512-AS4-REPROT

Chen Xu

A20377739

1. Problem statement

1. Implement a camera calibration algorithm under the assumption of noisy data.
2. Implement non-coplanar calibration.
3. Extract feature points from the calibration target and show them on image.
4. Output extracted points to a file with correspondence points(3D-2D).
5. Use a point correspondence file as argument to compute the camera parameter.
6. Compute and display the mean square error between the known and computed position of the image points.
7. Implement the RANSAC algorithm for robust estimation.
8. The final values of the estimation should be displayed by the program.
9. Parameters used in the RANSAC algorithm should be read from a text file named “RANSAC.config”.

2. Proposed solution

1. Use OpenCV functions: cvFindChessboardCorners, cvFindCornerSubPix, cvDrawChessBoardCorners, to extract feature points from the calibration target.

2. non-planar-calibration parameter equations:

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

3. mean square error:

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

4. Implement the RANSAC algorithm for robust estimation.

RANSAC algorithm:

Step 1: Draw n points uniformly at random with replacement

Step 2: Fit model to points

Step 3: Find all inliers (distance smaller than t)

Step 4: If there are d inliers, recompute the model.

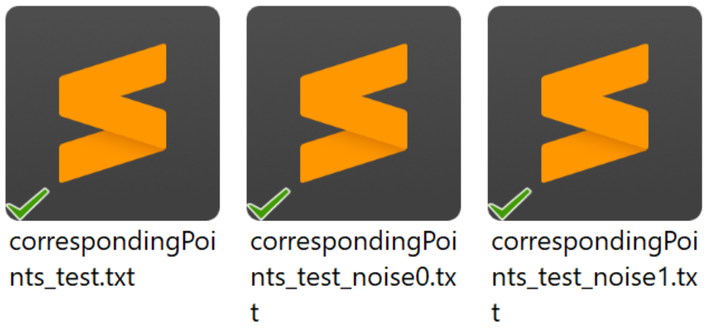
Step 5: Finally choose best solution, when it has the largest number of inliers.

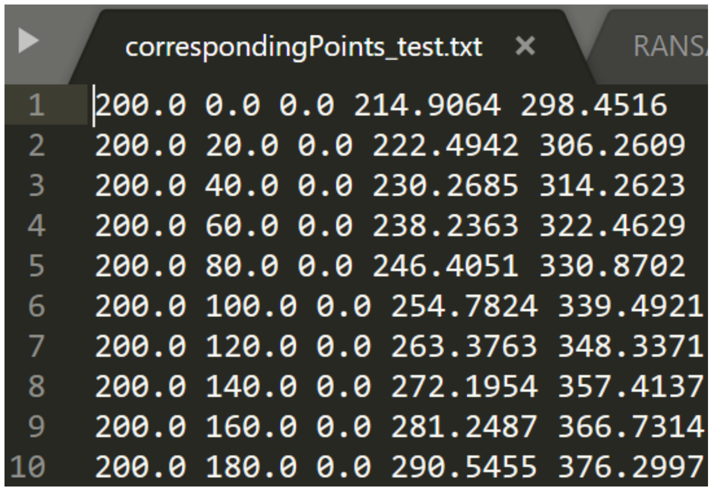
3. Implement details

1. Use zip() function to iterate two list at same time ,help you compute easier.
2. When process numpy array as matrix reshape() function may cause more bucket in following data processing.
3. When take parameter into some function, we need to know the type and change it into right type to compute.
4. I find out we don't have to write a help function, it also work, if I write comment and use print(__doc__) to display it at program running.
5. np.concatenate() make it easier to combine two np array.

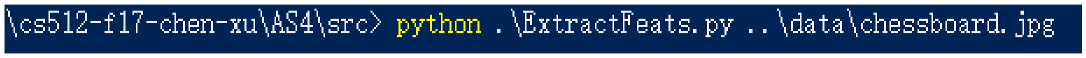
4. Results and discussion

0. combine the data professor provided to test the program. Change them into 3D-2D correspondence file.


correspondingPoints_test.txt
correspondingPoints_test_noise0.txt
correspondingPoints_test_noise1.txt

