

Homework 1

Camera Calibration

Group 12

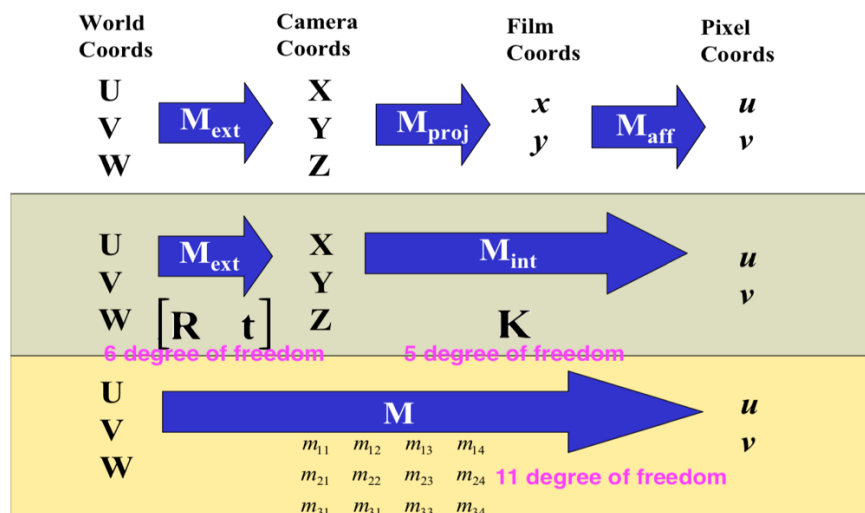
0786031 廖俊凱 0516044 陳思妤 0856733 黃明翰

I. Introduction

Camera calibration is the process of estimating intrinsic and/or extrinsic parameters. Intrinsic parameters deal with the camera's internal characteristics, such as, its focal length, skew, distortion, and image center. Extrinsic parameters describe its position and orientation in the world. Knowing intrinsic parameters is an essential first step for 3D computer vision, as it allows us to estimate the scene's structure in Euclidean space and removes lens distortion, which degrades accuracy.

II. Implementation Procedure

- A. The following figure shows the mathematical procedure of camera calibration in which our target is to derive every parameters and coefficients between each coordinate based on backward projection equation recovering 3D scene structure from images we provided.



B. Figure out the H_i of each image we provided.

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

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

First, we consider $\mathbf{x} = \mathbf{K}[\mathbf{R} \ \mathbf{t}] \mathbf{X}$ for each image, that is

$$p_i(\text{corners}) = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = MP_i(\text{Object point}) = \begin{bmatrix} m_1 P_i \\ m_3 P_i \\ m_2 P_i \\ m_3 P_i \end{bmatrix}$$

$$\rightarrow \begin{cases} u_i(m_3 P_i) - m_1 P_i = 0 \\ v_i(m_3 P_i) - m_2 P_i = 0 \end{cases}$$

Now we can construct \mathbf{P} by:

$$\begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ & \vdots & \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{bmatrix} \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} = \mathbf{P} \mathbf{m} = 0$$

Solved by SVD to get $\mathbf{P} = \mathbf{U} \mathbf{D} \mathbf{V}^T$ and set \mathbf{m} equal to the last column of \mathbf{V} . Reformatting \mathbf{m} into \mathbf{H} , which is the homography we desire.

```

1. # Build Homography H
2.     H = []
3.     for objp, imgp in zip(objpoints, imgpoints):
4.         P = np.zeros((corner_x*corner_y*2,9))
5.         for i in range(corner_x*corner_y):
6.             P[i*2] = [objp[i,0],objp[i,1],1,0,0,0,-imgp[i,0,0]*objp[i,0],-
imgp[i,0,0]*objp[i,1],-imgp[i,0,0]*1]
7.             P[i*2+1] = [0,0,0,objp[i,0],objp[i,1],1,-imgp[i,0,1]*objp[i,0],-
imgp[i,0,1]*objp[i,1],-imgp[i,0,1]*1]
8.         U,D,V = np.linalg.svd(P)
9.         m = V.T[:,-1]
10.        m = m/m[-1]
11.        H.append(m.reshape((3,3)))

```

C. Use H_i to find B , and calculate intrinsic matrix K from B by using Cholesky factorization.

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

Note: K can be calculated from B using Cholesky factorization

We learn from class that

$$\begin{cases} h_1^T K^{-T} K^{-1} h_2 = 0 \\ h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \end{cases} \text{ where } H = (h_1, h_2, h_3) \text{ and } B: \\ = K^{-T} K^{-1}$$

