Hw5 report

Subowen Yan
A20430537
CS512-Summer

1 problem statement

a) Write a program to extract feature points from the calibration target and show them on the image. You may use the OpenCV functions to do so.
b) Compute camera parameters using non-planar calibration or planar calibration. Compute MSE.
c) Implement the RANSAC algorithm for robust estimation.

2 proposed solution

For extract feature points, I will use OpenCV function, which contains cv2.findChessboardCorners, cv.cornerSubPix, cv.drawChessboardCorners, and so on.

For compute parameters, I will implement non-planar calibration. I will use following formula:

Non-coplanar calibration

Parameter equations

$$
\begin{aligned}
|\rho| &= 1/|a_3| \\
u_0 &= |\rho|^2 a_1 \cdot a_3 \\
v_0 &= |\rho|^2 a_2 \cdot a_3 \\
\alpha_v &= \sqrt{|\rho|^2 a_2 \cdot a_2 - v_0^2} \\
s &= |\rho|^4/\alpha_v (a_1 \times a_3) \cdot (a_2 \times a_3) \\
\alpha_u &= \sqrt{|\rho|^2 a_1 \cdot a_1 - s^2 - u_0^2} \\
K^* &= \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\
\epsilon &= \operatorname{sgn}(b_3) \\
T^* &= \epsilon |\rho| (K^*)^{-1} b \\
r_3 &= \epsilon |\rho| a_3 \\
r_1 &= |\rho|^2/\alpha_v a_2 \times a_3 \\
r_2 &= r_3 \times r_1 \\
R^* &= [r_1^T \;\; r_2^T \;\; r_3^T]^T
\end{aligned}
$$

For finding MSE, I will use following formula:

$$mean\ square\ error = \frac{\sum_{i=1}^{n}(x_i - \frac{m_1^T p_i}{m_3^T P_i})^2 + (y_i - \frac{m_2^T p_i}{m_3^T P_i})^2}{n}$$

For RANSAC algorithm, I will do following:

1) Compute matrix with random points.
2) Estimate the image point using the matrix.
3) Compute the distance between estimated points and the true one.
4) Find inliers that is smaller than 1.5 times the median
5) Recompute the inliers.

3 implementation detail

For extract feature points, I will use OpenCV function, which contains cv2.findChessboardCorners, cv.cornerSubPix, cv.drawChessboardCorners, and so on. I will save 3D and 2D points into two files.

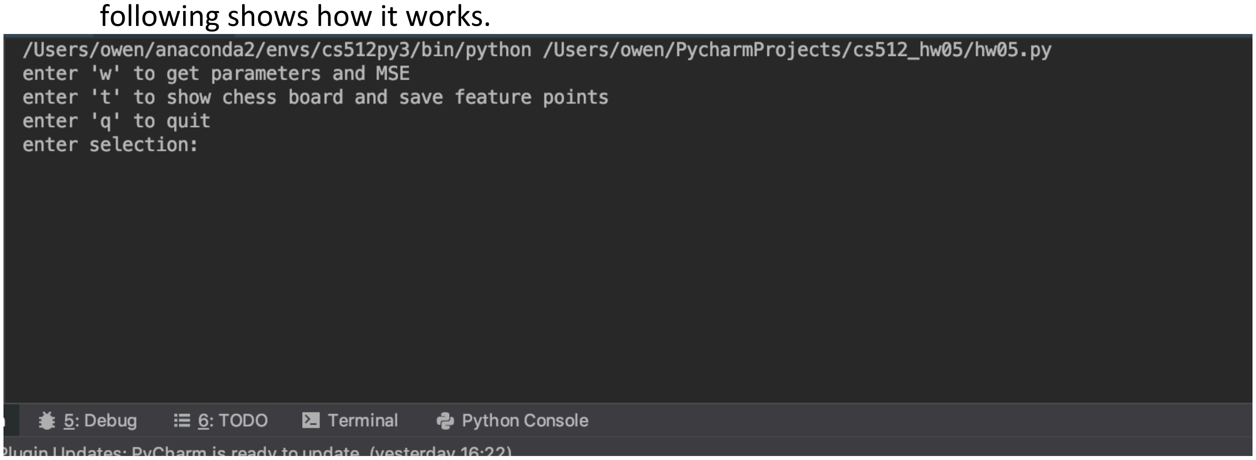
When implement the non-planar calibration, I sometimes forget to put transpose. It takes some time to figure out the right matrix to put. I will read two files separately. Put it into an array. I will make it as matrix as follow.

