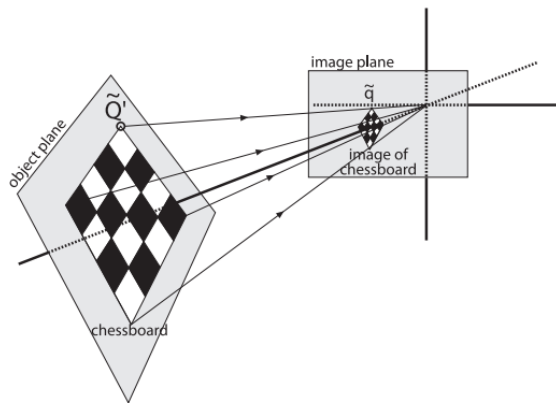


# Computer Vision - HW1 Camera Calibration

## 1. Introduction

In this assignment, we practice how to implement camera calibration. We take photos from different angles of the 2D chessboard. Derive intrinsic matrix and extrinsic matrices from these pictures.

## 2. Implementation procedure



The chessboard corners of images are found by “findChessboardCorners” function from cv2 as the points on the image plane denoted as  $x_2 y_2$ , and they are mapping to  $x_1 y_1$  as the points on the object plane. The corners of the chessboard on the object plans are defined as the same coordinate system(  $[0, 0, 0]$ ,  $[0, 1, 0]$ ... $[6, 6, 0]$ ).

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} * \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = K * [R|T] * W$$

K : Intrinsic Parameter

[R|T] : Extrinsic parameters

W : World coordinate

### -----Step 1-----

Due to chessboard is flatten , so the z-axis coordinate would be 0 , and

$$\begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

the extrinsic parameters would become

Then , we define a matrix  $H = K * [R|T]$  , so the equation could be

$$K * [R|T] * W = H * W \rightarrow \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\therefore x_2 = \frac{x_1 h_{11} + y_1 h_{12} + h_{13}}{x_1 h_{31} + y_1 h_{32} + 1}, y_2 = \frac{x_1 h_{21} + y_1 h_{22} + h_{23}}{x_1 h_{31} + y_1 h_{32} + 1}$$

Second , we choose array

$$P = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_2 x_1 & -x_2 y_1 & -x_2 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_2 x_1 & -y_2 y_1 & -y_2 \end{bmatrix}$$

$$\text{s.t } P * \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ \vdots \\ h_{33} \end{bmatrix} = P * h$$

The answer of the h is the eigenvector of  $P^T P$  associated with the smallest eigenvalue , so we could get h .

Proof :

Solve equation  $Ph = 0$  is equivalent to solve  $\min ||Ph|| \text{ s.t } ||h|| = 1$

By doing SVD ,  $||Ph|| = ||U\Sigma V^T h||$

$\because$  U is orthonormal matrix  $||U\Sigma V^T h|| = ||\Sigma V^T h||$

Let  $y = V^T h \therefore$  the equation would be  $\min ||\Sigma y|| \text{ s.t } ||y|| = 1$

Because  $\Sigma$  is diagonal matrix and the element in this array would sorted

So the optimal answer happened when  $y = (0,0,0, \dots, 1)^T$

And h is the vector in last column of V . (V is the eigenvector of  $P^T P$ )

(Eigenvector is perpendicular to each other)

## -----Step 2-----

$$\because H = [h_1 \ h_2 \ h_3] = s * K * [r_1 \ r_2 \ t]$$

$$\therefore r_1 = \frac{1}{s}K^{-1}h_1, r_2 = \frac{1}{s}K^{-1}h_2, t = \frac{1}{s}K^{-1}h_3$$

$r_1 \ r_2$  are rotation matrix and orthonormal basis.

$$\therefore r_1^T r_2 = 0 \text{ and } ||r_1|| = ||r_2|| = 1$$

$$\Rightarrow h_1^T K^{-T} K^{-1} h_2 = 0 \text{ and } h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2$$

Then we define matrix

$$B = K^{-T} K^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \text{ ( diagonal matrix )}$$

Let

$$h_i \text{ be the } i - \text{column in } H, h_i = \begin{bmatrix} h_{1i} \\ h_{2i} \\ h_{3i} \end{bmatrix}, b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$$

Recall that we have

$$h_1^T K^{-T} K^{-1} h_2 = 0 \Rightarrow h_i^T B h_j = v_{ij}^T b = 0$$

By solving the equation , we can get

$$v_{ij} = \begin{bmatrix} h_{1i} h_{1j} \\ h_{2i} h_{1j} + h_{1i} h_{2j} \\ h_{2i} h_{2j} \\ h_{3i} h_{1j} + h_{1i} h_{3j} \\ h_{2i} h_{3j} + h_{3i} h_{2j} \\ h_{3i} h_{3j} \end{bmatrix}$$

$$\begin{cases} h_1^T K^{-T} K^{-1} h_2 = 0 \\ h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \end{cases} \Rightarrow \begin{cases} v_{12}^T b = 0 \\ v_{11} b = v_{22} b \end{cases} \Rightarrow \begin{cases} v_{12}^T b = 0 \\ v_{11} b - v_{22} b = 0 \end{cases}$$