By expanding the first equation $h_1^T K^{-T} K^{-1} h_2$, we will get:

$$\begin{aligned} & (h_{11}b_{11}h_{12} + h_{21}b_{12}h_{12} + h_{31}b_{13}h_{12}) + \\ & (h_{11}b_{12}h_{22} + h_{21}b_{22}h_{22} + h_{31}b_{23}h_{22}) + \\ & (h_{11}b_{13}h_{32} + h_{12}b_{23}h_{32} + h_{31}b_{33}h_{32}) = 0 \end{aligned}$$

(by substituting b_{21} to b_{12} , b_{31} to b_{13} , b_{32} to b_{23})

Then we can derive that

$$(h_{11}h_{12})b_{11} + (h_{21}h_{12} + h_{11}h_{22})b_{12} + (h_{31}h_{12} + h_{11}h_{32})b_{13} + (h_{21}h_{22})b_{22} + (h_{31}h_{22} + h_{12}h_{32})b_{23} + (h_{31}h_{32})b_{33} = v_{12}b = 0$$

Thus, we have v in general:

$$v_{ij} = (v_{1i}v_{1j}, v_{2i}v_{1j} + v_{1i}v_{2j}, v_{3i}v_{1j} + v_{1i}v_{3j}, v_{2i}v_{2j}, v_{3i}v_{2j} + v_{1i}v_{3j}, v_{3i}v_{3j})$$

Now we can build V by $\begin{bmatrix} v_{12} \\ v_{11} - v_{22} \end{bmatrix} b = Vb = 0$, solving V by SVD to get b , and reformatting b into B .

In the end, we can find **intrinsic matrix K** from B by using Cholesky factorization.

```

1. # V for Vb=0
2.     V = np.zeros((len(H)*2,6))
3.     for i,h in enumerate(H):
4.         V[i*2] = [
5.             h[0,0]*h[0,1],
6.             h[0,0]*h[1,1]+h[1,0]*h[0,1],
7.             h[1,0]*h[1,1],
8.             h[2,0]*h[0,1]+h[0,0]*h[2,1],
9.             h[2,0]*h[1,1]+h[1,0]*h[2,1],
10.            h[2,0]*h[2,1]
11.        ]
12.        V[i*2+1] = [
13.            h[0,0]*h[0,0]-h[0,1]*h[0,1],
14.            (h[0,0]*h[1,0]-h[0,1]*h[1,1])*2,
15.            h[1,0]*h[1,0]-h[1,1]*h[1,1],
16.            (h[2,0]*h[0,0]-h[2,1]*h[0,1])*2,
17.            (h[2,0]*h[1,0]-h[1,1]*h[2,1])*2,
18.            h[2,0]*h[2,0]-h[2,1]*h[2,1]
19.        ]

```

```

20.     U,D,V = np.linalg.svd(V)
21.     b = V.T[:, -1]
22.     B = np.array([
23.         [b[0], b[1], b[3]],
24.         [b[1], b[2], b[4]],
25.         [b[3], b[4], b[5]]
26.     ])
27.     # intrinsic matrix K
28.     K_inv = np.linalg.cholesky(B).T
29.     K = np.linalg.inv(K_inv)
30.     K = K/K[-1, -1]

```

D. Get extrinsic matrix $[R|t]$ for each image by K and H

$$H = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \rho \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

After getting intrinsic matrix K, we can derive extrinsic matrix by K and H through:

$$\begin{aligned}
 r_1 &= \lambda K^{-1} h_1 \\
 r_2 &= \lambda K^{-1} h_2 \\
 r_3 &= r_1 \times r_2 \\
 t &= \lambda K^{-1} h_3 \\
 \lambda &= 1 / \|K^{-1} h_1\|
 \end{aligned}$$

