HW 1 report

Group 5

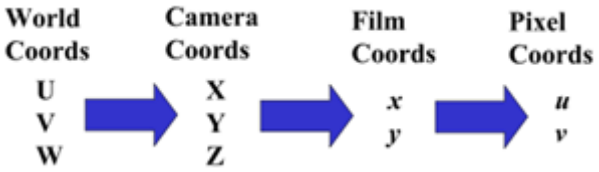
0516222 許芳瑀 0516209 呂淇安 0516326 陳廷達

1. Introduction

Camera calibration is the process of estimating the parameters of a camera that produced images or video. By using these parameters, we can correct distortion, measure the size of an object in world units, or determine the location of the camera in the scene. We use this technology in applications such as machine vision (for detecting and measuring objects) or navigation systems, and 3-D scene reconstruction.

2. Implementation

(a) figure out the Hi of each images



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f/s_x & s_k & o_x & 0 \\ 0 & f/s_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

$$\mathbf{x} = \underbrace{\mathbf{K}}_{\text{intrinsic matrix}} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}}_{\text{extrinsic matrix}} \mathbf{X}$$

We can represent K[R t] with H and get two equations from 1 point (u,v,1)

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

$$u.(h_{20}.X + h_{21}.Y + h_{22}) - (h_{00}.X + h_{01}.Y + h_{02}) = 0$$

$$v.(h_{20}.X + h_{21}.Y + h_{22}) - (h_{10}.X + h_{11}.Y + h_{12}) = 0$$

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

Then we write the equations as multiplecation of two matrixs and MH=0.

Since we have known M, we can get H by using SVD, below is the code how we get H.

```
## get H
###
H = np.zeros((len(images), 9))

for iter_H in range(len(objpoints)):
    Obj_Point = objpoints[iter_H]
    Img_point = imgpoints[iter_H]
    M = np.zeros((2 * len(Obj_Point), 9), dtype = np.float64)

    Img_point = Img_point.reshape(49, 2)
    Img_point = np.concatenate((Img_point, np.ones(shape = (49, 1))), axis = 1)

    for iter_M in xrange(len(Obj_Point)):
        x = Obj_Point[iter_M, 0]
        y = Obj_Point[iter_M, 1]
        u = Img_point[iter_M, 0]
        v = Img_point[iter_M, 1]

        M[iter_M * 2] = np.array([-x, -y, -1, 0, 0, 0, x*u, y*u, u])
        M[iter_M * 2 + 1] = np.array([0, 0, 0, -x, -y, -1, x*v, y*v, v])

    U_H, Sigma_H, Vt_H = np.linalg.svd(M)
    index_Min = np.argmin(Sigma_H)

    ##Norm it
    for iter_Vt in range(9):
        Vt_H[index_Min, iter_Vt] = Vt_H[index_Min, iter_Vt] / Vt_H[index_Min, 8]

    H[iter_H] = Vt_H[index_Min]


H = np.reshape(H, (len(images), 3, 3))
```

(b)Use Hi to find B

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

$$v_{ij} = \begin{bmatrix} h_{i0}.h_{j0} \\ h_{i0}.h_{j1} + h_{i1}.h_{j0} \\ h_{i1}.h_{j1} \\ h_{i2}.h_{j0} + h_{i0}.h_{j2} \\ h_{i2}.h_{j1} + h_{i1}.h_{j}2 \\ h_{i}2.h_{j2} \end{bmatrix}$$

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

After we have H, we can find b which satisfies the equation Vb = 0. Again, V is known, we can get b by doing SVD calculation.

Here is the code to get b.

```
## get B
MagicV = np.zeros([2 * len(H), 6])
index = 0
for iter_H in H:
    MagicV[index] = Magic(0, 1, iter_H)
    MagicV[index + 1] = Magic(0, 0, iter_H) - Magic(1, 1, iter_H)
    index = index + 2

U_B, Sigma_B, VT_B = np.linalg.svd(MagicV)
index_Min_B = np.argmin(Sigma_B)
b = VT_B[index_Min_B]
print("b; ", b)
B = np.array([[b[0],b[1],b[3]],
              [b[1],b[2],b[4]],
              [b[3],b[4],b[5]]])

Oy = (b[1] * b[3] - b[0] * b[4]) / (b[0] * b[2] - b[1] * b[1])

if b[0] == 0:
    b[0] = 1.0
Lda_B = b[5] - (b[3] * b[3] + Oy * (b[1] * b[3] - b[0] * b[4])) / b[0]
B = B / Lda_B
```

