

Computer Vision Assignment #1

0516234 黃聖祺 0516234 林恭白 0510532 楊上萱

A. Introduction

In this assignment, we are going to implement camera calibration. We have several photos of a certain chessboard picture taken from different poses. Our goal is to find out the extrinsic matrix and intrinsic matrix. At the end, we will plot the extrinsic matrixes.

B. Implementation Procedure

1. First of all, we have multiple photos from different poses. Then, for each photo, we have (u, v) from image plane, and (U, V) from object plane. The relationship between these two planes can be described as following:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \rho \underset{\substack{\text{up to a} \\ \text{scale}}}{H} \begin{bmatrix} U \\ V \\ 1 \end{bmatrix} \quad (\text{where } H \text{ is Homography})$$

2. We can remodel the above equation in a simpler way.

$$\begin{pmatrix} -X & -Y & -1 & 0 & 0 & 0 & u.X & u.Y & u \\ 0 & 0 & 0 & -X & -Y & -1 & v.X & v.Y & v \end{pmatrix} \begin{pmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{pmatrix} = 0$$

Now, we can find out H_i using the points in each images. Here, to find H , we first define *Matrix P* using the above equation:

$$P \cdot H = 0$$

Then, apply SVD of P: $P = U\Sigma V^T$. Thus, H can be found by the last column of V. Note that in Line97, we apply normalization to get an optimal solution.

```
87 H_list = []
88 for i, fname in enumerate(images):
89     print(i)
90     P = []
91     for j in range(corner_x * corner_y):
92         append_p(P, objpoints[i][j], imgpoints[i][j])
93     P = np.array(P)
94     print(P.shape)
95     u, s, vh = np.linalg.svd(P)
96     H = vh[-1]
97     H = H / H[-1]
98     H = np.array([H[0], H[1], H[2]], [H[3], H[4], H[5]], [H[6], H[7], H[8]])
99     H_list.append(H)
100    print(H)
```

```

68 def append_p(P, obj, img):
69     P.append([
70         obj[0], obj[1], 1, 0, 0, 0, -img[0] * obj[0], -img[0] * obj[1], -img[0]
71     ])
72     P.append([
73         0, 0, 0, obj[0], obj[1], 1, -img[1] * obj[0], -img[1] * obj[1], -img[1]
74     ])

```

3. Once we get H , and we know that H is composed of $K(r1, t2, t)$, we can conduct the equation as below:

$$\mathbf{h}_1^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{h}_2 = 0$$

$$\mathbf{h}_1^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{h}_1 = \mathbf{h}_2^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{h}_2$$

Now, we define B as:

$$\mathbf{B} := \mathbf{K}^{-\top} \mathbf{K}^{-1}$$

4. The next step is to build a matrix v , such that

$$v_{ij} = \begin{bmatrix} h_{i0} \cdot h_{j0} \\ h_{i0} \cdot h_{j1} + h_{i1} \cdot h_{j0} \\ h_{i1} \cdot h_{j1} \\ h_{i2} \cdot h_{j0} + h_{i0} \cdot h_{j2} \\ h_{i2} \cdot h_{j1} + h_{i1} \cdot h_{j2} \\ h_{i2} \cdot h_{j2} \end{bmatrix}$$

Therefore, using the dot production constraint for B , we can get

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22}) \end{bmatrix} \cdot b = V \cdot b = 0$$

where b is a representation of B as $(b_{11}, b_{12}, b_{13}, b_{22}, b_{23}, b_{33})$.

```

101 v12 = get_v(H[:, 0], H[:, 1])
102 v11 = get_v(H[:, 0], H[:, 0])
103 v22 = get_v(H[:, 1], H[:, 1])
104 V.append(v12)
105 V.append(v11 - v22)

```

```

77 def get_v(Hi, Hj):
78     return np.array([
79         Hi[0] * Hj[0], Hi[0] * Hj[1] + Hi[1] * Hj[0], Hi[1] * Hj[1],
80         Hi[2] * Hj[0] + Hi[0] * Hj[2], Hi[2] * Hj[1] + Hi[1] * Hj[2],
81         Hi[2] * Hj[2]
82     ])