And then we can have extrinsic matrix $[R|t]$.

```

1.     # extrinsic matrix
2.     K_inv = np.linalg.inv(K)
3.     rvecs = []
4.     tvecs = []
5.     for h in H:
6.         h1 = h[:, 0]
7.         h2 = h[:, 1]
8.         h3 = h[:, 2]

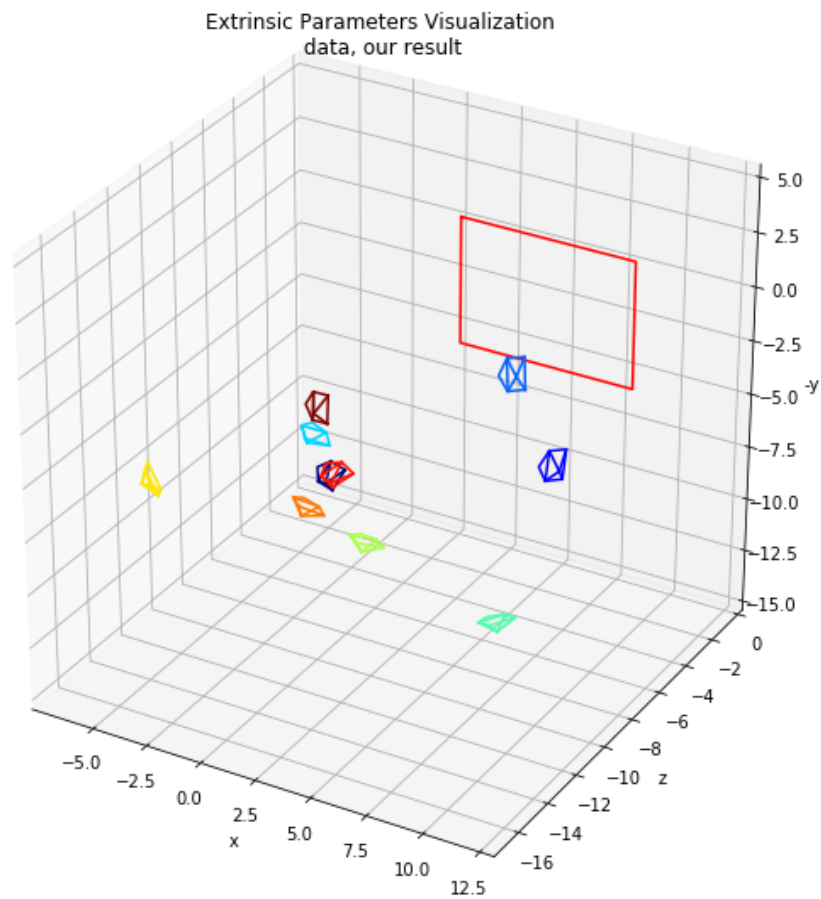
```

```

9.         lamda = 1/(np.linalg.norm(K_inv.dot(h1),2))
10.        r1 = lamda*K_inv.dot(h1)
11.        r2 = lamda*K_inv.dot(h2)
12.        r3 = np.cross(r1,r2)
13.        rotation_matrix = np.zeros((3, 3))
14.        rotation_matrix[:,0] = r1
15.        rotation_matrix[:,1] = r2
16.        rotation_matrix[:,2] = r3
17.
18.        r,_ = cv2.Rodrigues(rotation_matrix)
19.        t = lamda*K_inv.dot(h3).reshape(-1,1)
20.        rvecs.append(r)
21.        tvecs.append(t)
22.    return K, rvecs, tvecs

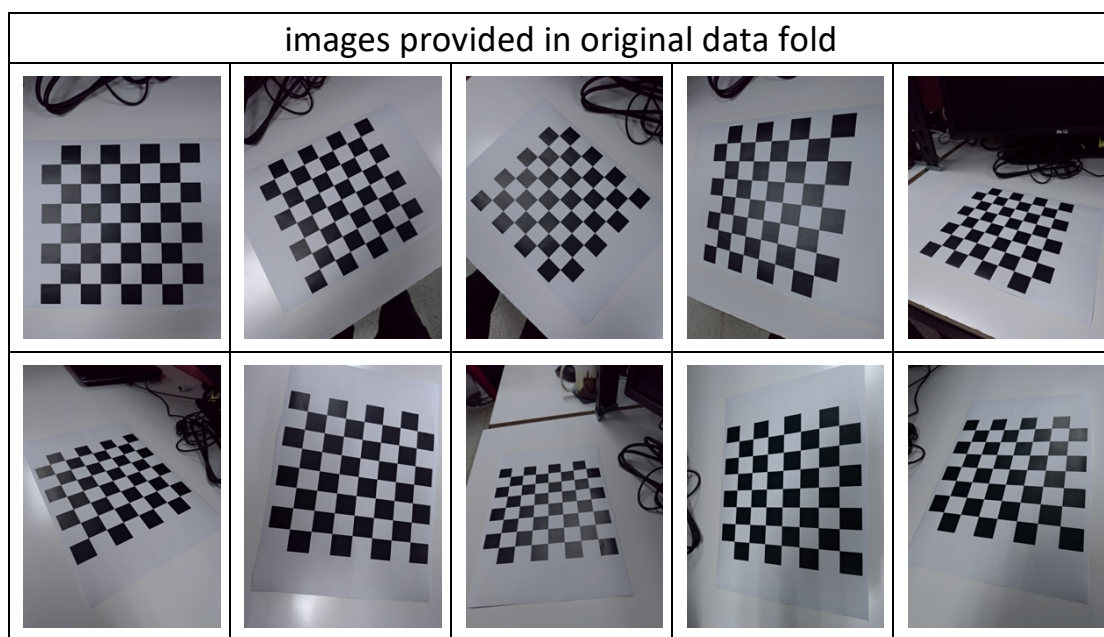
```