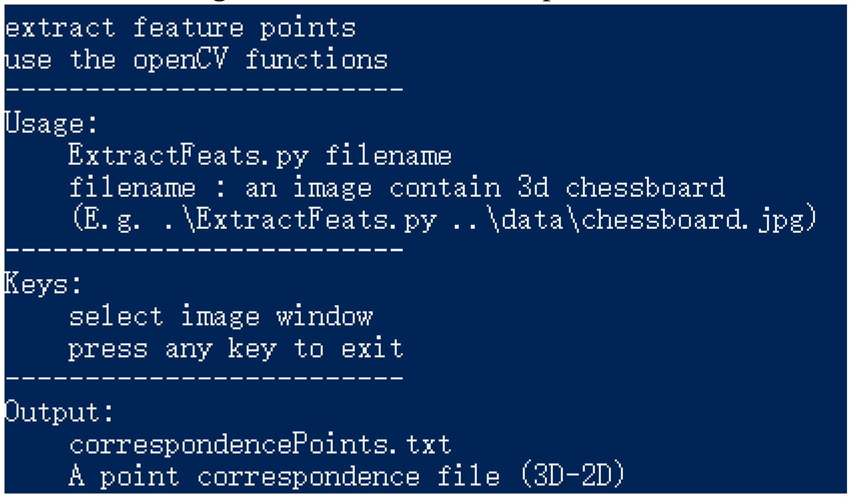


```
200.0 0.0 0.0 214.9064 298.4516
200.0 20.0 0.0 222.4942 306.2609
200.0 40.0 0.0 230.2685 314.2623
200.0 60.0 0.0 238.2363 322.4629
200.0 80.0 0.0 246.4051 330.8702
200.0 100.0 0.0 254.7824 339.4921
200.0 120.0 0.0 263.3763 348.3371
200.0 140.0 0.0 272.1954 357.4137
200.0 160.0 0.0 281.2487 366.7314
200.0 180.0 0.0 290.5455 376.2997
```

1. 

```
\cs512-f17-chen-xu\AS4\src> python .\ExtractFeats.py ..\data\chessboard.jpg
```

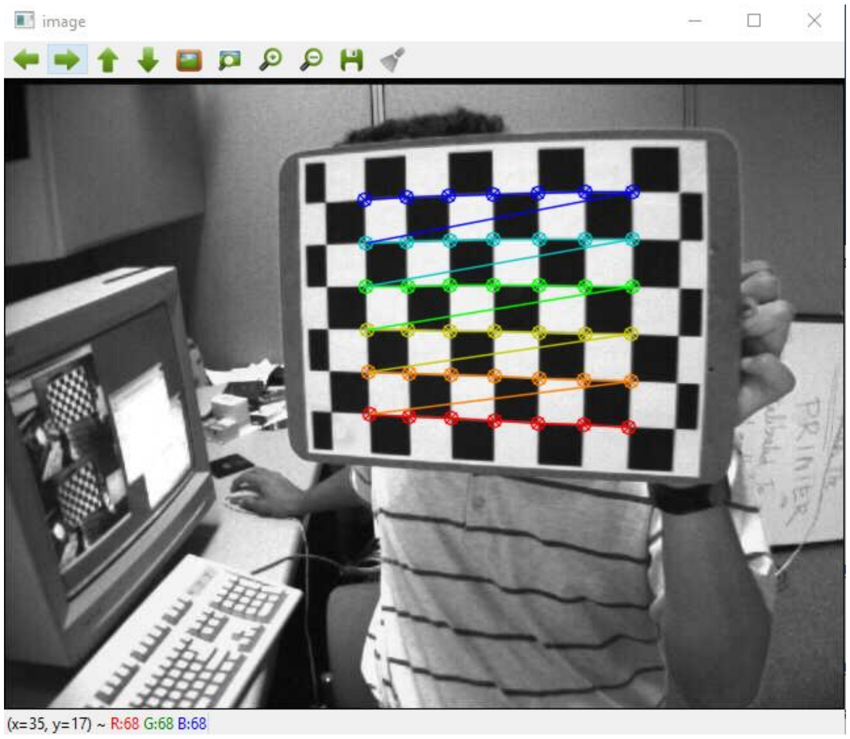
Take in one image and extract the feature points