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

, where V is 2n\*6 .  $h_1, h_2, h_3 = H.T$

$$K\_inv = \text{np.linalg.inv}(K)$$

```
s = 1/np.linalg.norm((np.dot(K_inv,h1)))
```

```
r1 = s * np.dot(K_inv , h1)
```

```
r2 = s * np.dot(K_inv , h2)
```

```
r3 = np.cross(r1,r2)
```

```
t = s * np.dot(K_inv , h3 )
```

```
Extrinsic_temp = np.zeros((3,3))
```

```
Extrinsic_temp[:,0] = r1
```

```
Extrinsic_temp[:,1] = r2
```

```
Extrinsic_temp[:,2] = r3
```

```
rotation_vector , _ = cv2.Rodrigues(Extrinsic_temp)
```

```
Extrinsic = np.concatenate((rotation_vector,t.reshape(-1,1)),axis = 0)
```

```
Extrinsic = Extrinsic.reshape(6)
```

Using the same method (SVD) to get vector b, and we can use b to reduction

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

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

By using cholesky factorization, we can get

$$G = K^{-T} \quad s, t \quad B = G * G^T$$

Finally ,we can get K from  $K^{-T}$  (Intrinsic Parameter).

### -----Step 3 -----

$$H = [h_1 \ h_2 \ h_3] = s * K * [r_1 \ r_2 \ r_3]$$

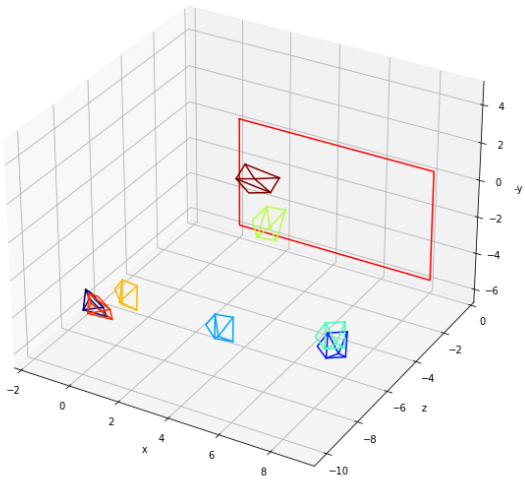
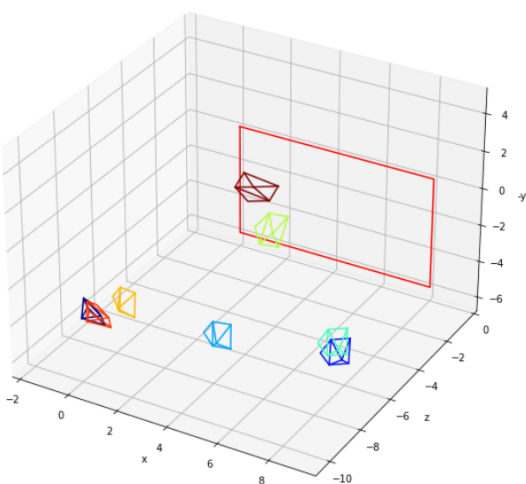
$$\therefore s = \frac{1}{\|K^{-1}h_1\|}$$

$$r_1 = \frac{1}{s} K^{-1} h_1, r_2 = \frac{1}{s} K^{-1} h_2, r_3 = r_1 * r_2, t = \frac{1}{s} K^{-1} h_3$$

From the equations above, we get Extrinsic parameters  $\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$ .

## 3. Experiment Result

- our own images

	cv2	our method
extrinsic visualization	<p>Extrinsic Parameters Visualization</p> 	<p>Extrinsic Parameters Visualization</p> 
intrinsic matrix K	<pre>[[ 3.04761700e+03  0.00000000e+00  1.53188937e+03]  [ 0.00000000e+00  3.05150952e+03  2.01890180e+03]  [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]</pre>	<pre>mtx [[ 3.12811746e+03  7.89253204e+00  1.53435934e+03]  [ 0.00000000e+00  3.13285494e+03  1.99525299e+03]  [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]</pre>

