# CV HW1 - Camera Calibration

**Group 18**

## Introduciton

We try to implement camera calibration without the cv2 function. We use the data already given by TA and take pictures from different angles of 2D chess board as our own data. Then after the practice, we get the intrinsic matrix and extrinsic matrices.
Intrinsic matrix is composed by camera's focal length (*f*), skew effect, scale and offset of x & y coordinate. Extrinsic is composed by rotation (r) and translation (t). In the given data, they all share the same intrinsic matrix due to the same camera. Yet, different images have different extrinsic metrics.

## Implementation Procedure

### 1. Get the Homography Matrix

For 2D camera calibration, we use 2D chess board pattern so that we can set all the corner points in world coordinate system on one plane. The z component of object points is 0. Now we have

$$\mathrm{p_i} = \begin{bmatrix} \mathrm{u} \\ v \\ 1 \end{bmatrix} \ \mathrm{P_i} = \begin{bmatrix} U \\ V \\ 1 \end{bmatrix}$$

and according to the relation between 2D space and 3D space

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

So the equation is

$$\mathrm{p_i} = HP_i \ \text{ where } \ \mathrm{H} = \begin{bmatrix} \mathrm{h1} \\ h2 \\ h3 \end{bmatrix}$$

h1, h2, h3, are the row of H. After rewriting the equation, for one point, we will get

$$\begin{cases} u(h3P) - h1P = 0 \\ v(h3P) - h2P = 0 \end{cases}$$

Now we can build the equation with all n points

$$\begin{bmatrix} U_1 & V_1 & 1 & 0 & 0 & 0 & -u_1U_1 & -u_1V_1 & -u_1*1 \\ 0 & 0 & 0 & U_1 & V_1 & 1 & -v_1U_1 & -v_1V_1 & -v_1*1 \\ & & & & \dots & & & & \\ U_n & V_n & 1 & 0 & 0 & 0 & -u_nU_n & -u_nV_n & -u_n*1 \\ 0 & 0 & 0 & U_n & V_n & 1 & -v_nU_n & -v_nV_n & -v_n*1 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = Ah = 0$$

$$\text{minimize } \|Ah\| \text{ , subject to } \|Ah\|^2 = 1 \text{ to avoid trivial solution}$$

We use singular value decomposition(SVD) method to solve it.

$$Ah = UDV^T$$

We set h equal to the last column of $V$ , in other words, the last row of $V^T$ , then reshape the vector h(1,9) to H(3,3).

## 2. Get the intrinsic matrix K

After we get the homography matrix H from each images, we can decompose H into intrinsic matrix and extrinsic matrix.

$$\mathrm{H} = (\mathrm{h1}, \mathrm{h2}, \mathrm{h3}) = \begin{bmatrix} f/s_x & 0 & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} = \mathrm{K}(\mathrm{r1}, \mathrm{r2}, \mathrm{t})$$

So we can get:

$$\mathrm{r1} = \mathrm{K}^{-1}\mathrm{h1},\ \mathrm{r2} = \mathrm{K}^{-1}\mathrm{h2}$$

Since r1, r2, r3 are orthonormal, we can derive:

$$\begin{cases} {r_1}^T r_2 = 0 \\ \|r_1\| = \|r_2\| = 1 \end{cases}$$

$$\begin{cases} {h_1}^T K^{-T} K^{-1} h_2 = 0 \\ {h_1}^T K^{-T} K^{-1} h_1 = {h_2}^T K^{-T} K^{-1} h_2 \end{cases}$$

To solve K, we define a positive definite B:

$$\mathrm{B} = K^{-T} K^{-1} = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_2 & b_4 & b_5 \\ b_3 & b_5 & b_6 \end{bmatrix}$$