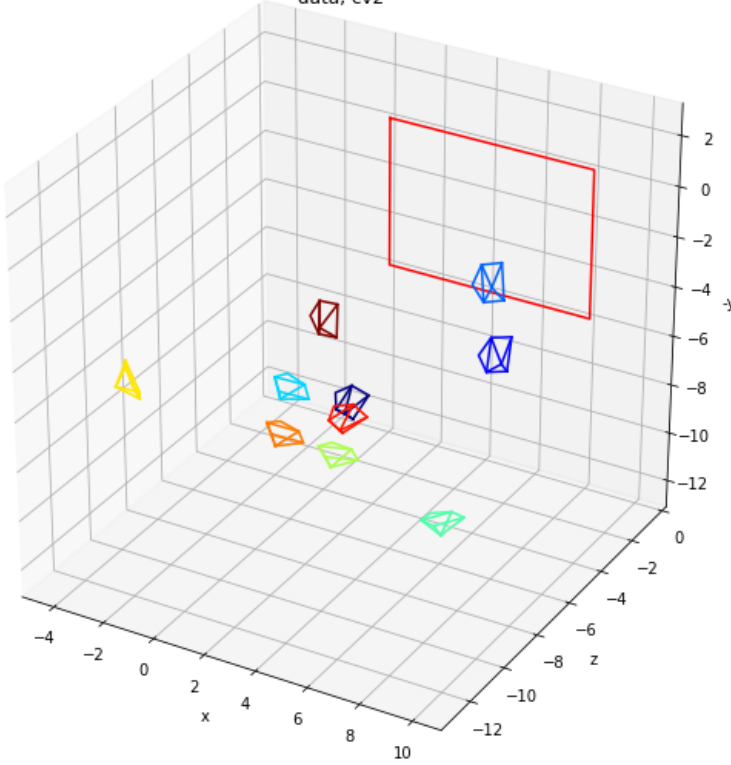
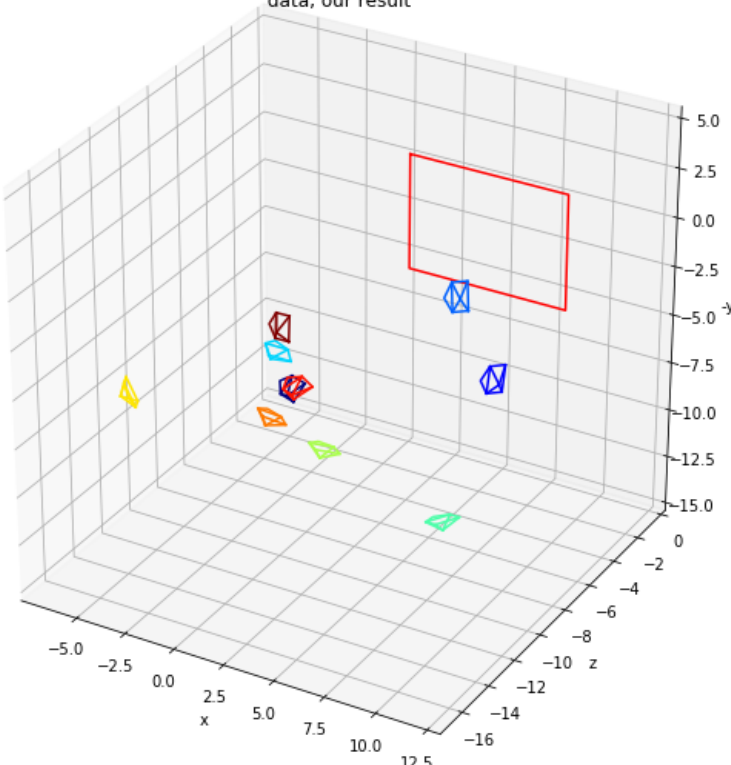
E. Plot the result of camera calibration



