CS6643 Computer Vision

Assignment 1 - Geometric camera models and calibration

Name: Yingnan Li NetId: yl3651

Pratical Problem Report

Part 1 Experimental procedure

In this experimental, I use iphone 6s to take the picture and Matlab to perform camera calibration. The sensor of iphone 6s is sony IMX 315, more detail specification of this sensor will be shown in the following part. Then I printed two checkboards and choosed 22 points from them. After that I measured world coordinates of those 22 points and marked them on the checkboard, so that I can use those coordinates in the next step. Next, I attached two checkboards into a corner of my room, so that they are perpendicular. See figure 1.

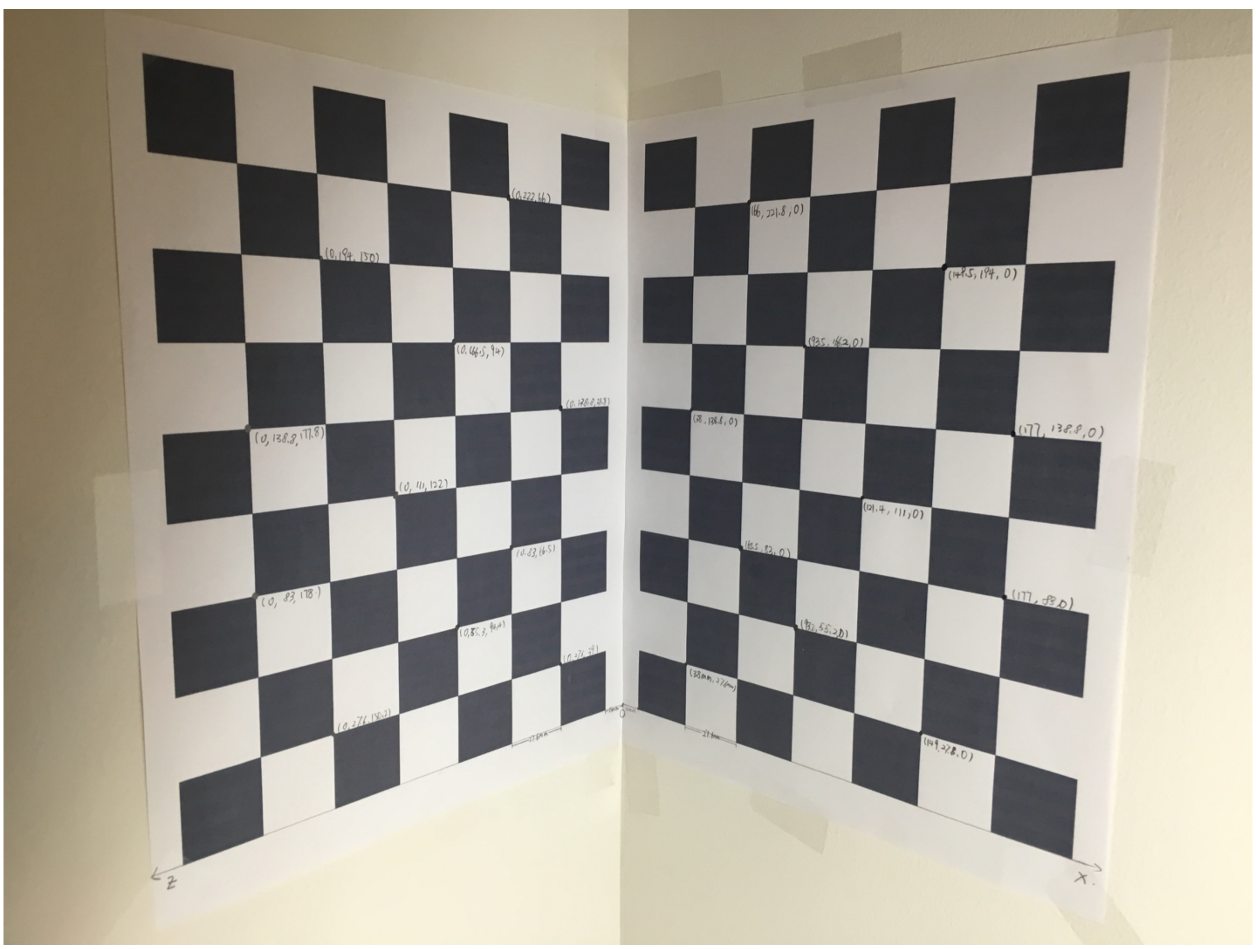

Figure 1, the checkboard I use to do camera calibration

For the calibration part, firstly, I use ginput() function get the pixel coordinates of each point and store x coordinates into **x** matrix, y coordinates into **y** matrix. Then I store the world coordinates of each point into **pw** matrix. See figure 2, figure 3, figure 4.