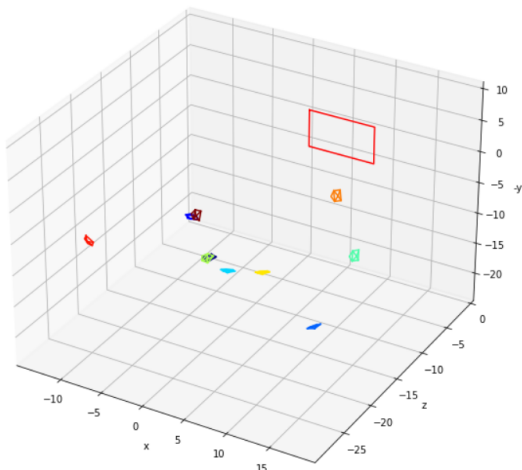
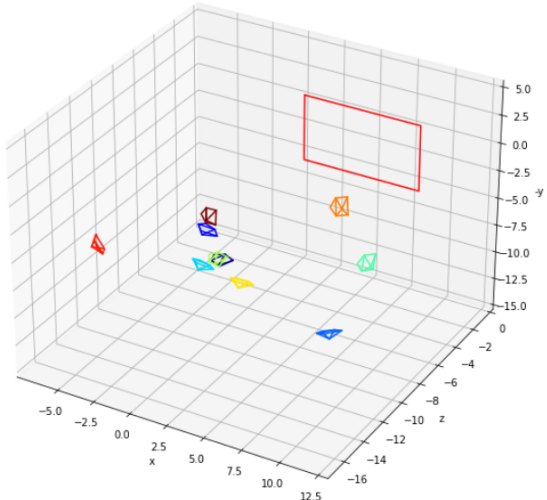
### extrinsic matrix of each images

cv2	<pre>[[ -2.65586446e-01 -4.74479647e-01 -7.55259458e-03 -2.27459045e+00   -3.64188619e+00  8.66071935e+00]  [ -4.70400433e-01  5.34364328e-01  1.48532834e+00  2.73823803e+00   -2.28364328e+00  1.27269278e+01]  [ -1.61231692e-01  1.69118506e-01  1.55745411e+00  2.91962379e+00   -3.09211547e+00  1.03312568e+01]  [ -4.29866273e-01  5.52104788e-01  1.51532697e+00  2.94624389e+00   -2.40460693e+00  1.23796646e+01]  [  5.73504724e-02  6.75402016e-01  1.41665484e+00  1.68783664e+00   -2.53329817e+00  1.03853455e+01]  [  9.24720092e-02 -5.02335695e-02  1.55113365e+00  2.75569183e+00   -2.26048834e+00  9.38975069e+00]  [ -1.76579176e-01 -3.45646037e-01 -2.48303080e-02 -2.34639137e+00   -3.36239896e+00  8.93972569e+00]  [ -2.81102922e-01  9.55669674e-01  2.75305037e+00  3.44023147e+00   8.99439020e-01  1.03117657e+01]]</pre>
-----	--

our method	<pre> extrinsic [ [-2.67016234e-01 -4.71028565e-01 -4.40460108e-03 -2.27755039e+00   -3.58614700e+00  8.79278942e+00] [ -4.79787483e-01  5.50292350e-01  1.48285149e+00  2.72050719e+00   -2.17121330e+00  1.28325374e+01] [ -1.58448157e-01  1.76526083e-01  1.55838115e+00  2.92393477e+00   -3.01549062e+00  1.04320621e+01] [ -4.36654198e-01  5.68128871e-01  1.51308360e+00  2.93325881e+00   -2.29807328e+00  1.24866621e+01] [  4.92811968e-02  6.91316689e-01  1.41309563e+00  1.68074695e+00   -2.44519278e+00  1.05158583e+01] [  9.27640910e-02 -4.34779702e-02  1.55217874e+00  2.75219768e+00   -2.18813556e+00  9.48540708e+00] [ -1.77831771e-01 -3.42452402e-01 -2.22326643e-02 -2.34586061e+00   -3.29562700e+00  9.05623624e+00] [ -2.83152839e-01  9.62434794e-01  2.75203354e+00  3.43125793e+00   9.79289260e-01  1.04462873e+01]] </pre>
------------	---

( the order of the vectors of extrinsic matrix are rotation vectors and translation vectors )