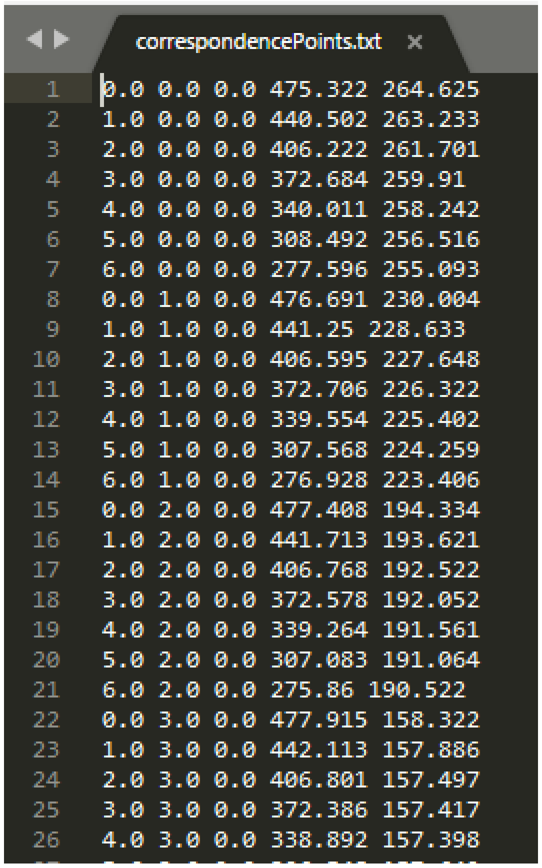
```
extract feature points
use the openCV functions
-------------------------
Usage:
    ExtractFeats.py filename
    filename : an image contain 3d chessboard
    (E.g.  .\ExtractFeats.py ..\data\chessboard.jpg)
-------------------------
Keys:
    select image window
    press any key to exit
-------------------------
Output:
    correspondencePoints.txt
    A point correspondence file (3D-2D)
```

Helps displayed as the program running.

Output a txt file with correspondence points(3D-2D).

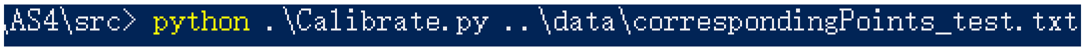


Displayed feature points.



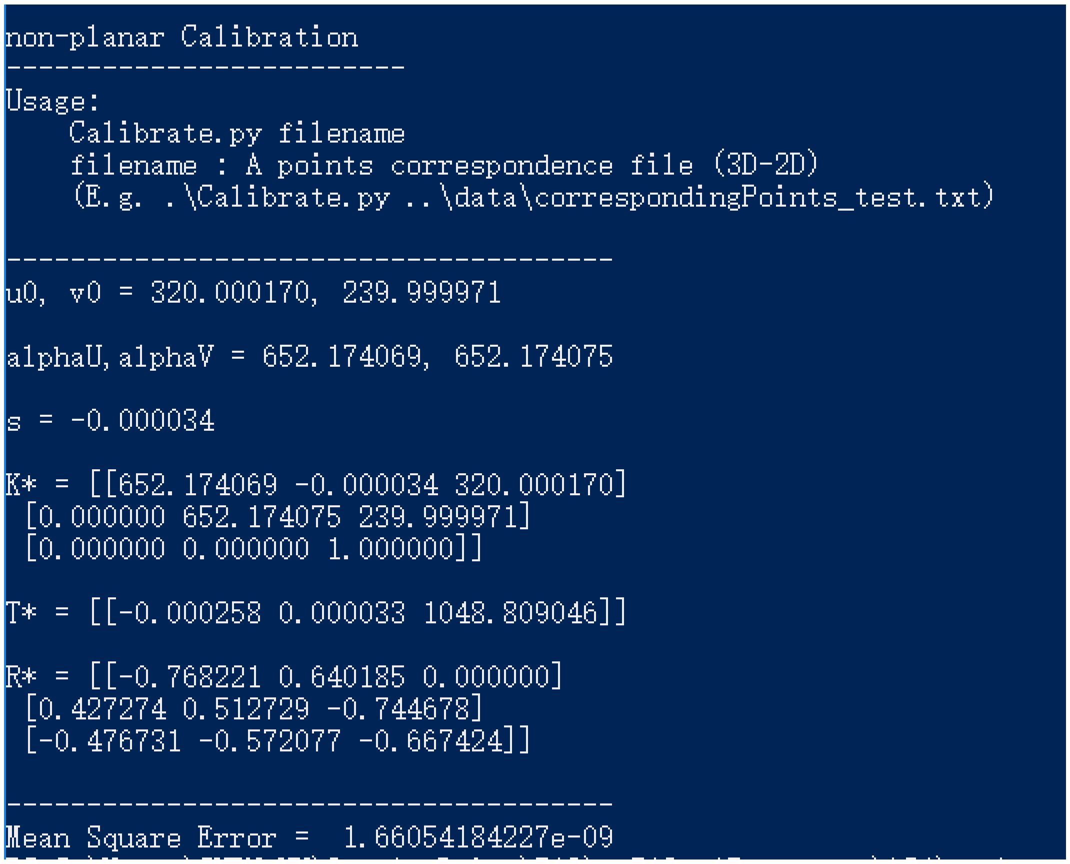
```
0.0 0.0 0.0 475.322 264.625
1.0 0.0 0.0 440.502 263.233
2.0 0.0 0.0 406.222 261.701
3.0 0.0 0.0 372.684 259.91
4.0 0.0 0.0 340.011 258.242
5.0 0.0 0.0 308.492 256.516
6.0 0.0 0.0 277.596 255.093
0.0 1.0 0.0 476.691 230.004
1.0 1.0 0.0 441.25 228.633
2.0 1.0 0.0 406.595 227.648
3.0 1.0 0.0 372.706 226.322
4.0 1.0 0.0 339.554 225.402
5.0 1.0 0.0 307.568 224.259
6.0 1.0 0.0 276.928 223.406
0.0 2.0 0.0 477.408 194.334
1.0 2.0 0.0 441.713 193.621
2.0 2.0 0.0 406.768 192.522
3.0 2.0 0.0 372.578 192.052
4.0 2.0 0.0 339.264 191.561
5.0 2.0 0.0 307.083 191.064
6.0 2.0 0.0 275.86 190.522
0.0 3.0 0.0 477.915 158.322
1.0 3.0 0.0 442.113 157.886
2.0 3.0 0.0 406.801 157.497
3.0 3.0 0.0 372.386 157.417
4.0 3.0 0.0 338.892 157.398
```