```

5. Again, the solution is computed using SVG of V which yields us b , and by extension B . Note that in Line121, we also apply normalization to make sure correction.

```

110 u, s, vh = np.linalg.svd(V)
111 # print(V.T.dot(V))
112 # u, s, vh = np.linalg.svd(V.T.dot(V))
113 print("s, vh")
114 print(s)
115 print(vh)
116 b = vh[-1]
117 print("b")
118 print(b)
119 B = np.array([[b[0], b[1], b[3]], [b[1], b[2], b[4]], [b[3], b[4], b[5]]])
120
121 B = B / B[-1, -1]

```

6. Once we get B , we apply Cholesky factorization to find K .

$$B := K^{-T} K^{-1}$$

```

126 K_inv_T = np.linalg.cholesky(B)
127 K = np.linalg.inv(K_inv_T).T
128 K_inv = K_inv_T.T
129 print("K")
130 print(K)

```

7. Now, we can easily find r_1, r_2, r_3, t .

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

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

$$r_3 = r_1 \times r_2$$

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

$$\lambda = 1 / \|K^{-1} h_1\|$$

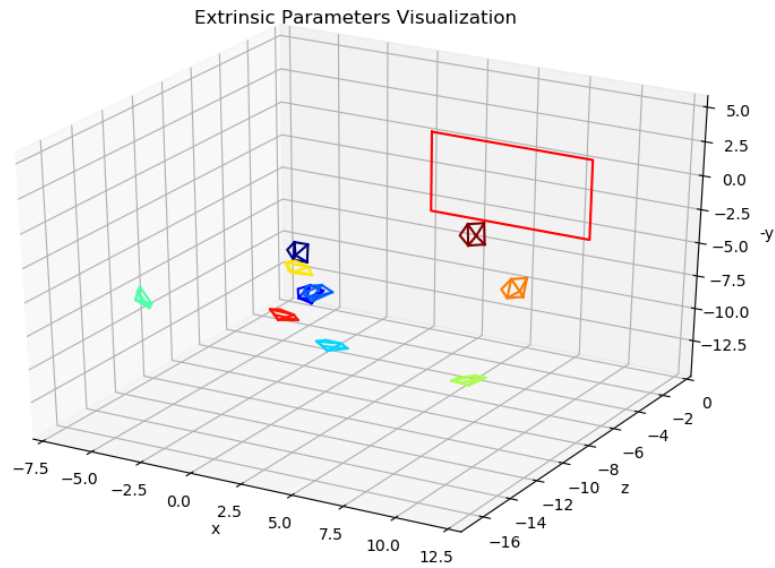
```

132 extrinsics = []
133 for i, fname in enumerate(images):
134     h1 = H_list[i][:, 0]
135     h2 = H_list[i][:, 1]
136     h3 = H_list[i][:, 2]
137     lam = 1 / np.linalg.norm(K_inv.dot(h1))
138     r1 = lam * K_inv.dot(h1)
139     r2 = lam * K_inv.dot(h2)
140     r3 = np.cross(r1, r2)
141     t = lam * K_inv.dot(h3)
142     extrinsic = np.zeros((3, 4), np.float32)
143     extrinsic[:, 0] = r1
144     extrinsic[:, 1] = r2
145     extrinsic[:, 2] = r3
146     extrinsic[:, 3] = t
147     print(r1)
148     print(r2)
149     print(r3)
150     print(t)
151     print(extrinsic)
152     extrinsics.append(extrinsic)

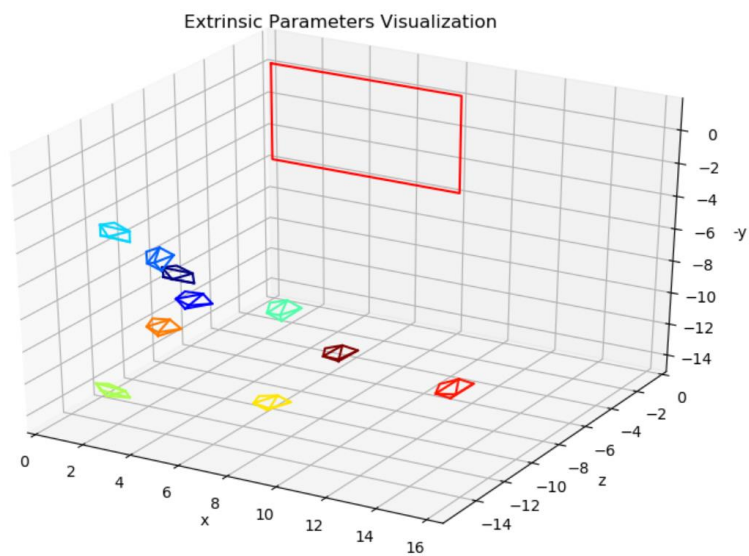
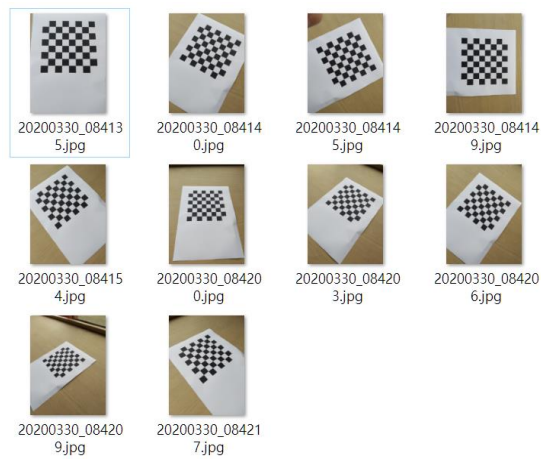
```