For calculate MSE, I will just follow the formula to get the result.

For RANSAC.config, I put 0.99 as probability of success. 600 as max number of drawing. 8 as minimum number of points to fit model. 12 as maximum number of points to fit model. The reason why I choose 8 as minimum number points is that there are 8 parameters for calibration. Therefore, I need at least 8 points for non-planar calibration

4 results and discussion

1) I put show feature points and calculate non-planar calibration into one file. The following shows how it works.



2) Press ‘w’ to get parameters and MSE. I will input world.txt, which is 3D point, and image.txt, which is 2D point.

```
/Users/owen/anaconda2/envs/cs512py3/bin/python /Users/owen/PycharmProjects/cs512_hw05/hw05.py
enter 'w' to get parameters and MSE
enter 't' to show chess board and save feature points
enter 'q' to quit
enter selection: w
please enter world file nameworld.txt
please enter image file nameimage.txt
```

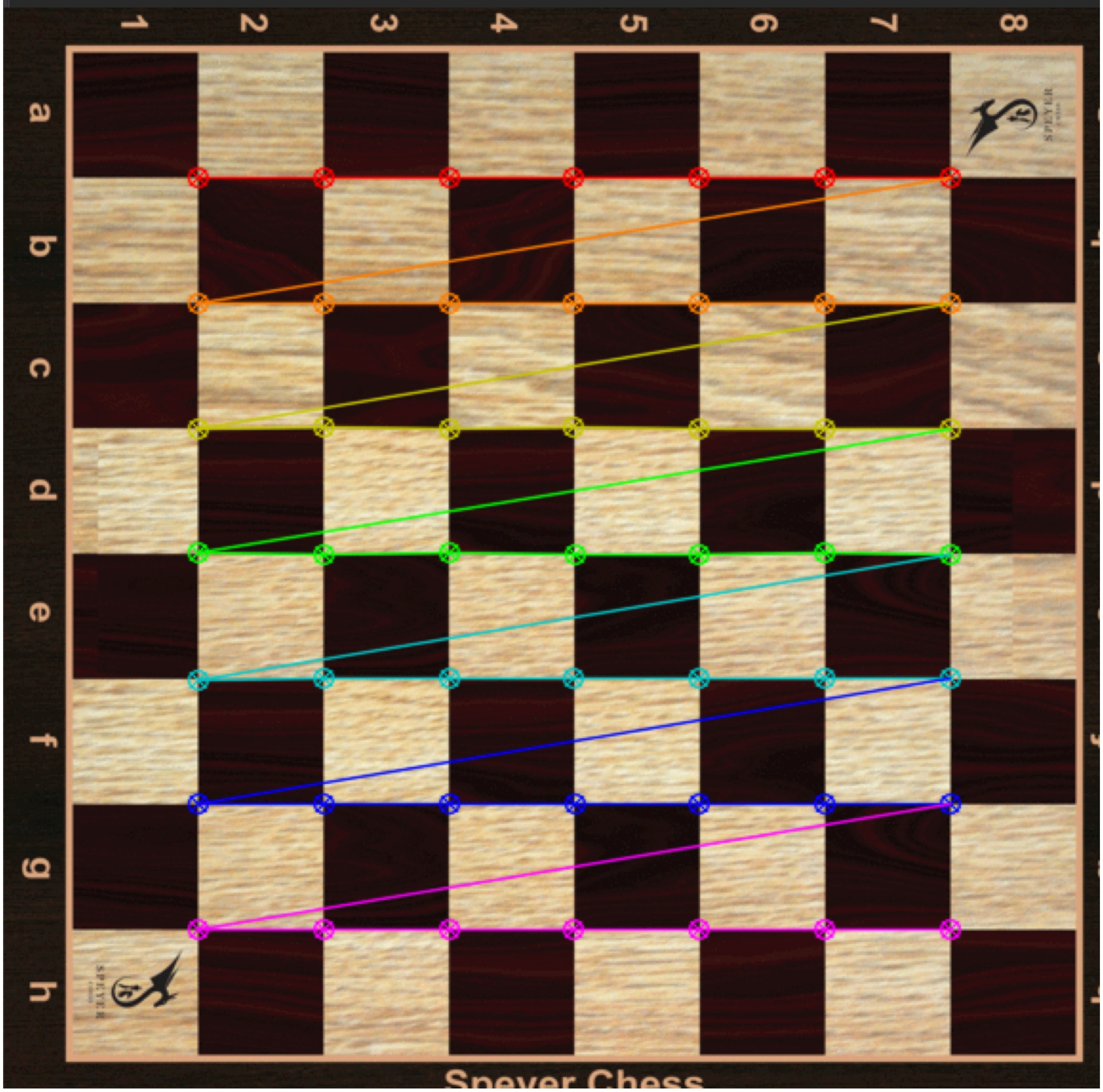
3) The following shows result from dataset that is provided by professor. It is almost identical with provided solutions.

```
--------------------------

u0 = 320.000170

v0 = 239.999971

Alpha_u = 652.174069

Alpha_v = 652.174075

s = -0.000034

T* = [[-2.57726950e-04  3.26846051e-05  1.04880905e+03]]

R* = [[-7.68221190e-01  6.40184508e-01  1.46359878e-07]
 [ 4.27274298e-01  5.12729182e-01 -7.44678091e-01]
 [-4.76731452e-01 -5.72077427e-01 -6.67423808e-01]]


--------------------------


--------------------------

MSE = 1.6605412420597685e-09


--------------------------
```

```
(u0,v0)           = (320.00,240.00)
(alphaU,alphaV)   = (652.17,652.17)
s                 = 0.0
T*                = (0.0,0.0,1048.81)
R*                = (-0.768221, 0.640184, 0.000000)
                    ( 0.427274, 0.512729,-0.744678)
                    (-0.476731,-0.572078,-0.667424)
```