- images from TA

	cv2	our method
extrinsic visualization		
intrinsic matrix K	<pre> [[ [6.11157968e+03  0.00000000e+00  1.36059698e+02]   [0.00000000e+00  5.35968515e+03  3.23038992e+02]   [0.00000000e+00  0.00000000e+00  1.00000000e+00]] </pre>	<pre> [[ [ 3.40033699e+03 -3.52543663e+01  1.47568440e+03]   [ 0.00000000e+00  3.34935944e+03  1.40822870e+03]   [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]] </pre>

extrinsic matrix of each images

cv2	<pre>[[-0.22506028  0.3523952  0.76555587  5.37716497  2.68616652  24.91315266]  [-0.24880747  0.05355021  0.14969047  2.70828861  1.43525853  19.71279256]  [-0.91983821  0.55101699  0.51510065  5.65226706  4.10567935  26.81666292]  [-0.63141065  0.29443618  0.06385003  4.20937496  1.68039714  21.63063317]  [-0.48689102  0.99437317  1.40727945  8.23672805  6.33365631  27.57556039]  [-0.20130972  0.33663926  1.07889769  6.63990516  2.62110568  23.85283453]  [-0.70686201  0.39361796  0.19414352  4.30319981  0.46999575  20.7136146 ]  [-0.19071516  1.06987434  1.59641728  8.51384893  2.4168957  20.53732639]  [-1.1563765  -0.4008397  -0.95264061  1.82456542  10.33748323  25.21802757]  [ 0.05821625  0.29413644  1.53013472  7.23774552  3.40493294  18.30962885]]</pre>
our method	<pre>[[-1.65500484e-01  7.62379918e-02  7.65082004e-01 -3.90640381e-01  -2.29354792e+00  1.50387073e+01]  [-1.48666266e-01 -1.44240108e-01  1.29022578e-01 -1.85202510e+00  -2.48724496e+00  1.16916296e+01]  [-7.97917056e-01  3.25813612e-01  4.55799915e-01 -5.58238104e-01  -1.24858916e+00  1.74106884e+01]  [-4.59216044e-01 -9.85870564e-02  2.99060383e-02 -7.53312586e-01  -2.69910545e+00  1.30981243e+01]  [-5.74262118e-01  7.00736274e-01  1.38257149e+00  1.68667387e+00  7.81464390e-01  1.80665268e+01]  [-1.71178942e-01  5.69162291e-02  1.07480213e+00  1.08607747e+00  -2.17294644e+00  1.46248918e+01]  [-5.47954436e-01  1.23339448e-02  1.50528909e-01 -4.63453035e-01  -3.82657123e+00  1.33249112e+01]  [-2.10209800e-01  7.15533667e-01  1.63054297e+00  3.71642772e+00  -1.87044393e+00  1.35969538e+01]  [-9.68519996e-01 -5.58888026e-01 -1.06417053e+00 -4.07213589e+00  4.81449331e+00  1.44851224e+01]  [-2.07081072e-02  6.77475165e-02  1.55357715e+00  2.86666666e+00  -3.92322647e-01  1.14575934e+01]]</pre>

( the order of the vectors of extrinsic matrix are rotation vectors and translation vectors )

#### 4. Discussion

- 4.1. When we implement our method with the images TAs provide, the program occurs error due to non positive definite B which is not able to apply on cholesky factorization. The vector  $b$  which makes up the B matrix is derived from vector  $V$  by SVD. When we apply the SVD function from the library we could get a set of “right singular vectors” all satisfying this  $V$  and the minimal one is chosen, However, we are not sure if it could make B a positive definite matrix or not. As the result, before executing cholesky factorization, we need to know the

eigenvalue of  $B$  we decide are positive or negative, if it is negative, we multiply  $B$  with  $-1$  to ensure that it becomes a positive definite matrix. As we know  $-B$  is still “right singular vectors” of  $V$ . Finally, it works well on both our images and TA's.

- 4.2. Noticing that our results and cv2 are different, we found out that cv2 always set the skew vector of  $K$  as zero. However, in our method, we derived all the values in  $K$  from the procedure above, and in other words, we don't limitate the skew vector. Nowadays, cameras are well designed and the skew effect is getting smaller. In cv2, it ignores the skew vector to do camera calibration which makes the different result from ours.
- 4.3. At the beginning, we found an obvious difference between the results of the extrinsic visualization by cv2 and our method. After we do normalization on both homography matrices and the intrinsic matrix by dividing each value by the value of the last block of the matrix. The results of extrinsic visualization get more similar between two methods.

## 5. Conclusion

We can mainly divide our work into 3 steps for accomplishing the 2D camera calibration task. First, we use the homography equation, composed of multiplication of intrinsic and extrinsic matrix, and SVD to get the homography matrix  $H$ . Second, we can find the intrinsic matrix  $K$  by using homography matrix  $H$  and some linear algebra theorems (Orthonormal basis, SVD and Cholesky factorization). Eventually, with the homography matrix  $H$  and intrinsic matrix  $K$ , we can easily find the extrinsic matrices by inverse matrix.

In conclusion, it's not only interesting to implement the 2D camera calibration on our own, but we also get to learn how the 3D world coordinate is actually transformed into 2D image coordinate.