C. Experimental Result

1. Result of provided data:



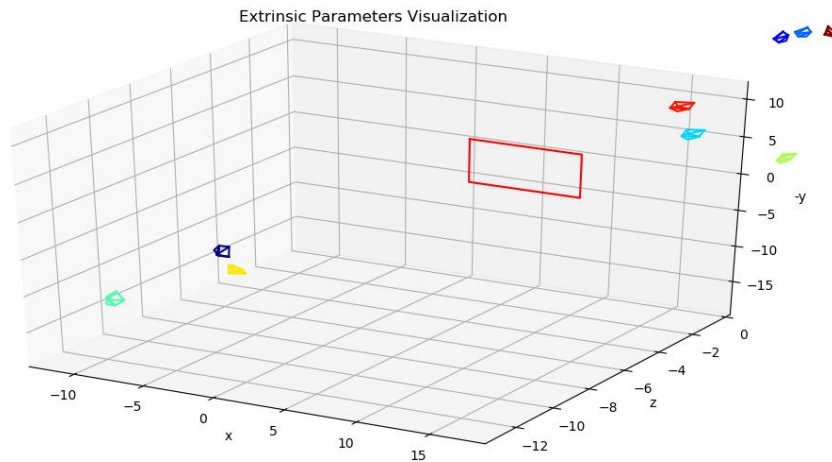
2. Result of our own data:



D. Discussion

1. The Normalization to keep accuracy:

Originally, we didn't add normalization terms in our code. And the result of provided data looked like this:

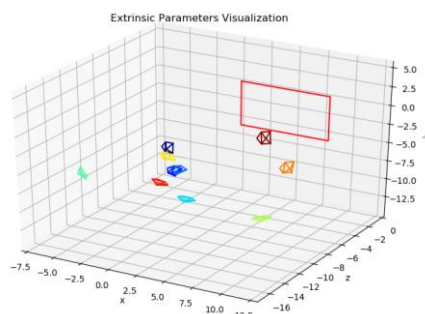


The result was so weird that we googled for it and found out that the only part our code missing was the normalization term. After we added it, the result looked correct.

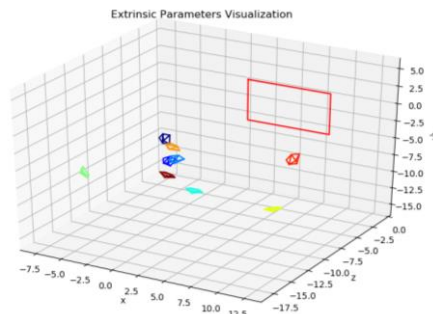
2. How many images do we need to achieve stability?