4) Press ‘t’ to show feature point and save the points.
   It will need to input the picture name.

```
enter 'w' to get parameters and MSE
enter 't' to show chess board and save feature points
enter 'q' to quit
enter selection: t
enter a filenamechessboard.jpg
press 'x' to quit
```



5) Following shows the 3D and 2D points that being saved from reading picture.

```
0. 0. 0.
1. 0. 0.
2. 0. 0.
3. 0. 0.
4. 0. 0.
5. 0. 0.
6. 0. 0.
0. 1. 0.
1. 1. 0.
2. 1. 0.
3. 1. 0.
4. 1. 0.
5. 1. 0.
6. 1. 0.
0. 2. 0.
1. 2. 0.
2. 2. 0.
3. 2. 0.
4. 2. 0.
5. 2. 0.
6. 2. 0.
0. 3. 0.
1. 3. 0.
2. 3. 0.
3. 3. 0.
```

```
101.97062   101.81464
167.7842    102.20011
233.51198   101.80129
299.4804    101.821304
365.34464   101.89763
431.34677   102.20421
497.06433   101.68491
101.9381    167.76044
167.72296   167.8817
233.53123   168.09457
299.4663    167.89478
365.44785   167.81993
431.3179    167.61548
496.95724   167.67838
101.82289   233.6959
167.79456   233.31598
233.55511   233.58151
299.4207    233.47214
365.45483   233.56068
431.28812   233.60762
497.13657   233.74062
101.88363   299.19797
167.73453   299.58664
233.56876   299.45837
299.51657   299.54575
365.4016    299.57367
431.25317   299.45566
497.0172    299.5348
101.84847   365.605
```

6) Following shows the parameter of RANSAC.config

```
0.99    #probability
600     #max number of drawing
8   #minimum number of points to fit model
12  #maximum number of points to fit model
```

7) Following shows the first noise file's results. As we can see, some numbers are similar to the result that provided by professor. However, some numbers are dramatically different.

```
please enter world file namencc-worldPt.txt
please enter image file namenoise-0.txt
please enter config file nameRANSAC.config

--------------------------

u0 = 356.133342

v0 = 256.114362

Alpha_u = 653.210798

Alpha_v = 664.264595

s = 3.074968

T* = [[ -55.88612839  -28.60532729 1063.27851955]]

R* = [[-0.75105233  0.65927652  0.03570524]
 [ 0.43743163  0.53737472 -0.72102842]
 [-0.4945442  -0.52591147 -0.69198494]]


--------------------------
```

```
(u0,v0)          = (320.00,240.00)
(alphaU,alphaV) = (652.17,652.17)
s                = 0.0
T*               = (0.0,0.0,1048.81)
R*               = (-0.768221, 0.640184, 0.000000)
                   ( 0.427274, 0.512729,-0.744678)
                   (-0.476731,-0.572078,-0.667424)
```