Then we can get 2 equations for each Hi:

$$\begin{cases} \begin{bmatrix} h_{00} & h_{10} & h_{20} \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_2 & b_4 & b_5 \\ b_3 & b_5 & b_6 \end{bmatrix} \begin{bmatrix} h_{01} \\ h_{11} \\ h_{21} \end{bmatrix} = 0 \\ \begin{bmatrix} h_{00} & h_{10} & h_{20} \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_2 & b_4 & b_5 \\ b_3 & b_5 & b_6 \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{10} \\ h_{20} \end{bmatrix} - \begin{bmatrix} h_{01} & h_{11} & h_{21} \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_2 & b_4 & b_5 \\ b_3 & b_5 & b_6 \end{bmatrix} \begin{bmatrix} h_{01} \\ h_{11} \\ h_{21} \end{bmatrix} = 0 \end{cases}$$

Define

$$\mathrm{b} = (\mathrm{b11}, \mathrm{b12}, \mathrm{b13}, \mathrm{b22}, \mathrm{b23}, \mathrm{b33})$$

And now we just need to solve

$$\mathrm{Vb} = \begin{bmatrix} v_{1,1} \\ v_{1,2} \\ \vdots \\ v_{n,1} \\ v_{n,2} \end{bmatrix} \mathrm{b} = 0, \text{where n is the number of h.}$$

$$\begin{cases} v_{i,1} = [h_{00}h_{01}, h_{10}h_{01} + h_{00}h_{11}, h_{20}h_{01} + h_{00}h_{21}, h_{10}h_{11}, h_{20}h_{11} + h_{10}h_{21}, h_{20}h_{21}] \\ v_{i,2} = [h_{00}h_{00} - h_{01}h_{01}, 2h_{00}h_{10} - 2h_{01}h_{11}, 2h_{00}h_{20} + 2h_{01}h_{21}, h_{10}h_{10} - h_{11}h_{11}, \\ \qquad 2h_{10}h_{20} - 2h_{11}h_{21}, h_{20}h_{20} - h_{21}h_{21}] \end{cases}$$

Then we can use SVD decomposition to solve

$$\mathrm{b} = \underset{b}{\operatorname{argmin}}\, Vb$$

$$\mathrm{V} = \mathrm{UD}V_h$$

We choose the last column of Vh as our b(because it forms null space of V), then we can reconstruct $B$ by b.

Last, we can get $L$ by using Cholesky factorization: $B = LL^H$ ,

and we get intrinsic matrix $K = L^{-T}$ .

### 3. Get the extrinsic matrix

After we got intrinsic matrix K and homography matrix H, we can derive extrinsic matrix on each column(r1, r2, t), and combine them into a 3 by 3 matrix.

$$\mathrm{H} = (\mathrm{h1}, \mathrm{h2}, \mathrm{h3}) = K \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} = \mathrm{K}(\mathrm{r1}, \mathrm{r2}, \mathrm{t})$$

$$\begin{aligned} \mathbf{r}_1 &= \lambda \mathbf{K}^{-1} \mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda \mathbf{K}^{-1} \mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda \mathbf{K}^{-1} \mathbf{h}_3 \end{aligned}$$

Since we're dealing with 2D camera calibration, the global coordinate matrix should be[U, V, 0, 1] = [U, V, 1]

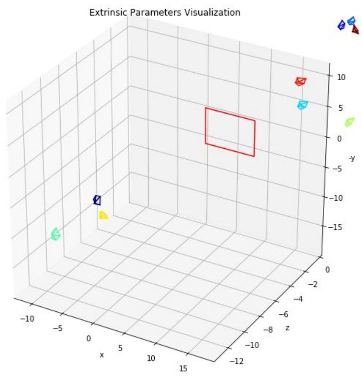
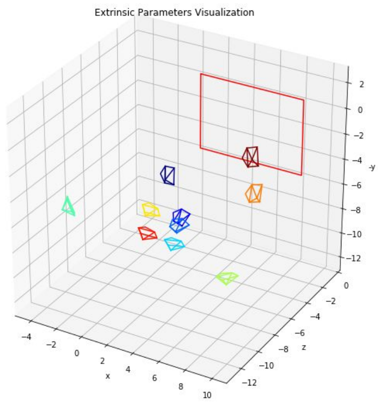
$$\begin{bmatrix} r_{11} & r_{12} & \cancel{r_{13}} & t_1 \\ r_{21} & r_{22} & \cancel{r_{23}} & t_2 \\ r_{31} & r_{32} & \cancel{r_{33}} & t_3 \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}$$