(c) calculate intrinsic matrix K from B by using Cholesky factorization

```
## Calculate K
K = np.linalg.inv(np.linalg.cholesky(B).transpose())
```

(d) get extrinsic matrix [R|t] for each images by K and H

Now we have everything we need, so we get extrinsic matrix [R|t] for each images by K and H

$$\mathbf{r}_1 = \lambda \mathbf{K}^{-1}\mathbf{h}_1$$
$$\mathbf{r}_2 = \lambda \mathbf{K}^{-1}\mathbf{h}_2$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$
$$\mathbf{t} = \lambda \mathbf{K}^{-1}\mathbf{h}_3$$
$$\lambda = 1/||\mathbf{K}^{-1}\mathbf{h}_1||$$

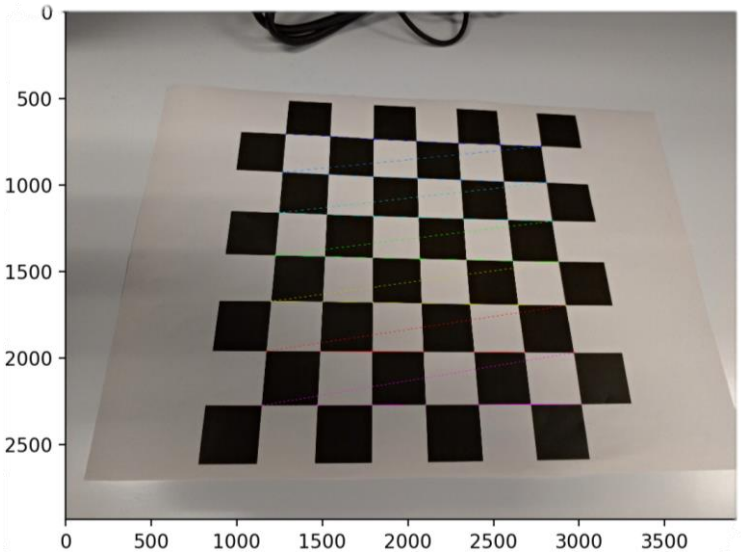
Here is the code.

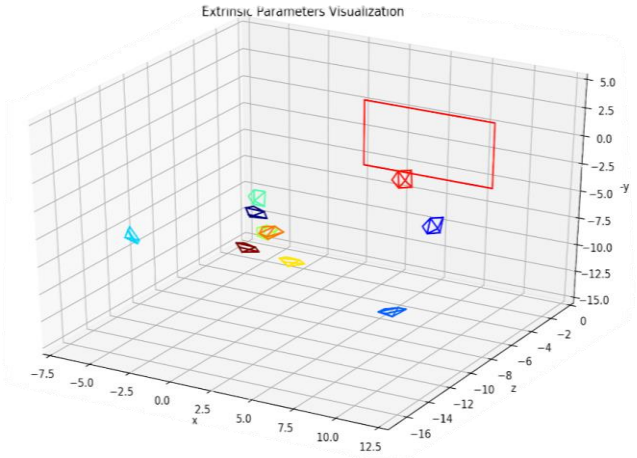
```
extrinsics = np.zeros((len(H), 3, 4))
K_inverse = np.linalg.inv(K)

for index in range(len(images)):
    h = H[index]
    h1 = h[:, 0]
    h2 = h[:, 1]
    h3 = h[:, 2]

    Lda_1 = 1 / np.linalg.norm(np.dot(K_inverse,h1), ord = 2)
    Lda_2 = 1 / np.linalg.norm(np.dot(K_inverse,h2), ord = 2)
    Lda_3 = (Lda_1 + Lda_2) / 2

    r1 = Lda_1 * (np.dot(K_inverse,h1))
    r2 = Lda_2 * (np.dot(K_inverse,h2))
    r3 = np.cross(r1, r2)
    t = Lda_3 * (np.dot(K_inverse,h3))
    extrinsics[index] = np.array([[r1[0],r2[0],r3[0],t[0]],
                                  [r1[1],r2[1],r3[1],t[1]],
                                  [r1[2],r2[2],r3[2],t[2]]])
```