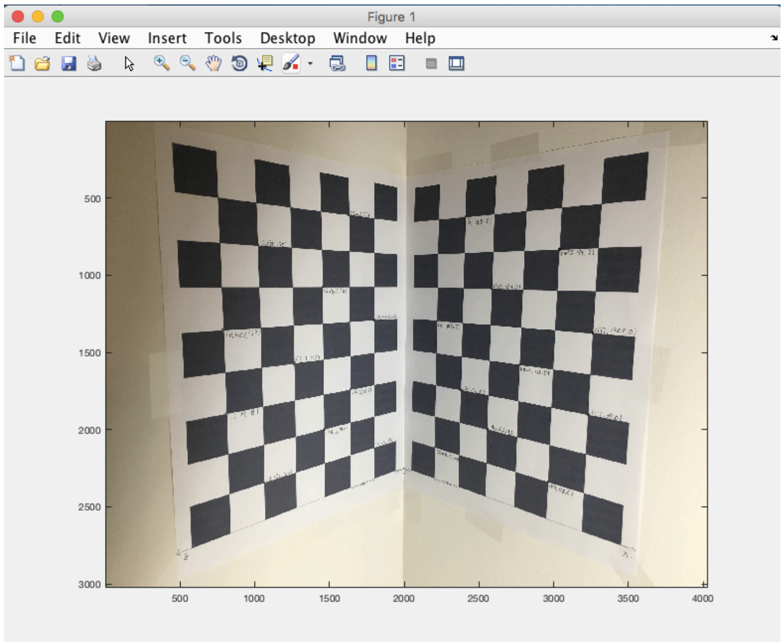

Figure 2, Read image into matlab and get pixel coordinates.

```
>> [x,y]

ans =

   1.0e+03 *

    2.2009    2.1234
    2.9603    2.3434
    2.5589    2.0017
    2.3817    1.7442
    3.2315    1.9034
    2.7723    1.5851
    2.2190    1.2995
    3.2605    1.3744
    2.5879    1.0935
    3.0399    0.8361
    2.4107    0.6301
    1.7995    2.1187
    1.0655    2.3434
    1.4668    1.9970
    1.6404    1.7349
    0.8123    1.8987
    1.2643    1.5710
    1.7995    1.2901
    0.7906    1.3557
    1.4560    1.0795
    1.0221    0.8080
    1.6296    0.6067
```

Figure 3, pixel coordinates

```
>> pw

pw =

   38.0000   27.6000         0    1.0000
  149.0000   27.8000         0    1.0000
   93.2000   55.2000         0    1.0000
   65.5000   83.0000         0    1.0000
  177.0000   83.0000         0    1.0000
  121.4000  111.0000         0    1.0000
   38.0000  138.8000         0    1.0000
  177.0000  138.8000         0    1.0000
   93.5000  166.2000         0    1.0000
  149.5000  194.0000         0    1.0000
   66.0000  221.8000         0    1.0000
         0   27.6000   39.0000    1.0000
         0   27.6000  150.2000    1.0000
         0   55.3000   94.4000    1.0000
         0   83.0000   66.5000    1.0000
         0   83.0000  178.0000    1.0000
         0  111.0000  122.0000    1.0000
         0  138.8000   38.8000    1.0000
         0  138.8000  177.8000    1.0000
         0  166.5000   94.0000    1.0000
         0  194.0000  150.0000    1.0000
         0  222.0000   66.0000    1.0000
```