Finally, we have all values needed for 2D calibration. (Intrinsic matrix K, Extrinsic matrix RT, and Global coordinate)

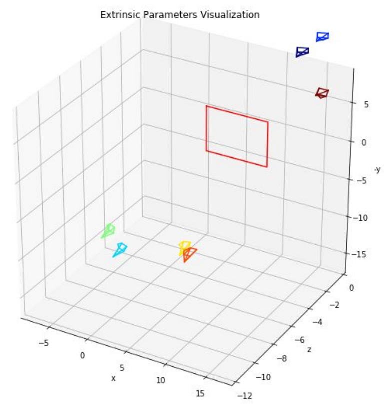
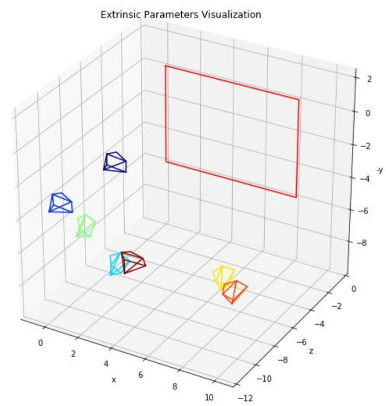
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathrm{K} \begin{bmatrix} \mathrm{R} & & t \\ 0 & 0 & 1 \end{bmatrix}_{3\times 3} \begin{bmatrix} U \\ V \\ 1 \end{bmatrix}$$

## Experimental Result

- **Data given by TA**

<table>
<tr><th></th><th>Our method</th><th>cv2.calibrateCamera</th></tr>
<tr><td>Image</td><td>
</td><td>
</td></tr>
<tr><td>Matrix K</td><td>$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 3848.06 & 94.9804 & 1803.51 \\ 1 & 0 & 3741.5 & 929.213 \\ 2 & 0 & 0 & 1 \end{array}$</td><td>$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 2701.93 & 0 & 1538.21 \\ 1 & 0 & 2738.09 & 1960.13 \\ 2 & 0 & 0 & 1 \end{array}$</td></tr>
<tr><td>Matrix B</td><td>$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 5.29408e-08 & -1.34394e-09 & -9.42304e-05 \\ 1 & -1.34394e-09 & 5.60336e-08 & -4.96433e-05 \\ 2 & -9.42304e-05 & -4.96433e-05 & 1 \end{array}$</td><td></td></tr>
<tr><td>H of first picture</td><td>$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & -0.00173318 & -0.108261 & 0.864022 \\ 1 & 0.105727 & 0.00286299 & 0.480162 \\ 2 & -1.83867e-06 & 9.59024e-07 & 0.000371199 \end{array}$</td><td></td></tr>
</table>

- **Our own data**

<table>
<tr><th></th><th>Our method</th><th>cv2.calibrateCamera</th></tr>
<tr><td>Image</td><td>
</td><td>
</td></tr>
<tr><td>Matrix K</td><td>
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th></tr>
<tr><td>0</td><td>1694.79</td><td>6.64243</td><td>789.471</td></tr>
<tr><td>1</td><td>0</td><td>1698.88</td><td>986.127</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>1</td></tr>
</table>
</td><td>
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th></tr>
<tr><td>0</td><td>1596.91</td><td>0</td><td>781.42</td></tr>
<tr><td>1</td><td>0</td><td>1600.01</td><td>1010.54</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Matrix B</td><td>
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th></tr>
<tr><td>0</td><td>2.24352e-07</td><td>-8.77191e-10</td><td>-0.000176255</td></tr>
<tr><td>1</td><td>-8.77191e-10</td><td>2.23276e-07</td><td>-0.000219486</td></tr>
<tr><td>2</td><td>-0.000176255</td><td>-0.000219486</td><td>1</td></tr>
</table>
</td><td></td></tr>
<tr><td>H of first picture</td><td>
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th></tr>
<tr><td>0</td><td>-0.299076</td><td>-0.00355159</td><td>-0.339439</td></tr>
<tr><td>1</td><td>-0.0274448</td><td>-0.290087</td><td>-0.842862</td></tr>
<tr><td>2</td><td>-2.61052e-05</td><td>-7.89889e-06</td><td>-0.00145027</td></tr>
</table>
</td><td></td></tr>
</table>