8) Following shows the second noise file's results. As we can see, all of numbers are similar to the result that provided by professor.

```
please enter world file namencc-worldPt.txt
please enter image file namenoise-1.txt
please enter config file nameRANSAC.config

--------------------------

u0 = 320.000615

v0 = 239.999447

Alpha_u = 652.171480

Alpha_v = 652.171629

s = -0.000549

T* = [[-1.01810619e-03  7.06938255e-04  1.04880595e+03]]

R* = [[-7.68220532e-01  6.40185297e-01  2.56265067e-07]
 [ 4.27274658e-01  5.12728669e-01 -7.44678238e-01]
 [-4.76732191e-01 -5.72077003e-01 -6.67423645e-01]]


--------------------------
```

```
-----------------
(u0,v0)         = (320.00,240.00)
(alphaU,alphaV) = (652.17,652.17)
s               = 0.0
T*              = (0.0,0.0,1048.81)
R*              = (-0.768221, 0.640184, 0.000000)
                  ( 0.427274, 0.512729,-0.744678)
                  (-0.476731,-0.572078,-0.667424)
```

9) I dig into the data that provided by professor. The first one is no noise data. The second one is first noise data. The third one is second noise data. As we can see, the first and third data almost identical. There is a huge amount of difference between first and second. This is the reason why the second noise results are preforming better.

Following shows Original data

```
214.9064 298.4516
222.4942 306.2609
230.2685 314.2623
238.2363 322.4629
246.4051 330.8702
254.7824 339.4921
263.3763 348.3371
272.1954 357.4137
281.2487 366.7314
290.5455 376.2997
300.0959 386.1290
312.1278 377.9196
323.8924 369.8925
335.3983 362.0418
346.6542 354.3619
357.6679 346.8470
368.4474 339.4921
378.9999 332.2920
389.3326 325.2419
399.4522 318.3372
409.3653 311.5734
211.8791 279.1739
219.6502 286.9701
227.6182 294.9635
235.7904 303.1620
```

Following shows noise 0 data

```
214.2627 297.7435
226.6419 306.8645
228.3564 314.3918
239.4772 323.9749
245.2069 329.7931
251.7734 339.0150
261.5213 347.0148
272.9432 357.1872
281.9947 368.0102
291.8626 375.3146
298.1362 388.3383
311.3673 379.1758
323.2379 365.9815
335.3904 361.6404
346.5074 354.8716
360.2696 347.9236
369.5172 338.1977
379.8929 333.6778
387.4355 325.3209
400.9237 318.4992
407.9839 310.0333
212.0901 278.1616
219.1443 287.6034
227.2039 295.1391
234.1477 302.5981
```

Following shows noise 1 data

```
214.9064 298.4516
227.8209 309.2874
230.2685 314.2623
238.2363 322.4629
246.4051 330.8702
254.7824 339.4921
263.3763 348.3371
272.1954 357.4137
281.2487 366.7314
290.5455 376.2997
294.6623 387.4211
312.2952 401.1928
323.8924 369.8925
335.3983 362.0418
346.6542 354.3619
357.6679 346.8470
368.4474 339.4921
378.9999 332.2920
389.3326 325.2419
399.4522 318.3372
409.3653 311.5734
211.8791 279.1739
219.6502 286.9701
227.6182 294.9635
235.7904 303.1620
244.1749 311.5734
252.7801 320.2062
261.6147 329.0691
270.6881 338.1716
```

5 references

https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/numpy.random.choice.html

https://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

https://www.geeksforgeeks.org/numpy-dot-python/

https://www.pythonforbeginners.com/files/with-statement-in-python

https://www.geeksforgeeks.org/zip-in-python/

https://www.geeksforgeeks.org/reading-writing-text-files-python/