Figure 4, world coordinates

Next step I use x, y and pw matrix to form the big **p** matrix, which is a 44*12 matrix. See figure 5.

```
p =

   1.0e+05 *

    0.0004    0.0003         0    0.0000         0         0         0         0   -0.8364   -0.6075         0   -0.0220
         0         0         0         0    0.0004    0.0003         0    0.0000   -0.8069   -0.5861         0   -0.0212
    0.0015    0.0003         0    0.0000         0         0         0         0   -4.4109   -0.8230         0   -0.0296
         0         0         0         0    0.0015    0.0003         0    0.0000   -3.4917   -0.6515         0   -0.0234
    0.0009    0.0006         0    0.0000         0         0         0         0   -2.3849   -1.4125         0   -0.0256
         0         0         0         0    0.0009    0.0006         0    0.0000   -1.8656   -1.1049         0   -0.0200
    0.0007    0.0008         0    0.0000         0         0         0         0   -1.5600   -1.9768         0   -0.0238
         0         0         0         0    0.0007    0.0008         0    0.0000   -1.1425   -1.4477         0   -0.0174
    0.0018    0.0008         0    0.0000         0         0         0         0   -5.7198   -2.6822         0   -0.0323
         0         0         0         0    0.0018    0.0008         0    0.0000   -3.3690   -1.5798         0   -0.0190
    0.0012    0.0011         0    0.0000         0         0         0         0   -3.3655   -3.0772         0   -0.0277
         0         0         0         0    0.0012    0.0011         0    0.0000   -1.9243   -1.7594         0   -0.0159
    0.0004    0.0014         0    0.0000         0         0         0         0   -0.8432   -3.0800         0   -0.0222
         0         0         0         0    0.0004    0.0014         0    0.0000   -0.4938   -1.8037         0   -0.0130
    0.0018    0.0014         0    0.0000         0         0         0         0   -5.7710   -4.5255         0   -0.0326
         0         0         0         0    0.0018    0.0014         0    0.0000   -2.4327   -1.9077         0   -0.0137
    0.0009    0.0017         0    0.0000         0         0         0         0   -2.4196   -4.3010         0   -0.0259
         0         0         0         0    0.0009    0.0017         0    0.0000   -1.0225   -1.8175         0   -0.0109
    0.0015    0.0019         0    0.0000         0         0         0         0   -4.5446   -5.8973         0   -0.0304
         0         0         0         0    0.0015    0.0019         0    0.0000   -1.2499   -1.6220         0   -0.0084
    0.0007    0.0022         0    0.0000         0         0         0         0   -1.5910   -5.3468         0   -0.0241
         0         0         0         0    0.0007    0.0022         0    0.0000   -0.4159   -1.3976         0   -0.0063
         0    0.0003    0.0004    0.0000         0         0         0         0         0   -0.4967   -0.7018   -0.0180
         0         0         0         0         0    0.0003    0.0004    0.0000         0   -0.5848   -0.8263   -0.0212
         0    0.0003    0.0015    0.0000         0         0         0         0         0   -0.2941   -1.6003   -0.0107
         0         0         0         0         0    0.0003    0.0015    0.0000         0   -0.6468   -3.5198   -0.0234
         0    0.0006    0.0009    0.0000         0         0         0         0         0   -0.8112   -1.3847   -0.0147
         0         0         0         0         0    0.0006    0.0009    0.0000         0   -1.1043   -1.8852   -0.0200
         0    0.0008    0.0007    0.0000         0         0         0         0         0   -1.3615   -1.0909   -0.0164
         0         0         0         0         0    0.0008    0.0007    0.0000         0   -1.4399   -1.1537   -0.0173
         0    0.0008    0.0018    0.0000         0         0         0         0         0   -0.6742   -1.4459   -0.0081
```

Figure 5, part of the p matrix

Then I use **p** matrix to get **M** matrix and make M matrix from 12*1 to 3*4 matrix. See figure 6.

```
>> m

m =

    0.0008
   -0.0002
   -0.0026
    0.6645
   -0.0004
   -0.0024
   -0.0005
    0.7473
   -0.0000
   -0.0000
   -0.0000
    0.0003
```

```
>> [m1;m2;m3]

ans =

    0.0008   -0.0002   -0.0026    0.6645
   -0.0004   -0.0024   -0.0005    0.7473
   -0.0000   -0.0000   -0.0000    0.0003
```

Figure 6, M matrix

Next I extract intrinsic and extrinsic parameters (in my implementation ε is 1), including α, β, θ, u_0, v_o and R, t matrix.

At last, I use those parameters and function: $\begin{cases} u = \frac{m_1 p}{m_3 p} \\ v = \frac{m_2 p}{m_3 p} \end{cases}$ to reconstruct image coordinates using my estimate of the calibration matrix. More detail will be showed in following part.

Part 2 Intrinsic parameter

Specification of camera sensor	
sensor model	Sony IMX315 Exmor Rs
sensor size	4.8 * 3.6 mm
image resolution	4032 * 3024 pixels
focal length	4.02 mm

Name of parameter	Value	Assessment
u_0 (unit: mm)	2003.5	very close to image pixel center 's x coordinate 2000
v_0 (unit: mm)	1513.9	very close to image pixel center 's y coordinate 1500
α (unit: pixel/mm)	3447.7	α is the magnification of the density of the pixel, so α = kf, according to the table above, we know k = 4032/4.8 = 840, so the α value I calculated is 840 * 4.02 = 3376.8 which is close to 3447.7.
β (unit: pixel/mm)	3441.9	β is the magnification of the density of the pixel, so β = lf, according to the table above, we know l = 3024/3.6 = 840, so the β value I calculated is 840 * 4.02 = 3376.8 which is close to 3441.9.
θ	1.5759	θ is 90.29°, very close to 90°

So every parameter I got is close to the value they should be, I am feel good about my result.