## Extrinsic matrix (Ours)

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0.246175 | 0.0757825 | -3.12122 | 2.84869 | 2.07787 | -8.71727 |
| 1 | 0.243775 | -0.0875567 | -3.09893 | 2.26114 | 2.28562 | -12.6486 |
| 2 | -0.178405 | -0.114318 | 1.01365 | 1.19093 | -4.70441 | 13.0638 |
| 3 | -0.118455 | -0.152885 | 1.17144 | 1.7502 | -2.74754 | 12.2269 |
| 4 | -0.414272 | 0.117337 | 1.21261 | 1.5598 | -5.96723 | 13.5417 |
| 5 | -0.359138 | 0.240516 | 0.916392 | 0.46861 | -6.78387 | 14.0818 |
| 6 | -0.02429 | -0.655046 | -3.05822 | 3.03618 | 2.67058 | -12.2674 |

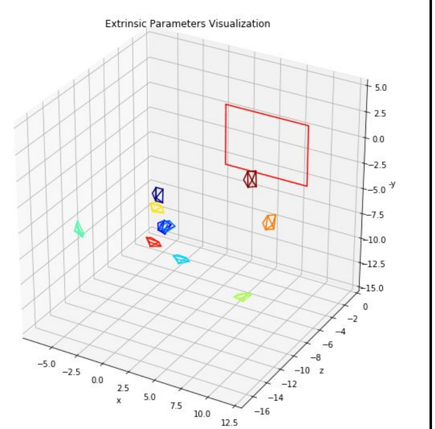
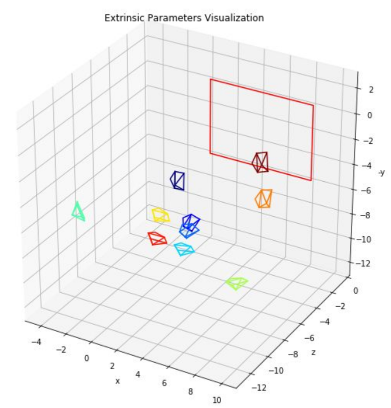
## Extrinsic matrix (cv)

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0.0419315 | -0.154263 | 0.00672387 | -2.81415 | -2.20247 | 8.33012 |
| 1 | -0.0495162 | -0.162107 | 0.0291525 | -2.2071 | -2.46508 | 11.988 |
| 2 | -0.192389 | -0.105229 | 1.01033 | 1.23451 | -4.89121 | 12.5399 |
| 3 | -0.112862 | -0.153066 | 1.16854 | 1.7961 | -2.91931 | 11.6367 |
| 4 | -0.411958 | 0.113217 | 1.20998 | 1.59741 | -6.14645 | 12.9952 |
| 5 | -0.360945 | 0.231768 | 0.915401 | 0.504837 | -6.97288 | 13.527 |
| 6 | -0.404506 | 0.012341 | 0.0133723 | -2.99065 | -2.84771 | 11.711 |

## Discussion

In the visualization image of TAs' data, we can observe that some extrinsic matrix of specific data are not matched. We also can see both intrinsic matrix are different. The problem also appear in the practice of our own data, so we check the extrinsic matrix of each input picture(above table). We observe that picture0, picture1, picture6 are different in two methods.
After some discussions, we find out that the problem of out original code is we didn't normalize our homography matrices, that is, divide each homography matrix by the value of last block. This will lead to some miscorrect answers because the value in the extrinsic matrix is correlated to the homography matrix, so if some of the homography matrix is below 0, then our extrinsic matrix will also be minus.So after the normalization, we solve most of the problem that our result is largely different to openCV's.