Output a txt file with correspondence points(3D-2D).

2. 

```
,AS4\src> python .\Calibrate.py ..\data\correspondingPoints_test.txt
```

Input a file with correspondence points(3D-2D).

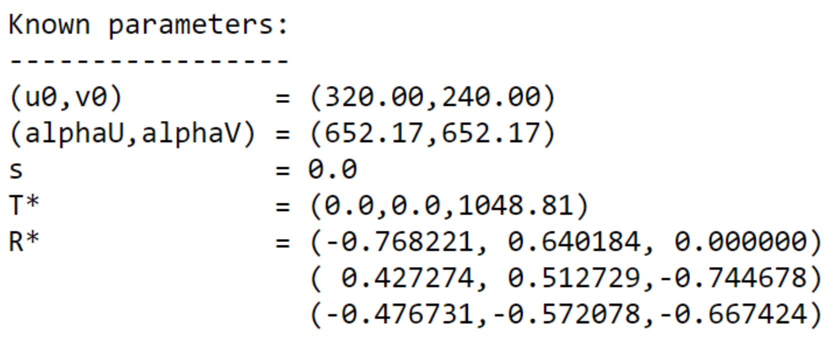


```
non-planar Calibration
-------------------------
Usage:
    Calibrate.py filename
    filename : A points correspondence file (3D-2D)
    (E.g. .\Calibrate.py ..\data\correspondingPoints_test.txt)

--------------------------------------
u0, v0 = 320.000170, 239.999971

alphaU,alphaV = 652.174069, 652.174075

s = -0.000034

K* = [[652.174069 -0.000034 320.000170]
 [0.000000 652.174075 239.999971]
 [0.000000 0.000000 1.000000]]

T* = [[-0.000258 0.000033 1048.809046]]

R* = [[-0.768221 0.640185 0.000000]
 [0.427274 0.512729 -0.744678]
 [-0.476731 -0.572077 -0.667424]]

--------------------------------------
Mean Square Error =  1.66054184227e-09
```

1). Display the usage while the program running.

2). Compute and display camera parameters

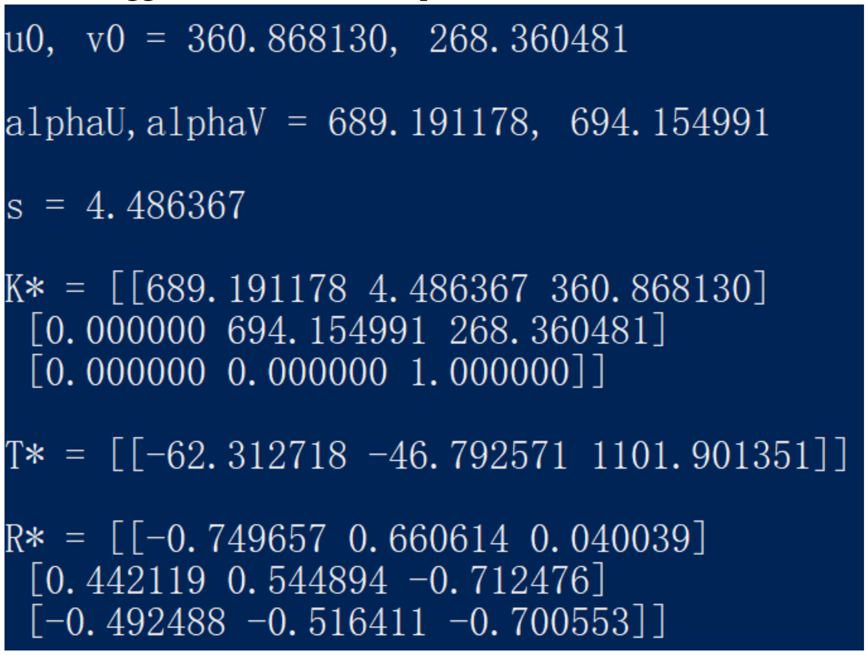
3). Compute and display the mean square error.



```
Known parameters:
-----------------
(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)
```

