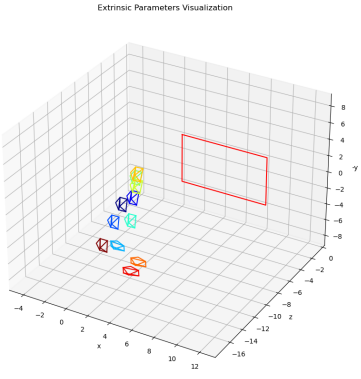
AF w/
ultra wide
lens



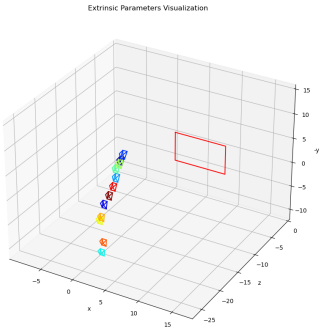


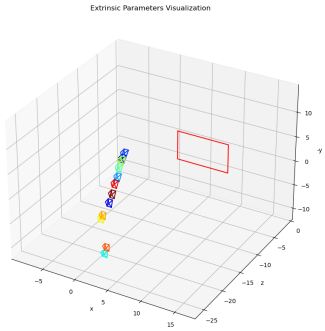
AF w/
wide lens





AF w/
telephoto
len

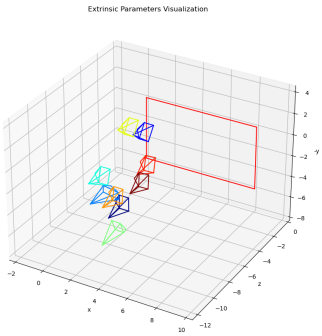


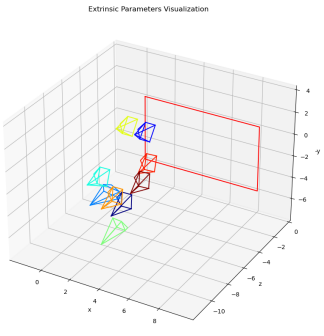


Ours OpenCV

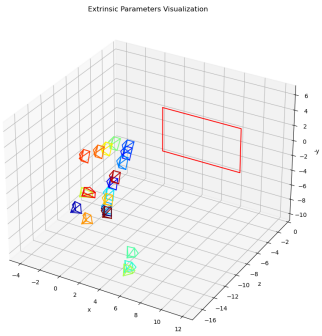
Figure 3: The camera calibration results with auto focusing iOS camera app