- **Data given by TA**

<table>
<tr><th></th><th>Our method</th><th>cv2.calibrateCamera</th></tr>
<tr><td>Image</td><td>
</td><td>
</td></tr>
<tr><td>Matrix K</td><td><table><tr><th></th><th>0</th><th>1</th><th>2</th></tr><tr><td>0</td><td>3400.34</td><td>-35.2543</td><td>1475.68</td></tr><tr><td>1</td><td>0</td><td>3349.36</td><td>1408.23</td></tr><tr><td>2</td><td>0</td><td>0</td><td>1</td></tr></table></td><td><table><tr><th></th><th>0</th><th>1</th><th>2</th></tr><tr><td>0</td><td>2701.93</td><td>0</td><td>1538.21</td></tr><tr><td>1</td><td>0</td><td>2738.09</td><td>1960.13</td></tr><tr><td>2</td><td>0</td><td>0</td><td>1</td></tr></table></td></tr>
</table>

## Extrinsic matrix (Ours) - Adapted

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | -0.0207081 | 0.0677475 | 1.55358 | 2.86667 | -0.392323 | 11.4576 |
| 1 | -0.171179 | 0.0569162 | 1.0748 | 1.08608 | -2.17295 | 14.6249 |
| 2 | -0.1655 | 0.076238 | 0.765082 | -0.39064 | -2.29355 | 15.0387 |
| 3 | -0.547954 | 0.0123339 | 0.150529 | -0.463453 | -3.82657 | 13.3249 |
| 4 | -0.96852 | -0.558888 | -1.06417 | -4.07214 | 4.81449 | 14.4851 |
| 5 | -0.797917 | 0.325814 | 0.4558 | -0.558238 | -1.24859 | 17.4107 |
| 6 | -0.148666 | -0.14424 | 0.129023 | -1.85203 | -2.48724 | 11.6916 |
| 7 | -0.574262 | 0.700736 | 1.38257 | 1.68667 | 0.781464 | 18.0665 |
| 8 | -0.459216 | -0.0985871 | 0.029906 | -0.753313 | -2.69911 | 13.0981 |
| 9 | -0.21021 | 0.715534 | 1.63054 | 3.71643 | -1.87044 | 13.597 |

## Extrinsic matrix (cv) - Adapted

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | -0.0742517 | 0.122381 | 1.55744 | 2.67729 | -2.24429 | 9.3694 |
| 1 | -0.195392 | 0.0946248 | 1.07294 | 0.841099 | -4.45105 | 11.9965 |
| 2 | -0.199553 | 0.091138 | 0.766483 | -0.633143 | -4.64136 | 12.3797 |
| 3 | -0.416396 | 0.0772206 | 0.158608 | -0.710885 | -5.68885 | 10.7043 |
| 4 | -0.811644 | -0.484697 | -1.10082 | -4.40813 | 2.39703 | 11.4642 |
| 5 | -0.681255 | 0.250056 | 0.469762 | -0.844528 | -4.03625 | 14.3471 |
| 6 | -0.177043 | -0.0955862 | 0.122653 | -2.0564 | -4.31872 | 9.73177 |
| 7 | -0.482413 | 0.575357 | 1.41393 | 1.34146 | -2.13339 | 14.5061 |
| 8 | -0.364156 | -0.00727122 | 0.0285866 | -1.02352 | -4.7072 | 10.7871 |
| 9 | -0.204662 | 0.610053 | 1.66984 | 3.39957 | -3.86711 | 10.7895 |

## Conclusion

We can divide 2D camera calibration into three steps in our work. Firstly, we use homography equation and SVD to solve out the H (homography matrix). Secondly, with H and linear algebra theorem (Orthonormal, SVD and Cholesky factorization), we can get intrinsic matrix K. Last but not least, with homography matrix and intrinsic matrix, we can solve extrinsic matrices by inverse matrix.
Although there’s an issue about the homography matrices, resulting in some of the cameras appear in the opposite side. Thankfully, we found the problem and solve it by normalization.

## Work assignment plan between team members

We finish this homework together.

## Additional references


1. How to compute homography matrix H from corresponding points
2. Homography in computer vision explained