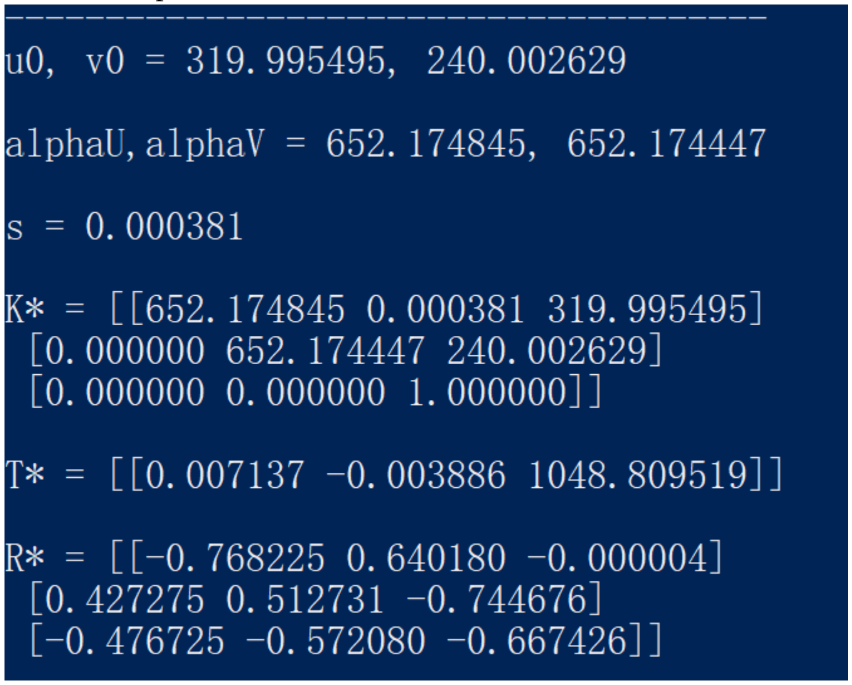
This result is same as result that professor provided.

3. Parameter influences (RANSAC.config):
   Prob: higher p comes with higher k.
   Nmin: too small cause error, too big will contain too many outliers.
   Nmax：too big will contain too many outliers, comes with wrong results.
   Kmax: bigger make result more precise.



```
u0,  v0 = 360.868130,  268.360481

alphaU,alphaV = 689.191178,  694.154991

s = 4.486367

K* = [[689.191178 4.486367 360.868130]
 [0.000000 694.154991 268.360481]
 [0.000000 0.000000 1.000000]]

T* = [[-62.312718 -46.792571 1101.901351]]

R* = [[-0.749657 0.660614 0.040039]
 [0.442119 0.544894 -0.712476]
 [-0.492488 -0.516411 -0.700553]]
```

For "noise_0" the result comes randomly, and can't come to the same result like known parameter.



```
-------------------------------------
u0,  v0 = 319.995495,  240.002629

alphaU,alphaV = 652.174845,  652.174447

s = 0.000381

K* = [[652.174845 0.000381 319.995495]
 [0.000000 652.174447 240.002629]
 [0.000000 0.000000 1.000000]]

T* = [[0.007137 -0.003886 1048.809519]]

R* = [[-0.768225 0.640180 -0.000004]
 [0.427275 0.512731 -0.744676]
 [-0.476725 -0.572080 -0.667426]]
```

For "noise_1" the result comes to the same as the professor provided parameter.

The “RANSAC” not handling well with the “noise_0”, but work well with “noise_1”.

Because the noise of “noise_0” is more than 50%, therefore, the robust RANSAC not handling well with it.

5. Reference

[1]. Camera Calibration

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

[2] non-planar-calibration

http://www.cs.iit.edu/~agam/cs512/share/non-planar-calibration.pdf

[3] Camera Calibration and 3D Reconstruction

https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html