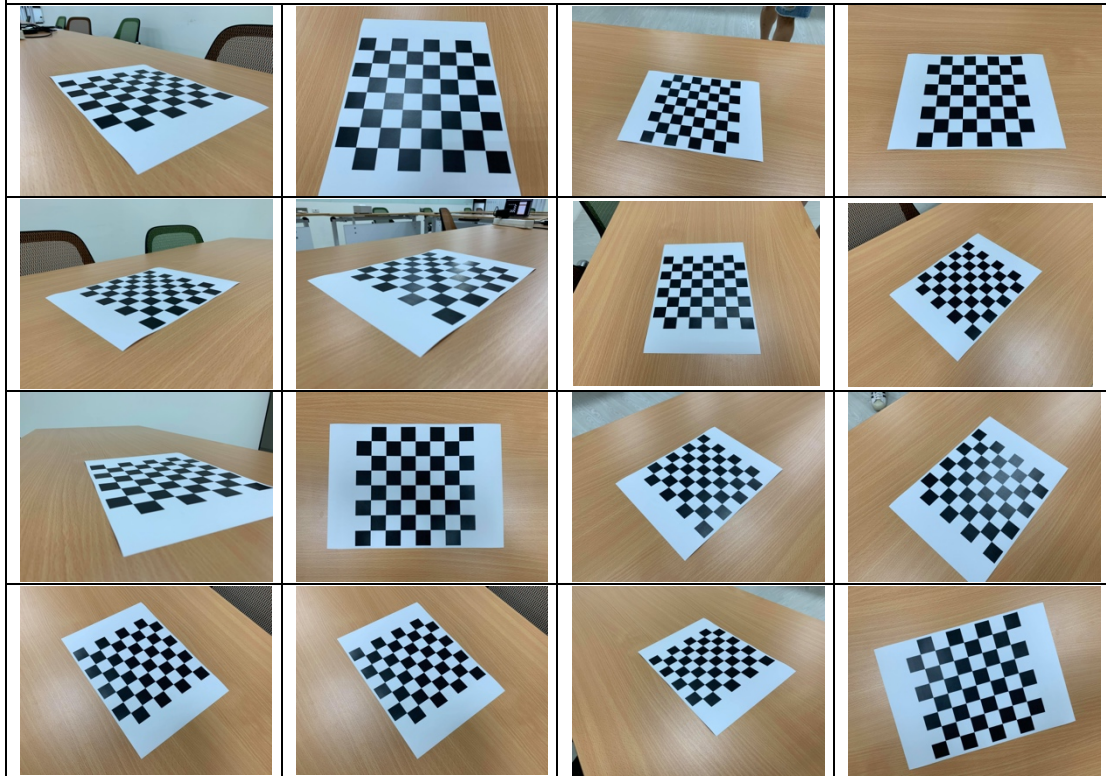
III. Experimental Result

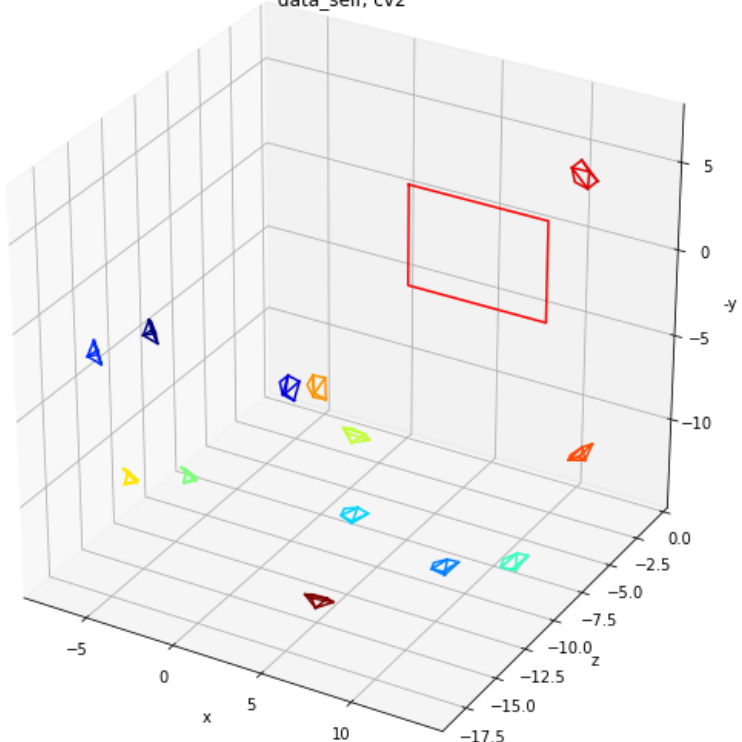
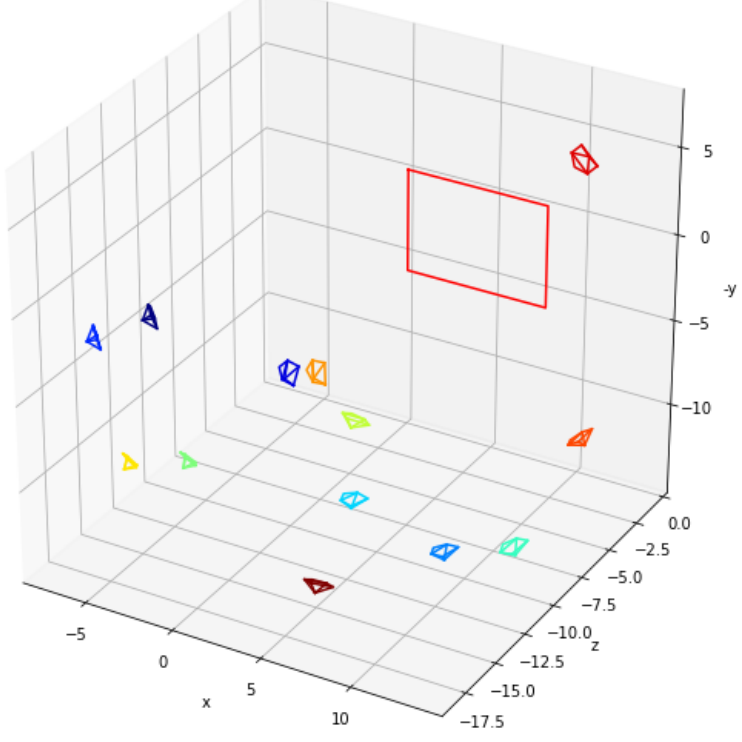
For the dataset of chessboard images, we use three kinds of different types of images, including of images provided in original data fold, images captured by our smartphones from random orientation and angels with fixed focus and images captured from one side to verify the