3. Experimental result

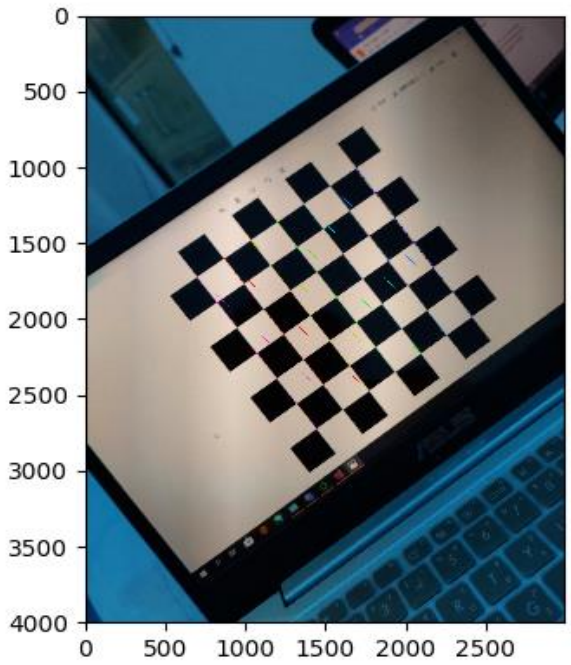
We run our code with two data set. One is comprised of 10 pictures from TA, named as 0000.png~0009.jpg. The other has 10 pictures took by our member's smartphone, named as img0.jpg-img9.jpg.

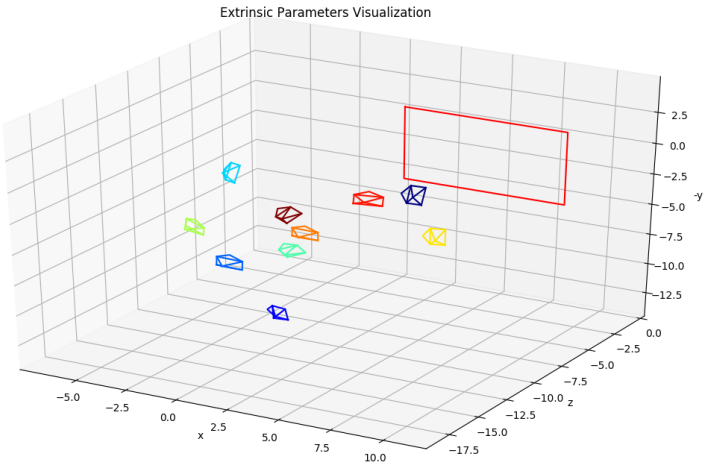
(a) pictures provided from TA





(b) pictures we made





4. Discussion

(a) find homography without built in function

We first find homography with building in function *cv2.findHomography*, but it turned out not working. So we refer to another method post online and find H by ourselves.

(b) get wrong extrinsic matrix

First, we use same lambda to calculate r1, r2, ,r3, t and extrinsic matrix. But we found that it doesn't work, it means that we cannot get the final pictures like examples. Thus we calculate 3 different lambda to get new r1, r2, ,r3, t and extrinsic matrix.

(c) use photos in different condition lead to error

After experiment, we found that if we use photos captured in different condition (like different scene and light) will let H matrics get error.

5. Conclusion

After this process we get the intrinsic and extrinsic matrix of our camera. We also found another way to calibrate camera, which use circular grid rather than chessboard.

Though our intrinsic matrix is a little different from TA's, it still works.

```
('our mtx :', array([[ 3.40033676e+03, -3.52543904e+01,  1.47568441e+03],
       [ 0.00000000e+00,  3.34935940e+03,  1.40822924e+03],
       [ 0.00000000e+00,  0.00000000e+00,  1.00000000e+00]]))
('mtx From TA: ', array([[2.70192979e+03, 0.00000000e+00, 1.53820705e+03],
       [0.00000000e+00, 2.73809172e+03, 1.96013469e+03],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00]]))
```

With an calibrated camera, we can do 3-D scene reconstruction or other further application.

6. Work plan

Slide reading：許芳瑀、陳廷達、呂淇安

Take course notes(with hackmd)：陳廷達、呂淇安

Reference finding：許芳瑀、陳廷達

Main coding : 陳廷達

Report : 許芳瑀、呂淇安

7. Reference

Computer Vision Ch2 PPT

Course Video

Demystifying Geometric Camera Calibration for Intrinsic Matrix. Retrieved from: https://kushalvyas.github.io/calib.html

相機內外參數校正與實作 - 張正友法 Retrieved from: https://abcd40404.github.io/2018/09/16/zhang-method/

相機標定（具體過程詳解）張正友、單應矩陣、B、R、T Retrieved from: https://www.itread01.com/content/1546857202.html