In class, professor mentioned that in theory, we need at least 2 images to get enough information of homography. But in practice, we need more images to achieve better stability. Therefore, we do some experiments to check how many images in our case can we achieve stability. The results are shown below:

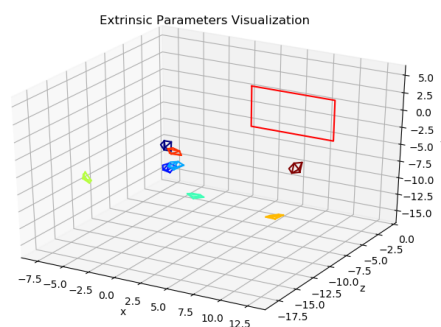
(M = 10)



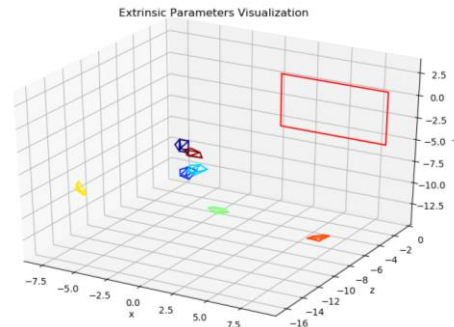
(M = 9)



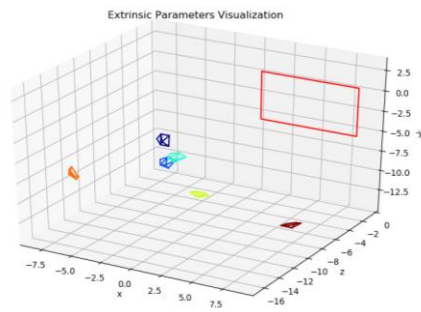
(M = 8)



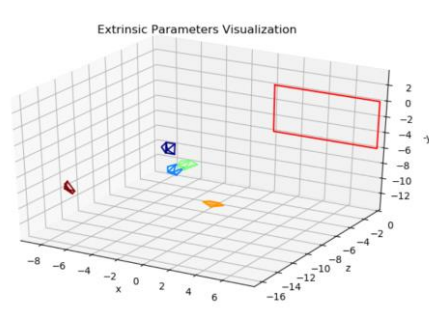
(M = 7)



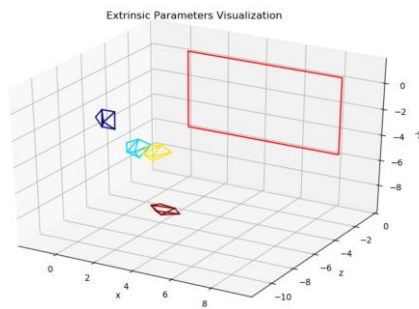
(M = 6)



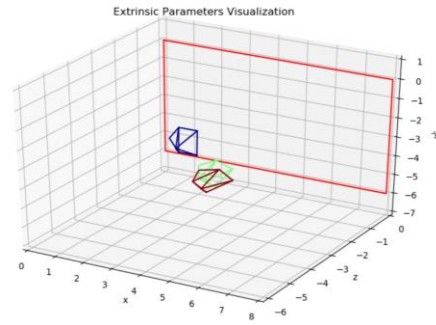
(M = 5)



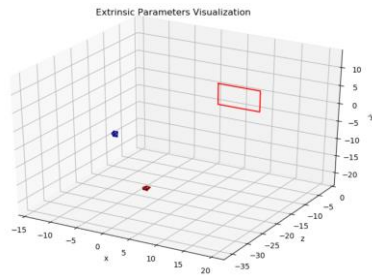
(M = 4)



(M = 3)



(M = 2)



From the experiments results, we find out that when $M \geq 3$, the result start to be stable.

E. Conclusion

1. Normalization on H and B truly affects the accuracy.
2. In our case, at least 3 images are needed to get enough information.

F. Work Assignment Plan

1. 黃聖祺-Coding
2. 林宮白-Coding
3. 楊上萱-Report