camera calibration result. For each dataset of the images, we develop our camera calibration function comparing with the result from the `cv2.calibrateCamera` function, and the error rate between our function and the counterpart's one.



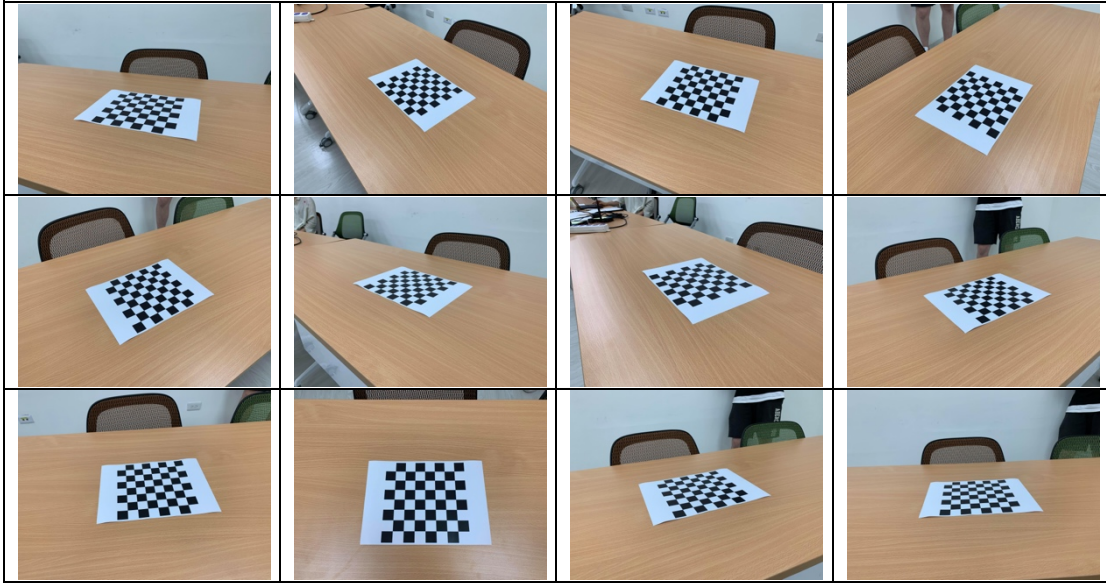
	result
cv2 function	<p>Extrinsic Parameters Visualization data, cv2</p> 
our function	<p>Extrinsic Parameters Visualization data, our result</p> 