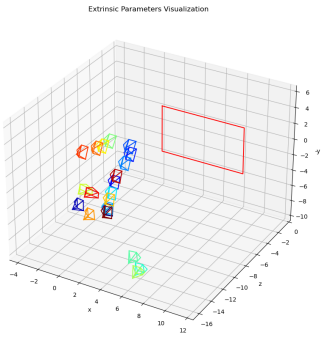
MF w/ ultra wide lens



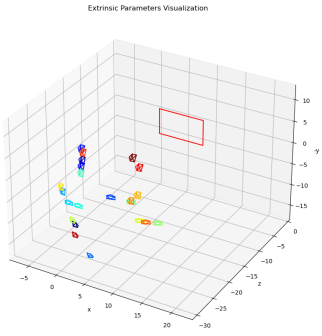


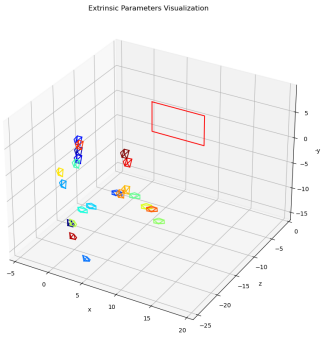
MF w/ wide lens





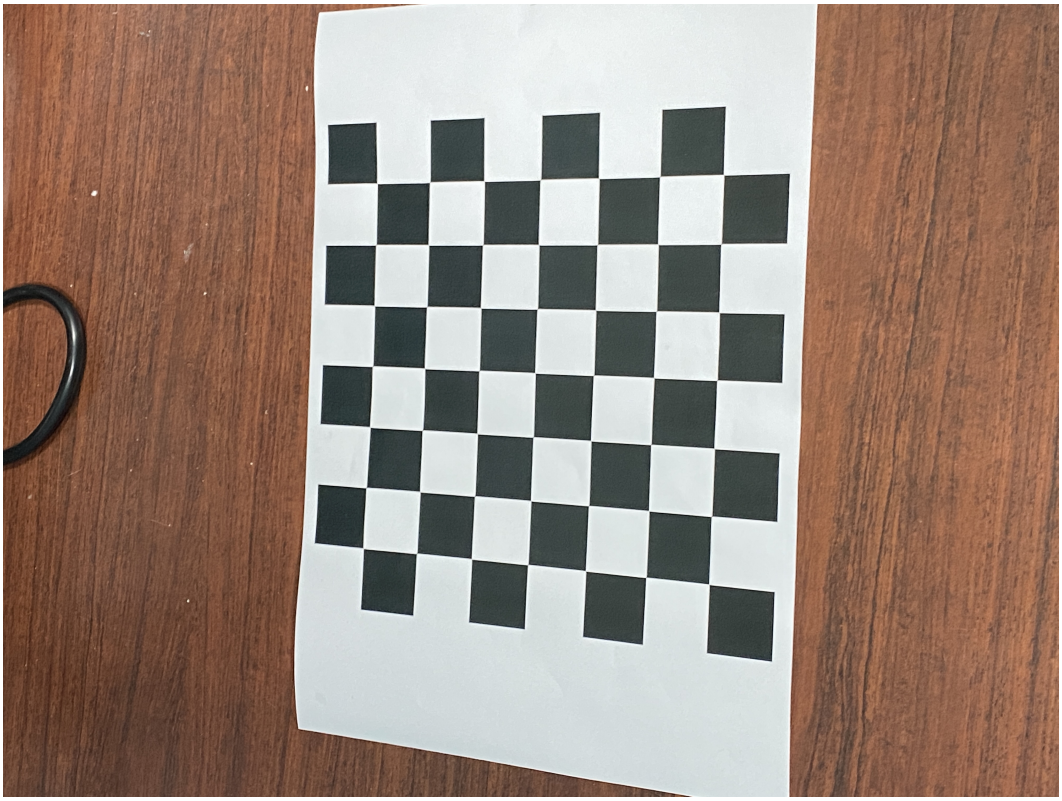
MF w/ telephoto lens



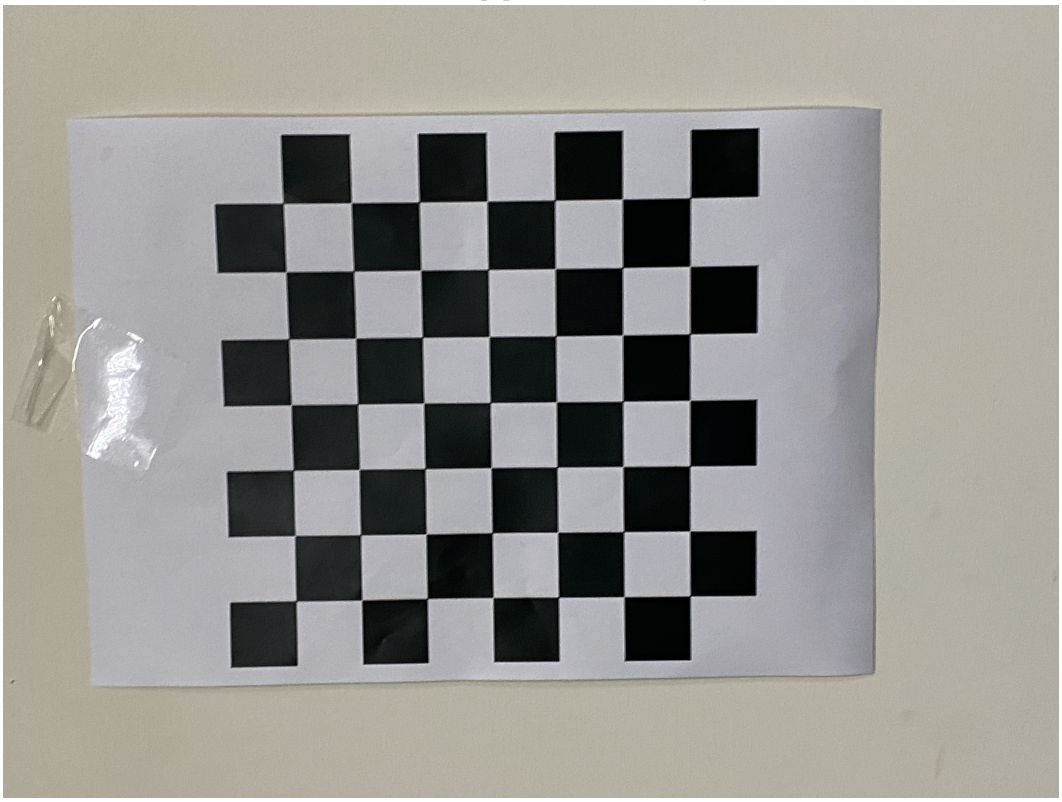


Ours OpenCV

Figure 4: The camera calibration results with manually focusing Adobe Lightroom app



(a) The chessboard paper with brown background



(b) The chessboard paper with white background

Figure 5: The background of chessboard will effect the results of camera calibration