Part 3 Extrinsic parameter

- R matrix = $\begin{bmatrix} 0.7282 & 0.0130 & -0.6852 \\ 0.1046 & -0.9902 & 0.0925 \\ -0.6773 & -0.1391 & -0.7225 \end{bmatrix}$

According to the rotation matrix decomposing function:

$$\theta_x = atan2(r_{32}, r_{33})$$

$$\theta_y = atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2})$$

$$\theta_z = atan2(r_{21}, r_{11})$$

So $\theta_x = -2.9514$, $\theta_y = 0.7441$, $\theta_z = 0.1427$

$\Rightarrow \theta_x = -169.1^{\circ}$ $\quad \theta_y = 42.63^{\circ}$ $\quad \theta_z = 8.176^{\circ}$

Those result means that my camera rotate −169.1 degree on x axis, rotate 42.63 degree on y axis and almost no rotation on z axis, which is good, because my camera almost face the checkboard, so there should be a small angle on z axis. (Note, in my frame, x axis is towards left wall, y axis is towards up to the wall and z axis is towards to the right of wall, you can check it in figure 1)

- $t = [68.6423, 104.9504, 493.0914]^T$

  Note: camera was roughly 10.5cm from the floor, 50 cm back from pattern. In the practical operation, the high of the camera is 11 cm, and 47 cm back from pattern, so they are very close.

Part 4 Reconstruct image coordinates

In order to reconstruct image coordinates from world coordinated with my estimate parameters, I still use the 22 point I choosed at beginning. And I use function $\begin{cases} u = \frac{m_1 p}{m_3 p} \\ v = \frac{m_2 p}{m_3 p} \end{cases}$ to recalculate image coordinate of each point. See figure 7 to check the result. (continues in next page)

```
>> [x,y]

ans =

   1.0e+03 *

    2.2009    2.1234
    2.9603    2.3434
    2.5589    2.0017
    2.3817    1.7442
    3.2315    1.9034
    2.7723    1.5851
    2.2190    1.2995
    3.2605    1.3744
    2.5879    1.0935
    3.0399    0.8361
    2.4107    0.6301
    1.7995    2.1187
    1.0655    2.3434
    1.4668    1.9970
    1.6404    1.7349
    0.8123    1.8987
    1.2643    1.5710
    1.7995    1.2901
    0.7906    1.3557
    1.4560    1.0795
    1.0221    0.8080
    1.6296    0.6067
```

```
>> [x_new, y_new]

ans =

   1.0e+03 *

    2.2049    2.1199
    2.9621    2.3384
    2.5547    2.0033
    2.3785    1.7471
    3.2314    1.9069
    2.7752    1.5813
    2.2186    1.2949
    3.2626    1.3781
    2.5853    1.0921
    3.0396    0.8394
    2.4082    0.6256
    1.7992    2.1203
    1.0666    2.3412
    1.4660    2.0001
    1.6399    1.7435
    0.8111    1.8965
    1.2644    1.5698
    1.8000    1.2908
    0.7906    1.3540
    1.4556    1.0754
    1.0204    0.8090
    1.6353    0.6108
```

Figure 7, the rough compare between original and new image coordinates

In order to get a more precise comparison between original image coordinates and new image coordinates, I calculate the percentage difference between each point pair. See figure 8.

```
>> 100*error_ratio

ans =

    0.1787   -0.1658
    0.0603   -0.2124
   -0.1637    0.0833
   -0.1358    0.1669
   -0.0037    0.1830
    0.1062   -0.2348
   -0.0183   -0.3538
    0.0671    0.2694
   -0.1003   -0.1292
   -0.0082    0.3981
   -0.1003   -0.7190
   -0.0205    0.0759
    0.1033   -0.0922
   -0.0574    0.1539
   -0.0315    0.4996
   -0.1458   -0.1162
    0.0075   -0.0765
    0.0275    0.0498
    0.0011   -0.1216
   -0.0249   -0.3798
   -0.1645    0.1193
    0.3510    0.6794
```

Figure 8, the percentage difference matrix, eg, 0.1787 means 0.1787%

From the matrix above, we can see no percentage difference is lager than 1%. So the calibration is plausible.