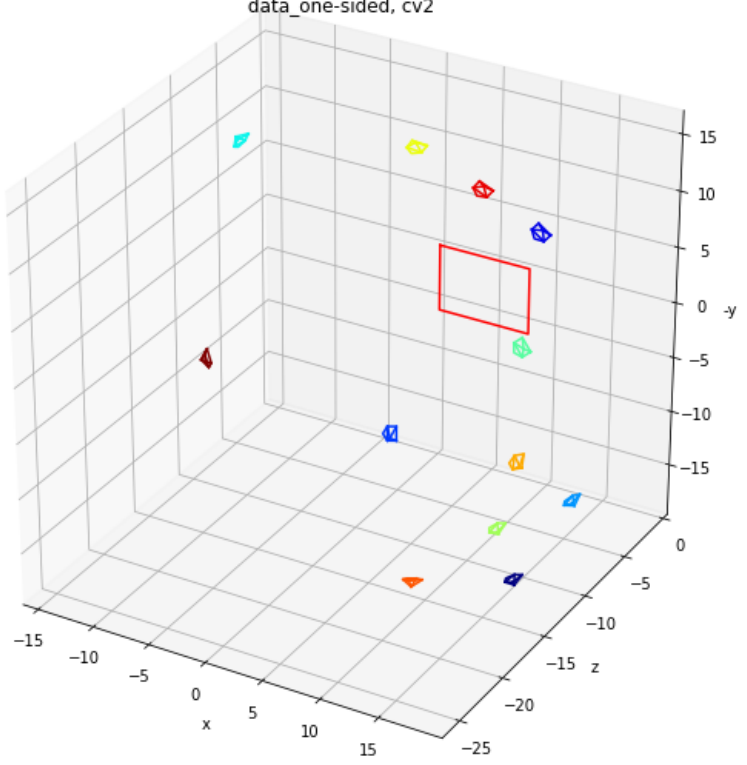
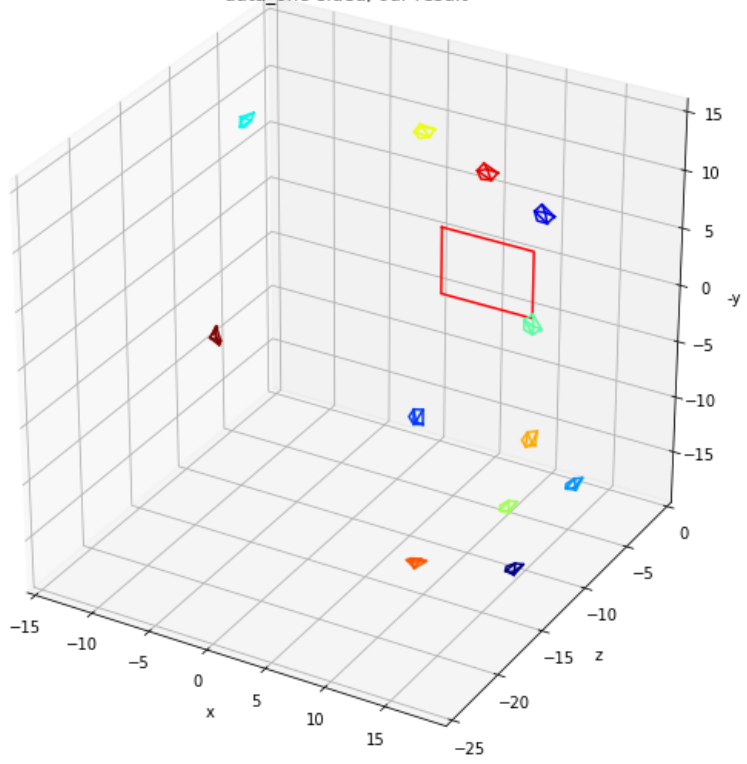
images captured by our smartphones with fixed focus



	result
cv2 function	<p>Extrinsic Parameters Visualization data_self, cv2</p> 
our function	<p>Extrinsic Parameters Visualization data_self, our result</p> 

images captured from one side



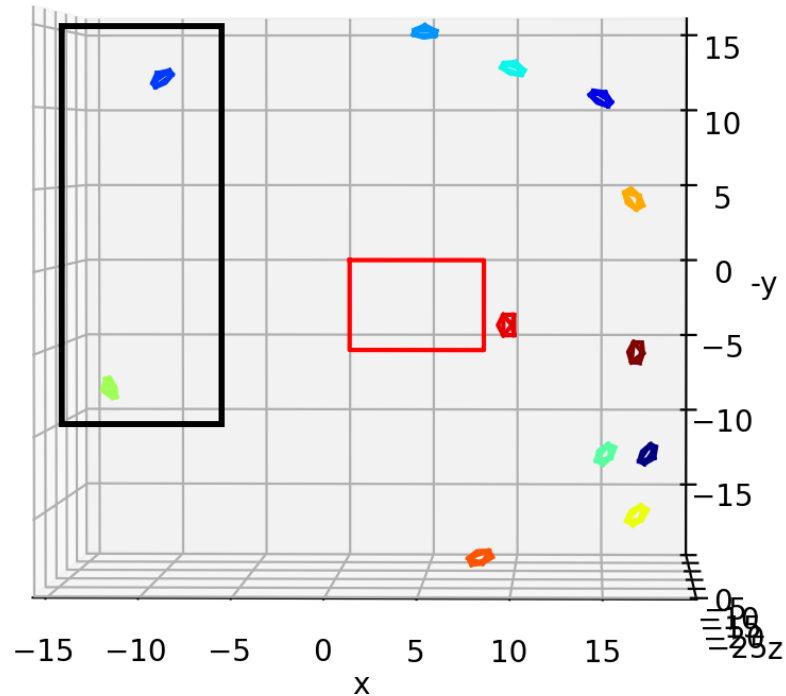
	result
cv2 function	<p>Extrinsic Parameters Visualization data_one-sided, cv2</p> 
our function	<p>Extrinsic Parameters Visualization data_one-sided, our result</p> 

IV. Discussion

- A. Comparing the result conducted by our function and cv2 function, the outcomes are basically the same. Since we don't have the ground truth of the dataset, we take the result of cv2 function as the correct result, and we estimate the performances by observing the difference between two camera extrinsic visualization images. Using re-projection error is a proper way to quantify how accurate the found parameters is. In our implementation, distortion of the image is not taken into consideration, so the error rate conducted by this method is not accurate, either. So, It may be the future work we could do to get better and thorough result on our project.
- B. According to the results of the images captured from only one every point (for example the points in black frame in the following figure) of the result is located on the same side which seems to be a discrepancy in the result of the truth and camera calibration calculated by ourselves.

We speculate that during the camera calibration, it is conducted by backward calculation contributing to the points of the result located on both side of our chessboard sheet since all the target images are line symmetric graphs where we or the computer could not verify the both sides of the chessboard sheet without any reference point. We think this phenomenon may be the practical issue of camera calibration which may be solved by optimizing our code by detail, taking some reference points into consideration or using non-line symmetric graphs.

Extrinsic Parameters Visualization



V. Conclusion

Camera calibration is a necessary process in the field of vision measurement playing an important part of the application of photogrammetry that aims to compute the camera model parameters from two-dimensional images. In this assignment, we learned how the camera calibration works by implementing camera calibration function. Besides, we know that camera model parameters in mathematical calculation including intrinsic and extrinsic parameters which describe the geometry of imaging process and camera position and attitude in the world coordinate system respectively.

VI. Reference

- A. <https://answers.opencv.org/question/3018/how-to-verify-the-correctness-of-calibration-of-a-webcam/>
- B. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html
- C. <https://silverwind1982.pixnet.net/blog/post/153218861-pinhole-camera%E7%9B%B8%E6%A9%9F%E6%A0%A1%E6%AD%A3-%28camera-calibration%29>
- D. https://boofcv.org/index.php?title=Tutorial_Camera_Calibration

VII. Work Assignment Plan Between Team Members

0786031 廖俊凱	Report
0516044 陳思妤	Result verification
0856733 黃明翰	Camera calibration function